A Exercises

1 Exercises for “Introduction”

1-1 Link between spatial distribution and distribution in terms of control- and implementation structure.

Information technical systems (IT-systems) that are spatially distributed are also exclusively realized in a distributed way regarding their control- and implementation structure. Why?
(As explanation: *control structure* = Which instances are issuing orders, instructions etc. to other instances; Convince yourself whether or not to execute and how. *implementation structure* = How are these instances implemented, e. g., many on one computer, e. g., single instances on many cooperating computers, etc.)
Solution at page 447.

1-2 Advantages and disadvantages of a distributed system in terms of security

a) Which advantages and disadvantages does a distributed system offer in terms of security?

b) Which security properties are supported by spatial distribution, and which by distribution in terms of control- and implementation structure?

Solution at page 447.

1-3 Advantages and disadvantages of an open system in terms of security

a) Which advantages and disadvantages does an open system offer in terms of security?

b) How are these advantages and disadvantages combined in terms of security in open distributed systems?

Solution at page 448.

1-4 Advantages and disadvantages of a service-integrating system in terms of security

a) Which advantages and disadvantages does a service-integrating system offer in terms of security?

b) How are these advantages and disadvantages in terms of security combined in open distributed service-integrating systems?

Solution at page 448.

1-5 Advantages and disadvantages of a digital system in terms of security

a) Which advantages and disadvantages does a digital system offer in terms of security? Differentiate between digital transmission/storage on the one hand and computer-driven transmission, computer-driven processing on the other hand. What is changing if computers can be programmed freely (program being in the RAM instead of being in the ROM)?

b) What relationships exist between digital systems and the properties 1) distributed, 2) open and 3) service-integrating?

Solution at page 449.

1-6 Definition of requirements for four exemplary applications in the medical domain

Create protection goals that are as complete as possible for the following applications!

a) Medical records of many patients are stored in a database and are accessed by physicians from all over the world.

b) During medical surgeries external experts are consulted via videophone. Which additional requirements arise if the surgery team is dependent on advice from the site?

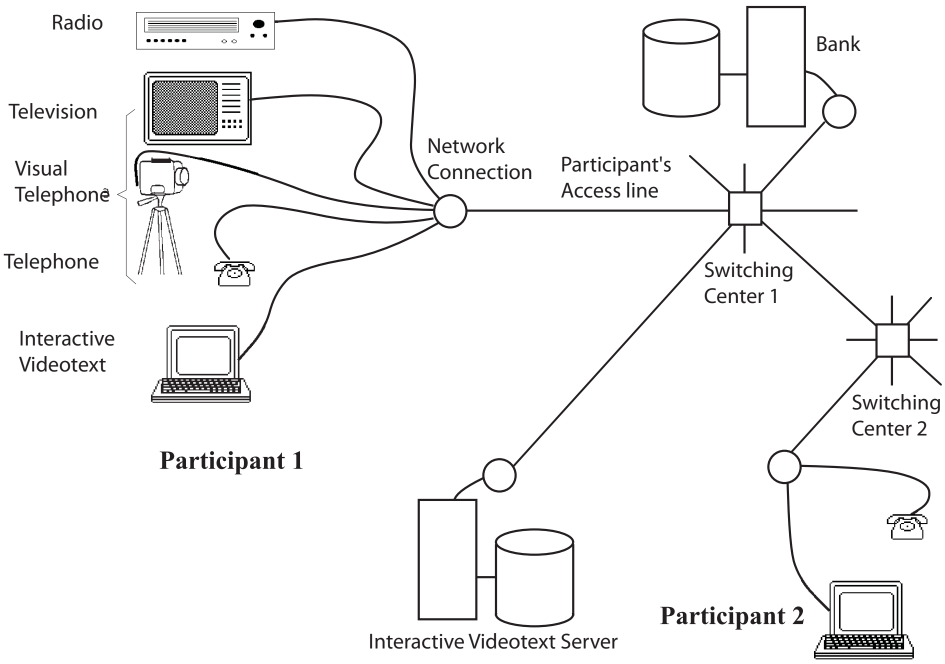
c) Patients are to be medically monitored in their private apartments. So that, on the one hand,they are able to spend less time in the hospital and on the other hand, it can be used to quickly and automatically call for help, e. g., in case of unconsciousness.

d) Medical and psychological fact-databases can be consulted by patients so as to become informed in a more comfortable, detailed, and up-to-date way than through books and to save the doctor's time (which they often do not take regardless).

Are you sure of the completeness of your protection goals?
Solution at page 450.

1-7 Where and when can computers be expected?

Mark the locations in the following figure by filling out the number n where you expect computers as of year n. Mark by adding / <role declaration >, who programmed this computer and who is able to program this computer. What does this mean for security?



Solution at page 451.

1-8 Deficits of legal provisions

Why do legal provisions not suffice for, e. g., legal certainty and data protection?
Solution at page 451.

1-9 Alternative definitions/characterizations of multilateral security

In §1.4 a definition of multilateral security is given. Since this definition is rather a colloquial characterization than a strict definition of a scientific issue: Come up with at least 3 alternative definitions/characterizations or collect them from literature. You can make a find in [Cha8_85, Chau_92, PWP_90, MüPf_97, FePf_99].
Solution at page 452.

2 Exercises for "Security in single computers and its limits"

2-1 Attackers beyond the wisdom of textbooks

Think of attackers that do not obey the currently known laws of nature, as demanded in §2.1.1, and against whom security cannot be achieved in principle.
Solution at page 454.

2-2 (In)Dependence of confidentiality, integrity, availability with identification as example

In §1.2.1 it was differentiated between the protection goals confidentiality, integrity, and availability. Discuss on the basis of the example identification (§2.2.1), if "secure" IT-systems can be expected for which no requirements in terms of at least one of the three protection goals exist.
Solution at page 454.

2-3 Send random number, expect encryption – does this work conversely, too?

In §2.2.1 a small example protocol is given for identification:

- generate a random number, send it;
- await encryption of the random number. Check it.

Is it possible to execute this protocol conversely:

- generate random number, encrypt it, send the encrypted random number;
- await decrypted random number. Check it.

What will be changed by this inversion? (Hint: Which assumptions about the generation of the random numbers must be made in each version?)
Solution at page 455.

2-4 Necessity of logging in case of access control

a) Log data (secondary data) are personal data as well as primary data. Therefore log data must be protected against unauthorized access like the primary data. It is a vicious circle – we are now again supposed to log the access to the secondary data (tertiary data) and so on. Why is this problem solved in a satisfactory way in spite of recursivity?

b) What should one aspire?

c) In terms of authorization, similar things as described in a) apply for the log data: To control who is allowed to set rights (authorization), rules themselves must be set (and enforced), who is allowed to do this, etc. It looks suspiciously like a vicious circle again: Do similar things (as in a) apply regarding the solution?

Solution at page 455.

2-5 Limiting the success of modifying attacks

In §1.2.5 it was mentioned that modifying attacks can be recognized in principle, but their success can not be prevented completely. Which measures must be taken to make modifying attacks as ineffective as possible? Consider especially how you would organize backups for data and programs.
Solution at page 456.

2-6 No hidden channel anymore?

Some top-ranking officers are extremely worried about keeping their plans for the defense of their country secret. So they decide after having read the first two chapters of this script: We will never trust that delivered devices are free of Trojan horses that want to reveal our defense secrets.Hence we will put these devices in airtight rooms and close all channels that go outside by

a) disconnecting wires with physical switches (reasons are clear!) in times of peace,

b) keeping the current power consumption constantly in the computer center (thus information cannot escape outside by modulating power consumption),

c) not giving the devices back to the manufacturer for repair (because information can be stored in them, too).

They announce to their minister of defense: Finally, no bit will escape from our computers in times of peace! Are they right?
Solution at page 457.

2-7 Change of perspective: the diabolical terminal

a) The security of each participant can not be better than the terminal, i.e., the device one is interacting with directly, is secure for him/her. Suppose you are Advocatus Diaboli: Come up with a terminal with as many properties as possible that undermine the security of its user, but which the user still wishes to use.

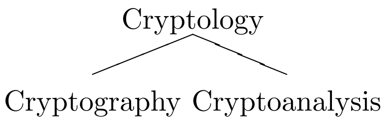
b) Compare your suggestions from a) with PCs that are offered to you today.

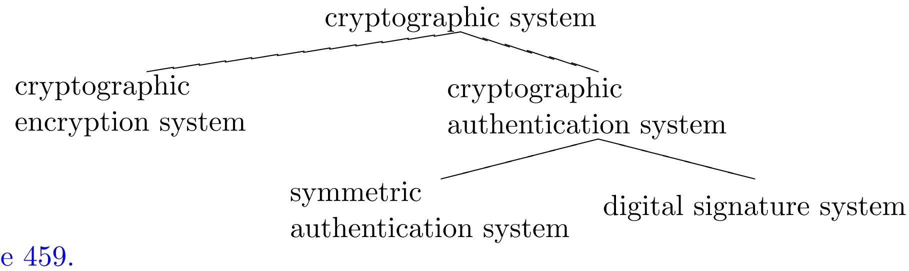
Solution at page 457.

3 Exercises for "Cryptography"

3-1 Terms

Trees of terms are shown in the following figure. Explain the textual structure of those terms and the advantages and disadvantages of using the short terms *crypto-system* resp. *encryption* and *decryption* in general for cryptographic systems or especially only for encryption systems.




cryptographic system
cryptographic encryption system
cryptographic authentication system
symmetric authentication system
digital signature system

Solution at page 459.

3-2 Domains of trust for asymmetric encryption systems and digital signatures

Domains of trust and areas of attack are marked in a way in Figure 3.1 that no statement is made about the generation of the key. It is clear that it must take place within a domain of trust. Whether in the domain of trust of the encrypter, of the decrypter or in an additional one stays open and should be different depending on the application used:

In case of encryption of a local storage medium the generation of the key, encryption, and decryption should take place in one and the same device – then the key does ever not need to leave the device. Thus the key can be optimally protected in accordance with confidentiality and integrity. (Please remember that encryption does not contribute anything to availability and that backup copies, which are stored on this medium, of the plaintext or of the ciphertext as well as of the decryption key, must be made.) In case of direct exchange of the keys between encrypter and decrypter outside of the considered communication network, e. g., via USB stick, the key should be generated within one of the domains of trust.

In case of key exchange within the communication network according to Figure 3.3, the generation of the key can take place at the encrypter, the decrypter, or at the key exchange center. It does not matter in which of the three domains of trust this takes place because the key exists unencrypted in each domain of trust anyway while the keys are distributed.

Everything that has been said up to now is of course not only valid for symmetric encryption systems, but also for symmetric authentication.

And now the question: How would you map the domains of trust in case of asymmetric encryption systems and digital signature systems? Do it – and explain your decision.
Solution at page 459.

3-3 Why keys?

It is stated in §3.1.1 that cryptographic systems use keys.

a) Would it work without keys, too? If yes, how?

b) What disadvantages would arise?

c) What are the advantages of using keys, in your opinion?

Solution at page 461.

3-4 Adjustment of keys in case of autonomous key generation by participants?

Exercise 3-2 illustrates why the key generation should take place in the device where the secret part of the key pair is used in case of asymmetric cryptographic systems.

Occasionally there are objections: Since the generated key pairs must be unique (obviously this is especially important when using digital signature systems) they can not be generated autonomously in a decentralized way by participants because doubled keys (two participants generate independent of each other the same key pair) can not be eliminated completely. In order to achieve this, so-called TrustCenters should– technically-organizational units that provide different services that are (hopefully) trustworthy and beneficial for security – also take over the function of key generation, too.

What do you think about this objection and suggestion?
Solution at page 462.

3-5 Increasing security by using multiple cryptographic systems

What can you do if you have multiple cryptographic systems, all based on different security assumptions, but you do not trust any one of them (resp. any one of the security assumptions) enough to rely on it alone? Differentiate between the protection goals secrecy (i.e., you have several encryption systems) and authentication (i.e., you have several authentication systems). Which effort for computation, storage, and transmission does this measure cause? (The last one only roughly for encryption.)
Solution at page 462.

3-6 Protocols for key exchange

a) All described protocols for key exchange assume that both participants that want to exchange a key have exchanged keys for a symmetric or asymmetric cryptosystem with a *common* key exchange center or a *common* public-key register, respectively. What must be changed if this is not the case? For instance, both participants could live in different countries and could have exchanged their keys only with key exchange centers resp. public-key registers in their own countries.

b) *Optional:* How should your solution for the exchange of symmetric keys be designed if each participant wants to use multiple key exchange centers in order to put as little trust in each individual key exchange center as possible?

c) Parallel use of key exchange centers was demonstrated and explained for symmetric cryptosystems. Is such a parallel use useful for asymmetric cryptosystems as well? How should it happen? What does it accomplish?

d) *Optional:* Does a protocol for key exchange exist that is secure despite of the following attacker model: Either the attacker receives passive assistance of all key exchange centers, i.e., they reveal all exchanged keys, or (exclusive!) the attacker is complexity-theoretically unlimited, i.e., he is able to break any asymmetric encryption system (and any normal digital signature system). It is implied additionally that the attacker wiretaps all communication in the network. How does a protocol for key distribution, if applicable, look like? Which cryptographic system should be used for encryption, which for authentication after key exchange?

e) What happens if an attacker commits a modifying attack during symmetric key exchange? This could happen by changing messages on the wire or by a key exchange center that distributes inconsistent symmetric keys. What should be done to defend participants from this? Treat the case of one key exchange center first, then the case of multiple centers in sequence (for instance in case of multiple hierarchical key distribution, cp. exercise part a)) and then the case of multiple centers in parallel (cp. §3.1.1.1).

f) Freshness of keys: How far and how can it be guaranteed that an attacker is not able to persuade participants to use old (and for instance compromised) keys that have already been replaced by new keys? (It is implied that keys of the "right" partner are involved, too.)

g) Exchange of public keys without public-key register: anarchic key exchange à la PGP
We assume that we can use no public-key registers (be it that the country or the telecommunication provider are not operating one, be it that some citizens do not want to trust central public-key registers, be it that there are public-key registers, but these are generating the key pairs themselves – Honi soit qui mal y pense). Nevertheless (or hence) public keys are supposed to be exchanged in a participant group everybody can join.
As repetition: What must be payed attention to in the case of exchanging public keys? Who could provide the function of the trustworthy public-key register? How could key exchange take place then?
What must be done if not everybody trusts everybody else in the same way?
What must be done when a participant discovers that his secret key is known by somebody else?
What should a participant do when his secret key is destroyed?
Optional: What must be done when a participant discovers that he had certified a false assignment of participant ⇔ public key?
Is it possible to combine the method created in this part of the exercise with the method created in part b) of this exercise?

Solution at page 465.

3-7 What kind of Trust is necessary for which function of a TrustCenter?

Some authors (as we will see in a moment) assign the following functions to so-called TrustCenters:

a) *key generation:* The generation of key pairs for participants

b) *key certification:* Certification of the relation between public key and participant, cp. §3.1.1.2

c) *directory service:* Keeping certified public keys ready for request from arbitrary participants.

d) *key revocation:* Blocking public keys, among others on request of the key owner, i.e., issuing a certificated statement that (and from when on etc.) the public key is blocked.

e) *time stamp service:* Digitally certification of submitted messages incl. current date and time.

Discuss which kinds of trust of the participant this requires. The distinction between **blind trust** (there is no way for the participant to check whether his/her trust is justified) and **checking trust** (participant is able to check whether his trust is justified) could be useful. How are these properties related to the attacker model described in §1.2.5, and especially to the distinction between *observing* and *modifying attacks*?
Solution at page 476.

3-8 Hybrid encryption

You want to exchange huge files and secure this exchange – for efficiency reasons – with a hybrid cryptosystem. How will you proceed if

a) only secrecy

b) as well as authentication

c) only authentication

is important for you?
Contemplate case a) first, where a sender S sends files to a recipient R, then case b), where S sends files one after another to R, and finally case c), where R answers, i.e., sends files to S.
Solution at page 477.

3-9 Situations from practice (for systematics of cryptographic systems)

Specify for the situations which cryptographic system you would use (encryption/authentication, symmetric/asymmetric) and how you would carry out key distribution. Additionally you could specify roughly how critical in terms of security and time the system is. (E. g., Vernam cipher in hardware: 10 Gbit/s, in software 100 Mbit/s; information-theoretically secure authentication codes in hardware 5 Gbit/s, in software 100 Mbit/s; DES in hardware: 200 Mbit/s, in software up to 10 Mbit/s, depending on the PC you are using; RSA in hardware: 200 kbit/s, in software 6000 bit/s (GMR, $s^2 - \text{mod} - n$-generator probably likewise).)

a) Andreas looks for flats in Hildesheim. He wants to write Birgit an email with advantages and disadvantages of the flats and wants to receive a decision if he is supposed to sign a tenancy agreement. (Both are so hoarse that they are not able to make a phone call.)

b) David again has a brilliant idea for a new cryptographic system and wants to send it to a few trustworthy cryptographers (via file transfer). Unfortunately less ingenious cryptographers are lurking in most computer centers in order to steal David's ideas.

c) A secure operating system flushes out data to a removable disk. When the data is read back, the operating system has to make sure the data has not been modified.

d) A fan of computers wants to order a new built-to-order computer on the basis of a digital catalog with a digital message.

e) A mechanical engineering company sends a fax to another company with a non-legally binding inquiry, as to whether the company was able to manufacture a certain piece and at what price.

Solution at page 478.

3-10 Vernam cipher

a) Calculate a small example, whereby we agree upon for now and for the following that processed bits are written from left to right.

b) The security of the Vernam cipher (with the addition of modulo 2) was proven in the lecture. Prove this for addition in arbitrary groups.

c) Decrypt the ciphertext 01011101, which was encrypted with the key 10011011. Is the result unique?

d) An attacker intercepts the ciphertext 010111010101110101011101 and wants to invert the third bit from the right of the corresponding plaintext. What must the attacker do if the ciphertext is computed modulo 2? What must the attacker do if the ciphertext is computed modulo 4, modulo 8, modulo 16, resp., thus 2, 3, 4 bits, resp., are added "together". In which sense is this exercise for modulo 4 and modulo 16 solvable, in which sense not?

e) Why is no adaptive active attack possible on the *key* of a Vernam cipher?

f) There are 16 functions $\{0,1\} \times \{0,1\} \to \{0,1\}$. Which are suitable as encryption functions or decryption functions, resp., for the Vernam cipher? Specify necessary and sufficient conditions for functions of arbitrary Vernam ciphers first, then apply them to the binary case.

g) A younger cryptographer has the following idea for decreasing the effort of key exchange: Instead of a random key, I do use a part of a book which my partner owns, too. Then we only have to agree on book, page, line, and row of the beginning and we can add or subtract, resp., modulo the number of characters in the alphabet of the book as long as the book is long enough.
Which advice and why will you give as an older cryptographer to your younger friend?

h) In some countries it is allowed to send encrypted messages, but one has to store the key used and has to hand it out on request to an authority of law enforcement or the like. Does this make sense for the Vernam cipher? (It is implied of course that the authority wiretaps all ciphertexts and stores them in the right order and is able to assign them to sender and recipient.)

i) How long is it possible to encrypt with the information-theoretically secure Vernam cipher

1) text communication (typing velocity: 40 bit/s),
2) voice communication (with the normal ISDN-encoding: 64 kbit/s), or
3) high-resolution full-video-communication (140 Mbit/s), resp.,

if one has exchanged

1) a 3.5" floppy disk (1.4 MByte),
2) CD-R (700 MByte),
3) DVD one sided (4.7 GByte),
4) Blueray-ROM (50 GByte)

full of random bits?
As subsumption: The first two media are standard technology at least since 1990 and widespread in large quantities. DVD is very popular today, the next generation optical media is Blueray with a capacity of 50 GB. It serves as an example for estimating what will be available for the end-user in the next years.

See p. #50
See p. #74
See p. #74
Solution at page 479.

3-11 Symmetrically encrypted Message Digest = MAC ?

a) Every now and then one hears about the following recipe for generating the Message Authentication Code (MAC):

1) Create a check digit for the message using a publicly known procedure and encrypt the check digit with a symmetric encryption system.

The recipient decrypts the check digit and checks whether it fits by calculating the appropriate check digit for the received message using a publicly known procedure. If both check digits are equal, the message is regarded as authentic.

What do you think about this recipe?
(As a hint: Take the most possible secure procedure for creation of check digits, i.e., a collision-resistant hash-function, and the most secure symmetric encryption system, i.e., the Vernam cipher, and then contemplate whether the combination is secure.)

b) What do you think about the following improvement: Instead of encrypting the hash-value with a symmetric key, the symmetric key is hashed together with the message (as for example in the Transport Layer Security (TLS) protocol). If you have the opinion that this is not a secure construction for all collision-resistant hash-functions: Which additional property must the collision-resistant hash-function have? (As a starting point for your consideration: Could the hash-value reveal something about the symmetric key?)

c) If the construction given in b) for symmetric authentication is secure for a particular hash-function: Are you then able to construct a symmetric encryption system using this hash-function? (This would be exciting, because the following opinion is widespread in the secret service: good collision-resistant hash-functions are allowed to be known by enemies, while good encryption systems must be kept secret from them. In Germany, for instance, scientific publications in the cryptographic area by the BSI exclusively deal with hash-functions, cp. e. g., [Dobb_98].)

Solution at page 485.

3-12 Authentication codes

a) Calculate a small example for an authentication code with 4 bits by using the authentication code represented in the following table. (Figure 3.15). H (Head) and T (Tail) are the plaintext characters.

Text with MAC

	H,0	H,1	T,0	T,1
00	H	-	T	-
01	H	-	-	T
10	-	H	T	-
11	-	H	-	T

b) Why is the authentication code information-theoretically secure? (You can continue or summarize the proof shown in the lecture, or represent the code easier first.)

c) If one uses this code in order to authenticate a message that consists of several bits (as in exercise part a) or example 2 from the lecture), will an attacker then be able to change the message unnoticed just by permuting the order of the bits including their MACs?

d) Calculate a small example for secrecy and authentication of 4 bits using the authentication code given in the following table. (It effects secrecy and authentication as well.)
Why is this code information-theoretically secure for arbitrary attacks on authentication and only information-theoretically secure for observing attacks on secrecy? (It may be implied that the attacker gets to know if its attack was successful or not in case of a modifying attack.)

		ciphertext			
		00	01	10	11
k	00	H	-	T	-
e	01	T	-	-	H
y	10	-	T	H	-
s	11	-	H	-	T

e) *Optional:* Modify the given secrecy and authentication code in order to make it resist modifying attacks regarding to secrecy. Is your system more efficient than just using an authentication code and the Vernam cipher?

f) The given authentication codes have two disadvantages: Both are working on single bits (H and T) so that the MAC must be attached on each bit which is not efficient for many applications. Furthermore forging attempts are only detected and denied with a probability of 0.5. A probability close to 1 would be desirable. The following authentication code by Mark N. Wegman and J. Lawrence Carter [WeCa_81] does not have this disadvantage: Messages m are elements of a huge finite field K (for instance addition and multiplication modulo a huge prime number, which is known to everybody). Keys are pairs (a, b) of elements of K (and are only used once). MACs are created as:

$$MAC := a + b * m$$

Please prove: If an attacker observes m and MAC then he is not able to create the appropriate MAC' for another message m' more purposefully than by guessing. His success probability is at most $1/|K|$.

Solution at page 486.

3-13 Mathematical secrets

a) (*Only roughly:*) Generation of prime numbers is often programmed in the following way: Only at the beginning an odd random number p is chosen. If this number is not prime, 2 is added to that number, the primality test is repeated, and so on.

Is it clear that systems that are based on the factorization assumption are still secure in this case?

(If someone wants to make this more exactly, (s)he should use an assumption of Cramer which says $\pi(x + \log^2 x) - \pi(x) > 0$. Thereby $\pi(y)$ stands for the number of prime numbers smaller than or equal to y.)

b) Someone programs – because of the procedure mentioned in a) – prime number generation accidentally in this way: after having found a prime number p, the search for the second prime number q goes on by adding 2 to the first prime number. How will you then factorize the product n?

c) Calculate using Square and Multiply: $3^{19} \bmod 101$. (Sometimes it is useful that $100 \equiv -1$.)

d) Calculate $3^{123} \bmod 77$ using Fermat's little theorem.

e) Calculate the inverse of $24 \bmod 101$ using the extended Euclidean algorithm.

f) What about the inverse of $21 \bmod 77$.

g) Calculate $y \bmod 77$ with $y \equiv 2 \bmod 7 \wedge y \equiv 3 \bmod 11$, and z with $z \equiv 6 \bmod 7 \wedge z \equiv 10 \bmod 11$.

h) Test if the following residue classes are quadratic residues (Calculating with negative representatives of residue classes simplifies your work): $13 \bmod 19, 2 \bmod 23, -1 \bmod 89$. If yes and if the simple algorithm from the script is applicable, extract the root.

i) Extract all 4 roots from $58 \bmod 77$. (You know the secret $77 = 7 * 11$.)

j) Utilize that $2035^2 \equiv 4 \bmod 10379$ to factorize 10379.
(For enthusiasts: How to generate such an exercise by hand?)

k) Let be $n = p * q$, where p and q are prime numbers, $p \not\equiv q$ and $p \equiv q \equiv 3 \bmod 4$, and let x be a quadratic residue modulo n. Show that exactly one of the 4 roots of x modulo n is again a quadratic residue.

l) Show: If the factorization assumption would be false then the quadratic residuosity assumption would be false, too.

Solution at page 488.

3-14 Which attacks on which cryptosystems are to be expected?

a) A notary attaches a timestamp to arbitrary documents that are presented to him and signs the resulting document digitally in order to enable everybody to check the notarization of each document. Which kind of cryptographic system is he supposed to use, which kinds of attacks is the cryptographic system used supposed to resist?

b) A newspaper receives articles by its correspondents confidentially. Which cryptographic system can be used, which attacks must be resisted by each system? (We do ignore that in practical life articles should be authentic, too.)

Solution at page 492.

3-15 How long must files be protected? Dimensioning of cryptographic systems; reactions to false dimensioning; Publishing public keys of a signature system or of an asymmetric encryption system?

a) Which relation between the time frame within which data is supposed to be protected by cryptographic systems and the dimensioning of those cryptographic systems does exist? What does that mean for encryption systems for personal medical data? And what does that mean for digital signature systems if signatures are supposed to stay valid for a life time?

b) What must be done in case of encryption systems, what in case of digital signature systems when it is conceivable that the system will be broken soon. (Hint: Do not forget to specify how to proceed with too weakly encrypted resp. signed data.)

c) We suppose that the authentic publishing of public keys is costly, cp. for instance exercise 3-6c), so that you can only afford to publish either a public key of a digital signature system or a public key of an asymmetric encryption system. Which one will you publish and why?

d) Is your solution for c), which you probably find under the aspect of functionality of cryptographic protocols, also right in light of b)'s solution?

Solution at page 492.

3-16 crypto-policy by choice of a convenient length of key?

Often a convenient, fixed key length is proposed to simplify fighting crime (espionage too, but there it is not talked about that publicly), but also to enable citizens and companies to achieve "sufficient" protection of confidentiality of data by encryption systems for companies and citizens: As long as curious neighbors and competitive companies cannot afford the effort for breaking. But the state that has more financial power and more computing capabilities can afford this. Do you think that this proceeding is reasonable or hopeless in principle? Please give reasons for your answer. (As background information: States such as the USA and Germany surely have no legal restrictions for production and national sales of cryptographic products, but put their export under the provision of authorization. USA used to grant export licenses only if the key length of symmetric encryption systems was 40 bits or less. Even nowadays certain high-performance computers underlie export restrictions. For this purpose nothing is officially public in Germany — but the proceeding is probably the same.)
Solution at page 496.

3-17 Authentication and encryption – in which order?

Confidentiality and integrity must be guaranteed for many applications. In which order should encryption and authentication take place? Does it make a difference if symmetric or asymmetric cryptosystems are used? (It is assumed for the whole exercise that encryption and authentication run in the same devices. Thus both are implemented equally trustworthily from the user's point of view. Additionally, it is possible to configure the user interface independently from the order, for instance plaintext is always shown in case of authentication.)
Solution at page 497.

3-18 $s^2 - \text{mod} - n - \textit{generator}$

a) Calculate a small example for the $s^2 - \text{mod} - n - \textit{generator}$ used as a symmetric encryption system. Suggestion: The key be $n = 77$ and the seed $s = 2$, the plaintext be 0101_2.

b) Decrypt the message $1001_2, s_4 = 25$, which was encrypted with the $s^2 - \text{mod} - n - \textit{generator}$ as an asymmetric encryption system. Your secret decryption key is $p = 3, q = 11$ (Only use algorithms for decryption that are still efficiently executable for $|p| = |q| \geq 300$. Simply trying out all possible seeds for instance, only shows that you understood how it is possible to *break* the $s^2 - \text{mod} - n - \textit{generator}$ if a too small key length is chosen. This is also valid for passing on to exponentiate with 2 and observing how the cycle is closing. You are supposed to show how *decryption* works!)

c) What happens when one uses $s^2 - \text{mod} - n - \textit{generator}$ as a symmetric encryption system and sends the ciphertext across a wire that is not fault-tolerant, and a bit completely switches through physical error? And what will happen if a bit gets lost?

d) Are you able to construct a symmetric authentication system using the $s^2 - \text{mod} - n - \textit{generator}$ that needs less key exchange than a real authentication code? If applicable, how does it work? How secure is your system?

e) How many key exchanges are necessary if one wants to send several messages consecutively using the $s^2 - \text{mod} - n - \textit{generator}$ as a symmetric encryption system? What must be stored by everyone?

f) *Optional:* If the $s^2 - \text{mod} - n - \textit{generator}$ reaches the inner state $s_i = 1$ sometime then all next states are $= 1$ because $1^2 = 1$. Thus from there all bits are 1. It is clear that this is allowed to happen only very rarely otherwise the $s^2 - \text{mod} - n - \textit{generator}$ would be cryptographically insecure. Calculate the probability that the $s^2 - \text{mod} - n - \textit{generator}$ reaches the inner state 1 after i steps. (Tip: There is hardly anything to calculate, you just have to find a simple argument.)

Solution at page 497.

3-19 GMR

a) Let your secret key be $p = 11, q = 7$ Sign the message $m = 01$ with the reference $R = 28$. (The length of the message be always 2 bit.) Is this possible with $R = 28$? If no, take $R = 17$ instead.

b) Assume that altogether 8 messages are to be signed. Which parts of the reference tree and appropriate signatures is the signer supposed to newly create when reaching the 4th message? And at the 5th? Which parts will be sent, what is the receiver supposed to test? (It would be best to paint a picture and mark.)

c) Sketch a proof for the collision resistance of the family of $f_{pref(m)}$ mentioned in §3.5.3. Where is the prefix-free coding needed in your proof?

Solution at page 499.

3-20 RSA

a) Calculate a small example for RSA as an encryption system.

b) Construct a small example for RSA as a signature system.

c) How will the plaintexts 0 and 1 be encrypted, and how will the ciphertexts 0 and 1 be decrypted? What do we learn?

d) How would you make an indeterministically encrypting encryption system from RSA? How many random bits do you need? Why are, for instance, 8 random bits not enough? Why is it not a good idea to use time as random bits?

e) *Optional:* Can you see if your example from d) is secure against active attacks, cp. §3.6.3? What must be done against active attacks, if applicable? What would the message format then look like?

f) *Optional:* How would you create an indeterministically signing signature system from RSA? What would that be useful for? Or could this even cause problems?

g) How does the computation effort per encrypted bit in dependence of the security parameter l grow if the algorithms shown in §3.4.1.1 are used for implementation? Distinguish as borderline cases if extremely short or extremely long messages are involved.

h) Sketch why the secret operation of RSA, thus signing resp. decrypting, is 4 times faster when one calculates modulo p and q separately, such when calculating modulo n.

i) By which factor is the public operation of RSA faster if one uses $2^{16} + 1$ as an exponent instead of a random number?

j) What do you think about the following change of key generation of RSA (taken from [MeOV_97, page 287]): instead of $\phi(n) = (p-1)(q-1)$ the least common multiple of (p-1)(q-1) , i.e., $lcm(p-1, q-1)$? Does this change anything in terms of the security of RSA? What is changing in the computation effort of RSA?

k) *Optional:* Calculate a small example for the attack of Johan Håstad.

l) You can find a proof, for instance in [MeOV_97, page 287], that breaking RSA completely, i.e., finding the secret key d, is equivalent to factorizing the modulus. Why does this say unfortunately say little about the security of RSA?

Solution at page 504.

3-21 DES

a) A simple DES-like cryptosystem is given by: block length 6; 2 iteration rounds; no initial permutation; f be described by $f(R, K_i) = R * K_i \bmod 8$; K be (K_1, K_2).

The key be 011010. Encrypt the plaintext 000101 and decrypt it again.

b) How would you implement S-Boxes in software to be able to decrypt as fast as possible?

c) *Optional:* And how would you implement the 32-bit-permutations?

d) known-plaintext attack: If you know some plaintext-ciphertext-pairs that were encrypted using DES, how and at what cost (according to number of en- and decryptions of DES) could you figure out the key?

e) chosen-plaintext attack: Does something change in comparison d) if you are allowed to choose an additional plaintext block and receive the appropriate cipher block? How high is the cost now?

f) How can you break DES if DES contains no substitutions but only permutations and additions modulo 2? Contemplate if your attack would work more generally for each linear block cipher?

Solution at page 509.

3-22 Modes

a) Confidentiality and authentication is reachable when using CBC: the first by transmitting plaintext blocks exactly, the latter by transmitting the plaintext and the last ciphertext block only. One could think both is possible at the same time by simply transmitting the ciphertext block: On the one hand secrecy would be secured, but on the other hand the receiver would be able to compute the plaintext and thus he would have all pieces of information for authentication at his disposal. Why does that not suffice in principle?

b) Show with a simple attack that the scheme is still insecure when one attaches a block full of zeros to the plaintext before it is encrypted to receive at least a minimum of redundancy.

c) Contemplate the mode PCBC in §3.8.2.5. Show that your attack will not work here (when one proceeds in the same way, thus attaching a block full of zeros, then transmitting the ciphertext and wanting secrecy as well as authentication)

d) Determine for the modes described in §3.8.2 what an attacker is able to achieve when he has possibilities for *adaptive active attacks*. Assume that he is not able to break the used block cipher. (It is assumed as usual that the attacker knows the cryptosystems used. In particular he knows the length of the used cryptosystems and the mode used.)

e) You want to work with ECB, CBC (may it be for secrecy, may it be for authentication) or PCBC, but you do not know if you will receive plaintexts of appropriate length (i.e., length of plaintext is divisible by the block length without a remainder). The advice "Then we fill with zeros up to the block length" given often is not satisfying for you, because it is supposed to make a difference if one receives "I owe you € 10" or "I owe you € 100000". Design a coding

   1) that maps plaintexts of arbitrary length to such an appropriate length,
   2) that is uniquely invertible,
   3) whose expansion of length is minimal.

f) OFB allows neither random access nor ability of parallelizing because the state of the shift register must be computed from scratch each time. Design an obvious modification for OFB that allows random access and parallelizing.

Solution at page 511.

3-23 Mirror attack

We assume that we want to substitute passwords for LOGIN with something better, but we have stupid terminals and an insecure network between terminals and mainframe. Therefore each participant has a "pocket calculator" which implements a secure symmetric block cipher (cp. footnote at §2.2.1) though it does not have an electrical access to the terminal. Therefore all information transfer via Display and keyboard has to run over the participant. What will you do if you assume that the mainframe itself is secure? Which attacks will not be tolerated by your designed system?
Solution at page 514.

3-24 End-to-end encryption

Which problems do arise if one wants to carry out an end-to-end encryption using an office computer, thus a computer to which others also have access? Which approaches do you see for a solution?
Solution at page 516.

3-25 End-to-end encryption without exchanging keys before

Bob and Alice, our couple of cryptographers, are – as always – separated and this time Alice is traveling by air.

a) And once again the luggage got lost, thus key and crypto devices are not available. How can the two communicate with some measure of confidentiality and authenticity? (A tip: You have to assume that the attacker can not be everywhere at all times.)

b) The baggage was recovered and returned. But someone wanted to find out the keys in the crypto devices, thus they were erased automatically. How could Alice and Bob improve their communication now?

Solution at page 516.

3-26 Identification and authentication by an asymmetric encryption system

How can a participant P, whose public key c_P of an asymmetric encryption system is known by a participant U, be identified by U, cp. §2.2.1?

a) Design a simple cryptographic protocol. (The existence of such a protocol is amazing since asymmetric encryption systems were only defined with the aim *confidentiality*. But actually the protocol that is searched for exists.)

b) Against which attacks is the encryption system in your small cryptographic protocol supposed to be secure?

c) Are you able to improve your small cryptographic protocol in a way that it identifies not only T but also authenticates messages N_i? (As incentive: Such a protocol exists.) Is your authentication "system" asymmetric or symmetric, i.e., does it comply with digital signatures? Where is the disadvantage to "normal" authentication systems?

d) Is the asymmetric encryption system using your improved protocol supposed to resist such potent active attacks as in b)?

Solution at page 517.

3-27 Secrecy by symmetric authentication systems

How can a participant A who has only a symmetric authentication system in common with a participant B communicate with him secretly? Please do not just take the confidentially exchanged key k_{AB} for a symmetric encryption system, it is actually just a matter of an authentication system. (The existence of such a protocol is amazing since symmetric authentication systems were only defined with the aim *integrity*. But actually the protocol that is searched for exists.)
See p. #369
Solution at page 519.

3-28 Diffie–Hellman key exchange

a) *Main idea:* What is the main idea of the Diffie–Hellman key exchange?

b) *Anonymity:* Diffie–Hellman key exchange, as it was described in §3.1.1.1, distinguishes itself, according to §3.9.1, from asymmetric encryption systems in the ability of the sender to stay anonymously (or not) towards the receiver of the encrypted message.

In case of asymmetric encryption systems this is the trivial case: The sender obtains the public key for itself, encrypts the message using this key, sends it to the receiver. The receiver decrypts the message independently from the one who had encrypted it.

This is different in case of Diffie–Hellman key exchange. The secret key which is assigned to the relationship of sender and receiver is used for symmetric en- and decryption. Thus it seems that no anonymity is possible for sender to receiver.

Consider a modification of the Diffie–Hellman key exchange where anonymity for the sender to the receiver is possible. And what must be done if the sender wants to receive an answer anonymously?

c) *Individual choice of parameters:* Diffie–Hellman key exchange, as described in §3.9.1, distinguishes from asymmetric encryption systems according to §3.1.1.1 in the property that parameters that are critical for security (concretely p and g) are known in general and are therefore equal for all participants. The question of who chooses them and if that person is able to get special power arises instantly. Because it is not manageable if one chooses p and g randomly or somehow specially that extracting the discrete logarithm is particularly easy for one. If one lets a external instance choose p and g one needs a stronger security assumption than the discrete logarithm assumption. Alternatively many instances can choose p and g together, but this requires a cryptographic protocol again.

Consider in the scope of this exercise if you are able to modify the Diffie–Hellman key exchange in a way that each participant is able to choose all parameters that are critical for security.

d) DSA (DSS) as a basis: On May 19th, 1994 a standard for digital signatures within all and with all public places in the USA was set once and for all by the National Institute of Standards and Technology (NIST), which is the successor of the National Bureau of Standards that standardized DES in 1977: the **Digital Signature Standard (DSS)**, given by the **Digital Signature Algorithm (DSA)** [Schn_96, page 483-495]. The steps in the case of key generation, - certification and -publishing that are interesting for us are:

1) a huge prime number p and a randomly chosen number $g \in \mathbb{Z}_{|}^*$ are public (and possibly equal for all participants)

2) Each participant chooses a number x with a length of app. 159 bit and keeps it secretly as a part of his signature key.

3) Each participant computes $g^x \bmod p$ which will be certified and published as a part of his test key.

Is this enough to perform Diffie–Hellman key exchange?

Solution at page 520.

3-29 Blind Signatures with RSA

Calculate a small example for blindly achieved signatures with RSA. You can save calculating effort by using results of exercise 3-20b)).
Solution at page 521.

3-30 Threshold scheme

Analyze the secret $G = 13$ in 5 parts that $k = 3$ are necessary to reconstruct it. Use the smallest prime number p that is possible and as random coefficients $a_1 = 10, \text{and} a_2 = 2$. (In practice it is not allowed to choose p in such strong dependence to G. This choice of G is supposed to ease your calculating and to show that you understood the conditions for p in relation to G.)
Reconstruct the secret by use of the parts $(1, q(1)), (3, q(3))\text{and}(5, q(5))$.
Solution at page 522.

3-31 Provably secure usage of several encryption systems?

Now that you know that there are informational secure and efficient threshold schemes, it is worth paying attention to exercise 3-5 again. How do you combine multiple encryption systems that the resulting total system is at least provable as secure as the most secure used encryption system? (Thereby you are allowed to suppose that the used encryption systems are designed for uniform distributed plaintexts – as they arise in case of minimal length coding, and that “Security of encryption system” means security in case of uniform distributed random plaintexts. Likewise you are allowed to suppose that the parts that are generated by the information-theoretically secure and efficient threshold scheme are uniformly distributed random plaintexts.)

Compare the effort of the provable secure combination with the one described in exercise 3-5.

Compare each security achieved by the combination.

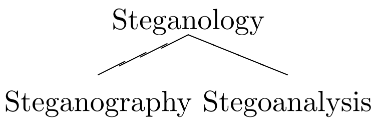
Does an especially simple implementation of a threshold scheme come into your mind for exactly this application?
Solution at page 523.

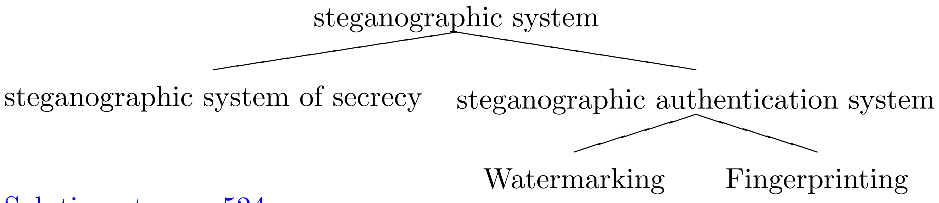
4 Exercises for “Basics of Steganography”

4-1 Terms

In the following picture, trees of terms are shown that were similarly created to the trees shown in the section about cryptography, cp. exercise 3-1. The term *Steganology* is newly created and in need of further exposure. I thought about taking the shorter term *Stegology* but decided to use the longer but grammatical correct term. Some authors use the term *information hiding* for the area of Steganology and therefore especially for the area of Steganography. They divide information hiding into two parts: *steganography* (that which I call steganographic secrecy) and *copyright marking* (that which I call steganographic authentication). Similar to refering to *cryptoanalysis* as *cryptanalysis* in German, the shorter term *steganalysis* is used for *stegoanalysis*.

Describe the textual structure of the trees of terms and the advantages and disadvantages of using the shorter term *stegosystem* in general for steganographic systems or especially for steganographic systems of secrecy.




steganographic system
steganographic system of secrecy
steganographic authentication system
Watermarking
Fingerprinting

Solution at page 524.

4-2 Properties of in- and output of cryptographic and steganographic systems

a) Draw a scheme (block diagram) which contains general parameters (all input and output values) of an encryption system. Where are the differences between symmetric and asymmetric encryption systems? Which assumptions about the inputs is a good cryptosystem allowed to make, which guidelines is it supposed to fulfill regarding to its outputs?

b) Draw a scheme (block diagram) which contains general parameters (all input and output values) of a steganographic system of secrecy. Where are the differences between symmetric and asymmetric encryption systems? Which assumptions about the inputs is a good stegosystem allowed to make, which guidelines is it supposed to fulfill in regards to its outputs?

c) What are the main differences between cryptographic and steganographic systems of secrecy?

Solution at page 524.

4-3 Is encryption superfluous when good steganography is used?

Why is it actually unnecessary to encrypt the secret message before it is embedded when a secure steganographic system is used? And why would you do it in practice despite this?
Solution at page 527.

4-4 Examples for steganographic systems

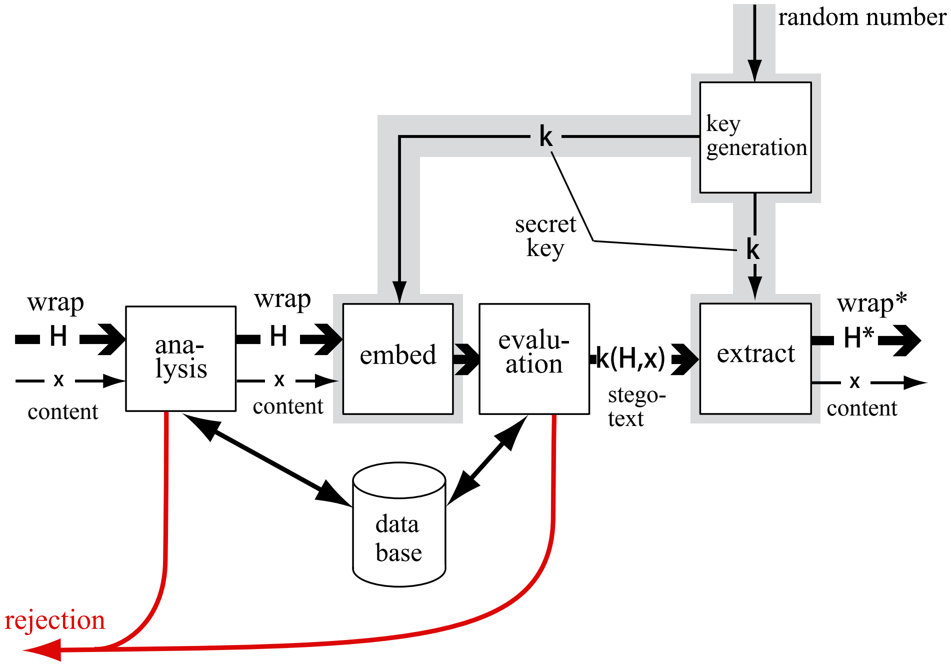
Name some examples for steganographic systems. How do you evaluate the security of your enumerated systems?
Solution at page 527.

4-5 Modeling steganographic systems

A group of students discusses the modeling of steganographic systems shown in the Figures 4.2, 4.3, 4.4, and 4.5: Substantial things were forgotten, says *Anja Foresight*. We need an analysis before embedding which is supposed to cooperate with a database of all so far used wrappers. This is the only way to avoid that improper wraps are used for embedding.

Right, agrees *Berta Viewback*. But it would be much better to do the analysis after embedding – let us call this algorithm **Evaluation**. The task of evaluation is to check if the stegotext is inconspicuous. If yes, the stegotext is outputted. If no, it is rejected.

Do not argue, notices *Caecilie Baseman*. We just do the analysis before as well as the evaluation after embedding. This is the general case and we do agree, right? Secure is secure! Let me plot it for you:



Doerthe Reinhardt raises her eyebrows and contradicts: No, I do not like any of your improvement suggestions. The original model is sufficient.

Who do you agree with – and above all: How do you justify your choice?
Solution at page 527.

4-6 Steganographic secrecy with public keys?

Ross Anderson describes in [Ande4_96, AnPe_98, page 480] the following procedure for steganographic secrecy:

> Given is a wrap in which absolutely any key can be embedded. Then usually a rate exists at which bits can be embedded without the stegoanalyzer noticing it. ⋯ Alice knows Bob’s public cipher key. She can take her message which is to be kept in secret, encrypt it with Bob’s public key, and then embed the resulting ciphertext. Each possible receiver will then simply try to decrypt the extracted message, but only Bob will succeed. In practice the value encrypted with the public key could be a control block consisting of a session key and some filling material. The session key would be used for operating a conventional steganographic system.

Ross Anderson calls this public key steganography. Do you agree with this term? Please give reasons for your decision.
Solution at page 528.

4-7 Enhancement of security by using several steganographic systems

What can you do if you have several steganographic systems, each underlying a different security assumption, and you do not want to trust one of these steganographic systems solely (resp. one of the security assumptions)? Differentiate between the protection goals secrecy (i.e., you have several encryption systems) and authentication (i.e., you have several authentication systems).
Solution at page 528.

4-8 Crypto- and steganographic secrecy combined?

Steganographic secrecy overlaps the functionality of cryptographic secrecy, so that, if steganographic secrecy can be assumed as secure, no combination with cryptographic secrecy is useful. Is a combination useful and how should it happen if applicable when doubts exist concerning the security of the steganographic secrecy?
Solution at page 529.

4-9 Crypto- and steganographic authentication in which order?

Steganographic authentication does not protect the plaintext from unrecognized small altering, a digital signature is doing this but can be easily removed by anyone. Can you combine both? If applicable, in which way?
Solution at page 529.

4-10 Stego-capacity of compressed and non-compressed signals

Is it possible to embed in terms of percent more undetected in compressed or in non-compressed signals? Please differentiate between lossy and lossless compression and give reasons for your answer.
Solution at page 529.

5 Exercises for chapter 5 “Security in communication networks”

5-1 Quantification of unobservability, anonymity, and unlinkability

In §5.1.2 definitions for unobservability, anonymity, and unlinkability are given and it is emphasized that their reliability depends on the underlying attacker model and where appropriate, on a classification. It is defined that unobservability, anonymity, as well as unlinkability are perfect if the relevant probability is the same for the attacker before and after his observations. Try to quantify unobservability, anonymity, and unlinkability, i.e., find useful a) parameters resp. b) gradations between each extreme:

- the attacker finds out nothing – the attacker finds out everything, and
- the attacker knows nothing – the attacker knows everything

Solution at page 529.

5-2 End-to-End or link encryption

Due to disclosures about plentiful wiretapping by secret services (e. g., former GDR secret service Stasi) the management of a company decides to encrypt all future telecommunication between the offices. How can and should this happen? Are your measures dealing rather with link encryption than with end-to-end encryption? Or do you suggest to even use both?
Solution at page 529.

5-3 First encrypting and then encoding fault-tolerance or the other way round?

In many communication networks error-detecting and even error-correcting codes are used that are adapted to the dumb error behavior of the transmission channels. According to the channel one has to reckon that zeros become ones or ones become zeros, that errors arise separately or in bursts, etc. Should one encrypt first and then encode fault-tolerant in case of networks with adapted fault-tolerance encoding, or should one prefer the reverse order? Explain your answer.
Solution at page 530.

5-4 Treatment of address- and error-recognizing fields in case of encryption

Given a plaintext message 00110110, which is supposed to be sent to the address 0111 (in this exercise each coding given is binary). The usual message format consists of a 4-bit-address, an 8-bit-message and a 2-bit-check-character for detecting transmission errors in the address or the message.

<table>
<tr><th colspan="4">Address</th><th colspan="8">Message</th><th colspan="2">Check-char.</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

The 2-bit-check-character is created by two-bit-wise addition modulo 4 of address and message. For the above address and plaintext-message the check character would result in 10, thus altogether the message 01110011011010.

a) Let the message above be encrypted end-to-end with a one-time pad. Let the one-time pad be 0110011111110001101010001101···. What does the end-to-end encrypted message look like if it is encrypted using one-time pad modulo 2?

b) Let the message above be connection encrypted using one-time pad. What does the encrypted message look like if it is encrypted using one-time pad modulo 2?

c) Let the message above be encrypted end-to-end and connection encrypted with the above one-time pad. What does the encrypted message look like if it is encrypted using one-time pad modulo 2?

Solution at page 530.

5-5 Encryption in case of connection-oriented and connection-less communication services

In the case of communication services there is a distinction made between *connection-oriented* and *connection-less* communication services. The first guarantee that messages are received in the same order as they were sent, the latter do not guarantee this. Do you have to consider this when choosing an encryption algorithm and its mode? If yes, what do you have to pay attention to?
Solution at page 530.

5-6 Requesting and Overlaying

a) Compute a small example for the method “Requesting and Overlaying”.

b) Why must there be encryption between participants and servers? Must the answer of the server to the participant also be encrypted?

c) How do you implement “Requesting and Overlaying” by use of usual cryptographic means if the transmission bandwidth from participant to server is very narrow? And what if the transmission bandwidth from server to participant, but not among the servers, is very narrow? Can you combine both solutions skillfully?

d) What do you think about the following suggestion: To confuse the servers the participant generates randomly an additional request-vector, sends it to an additional server, receives its answer and ignores this. None of the used $s + 1$ servers know if it is used as a normal or as an additional, ignored server.

Solution at page 530.

5-7 Comparison of “Distribution” and “Requesting and Overlaying”

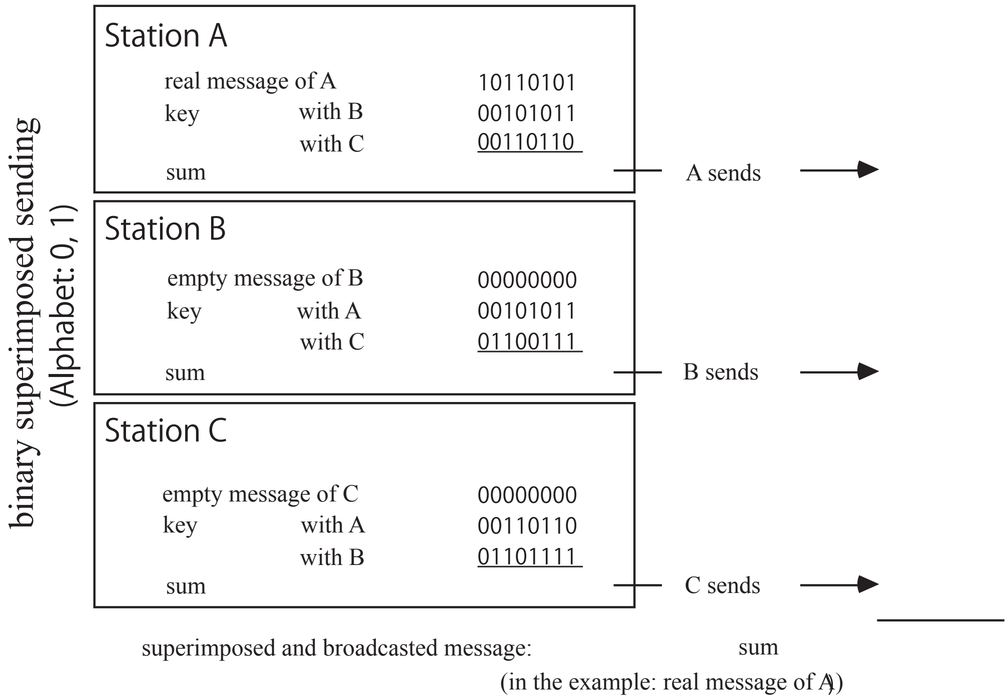
a) Compare “Distribution” and “Requesting and Overlaying”. Consider especially how opened implicit addresses can be implemented as efficiently as possible in case of “Distribution” and “Requesting and Overlaying”.

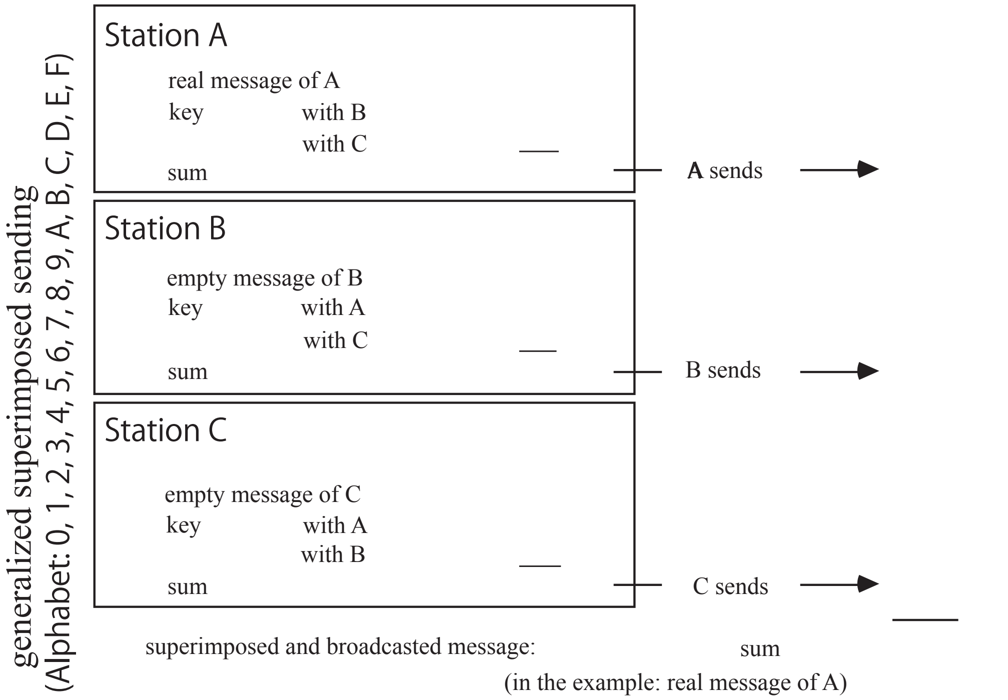
b) Can you imagine communication services where “Requesting and Overlaying” are necessary because “Distribution” is not applicable?

Solution at page 530.

5-8 Overlayed Sending: an example

Given the example for overlayed sending shown in the following figure. Inscribe the missing values in the upper part of the figure in which it is calculated modulo 2. After that, calculate the same example for modulo 16 in the bottom part of the figure. Which values would you be allowed/supposed to copy from the upper part of the figure? Why?




generalized superimposed sending
(Alphabet: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)
Station A
real message of A
key
with B
with C
sum
A sends
Station B
empty message of B
key
with A
with C
sum
B sends
Station C
empty message of C
key
with A
with B
sum
C sends
superimposed and broadcasted message:
sum
(in the example: real message of A)

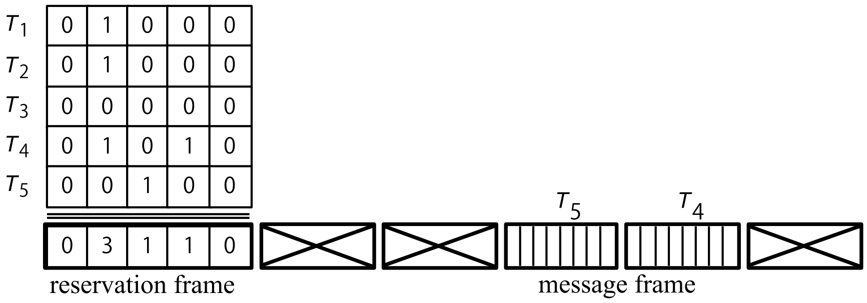
Solution at page 530.

5-9 Overlayed sending: Determining a fitting key combination for a alternative message combination

How must the keys look like in the example of the previous exercise to have the station A have sent the empty message 00000000 and station B have sent the real message 01000101 in the case of same local sum and output (and of course therewith with same global sum)? Which real message must station C have sent then? Is the key combination unique? Will it be if it is given additionally that the key between A and B stays the same because the attacker knows it? (A has given the key to B on a floppy, which B, after having copied its content to a disk, has not physically destroyed or at least not written on multiple times but he has deleted it, i.e., the operating system has signed the key as deleted. After that B has recorded the almost new floppy with very few data and gave it to a friendly man, which acted unexpectedly as an attacker.) What are the other keys in this case?
Hint: If you calculate this exercise only in regards to addition modulo 2 you should differentiate between + and - too. This of course only plays an important role with bigger modulus, thus in the exercise only with modulus 16.
Solution at page 530.

5-10 Several-access-methods for additive channels, e. g., overlayed sending

a) Reservation method
It is shown in the following figure how reservation in a message frame avoids collisions of message frames if after reservation message frames are only used by such stations that reservation resulted in 1 as sum. What would happen in this example if the reservation frames did not overlay modulo a greater number but modulo 2?



T_1	0	1	0	0	0
T_2	0	1	0	0	0
T_3	0	0	0	0	0
T_4	0	1	0	1	0
T_5	0	0	1	0	0
	0	3	1	1	0

b) pairwise overlayed receiving
In a DC-net in which pairwise overlayed receiving is not supported, two friends, Scrimpy and Scrapy, are agreeing to literally double the bandwidth of their communication by pairwise overlayed receiving. What do they have to find out first? Perhaps the following example will help you: Scrimpy has sent 01101100 and receives 10100001 as sum. He asks himself: What has Scrapy sent to me? What is

the answer if the overlayed sending takes place modulo 2, as with modulo 256? (We assume that the several-access-method does not make difficulties for Scrimpy and Scrapy. For instance one may allow that one of both reserves a (simplex-)channel which both are using for pairwise overlayed sending.)

c) Global overlayed receiving: collision detection algorithm with creating the average value
The participant stations 1 to 5 send messages $5, 13, 9, 11, 3$. How does the collision detection with creating the average value and overlayed receiving take place? Let the local decision criterion be $informationunit \leq \lfloor \oslash \rfloor$. Use the notation given in Figure 5.14 of the script.
How does it happen if the participant stations send $5, 13, 11, 11, 3$?

Solution at page 530.

5-11 Key-, overlaying-, and transmission topology in the case of overlayed sending

a) Why is it possible to specify the key topology independently from overlaying- and transmission topology?

b) Why should overlaying- and transmission topology be adjusted to each other?

Solution at page 530.

5-12 Coordination of overlaying- and transmissionalphabets in case of overlayed sending

With which "trick" was it possible to achieve a huge overlaying alphabet (useful, for example, for collision detection algorithm with average value creation) and a minimal delay of time, i.e., a minimal transmission alphabet, at the same time?
Solution at page 530.

5-13 Comparison of "Unobservability of adjacent wires and stations as well as digital signal regeneration" and "overlayed sending"

a) Compare the two concepts for sender anonymity "Unobservability of adjacent wires and stations as well as digital signal regeneration" and "overlayed sending' in regards to the strength of considered attackers, effort, flexibility. Which concept is the most elegant one? Why?

b) Do these concepts cover receiver's anonymity? If applicable, what does the cooperation of sender and receiver anonymity look like?

Solution at page 530.

5-14 Comparison of “overlayed sending” and “MIXes that change the encoding”

Compare the concepts “overlayed sending” and “MIXes that change the encoding” in regards to a security aim, strength of considered attackers, effort, flexibility. Which of these is the more practicable concept in your opinion? In which respect and why?
Solution at page 531.

5-15 Order of basic function of MIXes

To what extent is the order of basic functions of a MIX in Figure 5.22 inevitably?
Solution at page 531.

5-16 Does the condition suffice that output messages of MIXes must be the same lengths?

What do you think about the following suggestion (according to the maxims of the retired chancellor Kohl: The deciding factor is the result): MIXes can also mix input messages of different length together. The main point, is that all appropriate output messages have the same length.
Solution at page 531.

5-17 No random numbers → MIXes are bridgeable

Explain how an attacker could bridge a MIX if the direct scheme of changing the encoding to guarantee sender’s anonymity (cp. §5.4.6.2) is used without random numbers.
Solution at page 531.

5-18 Individual choice of a symmetric encryption system at MIXes?

MIXes publish their public cipher keys and therefore define the appropriate asymmetric encryption system. In §5.4.6.1 (symmetric secrecy at the first and/or at the last MIX) and in §5.4.6.3 (indirect scheme of changing the encoding) two applications of symmetric secrecy in the case of MIXes are introduced. Should each user of the MIX-network be able to choose the used symmetric encryption system or should it be determined by the MIX?
Solution at page 531.

5-19 Less memory effort for the generator of untraceable return addresses

It is described in §5.4.6.3 how untraceable return addresses can be created. Here it is costly for the generator that it has to memorize all symmetric keys under the unique name e of the return address until the return address is used. This can last a while and is therefore circumstantial. Consider how the generator of return addresses could avoid memorizing symmetric keys and even unique names.
Solution at page 531.

5-20 Why changing the encoding without changing the length of a message

Explain how an attacker could bridge the MIXes shown in the upper part of Figure5.26. (It is trivial, but just try to explain it precisely).
Solution at page 531.

5-21 Minimal expanding of length in a scheme for changing the encoding without changing length of a message for MIXes

a) Describe shortly the effort for communication and encryption of MIX nets.

b) You have an asymmetric encryption system, which is secure against adaptive active attacks, and an efficient symmetric encryption system at your disposal. Both are working (after having chosen keys) on blocks of the same length. Design a scheme for changing the encoding for MIXes which expands the message minimally at the sender and which does not change the length of the message during the steps of changing the encoding.

   Derive for simplicity this minimal expanding scheme for changing the encoding from symmetric encryption system with the properties $k^{-1}(k(x)) = x$ and $k(k^{-1}(x)) = x$, i.e., not only en- and decryption but also de- and encryption are inverse towards each other, given in §5.4.6.5, cp. Figure 5.31.

Solution at page 531.

5-22 Breaking the direct RSA-implementation of MIXes

What is the main idea of the described attack on the direct RSA-implementation of MIXes in §5.4.6.8?
Solution at page 531.

5-23 Use of MIX-channels?

a) Why and for what are MIX-channels necessary?

b) What is limiting their use resp. is leading to the "development" of time-slice-channels?

Solution at page 531.

5-24 Free order of MIXes in case of fixed number of used MIXes?

Ronny Freeroute and Andi Cascadius are arguing again if free choice of MIXes and their order should be preferred- or so-called MIX-cascades should be used instead, i.e., MIXes and their order are firmly stated. Against the usual scheme of discussion

> *Ronny Freeroute*: Free choice of the MIX-order would be better because everyone is free in his choice that way. (Confidence, usage of MIXes on the way) and, if only a few MIXes attack, greater groups of anonymity would come up.

> *Andi Cascadius*: MIX cascades must be preferred, because the largest anonymity set possible would come up in case of maximal number of attacking MIXes because the intersection of anonymity sets would be empty or the intersection would contain the whole anonymity set.

a pale Ronny says someday: "What if the attacker controls a maximal number of MIXes and knows that everyone uses a fixed number of MIXes in case of free choice of the order of MIXes. Then the attacker has good a chance to bridge the MIX that is not attacking, too." What does Ronny mean, i.e., what could a simple attack on a MIX-net with free choice of route and participants with a fixed chosen route-length look like? And what would you do against it?
Solution at page 531.

5-25 How to ensure that MIXes receive messages from enough different senders?

As it was explained at the beginning of §5.4.6.1 MIXes have to work on information units from sufficient senders in one batch – are they from too few the danger is high that one of them can be observed from the few others over the MIXes too.

How can the compliance of the claim "information units from sufficient senders" be ensured

a) at the first MIX of a MIX cascade,

b) at a later MIX of a cascade, i.e., not the first one?

Otherwise a MIX could substitute, for instance, in its output batch all information units except for one information unit produced by itself and trace the non-substituted information unit despite all following MIXes this way.
Solution at page 531.

5-26 Coordination protocol for MIXes: Wherefore and where?

For which schemes of encryption is a coordination protocol between MIXes necessary? Give reasons for your answer.
Solution at page 532.

5-27 Limitation of responsibility areas – Functionality of network terminals

Why should the necessary areas of confidence between participant and network provider overlay as little as possible? Explain this for all introduced security measures in communication networks. (Figure 5.48 might be a help for you).
Solution at page 532.

5-28 Multilaterally secure Web access

Consider multilaterally secure web access: Which security problems exist? Which participant has which protection interests? Then apply all procured concepts to this new problem. (Your curiosity is supposed to be created and the procured is supposed to be practiced with this larger scaled exercise. Therefore it is okay to have a look in the web for existing concepts after having analyzed the problem. And even I will allow it myself to link to literature when giving the solution.)
Solution at page 532.

5-29 Firewalls

Many companies and authorities control the traffic between public networks, e. g., the *Internet*, and their in-house network, e. g., LAN or *Intranet*, by so-called Firewalls (the name comes from the fire protection in buildings and means there is a wall which prevents the spreading of fire or at least delays the spreading). Do you think this is useful? What does it exactly bring? Where are the limits? Is this able to substitute admission and access controls for a workplace computer?
Solution at page 532.

6 Exercises for “value exchange and payment systems”

6-1 Electronic Banking – comfort version

a) What do you think about the following paper-banking comfort version: The customer receives pre-filled forms from his bank in which only the amount must be filled in. It especially saves time for the customer because he does not have to sign the forms anymore.

b) Does this paper-banking comfort version differ from the electronic banking version which is usual today?

c) Will this be changed if customers receive a chip card from their bank with preloaded keys for MACs or digital signatures?

Solution at page 532.

6-2 Personal relation and linkability of pseudonyms

Why do role-pseudonyms offer more anonymity than personal-pseudonyms and transaction-pseudonyms more anonymity than business-relation-pseudonyms?
Solution at page 532.

6-3 Self-authentication vs. foreign-authentication

Explain self-authentication and foreign-authentication and the difference between both. Give typical examples. Why is the distinction between self-authentication and foreign-authentication important for configuration of several-sided IT-Systems?
Solution at page 533.

6-4 Discussion about security properties of digital payment systems

The security properties described in §6.4 do only consider availability and integrity of payment procedures, thus only the most essential ones. Create a list of necessary protection goals that you consider as necessary. Order them by confidentiality, integrity, and availability.
Solution at page 533.

6-5 How far do concrete digital payment systems fulfill your protection goals?

Prove known digital payment systems against your protection goals that you elaborated in exercise 6-4. Examples could be

- tele-banking
- homebanking computer interface (HBCI); http://www.bdb.de/, search for HBCI

Solution at page 534.

9 Exercises for "Regulation of security technologies"

9-1 Digital signatures at "Symmetric authentication allows concealment"

Is it possible to use digital signatures instead of symmetric authentication codes in the procedure of "Symmetric authentication allows concealment" (§9.4)? If no, why not?
Solution at page 534.

9-2 "Symmetric authentication allows concealment" vs. Steganography

What do "Symmetric authentication allows concealment" (§9.4) and Steganography (§4) have in common? What is the difference?
Solution at page 535.

9-3 Crypto-regulation by prohibition of freely programmable computers?

Due to the procedures in §9.1 - §9.6 it is clear that crypto-regulation is not applicable as long as freely programmable computers are available. We assume that *all* states agree to prohibit the production of freely programmable computers and they could eventually succeed in enforcing this prohibition. We assume the decades in which freely programmable computers already sold still worked have elapsed – thus there are no freely

programmable computers anymore. Is it then possible to enforce a crypto-regulation? Asked more exactly: Is it then possible to enforce a crypto-regulation which allows to decrypt email traffic of a suspect after a permission for observation has been given – but not before?
Solution at page 535.

B Answers

B.1 Answers for "Introduction"

1-1 Link between spatial distribution and distribution in terms of control- and implementation structure.

Spatially distributed IT-systems usually allow inputs at more than one place. Those inputs (if not ignored) have effect on the system state. If you want to implement a central control structure through an instance with a global view, every incoming input needs to be communicated to it. Only after this may the state transition take place. Since information can only be transported at the speed of light and since today's computers work at more than 10^7 instructions per second you have:

$$\frac{300000\ km/s}{10^7\ \text{instruction}/s} = 30\ m/\text{instruction}$$

You would use more time for transmission than for processing[1]. While the computation performance is steadily increasing (and the structural size is decreasing) the speed of light remains constant, so this argument becomes an issue more and more. Physical maintenance in spatial distributed IT-Systems is quite costly especially right after a failure. But IT-Systems that have a distributed control- and implementation structure allow the postponement of certain maintenance tasks (even after a serious failure) since a failure will not bring down the whole system. Further arguments can be found in the next answer.

Exercise at page 409.

1-2 Advantages and disadvantages of a distributed system in terms of security

[1] You could argue that you have to consider the instructions of the users but not the computer which would result in a completely different picture (since a human being acts much slower than any computer). But from my point of view this is not right: Since it is not about whether it is a distributed system from the user's point of view but the implementation structure (and therefore at least one level deeper) machine operations must be regarded as units.

Another objection could be, that not machine instructions but hard disc accesses must be regarded as operations since they make the state transition persistent and they can be set back for fault tolerance reasons anyway. This is surely the right view for system shaping - by implicitly introducing a distributed control- and implementation structure for computers and systems.

a) Advantages:
Confidentiality can be supported by everybody can have their data under his physical control → End of external determination on information: It is unnecessary to give away your own data and hope for the best.
Integrity can be supported by everybody being able to see his own data before anyone else accesses them and confirming if they are valid or not. Of course this process can be automated to make a PC validate the data. Furthermore the logging can be distributed to complicate faking or suppressing of logging.
Availability can be supported by not having a single point of failure for explosive attacks or logical bombs.
Disadvantages:
Confidentiality can be endangered by the fact that data can be moved freely which makes controls difficult. It is doubtful that there is always a legally responsible entity, named “speichernde Stelle” (storing place) in privacy laws. *Integrity* and *availability* can be endangered by the mere existence of more possible points of attack. Just think of the many physical connections (especially wireless ones) that can hardly be secured or the many computers staying around unsupervised.

b) Spatial distribution supports availability against physical threats. Distribution as well as control and implementation structures support confidentiality and integrity.

Exercise at page 409.

1-3 Advantages and disadvantages of an open system in terms of security

a) Advantages:
Open systems can be connected more easily to cooperating distributed systems. Therefore the advantages of distributed systems mentioned above can be achieved more easily. The compatible interfaces of open (sub)systems allow an easier creation of over-all systems using different designers, producers, and maintainers. As seen in §1.2.2 diversity can heavily increase security:
Disadvantages:
Open systems can be easily connected to cooperating and distributed systems. So the disadvantages of distributed systems mentioned above can be achieved more easily. In particular a user of an open system usually does not know who has which rights and under which role name.

b) Therefore they amplify each other so anything said about distributed or open systems holds true for distributed open systems.

Exercise at page 409.

1-4 Advantages and disadvantages of a service-integrating system in terms of security

a) Advantages:
Service integrating systems can be more expensive since all efforts are shared by several services. Relatively seen this allows more effort regarding security. On the one hand this allows a more careful design and its validation (like testing for Trojan Horses). On the other hand you can employ more redundancy for increasing the availability, like additional alternative physical connections. Then maybe even the more expensive measures for protecting the anonymity of sender and recipient (cp. §5) are more likely to be achieved.
Disadvantages:
If a security property is violated this would surely have an impact on several concurrent services. This turns out to be a disaster if no spare systems are at hand (this would be a totally service integrated system [Pfit_89, pg. 213ff][RWHP_89]). Access must be granted even to those users that only make use of parts of the services offered. For the unused services those participants are an unnecessary security risk.

b) Service integration amplifies anything told here (independent from the type of the system) but this applies especially to distributed open systems.

Exercise at page 409.

1-5 Advantages and disadvantages of a digital system in terms of security

a) Digitally transmitted or stored information can be altered or copied perfectly (=without traces) if the attacker is willing to take a certain minimal amount of effort. This amplifies the problems regarding confidentiality and integrity as long as the encryption techniques from §3 are not used (which are available by actually using digital transmission). Availability is supported by digital transmission and storage in common.

Computer aided relaying and processing are vulnerable to problems of confidentiality, integrity, and availability. On the other hand its enormous capabilities allow protection measures that would not be feasible elsewhere. With an appropriate system configuration this disadvantage can be more than balanced out. If you are confident about the correctness and completeness of the protection measures they should rather be implemented in hardware than in software. Mark that: Protections implemented in software often turn out to be *soft*!

b) The arguments from 1-1 show that the property *digital* supports the construction of distributed systems.

As long as an open system is not complete (especially being incompletely specified) computer aided relaying and processing (both freely programmable) are necessary to make the systems adaptable for future developments.

Today it is inevitable that service integrated systems are digital.

Exercise at page 410.

1-6 Definition of requirements for four exemplary applications in the medical domain

a) *Availability*: Medium critical, as long as there are local originals printed on paper. In case of failure the situation remains the same.

   Integrity: Critical since nobody wants to make additional checks with the paper documents. Nor does anybody want to update them by hand or print them out regularly.

   Confidentiality against external doctors: critical; against third persons: highly critical. The patient may request admission or deletion rights that are to be granted. International harmonization would be difficult.

b) *Availability*: Uncritical, as long as the surgery team on site does not rely on the help. Otherwise: Highly critical.

   Integrity: Small integrity violations (i.e., some broken pixels) are uncritical. Larger violations (bad images) are critical. To make the patient (or his relatives!) able to prove professional blunder requires the reproduction of who saw what and who gave the advice (even after many years). In particular, a clear identification (cp. §2.2.1) of the external advisor is needed for court.

   Confidentiality: Confidentiality of the images: less critical; if transmitted at the beginning of the case history the content of course needs to be protected. The protection of the sender and recipient identity is superfluous. ([PfPf_92, §4.2]).

c) *Availability*: Medium critical, since failure "only" leads to the current situation. As long as patients remain at home who would be in a hospital otherwise: critical.

   Integrity: False alarm is uncritical but should not be triggered too often. The opposite: see availability.

   Confidentiality of an alarm is less critical, all other message contents as well as the patients as sender and recipient need to be protected ([PfPf_92, §4.3]).

d) *Availability*: Uncritical.

   Integrity: Not very critical since for critical cases a doctor is compulsory. Anyway, while accessing a medical database it must be clear who is responsible for its content. If doctors become dependent on such a database, requirements are heavily increased towards availability and integrity.

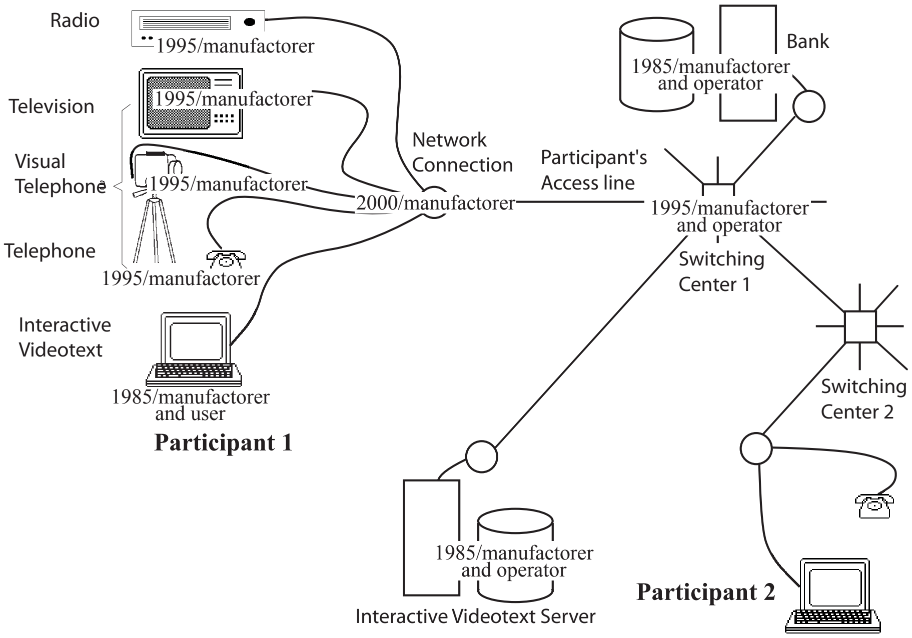
   Confidentiality of the message content and of the request itself must be secured by anonymity ([PfPf_92, §4.1]).

Of course I am not sure about the completeness of my protection goals.

Exercise at page 410.

1-7 Where and when can computers be expected?

The numbers show the year where a noteworthy spread of computer aided components (within it is component class) took place or is predicted. In the near future you must consider switching stations, “harmless” end user devices or even network connection units as potential attackers.



Exercise at page 410.

1-8 Deficits of legal provisions

The answer consists of a combination from the arguments of §1.2.3 and §1.2.5.

A prohibition is only useful if its adherence can be *verified* (in principle impossible with observing attacks) with a reasonable effort, if it is secured by criminal prosecution and if it is possible to *restore* the original state. All that is not true for IT-Systems.

“Data theft” in general, specifically the direct wiretapping of connections or the copying of data from computers, is hardly detectable since the original data will not change. Nor is the installing of Trojan Horses or the unallowed processing of data you legally (or illegally) obtained.

The reconstruction of the original state would be to delete all data created. But you can never be sure that there are not any further copies. Besides data can remain inside the human’s brain where deleting is not a worthwhile option.

Parallel to the problem of “data theft” is the problem of data loss: Maybe lost data can not be restored or reproduced so that the original state can not be brought back.

So more effective measures are needed.

Annotation For IT-Systems crossing the national borders an international agreement (expressed by laws) and its enforcement (by international courts) are needed. While there are many such IT-Systems, they are not covered by many international agreements.

Exercise at page 411.

1-9 Alternative definitions/characterizations of multilateral security

“The large-scale automated transaction systems of the near future can be designed to protect the privacy and maintain the security of both individuals and organizations. ... Individuals stand to gain in increased convenience and reliability; improved protection against abuses by other individuals and by organizations, a kind of equal parity with organizations, and, of course, monitorability and control over how information about themselves is used. ... the advantages to individuals considered above apply in part to organizations as well. ... Since mutually trusted and tamper-resistant equipment is not required with the systems described here, any entry point to a system can be freely used; users can even supply their own terminal equipment and take advantage of the latest technology.” from [Cha8_85]

“Meanwhile, in a system based on representatives and observers, organizations stand to gain competitive and political advantages from increased public confidence (in addition to the lower costs of pseudonymous record-keeping). And individuals, by maintaining their own cryptographically guaranteed records and making only necessary disclosures, will be able to protect their privacy without infringing on the legitimate needs of those with whom they do business.” from [Chau_92]

“Considering the growing importance of legal affairs using open digital systems, the need arises for making usage of those systems unobservable for non participants and anonymous for participants by keeping up the necessary legal certainty” from [PWP_90]

“The term multilateral security is therefore used here to describe an approach aiming at a balance between the different security requirements of different parties. In particular, respecting the different security requirements of the parties involved implies renouncing the commonly used precondition, that the parties have to trust each other, and, especially, renouncing the precondition that the subscribers have to place complete trust into the service providers. Consequently, each party must be viewed as a potential attacker towards the other and the safeguards have to be designed accordingly.” from [RDFR_97].

“Multilateral security means to respect all security needs of all participating parties.” from [Rann_97].

“Multilateral security means to include the security interests of *all* participants as well as to deal with the resulting conflicts during the creation of a communication connection.” from [FePf_97]

“Multilateral security means to include the security interests of *all* participants as well as to deal with the resulting conflicts during the creation of a communication connection.

The creation of multilateral security does not inevitably lead to the fulfilling of all the participant's interests. It possibly reveals incompatible interests the participant were not aware of since protection goals are formulated explicitly. But it inevitably leads to the interacting of the participants using multilateral communication systems to achieve a certain balance of powers." from [Fede_98]

Multilateral security is *security with minimal assumptions about others*:

- Every participant has security interests.
- Every participant may express his interests.
- Conflicts are to be recognized and solutions to them are to be negotiated.
- Every participant may enforce his security interests in negotiated solutions.

from [FePf_99]

"... new developments, e. g., the rising need and demand for multilateral security, which cover the requirements of users and uses as well as those of IT system owners and operators. ... The TCSEC had a strong bias on the protection of system owners and operators only. This bias is slowly losing strength, but can still be seen in all criteria published afterwards. Security of users and usees, especially of users of telecommunication systems is not considered. Therefore techniques, providing bi- or multilateral security, e. g., those protecting users in a way privacy regulations demand it, cannot be described properly using the current criteria." from [Rann_94]

"Multilateral security emphasizes the equal protection of all participants. The protection goals of privacy and liability are considered as important priorities with regard to personal freedom of man and obliging social behavior" from [PoSc_97]

"While for classic security models the operator's view stays in front, multilateral security respects the participating user's view. Besides the protection the networks offer to their users, the user needs mechanisms for self-protection for reducing his dependency on others. Not only are they now protected from attacks from the network but they also keep up their mutual interests like buyer and seller exchange money and receipts sometimes even with notary supervision." from [GGPS_97]

"Multilateral security is a user-centric feature. The protection demanded is finally for the people and organizations that do employ telecommunication systems to do their tasks. However security serves not only the communication partners but also all others that are somehow connected to the partners or the content transmitted..." from [RoSc_96]

"Multilateral security: Whereas the introduction of new communication media and services primarily targeted the *security of the provider* (like against unpermitted usage) now the *security of the participant* is to be emphasized." from [BDFK_95]

"It seems quite legitimate to demand that users be treated equally in terms of their security rights. If you promise every user that their security does not depend on the goodwill of others, but only on their own behavior, such a system would be truly multilateral." from [PSWW_96]

"Multilateral security means the upholding of morale in an information technical sense." a student in winter term 99/00.
Exercise at page 411.

B.2 Answers for "Security of single computers and its limits"

2-1 Attackers beyond the wisdom of textbooks

To begin with, an *attacker with unlimited capacity* in terms of computation power, time, and memory comes into mind. Fortunately we will see that there are corresponding information-theoretical techniques to achieve security (confidentiality and integrity) against such a complexly, theoretically unlimited attacker.
However it is challenging to fight against attackers that are owners of unknown technical, physical, or chemical innovations (like new measurement tools, explosives, . . .).
And it becomes highly critical if they posses the ability to control new physical phenomenons: new kinds of radiation, fields, etc.; Or applied to humans: Supernatural senses (X-Ray-viewing capability, soothsaying) or abilities (passing-through-the-wall, invulnerability, telekinesis). This is not meant to start a parapsychological discussion, but gives warning that perfect security can only be achieved in a *closed world* with no new perceptions. Luckily, as we are all curious students, we live in an open world. Here are some examples of closed world attacks:

- *Unpermitted information gaining*: Against an attacker with soothsaying abilities neither shielding (cp. §2.1) nor cryptographics (cp. §3) help to protect confidentiality.
- *Unpermitted modification of information*: Against an attacker with telekinetic abilities shielding (cp. §2.1) does not protect confidentiality. Against an attacker with telekinetic and soothsaying abilities cryptographics (cp. §3) will not help either.
- *Unpermitted affecting of functionality*: Against an attacker with telekinetic abilities neither shielding nor (finite) fault tolerance will help to maintain availability.

Exercise at page 411.

2-2 (In)Dependence of confidentiality, integrity, availability with identification as example

You can not expect any "secure" IT systems if you have at least one of the three protection goals without any requirements at all:

- If there are no requirements regarding *availability* then no IT systems at all needs to be realized. With that, all its (existing) pieces of information are good in terms of integrity and confidentiality.
- If there are no requirements regarding *integrity* an IT system can not distinguish between different persons since the information needed for that could have been modified without permission. The same applies for distinguishing different IT systems.
- If there are no requirements regarding *confidentiality* an IT systems can not identify itself towards others because an attacker could gather all necessary information from the IT system itself. The other way round is not that simple: “Since an IT-Systems can not keep any information confidential it can not recognize other instances since an attacker could obtain all data required to do so from the IT system to distinguish other instances.” This argument seems consistent but the IT system could apply a collision resistant hash function (cp. §3.6.4 §3.8.3) to the information expected from the other instance and save only the result of it. Later on the result is compared with the input applied on that hash function.

Exercise at page 412.

2-3 Send random number, expect encryption – does this work conversely, too?

The difference is subtle:

With the normal version you only have to assume that random numbers will not repeat (with a relevant probability) during random number generation.

With the opposite version you *additionally* have to assume that the random number generation is unpredictable for the attacker. Actually this should be true for any random number generator (just like what the name says) but it is always safety first.

The difference becomes clear with an extreme example: The normal version is secure if the random number generator is a counter that is incremented after every “random number”. With this implementation of a random number generator the opposite version would be completely insecure.

For further interest: In [NeKe_99] you will find, next to a detailed explanation of this example, a worthwhile introduction to a formalism (BAN-Logic) with which you can examine such protocols.

Exercise at page 412.

2-4 Necessity of logging in case of access control

a) The confidentiality problem is reduced with every level of the “recursion” since the amount of personal related data is (hopefully) getting fewer and less sensitive as well. Or the other way round: The problem is not really recursive since the access to the secondary data can be more restricted than for the primary data.

For example you could print out the secondary data and prevent the manipulation afterwards by spatially remote people. Furthermore it is not useful to let the secondary data be created too thoroughly; it should really get fewer. Finally you can reduce the circle of people that *may* access the secondary data to a few certain trustworthy ones. Hopefully this reduces the circle that *may* access. If you introduce tertiary data you can have them evaluated by other privacy commissioners as the secondary data. Then the controllers control each other. Assuming the use of nonalterable protocols you could realize a time staggered $2n$ eye principle (corresponding to the known - not time staggered - 4 eye principle) for n levels.

b) You should strive for a strict and detailed regulation concerning who may do what under which circumstances. Then the amount of logging can be reduced: As much logging as needed but no more.

c) Here the "sensitivity" does not decrease with every level but it increases: The regulation for who may grant rights is at least as security critical as the enforcing of the rights and their limits. The same as for a) is true regarding that the granting of rights may be subject to stronger security regulations and other security regulations than for the usage of rights, the granting of rights for rights granting to other and stronger security regulations and so and so on.

Exercise at page 412.

2-5 Limiting the success of modifying attacks

Modifying attacks should be a) *instantly detected* and all unpermitted modifications should be b) *localized as precisely as possible*. After that c) *the state* should be recovered as fully as possible compared to the original state *without the modifying attack*.

a) Besides the commonly applicable information technical techniques such as

- **Error detecting encoding** for transmission and storage
- **Validation of the authenticity of all messages** before their processing (authentication codes or digital signatures, cp. §3) so that faked commands in the messages will not be executed at all. The same applies for storage content that can be read from unprotected areas (like floppy discs, CDs, USB media devices).

special tests of plausibility exist for every single application. For example, a person can travel a maximum of 1000 km/h. This can be used to discover if an attacker pretends to be somebody else who reported earlier from somewhere else.

b) Here it helps to *log* who had accessed what and when (cp. §2.2.3).

c) Here you must differentiate between hardware and data (incl. programs). **Hardware** has to be repaired or replaced very quickly. Or you have alternatives like backup computer centers, alternative communication networks. The usage of hardware is mostly a financial problem.

For **data** (incl. programs) you will need *backups*. Here the following dilemma exists: On the one hand the last current state possible should be recovered so that a minimum of work is lost. On the other hand if unpermitted modifications took place the current state should be rolled back to the latest version that was unmodified. What now? The backup system must allow to set back the distance of recovery dynamically. This can be done by conducting a daily **full backup**, and weekly, monthly, and per year. The daily backup can be overwritten weekly, the weekly every month, and the year backup never, please - your company should be able to afford that many streamer tapes! An alternative would be to to make a backup from time to time but also to create a **logfile** where all changes (incl. the exact values) are recorded. Using the current state and the logfile you can rollback to any previous state desired - as well go forward with an old backup and the logfile. Please remember to conduct a backup of the logfile as well. And of course you can combine that too.

Exercise at page 412.

2-6 No hidden channel anymore?

Unfortunately not! If our military really needs its computers there is the following hidden channel: Every time a *computer fails* the military will order a new one. If they do so within the next 24 hours and if you assume that computers fail every 10 years (=3652.5 days with 2.5 intercalary days) so there is a hidden channel with almost (assuming a perfect computer that will not fail except for the usage of the hidden channel: exactly)

$$ld\ 3652,5 \text{ bit}$$

and therefore more than 11 bit in 10 years.

Of course you can object that *illoyal staff* represents a far more bigger hidden channel. But I am only considering the information technical point of view as well as the usually overlooked aspects.

By the way, the *loyal staff* may be a hidden channel of a bigger bandwidth too: Consider a Trojan horse that is placed in a computer where it can influence the video output of the terminal by simply reducing the image quality. Then the attacker only has to observe if the staff leaves the hermetically shielded rooms with reddish eyes.

Exercise at page 413.

2-7 Change of perspective: the diabolical terminal

The properties for a) are first followed by the motives of the participants to use such a device. The consequences are stated behind the implication arrow. Keywords found in the PC handbook are marked in *cursive*.

Using the device simultaneously for as many purposes as possible to make its inner workings as inscrutable as possible. The motivation for the user to tolerate that, is that he can use his data for many purposes which makes the device look like it has a good price performance relation → service integrated terminal device.

The device should count as useful, nifty, and advanced or even as inevitable → service integrated device, useful, nifty, advanced, small, colorful, communicative ... as requested by basic agreements for big companies or public institutions.

The device should be an alterable → programmable device; dynamical loading of software: *PC, PDA, ActiveX, BHO, Plugins*

The responsibility for changes should belong to someone who does not completely understand what he is doing. But he does not realize that and he has good reasons for doing the things. The user itself would be a perfect match. Then others could say: Blame yourself ... (A subtle variation would be a preinstallation by the seller according to the buyers' wishes)

As much computation power and storage as possible that are used for inscrutable purposes so that the resource usage for (universal) Trojan horses will not attract any attention.

Since Trojan horses will not work perfectly at first, system crashes should be seen as normal.

Inscrutable and complex closed source programs.

Proprietary, publicly undocumented data formats instead of openly documented or standardized ones, e. g., *Word* instead of *RTF*.

No access control (so that every program can influence every other). Not to mention access control employing the Least Privilege principle. → *Windows 98* and *Mac OS 9* do not have any access control. *Windows 2000* and *Linux* do not have access control following the Least Privilege principle.

The device should have universal, bidirectional interfaces so the user can connect to everything fairly easily. Especially good are public open networks → *Internet* On the one hand for dynamically reloading of software, on the other to enable attackers to access their Trojan horses as often as possible without being noticed.

Giving away user data (content, user ID, or the ID of the device) of which the user is not aware of → Transferring plain data over open networks like the Internet. Adding the creator device of every object into the digital representation of the object (DCE UUID of OLE or the Microsoft *GUID*=Global Unique Identifier). Device IDs like Ethernet addresses *mac address* or processor IDs (*Pentium III*) undermine the anonymity of the user since they can be transferred without being noticed.
Exercise at page 413.

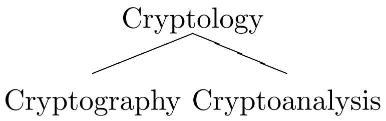
B.3 Answers for “Cryptography”

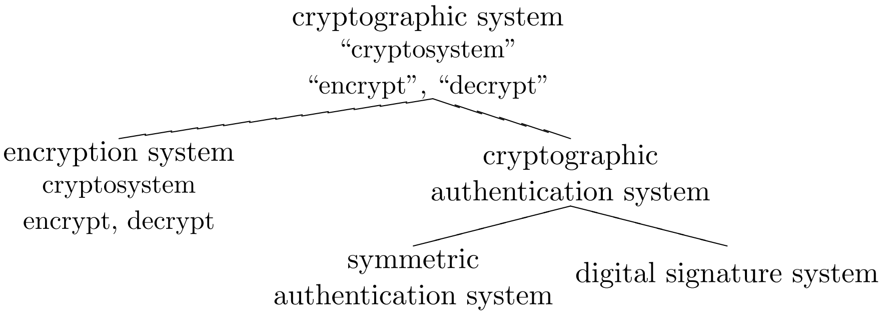
3-1 Terms

Cryptology encompasses cryptography (the science of algorithms of the “good” users) as well as cryptoanalysis (the science of algorithms used by attackers on cryptographic systems). Knowledge in the area of cryptoanalysis is necessary to evaluate the security of cryptographic systems and its applications.

Cryptographic systems are either encryption systems (protection goal: confidentiality) or authentication systems (protection goal: integrity). Authentication systems are either symmetric (and therefore not suitable as legal evidence) or asymmetric and therefore suitable as legal evidence towards a third person. Because of this important property asymmetric authentication systems are also called digital signature systems.

The advantage to place terms like *cryptosystem*, *encrypting*, and *decrypting* at the top of the tree of terms is to use short common terms almost everywhere. The disadvantage is that the distinction between encryption and authentication is pushed into the background so that I will use these terms only for cryptographic encryption systems – if used in a more common meaning I will use quotation marks. My tree of terms looks like this:

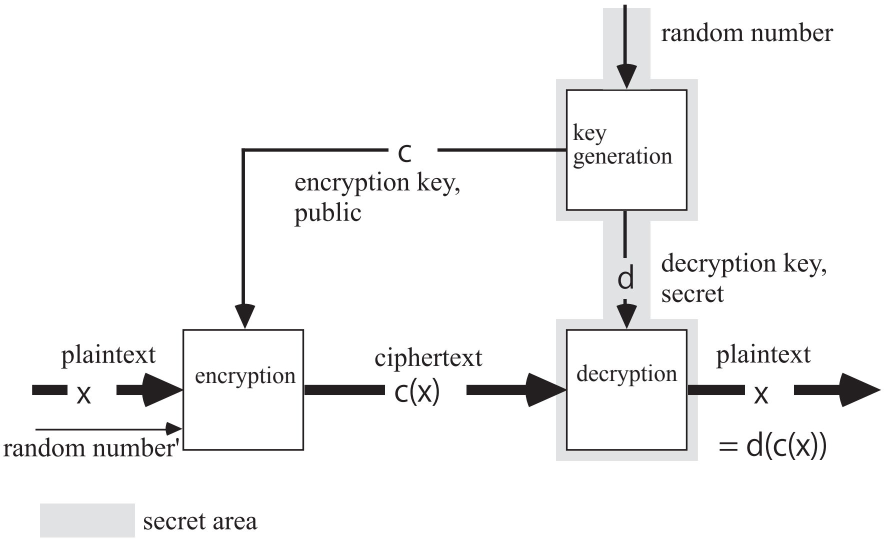


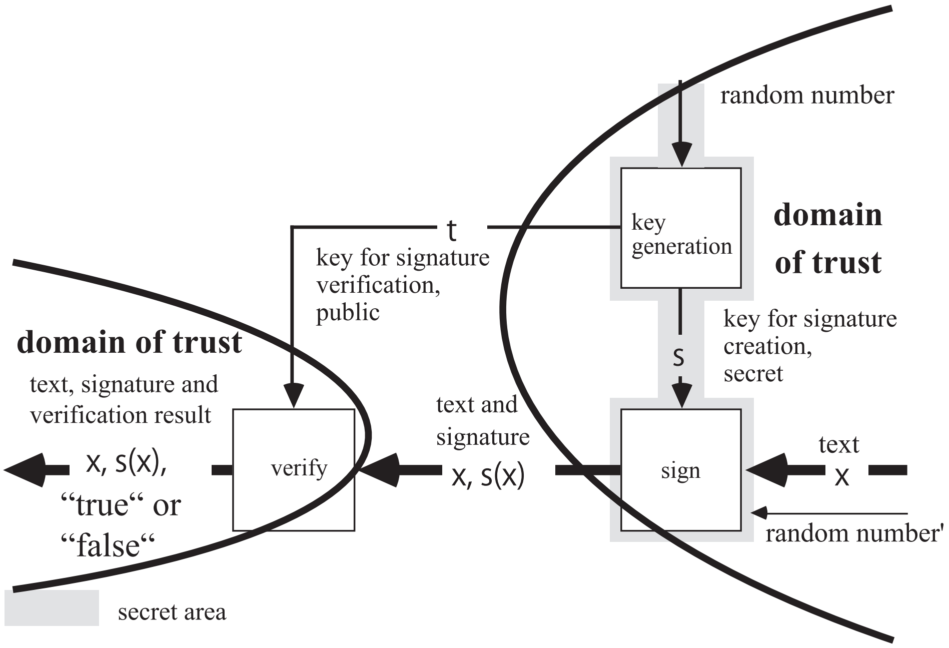


Exercise at page 413.

3-2 Domains of trust for asymmetric encryption systems and digital signatures

The secret key can be protected (in terms of confidentiality and integrity) by generating it inside the device (so the region of trust) in which the key is used. The following pictures arise:





A backup copy of the signing key is not necessary since signatures remain testable and therefore valid even if the key is destroyed. The signing key should never leave the signing device. If the key is not available anymore you will just create another key pair, get the public part certified - and there you go. The same applies to communication with asymmetric concealment: After generating and exchanging the keys the message is just encrypted once again. If using asymmetric concealment for storing data, you may either need plaintext backups of the data or the encrypted data and the decryption key. Exercise at page 414.

3-3 Why keys?

a) You can do without keys too: For symmetric cryptographics two users need to exchange a secret and exclusive cryptographic algorithm. For asymmetric cryptographics you will have to publish a public algorithm from which it is impossible to derive the secret one.

b) The disadvantages are numerous and fatal:

- Where do users get good cryptographic algorithms from? Eventually non-cryptographers want to communicate confidentially with integrity without anybody else (this includes the cryptographer as well) breaking its security. Finally, the cryptographer that created the algorithm would needed to be "removed" since he knows the secret. While this was done in the ancient times, today all people (even cryptographers) are subject to the Human Rights.
- How to analyze the security of cryptographic algorithms? Anybody analyzing a symmetric algorithm is just as much a security risk as the inventor. For asymmetric algorithms anybody who knows the secret algorithm for validating purposes is a security risk.
- Implementing the cryptographic systems in software, hardware, or firmware is possible only if the user is able to do this on his own. Otherwise the same applies for the implementor as for the algorithm inventor.
- In public systems where potentially everybody wants to communicate securely with everybody else, only software implementations with algorithm exchange instead of key exchange would be possible - firmware depends on the machine, and hardware exchange would require physical transportation. But even software implementations have the disadvantage of requiring (for symmetric systems with confidentiality and integrity and asymmetric systems full of integrity) transmissions of huge amounts of bits: Algorithms are usually larger than programs.
- A standardization of cryptographic algorithms would be impossible due to technical reasons.

c) The main advantage of keys is: The cryptographic algorithms can be published. This allows:

+ a broad and public validation of its security
+ its standardization
+ any implementation form desired
+ mass production
+ a validation of the implementation's security as well

Exercise at page 414.

3-4 Adjustment of keys in case of autonomous key generation by participants?

The objections against autonomous decentral key generation by participants are wrong (cp. below). Thus this proposal is unnecessary and in terms of the confidentiality of the secret keys - as you know - very dangerous.

Of course, you can not totally rule out that the random number used for initializing the key generation occurs more than once with the consequence that the secret part of the key can be generated multiple times. But the conclusion that therefore the key generation needs to be coordinated is pretty obscure. If the same random number occurs more than once either the random number generator is bad or the number is too short. In both cases it will not help to use the random number once and discard it if it occurs again. Because nothing would prevent an attacker from creating equally long random numbers with the same generator. Since this can not be prevented, only one thing helps: Carefully picking and validating the random number generator as well as making random numbers long enough. Anything else is decoration, which distracts from the real problem.

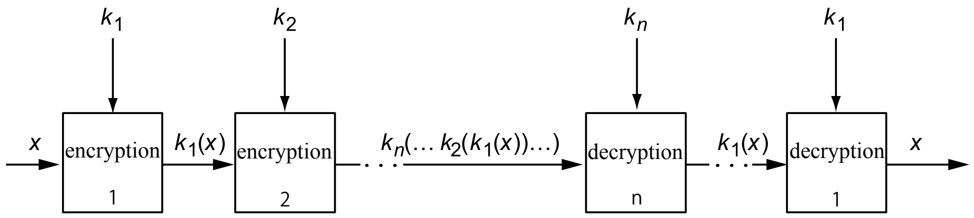
Besides: If keys are compared it is sufficient to do so with the *public keys*. So the key generation can stay autonomously - the key comparison becomes part of the key certification.

Exercise at page 415.

3-5 Increasing security by using multiple cryptographic systems

For the protection goal **confidentiality** you may use cryptographic systems sequentially. Then the plaintext is encrypted with encryption system 1, the result is encrypted with encryption system 2 and so on (cp. Figure 3.2 where k_i is the key of the $i-th$ encryption system: $S := k_n(\ldots k_2(k_1(x))\ldots)$. Of course every system needs its own unique key and the applying order must be the same.

Decrypting is the other way round; therefore as last with encryption systems 1 which reveals the plaintext: $x = k_1^{-1}(k_2^{-1}(\ldots k_n^{-1}(S)\ldots))$.



Of course asymmetric systems can be used that way too. Even a mixed usage of symmetric and asymmetric encryption systems is possible. In the example you could replace every i, $1 \le i \le n$ the k_i on the left with c_i and on the right with d_i.

This construction of course only works if the key text space of encryption systems i is a sub space of the plaintext of encryption system $i+1$. In practice this can be easily achieved (cp. §3.8.1,§3.8.2). If plaintext and key text do have the same length on all n systems this measure will not affect memory and transmission complexity (but at the most time it will increase due to the arrangement into blocks). The *computation complexity* is the *sum of all computation complexities* of every single encryption system.

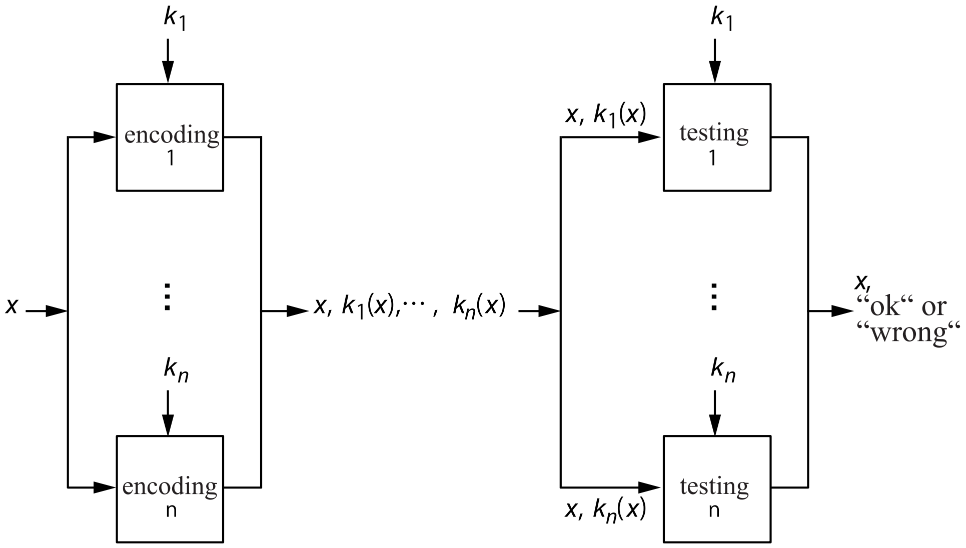
Besides: The encryption system created by using several sequential encryption systems is called **product cipher** of the encryption systems used.

For the protection goal authentication you can use the cryptographic systems in **parallel**: For every system the authenticator of the original message x is created and all of them are consecutively appended on x (with separators; cp. Figure 3.6: $x, k_1(x), \ldots, k_n(x)$). The recipient checks every authenticator separately: They all have to be valid.

Of course you may use digital signature systems that way too, even a mixed usage is possible. In the example you could replace every i, $1 \leq i \leq n$ the k_i on the left with s_i and on the right with t_i.

Here the keys used must be picked independently, too.

Computation complexity and *memory and transmission complexity* for the authenticator *are exactly the sum* of each single system. The computation can be done in parallel so that the whole system (properly implemented) is as fast as the slowest authentication system used.



For the protection goal concealment as well as for the protection goal authentication, you will need additional memory to house the software that implements the cryptographic systems.

About the *provability* of both constructions:

Systems parallel to the authentication can be trivially proved since nothing of the usage of the single systems would change, and especially not the distribution of messages and key texts/MACs.

Sequential Systems for concealment can not be proved at all. The distribution of the inputs of “encryption 2” up to “encryption n” is changed by the construction - instead of the plaintext the key text is encrypted. This may happen in such an unfortunate way that the encryptions 2 up to n will not support security at all so that Sequential Systems would be only as secure as “encryption 1”. In practice this would surely only occur if “encryption 2” up to “encryption n” are explicitly designed to be “unfortunate” (in Exercise 3-31 you will find a provably secure (in a certain sense), but more costly construction for using several encryption systems).

Reversely, the proof for *systems parallel to the authentication* also shows that the construction is *only* as secure as it is hard to break all the systems used. Also reversely, the proof for *sequential systems for concealment* fails, mainly because the attacker does not have the interim results (the key texts between the encryption systems used) - this is what makes his task unequally harder than to *merely* break all encryption systems used.

Discussion of *wrong* solutions:

To encrypt the plaintext with several encryption systems, where some of them are unsecure, the message is split into as many parts as you have encryption systems. Then every part is encrypted with another system. *Disadvantage*: The probability that an attacker can decrypt at least some parts of the message is usually increased and not decreased.

To authenticate the message with several authentication systems you create the 1. validation record with the first authentication system. The second system is used to authenticate the 1. validation record; so the 2nd validation record is created, ... *Disadvantage*: If an attacker can break the first by finding another message for the 1. validation record all other authentication systems become ineffective (because the 1. validation record for the other matching part is not altered, the remainder can stay untouched). Besides, the computation of the validation record would not be that easy to parallelize.

To save key exchange efforts the same key is used for several cryptographic systems (if possible at all). *Disadvantage*: In many cases the resulting system is not more secure, sometimes even weaker than the weakest of the systems used. This can be easily shown for authentication: Consider, all systems require the same key. Then it is clear that breaking one system results in the breaking of all systems. If some systems allow effective gaining of information about the key, the sum of all these information may be sufficient for a successful attack.

Discussion of *clumsy* solutions:

You use authentication system i not only to create a validation record for message x but also for the validation records 1 to $i - 1$. Of course this solution is secure but unnecessarily expensive:

a) The calculation effort of the validation records is increasing proportionally with the length of the string to be authenticated.

b) The calculation of the validation records is no longer parallelizable since validation record i depends on part $i-1$.

Exercise at page 415.

3-6 Protocols for key exchange

a) The functionality of *one* key exchange center has to be provided by *several* others. For this either:

- all key exchange centers know each other's keys directly or,
- the key exchange centers require a meta key exchange center.

In the latter case two key exchange centers need to exchange two keys like two normal participants do.

From now on we will assume that both the key exchange centers Z_A and Z_B, with which the participants A and B have exchanged keys, now have a common key.

Symmetric system: Now one of the two key exchange centers may generate a key and transmit it confidentially to the other (i.e., Z_A generates key k and sends it to Z_B encrypted with $k_{Z_A Z_B}$ (their common key)). Now every key distribution center securely tells “their” participants the key (i.e., Z_A sends $k_{AZ_A}(k)$ to A, and Z_B sends $k_{BZ_B}(k)$ to B).

$$\begin{tikzcd}
Z_A \arrow[r, "k_{Z_A Z_B}(k)"] \arrow[d, swap, "k_{AZ_A}(k)"] & Z_B \arrow[d, "k_{BZ_B}(k)"] \\
A & B
\end{tikzcd}$$

Annotation Authentication of public keys: The public key of every participant is authenticated by “its own” center and sent to the others center (Z_A sends the key c_A to Z_B, signed with s_{Z_A}). The center of the other one checks the authentication and authenticates the key itself if necessary so that the recipient can check this second authentication (Z_B checks with t_{Z_A} and signs c_A with s_{Z_B}).

$$\begin{tikzcd}
Z_A \arrow[r, shift left, "{c_A,\, s_{Z_A}(c_A)}"] \arrow[d, "{c_B,\, s_{Z_A}(c_B)}"'] & Z_B \arrow[l, shift left, "{c_B,\, s_{Z_B}(c_B)}"] \arrow[d, "{c_A,\, s_{Z_B}(c_A)}"] \\
A & B
\end{tikzcd}$$

The resulting key exchange is of course only as confidential as the weakest key exchange sub protocol.

Annotation: In the case that the authentication is being done by digital signature systems, an alternative would be that the center of the other participant is not authenticating the public key of the first participant but the public key of his center. Then the participant can validate the authentication of the public key of the other participant on his own.

$$\begin{tikzcd}
Z_A \arrow[r, shift left, "{c_A,\, s_{Z_A}(c_A)}"] \arrow[d, "{\begin{array}{r} c_B,\, s_{Z_B}(c_B), \\ t_{Z_B},\, s_{Z_A}(t_{Z_B}) \end{array}}"'] & Z_B \arrow[l, shift left, "{c_B,\, s_{Z_B}(c_B)}"] \arrow[d, "{\begin{array}{l} c_A,\, s_{Z_A}(c_A), \\ t_{Z_A},\, s_{Z_B}(t_{Z_A}) \end{array}}"] \\
A & B
\end{tikzcd}$$

The advantages of this alternative are

+ Centers can store less (in the previous alternative Z_B has to store $c_A, S_{Z_A}(c_a)$ since only then Z_B can justify the issuing of the $c_A, S_{Z_B}(c_a)$ certificate).
+ It is clear who has checked what (and who is legally responsible).
+ It is transparent to the participant how many steps the authentication of the public key took
+ This alternative can be applied more generally (cp. exercise 3-6 part g).

The disadvantages are:

− The participants have to check several signatures to prove the authenticity of the key.

From my point of view the advantages are significantly prevailing - and they will increase with the growth of the participant's computation power.

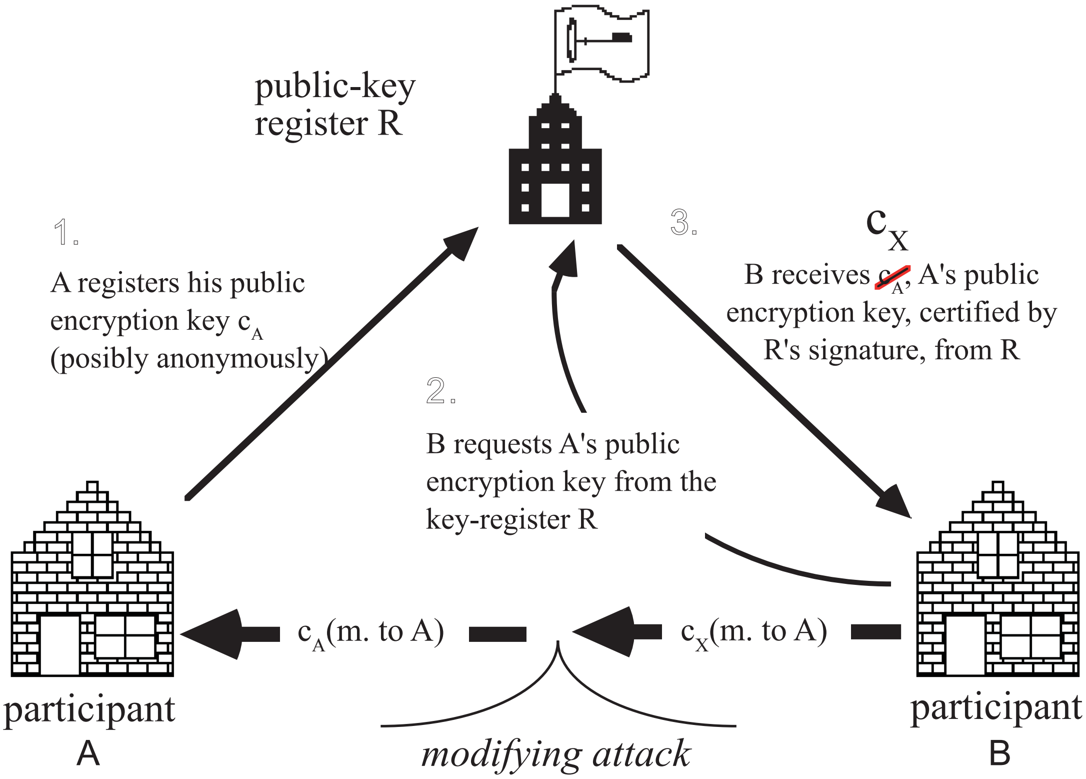
b) *Optional*: Likely your solution is made of several key parts that are exchanged with two different key distribution centers (one per country), and the participants are added modulo 2. This is better than to exchange the key as a whole. However not all key exchange centers per country need to cooperate to get the key. It is sufficient, if one of the two cooperates. This means that the decision of the participants which are the centers in their country that will not cooperate at all is circumvented. When we suppose that all key distribution centers have exchanged keys, then there is a better solution that respects the local decision of both participants:

   In the first phase one of the participants exchanges key parts over all his key exchange centers with all the others centers. Then the participant and the others participants key distribution centers add each of them modulo 2 to gain a secret key. If the participant has a key exchange centers and the other one has b then b-time a partial keys (so $a \cdot b$) are exchanged to gain b keys.

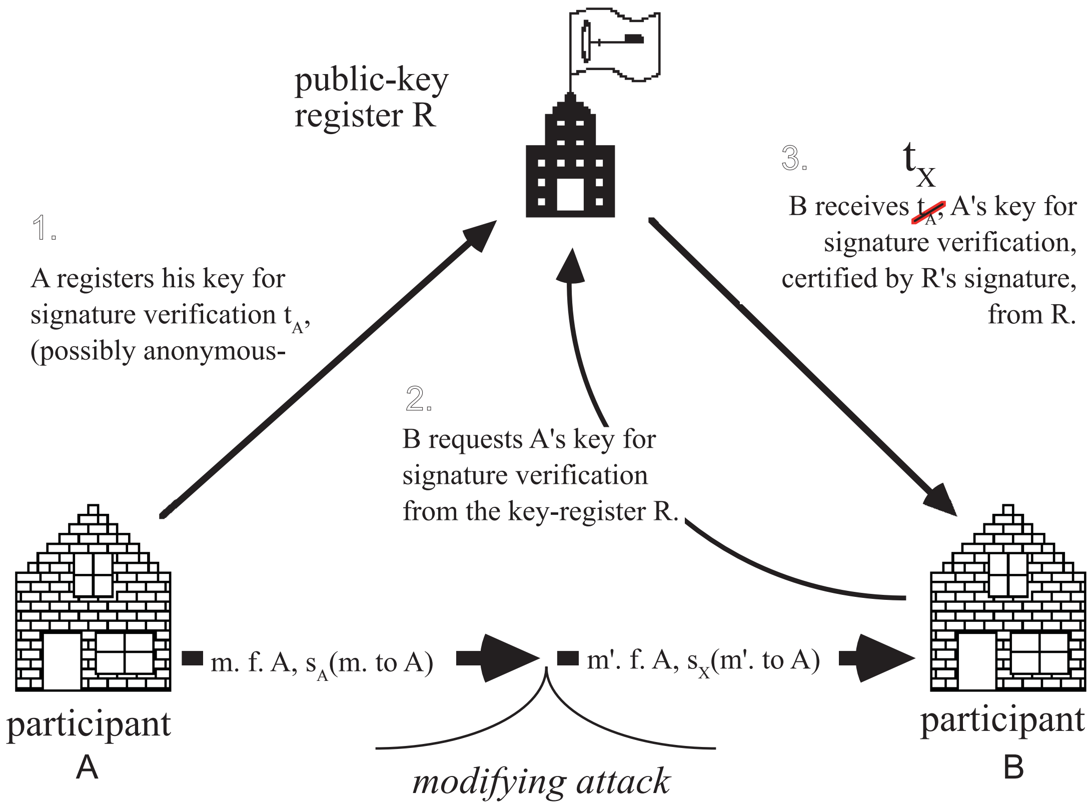
   In the second phase those b secret keys (exchanged with the key distribution centers of the other participant) along with the keys exchanged between the other participant and his key distribution centers are used to exchange b partial keys (which of course have nothing in common with partial keys from phase 1) between the participants (cp. Figure 3.3).

c) Yes, parallelizing is useful for asymmetric systems too. There are methods existing for an **altering attack** on a single (or a few) key registers R: It could hand out wrong keys to which the key register knows the secret keys. As seen on the following figures it could on the one hand break confidentiality of the messages unnoticed by reencrypting them. On the other hand it could fake them by resigning or newly signing.

Altering attack on concealment by reencryption of messages

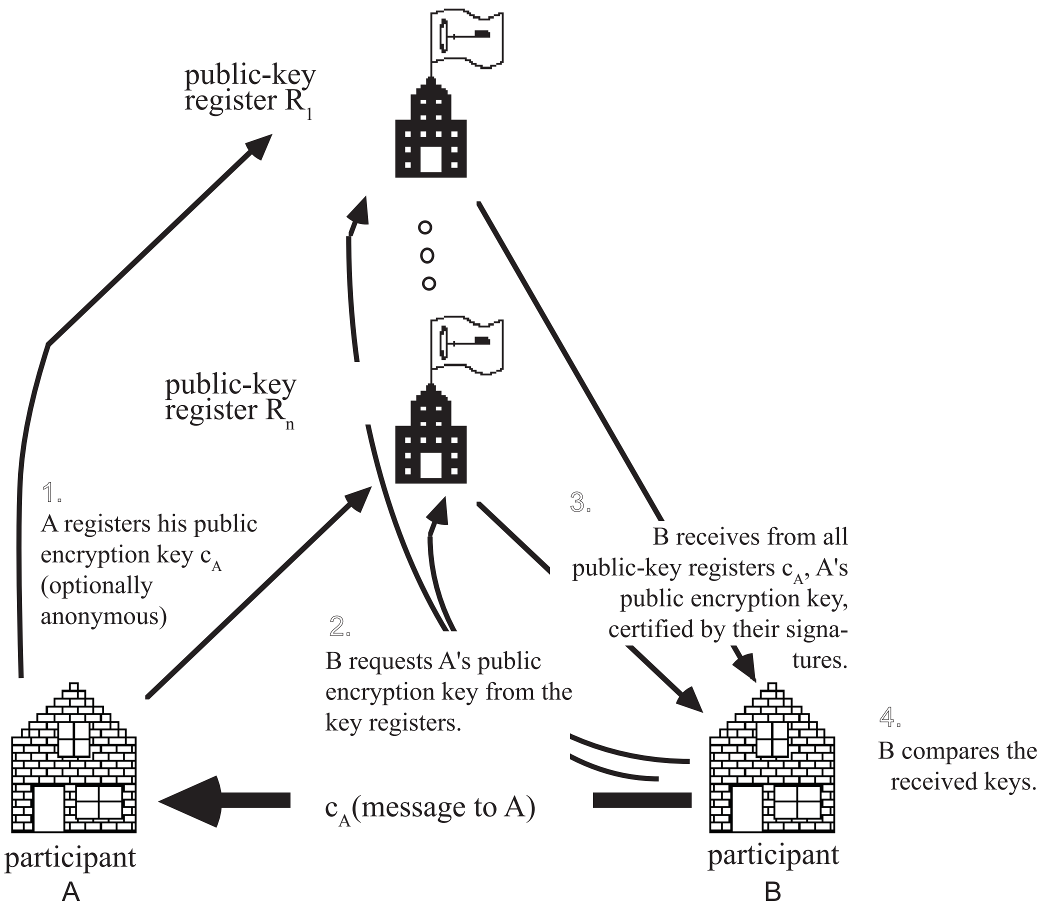


Altering attack on authentication by resigning of messages



The parallelizing is done as follows: Every participant A[2] publishes his own public key to several public key registers. Every participant B requests A's public key from these registers and compares them. If some of them are not equal either the signature system was broken (modification on its way to participant B) or at least one of the key registers is cheating, or participant A did not tell every register the same key. The latter case can be excluded by “honest” key registers by synchronizing the key entries. Again you can not distinguish all cases clearly so availability is not guaranteed.

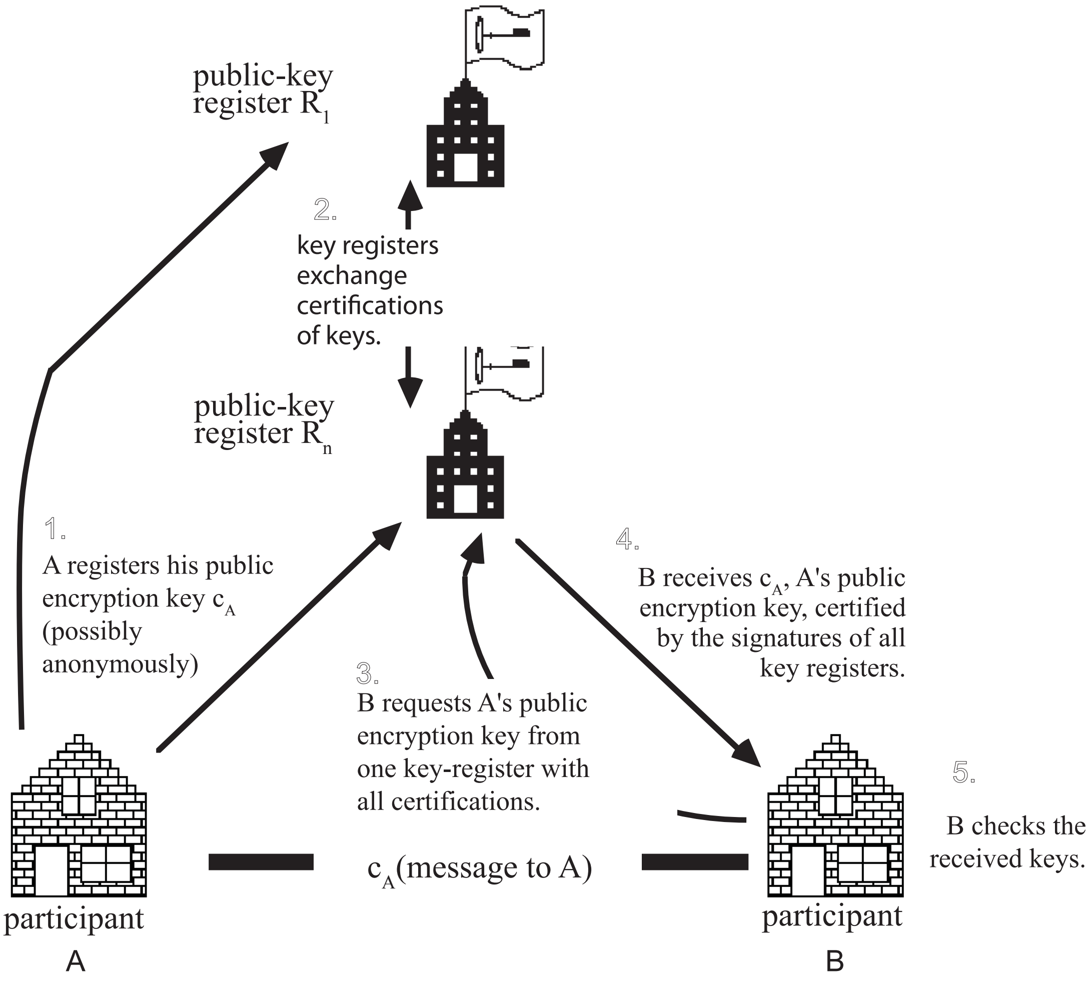
[2] This is pretty simple but not precise. It should be read as: Every participant in a role that has been used by participant A before ...


public-key register R_1
public-key register R_n
1.
A registers his public encryption key c_A (optionally anonymous)
2.
B requests A's public encryption key from the key registers.
3.
B receives from all public-key registers c_A, A's public encryption key, certified by their signatures.
4.
B compares the received keys.
c_A(message to A)
participant A
participant B

To make at least some cases distinguishable the participant who passes a public key to a key register should provide a written "statement" showing which key is his public key. Alternatively the participant should receive a statement from the register which key they have put into their database. This second signature can also be a digital signature since the register's public test key is known. After this exchange any disputes between both parties are as easy to solve as the weakest of the two systems is secure.

There are at least two variations to parallelize asymmetric cryptographic systems:

a) If public keys are to stay valid forever you do not have to check for their expiration. Hence every key register may save the other's key registers signatures with the keys. Then a participant may request the key and all the signatures associated with it of another participant from *one* key register. This saves additional communication. Nothing will change for the signature validation.


public-key register R_1
2.
key registers exchange certifications of keys.
public-key register R_n
1.
A registers his public encryption key c_A (possibly anonymously)
3.
B requests A's public encryption key from one key-register with all certifications.
4.
B receives c_A, A's public encryption key, certified by the signatures of all key registers.
5.
B checks the received keys.
c_A(message to A)
participant A
participant B

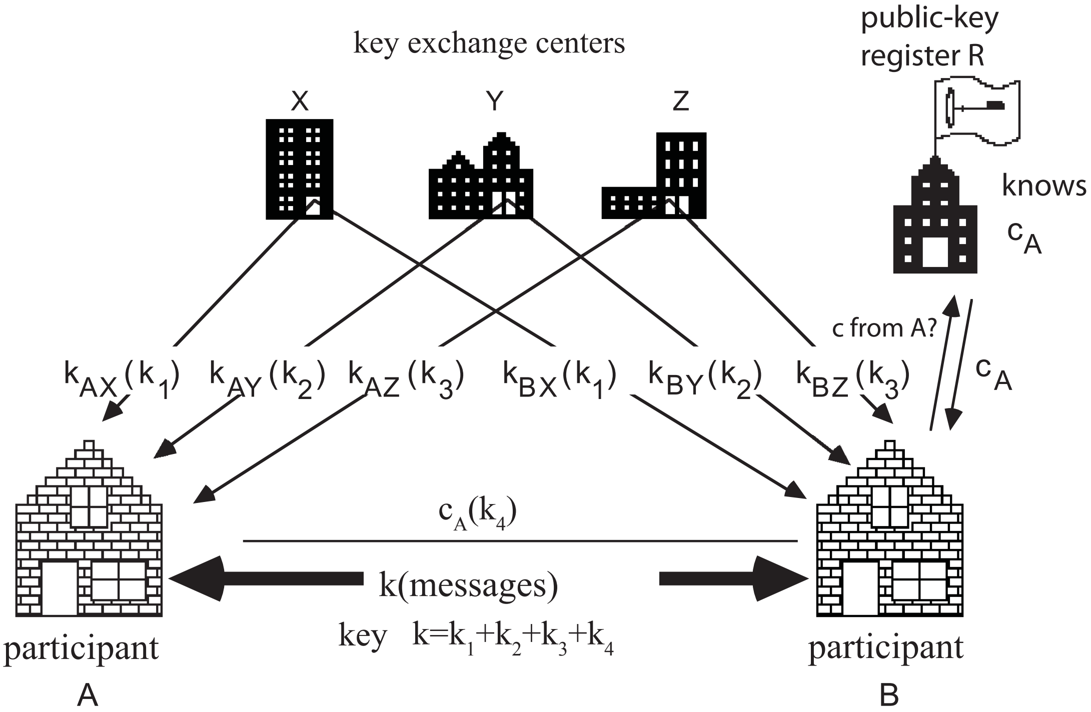
b) If you have doubts about the security of the asymmetric cryptographic system you may use several of them instead of one (cp. Exercise 3-5). And instead of using a single multi-authenticated key you would use multiple keys correlating to the numbers of systems used. For the concealment system KS_i the key text of KS_j is encrypted with KS_{j+1}. For the digital signature system the plaintext is signed and the signatures are concatenated.

d) Yes. As to be seen in the next figure you conduct the known key distribution protocol, each for symmetric and asymmetric concealment, in *parallel*:[3]

A participant B then sends a key k_4 of the symmetric encryption system to the other one. The key is then concealed with the public key c_A (that is known to

[3] If you want to resist modifying attacks on the public key register you can and should use the paralelizing (which is not shown in the figure to keep it simple) for the public key registers (cp. answer to Exercise 3-6c)

B) of his partner A. Both partners use the sum k of all the symmetric keys (k_1, k_2, k_3) they received from the key distribution protocol and the additional key k_4. To acquire that sum an attacker would have to gather all summands. And this would require the passive help of all key distribution authorities *and* to break the asymmetric encryption system.



After the key exchange you have to use a *Vernam cipher* (one-time pad) for concealment and an *information theoretical safe authentication code* for authentication. If you suspect *active* attacks from key registers the single key register should be extended by further ones (cp. Part c).

Annotation : The following variation of the question can be relevant *in practice*: We do not assume that the attacker (while not receiving passive assistance from the key distribution authorities) is complexly theoretically unlimited but “only” able to break asymmetric cryptographic systems. Then you will not need a Vernam cipher nor information-theoretically secure authentication codes.

e) If the key is used within a symmetric encryption system the partner can not decrypt correctly anymore. If the key is used in a symmetric authentication system the messages that were “correctly” authenticated would be classified as “fake” too.

To prevent any hidden alteration of the keys, they should also be authenticated instead of merely being concealed (in which order this is to be done is subject to

Exercise 3-17). If the authentication for the messages used for the key distribution is valid then the circle of suspects is limited to the key distribution authorities and both participants. The participants should make sure they have the same keys by using a *testing protocol for key equality* right after the key exchange. This can be done by each one sending encrypted random numbers and returning them in plaintext to the partner (This little crypto protocol is of course only secure if the symmetric concealment used can withstand at least one chosen ciphertext-plaintext attack (cp. §3.1.3.2) and if an altering attacker with knowledge of the inconsistent keys positioned between the participants can be excluded. Additionally you need to prevent the “mirror attack” from Exercise 3-22 part a) by having one participant sending the random numbers and waiting for the response before the other one does so. Finally you can not use the Vernam cipher with this protocol! Because then the attacker would gain the exact region of the key that is being tested for equality. In short: We need a better testing protocol or a very good block cipher. The first one is shown right next).

First of all we will look at a *wrong* solution. However this will lead us in the right direction:

Within the network it is known how to create checksums over strings (namely parity bits, Cyclic Redundancy Check (CRC), . . .). The participants create such a checksum and one of them sends the result to the other one. If the checksums are equal so both participants may think that (with a very high probability) they have the same key. So where is the error in the reasoning? Simply said, checksums are made to prevent stupid errors but not for preventing intelligent attacks. For the usual checksums it is easy to find many different values that have the same checksums. So this can be done by the attacker to distribute inconsistent keys. Then the two participants would recognize this only if they are using the key. A clever creation of a checksum would be a process with an outcome the attacker can not predict!

In [BeBE_92, pg. 30] (with reference to [BeBR_88]) you will find the following nice idea for an *information theoretically secure testing protocol for key equality*: The participants pick randomly (and independently from the previous pick) several times one half of all key bits and tell each other the sum mod 2 (If the keys do not match it is discovered with a probability of $\frac{1}{2}$). To prevent the wiretapping attacker from knowing anything about the forthcoming keys one bit of the sum mod 2 is removed (for example the last one). The key is shortened one bit for every iteration. This is no problem because you only need a few iterations: After n steps the probability that the result is $1-2^{-n}$ and for $n > 50$ sufficiently high (For this testing protocol for key equality too you have to exclude an altering attacker with knowledge of the inconsistent keys between the two participants: He could modify the sums sent to make them look consistent to the other participant).

One key distribution authority: If only symmetric authentication is used for the key exchange (so neither digital signatures nor unconditional secure pseudo-signatures;

cp. §3.9.3) then from each participant's view the other participant or the key distribution authority is lying (this has been proven for a long time within the context of the Byzantine Agreement protocols [LaSP_82]). Either the participants are giving up or they pick, if possible, another key exchange authority.

Several key distribution authorities in sequence: If the two participants detect inconsistencies in their keys, then the key distribution authorities should check the consistency with the same method as well and, if possible, circumvent attacking key authorities. This is equal to the changing of the key exchange authority for "one key exchange authority".

Several key exchange authorities in parallel: If the participants detect inconsistencies in their keys' sum they should repeat the testing protocol for equality for every single one of the added keys. All inconsistent ones are omitted from the sum. If there are not enough keys left to be added the participants should use additional key distribution authorities or even abstain from communication under these circumstances. Otherwise an altering attacker between the participants can prevent the inclusion of keys into the sum that are unknown to him.

f) Ideas (maybe incomplete; further ones are welcomed):

   a) Keys contain a part representing their expiration time. Then the participants only need some correctly working clocks for their security. Disadvantage: If a key is compromised before its expiration, the usage of it can not be stopped.

      A related solution would be to limit the amount of usages of the key. This only makes sense for symmetric cryptographic systems where both participants can count. Of course they have to keep permanently track of the keys expired.

   b) Explicit revoking of keys - either by the key owner or the key's certificate authority. This can be realized by a digitally signed message that contains, along with the revoked public key or certificate, the exact point of time of revoke and the revoker's identity (If the key owner is the revoker this authentication is the last action undertaken with the associated secret key).

      Of course, explicit revoking of keys only works with participants that can not be cut off by an altering attacker. This additional restriction in contrast to 1. is the price for higher flexibility.

      However there are protocols that enable participants to detect their isolation very fast: They send each other messages after a specific number of time units previously agreed upon. All messages are numbered and authenticated (including the enumeration). If there have not been any messages for some time or if some of them are missing, the participants can conclude that somebody tried or is trying, at least temporarily, to isolate them.

   c) We combine 1. and 2. to have the advantages of both worlds.

d) Prior to the usage of the key, the owner is asked if it is still valid. Disadvantage: If the participant is not always online, large delays may occur if the answer is being waited for without a timeout.

e) A variation of 4. by waiting only a relatively short amount of time. Since an altering attacker can suppress the answer for a short time unnoticed, this should be combined with variant 3.

g) What needs to be taken care of while exchanging public keys?
→ The public keys need to be *authentic*, therefore the recipient needs to be able to verify the authenticity.

With whom can the task of a trustworthy public key register be distributed?
→ The functionality of the trustworthy public key register could be distributed among some of the other *participants*.

When does the key exchange take place?
→ Instead of letting public key registers (one or more; cp. part c) certify the association between your identity and your public key by its digital signatures the participant certifies this association by all participants he knows (and with whom he can safely communicate, for example outside the system to be secured). Those certificates are passed along with the public key. The recipient of such a publicly certified key now decides if he wants to trust its authenticity. For this you could use certificates with known and authentic testing keys. To use those other certificates you need certificates for their testing keys and so on. Usually a public key t becomes more trustworthy to participant T the more participants (called *introducer* in [Zimm_93]) certified key t in a manner participant T can verify that. On the other hand the certificate will be more trustworthy to T the less certificates are needed by T for its authentication.

What is to be done if everyone is not trusting everybody in the same way?
→ Instead of considering all introducers equally trustworthy you can assign them individual probabilities for their trustworthiness. Consequently you can also calculate a probability for trustworthiness for every certificate chain as well as for every multiply certified public key. The participant can now decide if the probability of trustworthiness is sufficient for the application.

What is to be done if a participant discovers that his private key is known to somebody else?
→ If there is no central hierarchical key distribution then there is no central hierarchical list of all invalid keys. So the poor participant now needs to send his signed message "My key is invalid from now on" to all other participants. If legal stuff comes into play the receiving of this *key-revocation-message* needs to be acknowledged by all with date and time (An obvious but wrong solution would be: Everybody using a key must ask the owner if the key is still valid. Unfortunately

now everybody who knows the secret key can give the correctly signed answer: "Of course my key is still valid. I guard my secrets!!!").

What should a participant do if his secret key was destroyed?
→ Besides the creation of a key pair and the publication of the public one he should tell everybody not to use the old key for new messages. To prevent that not everybody is able to do so, this message should be signed - optimally with the key that is just considered as destroyed. So right after the generation of a key pair the message "Please do not use my key t anymore!" should be signed as first. This message does not need to be kept as secret as the secret key of which mostly no copies exist. But this message can be stored with several copies. Often it is useful to distribute it with a threshold scheme (cp. §3.9.6) among several other friends.

What is to be done if a participant discovers that he certified a wrong association Participant ↔ Public Key?
→ The participant revokes this certificate with a signed message. Regarding the distribution of this *Certificate-Revokation-Message* the same as for the revealing of the secret key (cp. above) applies.

How to combine the technique developed in this exercise part with the one from exercise part c)?
→ This key exchange method can be easily combined with the methods from c) and d). Central hierarchical key registers and anarchical key distribution à la PGP/GPG support each other. The safety is higher than with only one system.

See p. #466
See p. #471
Exercise at page 415.

3-7 What kind of Trust is necessary for which function of a TrustCenter?

Blind trust for an entity is always necessary if it can make observing attacks. If an entity can only make altering attacks you can shape the procedures in order to alert the attacked participant. Conversely, if the participant does not recognize an attack, then he can justify his trust.

- *Key generation* requires blind trust since the generator can always store a copy locally.
- *Key certification* only requires checkable trust since with a wrong key certification the affected participant would see it during the first conflict. With corresponding procedures (cp. §3.1.1.2) he can even prove this attack.
- *Directory services* only require checkable trust since the providing of false keys can be proven if answers to requests to the directory service are digitally signed.

- *Locking services* only require checkable trust (cp. directory services).
- *Timestamp services* only require checkable trust if a suitably distributed log file exists, since it prevents the service from back-dating (cp. [BaHS_92, HaSt_91]).

These ruminations allow the conclusion that if security functions are bundled into one entity you should only bundle key certification, directory service, locking service, and timestamp services but never the key generation (cp. Exercise 3-2,3-3 part a).

Exercise at page 416.

3-8 Hybrid encryption

Case 1:

a) Generation of a key k with a fast symmetric encryption system like AES. Encryption of k with the public cipher key c_E of the recipient E. Encryption of the file with k. Sending: $c_E(k)$, $k(file)$.

   Receiver decrypts the first part with decipher key d_E and gains k. He uses k to decrypt the file.

b) Generation of a key k with a fast symmetric encryption system that conducts concealment as well as authentication like DES running in PCBC mode (cp. §3.8.2.5; if you like to use two pure cryptographic systems you must replace k with k_1, k_2 and continue as seen in Exercise 3-17. Signing of k (maybe with additional date and time, etc..) with your own secret signing key s_S (S as in Sender) and then encryption of the signed k with the public cipher key c_E of the recipient E (to conceal first and then sign would work too). Encryption and authentication of the file with k. Transmission: $c_E(k, s_S(k)), k(file)$ (If the symmetric encryption system is less complex and if the signature $s_S(k)$ is not concealed with it then the message format $c_E(k), k(s_S(k), file)$ can be better.).

   Receiver E decrypts the first part with his decipher key d_E and gains the signed k and checks the signature with the testing key t_S. If the signature is valid E uses the key k to decrypt the file and to check the authenticity (For pure cryptographic systems: Let k_1 be the key for a symmetric encryption system, k_2 key for a symmetric authentication system. By replacing k with k_1, k_2 you send: $c_E(k_1, k_2, s_S(k_1, k_2)), k1(file, k_2(file))$. The authentication of k_1 is not necessary: $c_E(k_1, k_2, s_S(k_1)), k1(file, k_2(file))$ will do fine too.)

c) Like b) but the symmetric cryptographic system only needs to perform the authentication.

Case 2:

Here S can perform the same as in case 1 therefore creating a key k for every file, encrypt it asymmetrically and so on. E must proceed as with case 1, too. Alternatively S can reuse the key k created for the first file transmission - and tell E about this. This can save a lot of complexity: No additional key generation, key distribution, key transmission, and key decryption - and for b) and c) no key signing, no signature transmission and no signature validation.

Case 3:

Here E can perform as with case 1, therefore creating a key k for every file, encrypt it asymmetrically for S and so on. If E wants to reuse key k for the first file transmission as S did in case 2 you must be cautious: E did not create k on its own and needs to make sure that k is only known to the intended recipient of his file. For b) and c) it is assured by the digital signature from S. For a) E can not be sure who has generated k since here k is not authenticated at all, especially not by S. If at a) E would not create a new symmetric key k' but use the old k an attacker A that has generated the original k and sent the first file to E could intercept the message k(file from E addressed to S and decrypt it with k.

And what do we learn? A protocol that creates a random key can not be used (at least not without further deliberation) with an *existing* key. This is according to the fundamental proposition of cryptographics: All keys and initial values should be picked randomly and independently from each other - unless they need to be picked in dependence, or the picking is thoroughly analyzed and includes a proof of security.

Exercise at page 417.

3-9 Situations from practice (for systematics of cryptographic systems)

a) A *symmetrical authentication system* is sufficient to spot alterations at each other trusting partners. A key exchange between old friends is no problem. The system is neither very security critical nor time critical. To keep rental options and preferences of the crypto folk secret from curious renters that are (or their friends) employed as communication technicians you can additionally use symmetric concealment. So picking a cryptographic system for the Pfitzmann family is an agony for choice.

b) *Symmetric encryption system*, direct key exchange, security critical (cryptographers!), time is uncritical. David will use a one-time pad or a pseudo one-time pad - if it is only about stealing ideas. If David is also interested in having his data transmitted unaltered to the trustworthy cryptographer, then an additional authentication is necessary.

Since David has probably already met all cryptographers that are trustworthy to him a direct key exchange should be no problem. But the question arises if David wants to encrypt his messages for each cryptographer separately - which would be necessary with canonical key distribution (every cryptographer with another key). If David wants to occasionally send ideas to a certain group then he could give everybody of the group the same key - this would save quite some encryption effort. In terms of confidentiality this would not be a problem since the plaintext would be known to everybody from the group. However if David wants to exclude somebody from the group - then he must send everybody else from the group a new key. Such a group key is also problematic in terms of authentication. Because now everybody could authenticate the new idea with the symmetric authentication system and therefore tarnish David's reputation of having new and good ideas. With pairwise key knowledge this could not happen.

c) A *symmetric authentication* is sufficient with actually no need for a key exchange. But time and space are very critical. You could use DES (running in CBCauth mode cp. §3.8.2.2).

d) The computer freak should use a *digital signature system*. The key exchange has to take place via a central authority or by a phone book to make the seller know about the right testing key of the buyer. And to make sure that the association buyer → testing key is legally binding. Since the cryptographic system is time uncritical but security critical the computer freak should use GMR.

e) *encryption systems* for keeping business secrets (maybe even *authentication* cp. below), but symmetrical is sufficient because it is not a legally binding contract). The key exchange should run over a public key register. So because of the key exchange an asymmetric encryption system is needed, like RSA (or a hybrid, cp. §3.1.4, like RSA combined with DES). If the request is done via FAX, with 9600 bit/s or ISDN with 64 kbit/s, the application is not time critical. If the company uses broadband communication then the usage of a hybrid concealment system is necessary (cp. §3.1.4). The security of RSA and DES should be way sufficient for this not very security critical application.

   If authentication is not used at all, the following attack can circumvent the security of business secrets even if a perfect encryption system is used: To gain knowledge about the prices of the competitors the attacker just sends them an according request (including the key for the encryption system). If only network addresses are used as “sender” and therefore as authentication, the attacker just has to intercept the reply message to make the real recipient not suspicious. If the sender is generated by the network the attacker has to adapt it to the connection of the attacked one.

   For the reply you will need additional asymmetric authentication besides the concealment to make the offer legally binding (cp. d)).

Exercise at page 417.

3-10 Vernam cipher

a) The key 00111001 may be exchanged. The plaintext 10001110 is supposed to be encrypted. Using addition modulo 2 the encrypter receives the ciphertext 10110111. By addition of the key and ciphertext modulo 2 the encrypter receives the plaintext. Written down as a computation it looks like this::

encrypter:		*decrypter:*	
plaintext	10001110	ciphertext	10110111
⊕ key	00111001	⊕ key	00111001
ciphertext	10110111	plaintext	10001110

b) Let plaintext x, ciphertext S, and key k be elements of the group. By using the definition of the Vernam cipher the following is valid: The encrypter calculates $x + k =: S$. The decrypter calculates $S - k =: x$ (addition of the inverse element is written as subtraction here). If an attacker finds out S then for each plaintext x' exactly one key k' exists, because in each group the equation $x' + k' = S$ can be solved for k': $k' = -x' + S$. Since all keys have the same probability from the attacker's point of view, the attacker will not get to know anything about the plaintext when he knows the ciphertext S.
Annotations:

1) The proof uses no commutativity, thus it is valid for *arbitrary* groups. Check if your proof possibly is valid only for Abelian groups.
2) The proof ignores that according to the communication protocol, the attacker gets to know something about the plaintext: usually its length, at least his maximum length. The first can be avoided by padding messages to standard length and/or by adding meaningless messages, cp. §5.4.3. The best is to transmit a permanent stream of characters to ensure that an attacker can not recognize when no messages are being transmitted, cp. §5.2.1.1.

c) The result is unambiguous only if the coding operation is assumed as agreed. (This agreement can be considered as part of the definition of the encryption system or of the key.) If addition modulo 2 is agreed as operation, the following plaintext arises:

ciphertext	01011101
⊕ key	10011011
plaintext	11000110

If addition modulo 16 is agreed, however, as an operation, another plaintext arises:

ciphertext	01011101
$-_{(16)}$ key	10011011
plaintext	11000010

if it is calculated character wise (4-bit-wise), otherwise the following arises:

ciphertext	10111010
$-_{(16)}$ key	11011001
plaintext	11100001

written using our convention: 10000111.

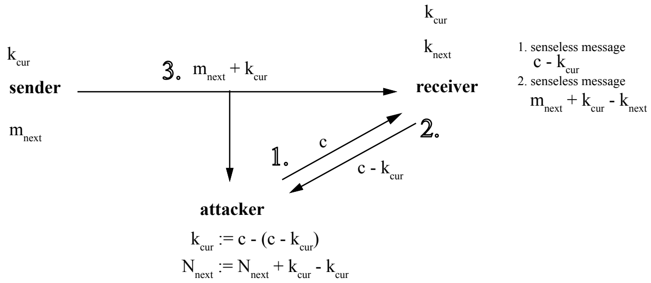
d) Calculating modulo 2: Since each bit is en– and decrypted, it is valid: Invert the third bit from the right, i.e., transmit 01011101010111010101011**0**01.

> (Modulo each number causes a change of the ciphertext from S to S' and a change of the plaintext about the same difference $d := S' - S$, because S is encrypted as $x' = S' - k = (S + d) - k = (S - k) + d = x + d$.)

The following calculations assume that it is written character-wise from the left to the right.
Calculation modulo 4: Because there are always two groups of bits being en– resp. decrypted, it has to be decided what has to happen with the third last and fourth-last bit. If 01 modulo 4 is added to them, the third last bit is inverted for sure, but the fourth-last bit too, provided that an add carry exists. If the attacker does not know the third last plaintext bit, he can not prevent the change of the fourth-last plaintext bit deterministically. Because nothing was said in the exercise about the fourth-last bit, one can accept the following as a solution: The attacker transmits 0101110101011101010101**00**01. Should the change of the fourth-last bit be avoided with unknown third last plaintext bit deterministically, the exercise is unsolvable for modulo 4.
Calculation modulo 8: Because there are always 3-groups of bits being en– resp. decrypted, it has to be decided what has to happen with the three last bits. If 100 modulo 8 is added to them, the third last bit is inverted for sure. Thus the attacker transmits 0101110101011101010101**0001**.
Calculation modulo 16: Because there are always 4-groups of bits being en– resp. decrypted, it has to be decided what has to happen with the four last bits. If 0100 modulo 16 is added to them, the third last bit is inverted for sure, but the fourth-last too, provided that an add carry exists. If the attacker does not know the third last plaintext bit, he can not prevent the change of the fourth-last plaintext bit deterministically. Because nothing was said in the exercise about the fourth-last bit, one can accept the following as a solution: The attacker transmits 0101110101011101010101**0001**. Should the change of the fourth-last bit be avoided with unknown third last plaintext bit deterministically, the exercise is also unsolvable for modulo 16.

e) Because every section of the key is used by the attacked person only once and all sections are generated stochastically independent of each other, an attacker can learn nothing from earlier reactions about future reactions. (In principle such an attack exists of course, but it has just no use).The same is true for active (not adaptive) attacks.
It is has been observed, however, that both sorts of active attacks in regards to *messages* can be successful: By an active attack (chosen ciphertext-plaintext-attack) on the recipient an attacker finds out the value of the just current key section k_{cur}. Then he is able to decrypt the next messages N_{next} encrypted by the sender. What we learn is: The partner who decrypts is not allowed to output decrypted messages and make an active attack possible in this way. It is allowed

to deviate from this restriction, if the received message is authenticated correctly. (For obvious reasons with an information-theoretically secure authentication code.)



An additional annotation: Vernam cipher keys (or at least key sections) must be applied by one partner only in order to encrypt and from the other partner only in order to decrypt. It is in no case enough that two partners have a common key and always the one who wants to send just a message to the other, uses for this as a "key" the next key section. Otherwise it can happen that both partners send messages at the same time, so that an attacker finds out the difference of their plaintexts.

f) Since this exercise is about properties of the en– and decryption function itself, i.e., independent of single keys, the abbreviated notation introduced in §3.1.1.1 is chosen clumsily (The functions themselves are just not present in this definition), so that we go over to the more detailed notation likewise described in the mentioned section. First the definition (1) is written down in the more detailed notation:

Definition for informational theoretically security in the case that all keys are chosen with same probability:

$$\forall S \in S \exists const \in \mathbb{N} \ \ \forall x \in X :\mid \{k \in K \mid enc(k,x) = S\} \mid = const \tag{1}$$

We show: At least if, as with all Vernam cipher, key space = plaintext space = ciphertext space, it is *necessary* and *sufficient*: The function *enc* has to run through the whole co-domain in case of retaining one of both input parameters and varying the other.

1 (retaining k) In the case of constant k each ciphertext must be encrypted uniquely, thus occurring only for one plaintext (i.e., enc(k, •) is injective). Since there are as many ciphertexts as plaintexts, it is equivalent that each ciphertext occurs at least once (i.e., dec(k, •) is surjective).

2a (retaining x, passing through is *necessary*) From definition (1) follows that behind each actually appearing ciphertext S each plaintext x is possible. It follows from 1. that each value S of the ciphertext space occurs actually. This is formally written in the following way: $\forall S \forall x \exists k : enc(k,x) = S$. This is equivalent to $\forall x \forall S \exists k : enc(k,x) = S$, i.e., that all S can occur in case of retained x.

2b (retaining x, passing through is *sufficient*) Do all S occur the other way around when retaining x security follows: If S is given, thus for each x exists a k, so that $enc(k,x) = S$. In order to show that all x have the same probability, too, we do need that again because of the same size of the spaces $enc(k,x) = S$ is only valid for one k.

For the binary case of the Vernam cipher we search 2x2 functional boards, in which in every line and column stands a 0 and 1. There are exactly two which represent the function XOR and NOT XOR. (That there are exactly two, one can see for example in this way: If one specifies the value for one combination of both input parameters the function is clear, because the result of the function must change each time when the parameter values are varied. Since one has 2 possibilities for the first function value it follows that there must be two functions.)

		x	
	$S = k(x)$	0	1
k	0	y	$\bar{y}$
	1	$\bar{y}$	y

$\bar{y}$ may be the inverse to y.

		x	
	$S = k(x)$	0	1
k	0	0	1
	1	1	0

For y=0: XOR

		x	
	$S = k(x)$	0	1
k	0	1	0
	1	0	1

For y=1: NOT XOR

g) Dear younger friend!
Firstly, if it is not intuitively clear to you, you can reread in historic sources [Sha1_49], that the natural language consists of *redundancy* of more than the half and is therefore no replacement for a randomly chosen key, i.e., a key of maximal *entropy*. Therefore, one can understand the sense of longer ciphertexts, if natural language is encrypted in this way, without determining book, side, and line. One can in fact understand even the sense of the book passage used for encryption and determine it if necessary. (An exact procedure to the entire breaking by means of a pure ciphertext-plaintext-attack is outlined in [Hors_85, page 107 f.]. Entire breaking by means of a plaintext-ciphertext-attack is essentially easier.)

Secondly, if you have not recently done it, you should inform yourself about the current and future expected computing power and storage capacity of a PC. Then you will also fear that it will soon be possible to store all the world's literature on some optical disks and then decrypt automatically each ciphertext, that has been encrypted using your procedure, that contains mechanically recognizable redundancy by entire search in some hours. (Very defensive explanation for plausibility: World literature may have 10^8 volumes that average 1 MByte. Thus $10^8/600$ disks are needed for 5,25"-optical disks with 600 MB each that are usual for 1990, thus circa 160 000 optical disks. To examine them from each bit position needs $10^8 * 2^{20} * 8 = 8 * 10^{14}$ steps, where each step might last 10^{-6} s. Thus entire rummage would last $8 * 10^8$s, thus 92.593 days when using a computer with today's performance. If one starts with the most renown works of the world, the search will end a lot faster.) Finally a small comfort: You are not the first one who creeps in such a mistake. Besides, you had the luck to find a friend by publishing your procedure who points out your mistake to you. Many of your colleagues who want to learn either nothing more or whose contractors want to protect copyrights keep their procedures secret. You can imagine what often happens then.

h) No, because one can think out an arbitrary different plaintext of the same length and then calculate the key so that he fits to this plaintext and the existing ciphertext. Thus the prosecution authority gets no meaningful information.

i) One exchanged storage medium is used for 2-way communication, so that for every direction only half of the storage capacity can be used. Then arises:

	text communication (typing speed 40bit/s)	speech communication (normal ISDN coding: 64kbit/s)	high-resolution full-video communication (140Mbit/s)
3.5" floppy (1.4 MByte)	1.62 days	87.5 s	0.04 s
CD-R (700 MByte)	850 days	12.7h	21 s
DVD one-sided (4.7 GByte)	5440 days	3.4 days	134 s
Blueray ROM (50 GByte)	57870 days	36.2 days	1429 s

Exchange of a 1.4 MByte floppy suffices for text communication for most applications "for life" (since you will only like a few so much that you type 1.62 days non-stop for them). Correspondingly, exchange of optical 700 MByte of disks suffices for many linguistic communication-applications (even if many people talk more perseveringly than typing). For uncompressed high-resolution full-video communication the current mass storages are still too small, but are absolutely useful for compressed full-video communication of low dissolution like intended for ISDN (an additional channel of 64 kbit/s for the picture): To talk and watch one's telephone partner information-theoretically secure with the help of a DVD for 3.4 days, might also reach here "for life". For volatile acquaintances suffices a CD-R newly – and for intense tele-relations, a Blueray ROM will suffice in the future.

Another question is whether the reading access to media goes quickly enough to be able to access the key in real time. For text communication and linguistic communication no problem. For high-resolution full-video communication with 140 Mbit/s for all media this is not possibly up to now – another reason, why lossy compression is used for full-video signals. The first obtains app. the factor 4, with the latter it depends on the fact which quality loss one would like to accept. If you have still unpacked your pocket calculator, you can calculate the numbers for high-resolution full-video communication once more with the typical values of 34 Mbit/s for lossless and 2 Mbit/s for lossy compression. What arises? DVD is absolutely suitable for high-resolution full-video communication – for the storage of the one-time pad as well as for the storage of full-video signal. No miracle: key, ciphertext, and plaintext are of the same length – and the medium was designed for the latter.

See p. #50
See p. #74
See p. #74
Exercise at page 418.

3-11 Symmetrically encrypted Message Digest = MAC ?

a) The recipe is completely insecure, as the example mentioned in the exercise shows:

> From the intercepted message everybody can calculate the check digit by means of the publicly known procedure. Everybody can determine the current key section from the check digit and its intercepted encryption with the Vernam cipher. The check digit can be calculated by means of the publicly known procedure for each message (which the attacker can freely select) and then be encrypted by means of the current key section to an "authentic" MAC.

A selective break is reached by this passive attack which is not acceptable, as already mentioned in §3.1.3.1.

b) For the usual collision-resistant hash-functions, this is a secure construction according to current knowledge. But these hash-functions are not only collision-resistant, but they additionally have a trait which is essential for the security of the construction applied with TLS:

> Their functional value says *nothing* about the input.

To make clear why this is essential, we assume the contrary: The collision-resistant hash-function may output 80 bits of its input as a part of its functional value. If the functional values as well as the inputs are long enough, such a collision-resistant hash-function can be a super collision-resistant one. However, used with the construction applied with TLS, it shortens the key length by about 80 bits if the key "unfortunately" stands at the corresponding place of the input. If a key length of 128 bits is used which usually is more than enough, then an attacker has to try at most 2^{48} times to determine the key. Today nearly every relevant attacker is able to do this.

c) Yes: Hash ($< key >, < number\ of\ byte >, < byte >$) and transfer the functional values (if necessary together with $< number\ of\ byte >$, if the recipient does not count bytes or the communication medium is unreliable). Now the recipient only has to try out the values of the byte, by hashing ($< key >, < number\ of\ byte >, < value\ of\ byte >$

and to compare this with the transmitted functional value to decrypt the functional value to the plaintext byte. Except for the sender and recipient nobody can do this, because nobody knows the key. The numbering of the bytes prevents that same bytes result in same functional values, a synchronous stream cipher is produced this way, cp. §3.8.2. In the example it is supposed that a byte is a meaningful compromise between the number of trials in case of decryption by the recipient and the message-expansion by essentially more values of the hash-function (typically 160 bits).

Exercise at page 419.

3-12 Authentication codes

a) The key 00111001 is exchanged. The plaintext $HTTT$ is to be secured. Using authentication code $H,0\ T,1\ T,0\ T,1$ is generated as a “ciphertext” and transmitted.

b) If an attacker intercepts H or T and the corresponding MAC, two (of four) ciphertexts remain for him indistinguishable for good, because H resp. T occur exactly twice in each column of the table in which they occur. Exactly these two values distinguish, whether the MAC has to have the value 0 or 1 for the other value of “tossing the coin”. (This claim can be easily verified for this simple authentication code by consideration of all possibilities.) Therefore the attacker has no better strategy than guessing, what will lead to the fact that the recipient will recognize the falsification with a probability of 0.5. An easier representation: If $k = k_1k_2$ then is $k(H) = k_1$ and $k(T) = k_2$. (This is illustrated in the following picture taken from the exercise, where k_1 is drawn bold.) Thus the authentication code can be pictured as shown in the figure on the right: Here the value of the MAC is represented in dependency of the key and text. Obviously one does not get to know anything about the MAC for T through the current MAC for H.

		Text with MAC			
		H,**0**	H,**1**	T,0	T,1
keys	**0**,0	H	-	T	-
	0,1	H	-	-	T
	1,0	-	H	T	-
	1,1	-	H	-	T

		Text	
		H	T
keys	**0**,0	**0**	0
	0,1	**0**	1
	1,0	**1**	0
	1,1	**1**	1

c) No, because the recipient uses his key sections in a fixed order.

d) *Example:* Let the key 00111001 be exchanged. The plaintext $HTTT$ is to be secured. Then the ciphertext 00110100 is generated and transmitted.
In terms of authentication the argumentation above applies analogously: For both key values that are possible after being observed by an attacker he has to take different ciphertexts in case of the intended change of plaintexts. Thus again he

has no better chance for success than guessing right with a probability of 0.5.
If an attacker executes a modifying attack, however, he can recognize in every case in the reaction of the attacked person (accepts faked message or not), which of both possible key values that are still possible after his first observation are present, and thereby he is able to determine exactly the key value. From this value and the observed ciphertext arises the clear text. This is explained with an example: The attacker intercepts the ciphertext 00. Then he guesses from the two possibilities the key value 00, thus the plaintext H. He changes the plaintext to T by forwarding the ciphertext 10. If the recipient accepts this, his guess was right, thus the original plaintext was really H. If the recipient does not accept, the value of the key was 01, and therewith the original plaintext is T.

e) An additional key bit is used. If it is 0 everything stays the same. If it is 1 all T are replaced by H in the table and vice versa. Even with the optimal attack over the key regarding to acquisition of information described above the attacker does not know the value of the additional key bit. This key bit acts as a Vernam cipher so that the encryption is still perfect even after this attack.
Now, unfortunately, the combined system is not more efficient than an authentication code and Vernam cipher, provided that only the profit bit is encoded first and afterwards authenticated. If one does it the other way round, one needs instead of 3 key bits 4.

f) It is to be shown that for each possible MAC' (i.e., $MAC' \in K$) there is exactly one value (a, b) compatible to the observation (m, MAC). The system of equations

$$MAC = a + b * m$$
$$MAC' = a + b * m'$$

with the unknown quantities a, b has exactly one solution if and only if $m \neq m'$, but $m \neq m'$ is exactly the aim of the attacker.

For those who do not remember linear algebra that exactly, here is a somewhat more precise explanation: A system of equations $C = M * X$ with the matrix M and the vectors C and X is exactly then uniquely solvable if M is regular. M is regular if and only if a triangular matrix can be created by the rows and columns of M (i.e., all diagonal elements are $\neq 0$ and either all values beneath the diagonal or all values above the diagonal are 0). This is shown for our matrix

$$\begin{pmatrix} 1 & m \\ 1 & m' \end{pmatrix}$$

shortly: Subtraction of the first from the second row results:

$$\begin{pmatrix} 1 & m \\ 0 & m' - m \end{pmatrix}$$

This is exactly then a diagonal matrix if $m \neq m'$.

To show a subtle error, another "proof" is shown in the following:

$$MAC := a + b * m \Leftrightarrow a := MAC - b * m$$

Thus b can be chosen arbitrarily, because the equation can be fulfilled by a. Thus the attacker gets to know nothing about b. In case of authentication of another message m', he ought to know b to be able to correct MAC to MAC' {because $b * m$ changes its value arbitrarily and is distributed uniformly in case of variation of m}.

Without the addition of the parentheses in the "proof", the authentication code $MAC := a + b * m^2$ would be a nice counterexample: Because the "proof" applies here, too, but here the message m can be replaced by $-m$ without having to change the MAC.

Exercise at page 420.

3-13 Mathematical secrets

a) It is not completely clear because the prime numbers are not chosen with the same probability any more, but such that primes after long gaps in the sequence of primes are chosen with higher probability and such that if prime numbers are only two larger than another prime number, they are only chosen if they were chosen as random number at the beginning.

> For specialists: At least under Cramer's assumption the procedure is secure nevertheless: This assumption (which I know from [GoKi_86]) implies $\pi(x * \log^2 x) - \pi(x) > 0$. This means that one gets the prime numbers after longest gaps only up to a factor of $\log^2 x \approx l^2$ more frequently than the above mentioned prime twins. For the products n probability differences up to l^4 arise. If there would be a successful factorization algorithm for this distribution it would be less successful only with a factor of at most l^4 for a correct distribution. Therefore its success probability decreases not faster than each polynomial, which contradicts the factorization assumption. (Without Cramer's assumption one would come artificially to the same result, if one proceeds the search by adding 2 in each step not arbitrarily long, but uses a criterion for stopping, e. g., after l^2 steps.)

b) One can expect that both prime numbers are very close together, thus both are about $\sqrt{n}$. Therefore, one can search the integers around $\sqrt{n}$ (i.e., divide n by each of them.)
More exactly: In most cases e. g., $q \leq p + 16l$ applies (prime number theorem with some leeway, because the distance to the next prime number averages $l * \ln(2)$. The particularly beautiful number 16 was chosen to alleviate the following calculations).

In these cases is $p^2 < n < p(p + 16l) < (p + 8l)^2$, i.e., $p < \sqrt{n} < p + 8l$, thus $\sqrt{n} - 8l < p < \sqrt{n}$. Thus one only has to examine this $8l$ numbers and has them factored with significant, even huge probability.

c) $19 = (10011)_2$. mod 101 we have:

$$3^{(1)_2} = 3; 3^{(10)_2} = 3^2 = 9; 3^{(100)_2} = 9^2 = 81 \equiv -20; 3^{(1001)_2} = (-20)^2 * 3 = 1200 \equiv -12;$$

$$3_2^{10011} = (-12)^2 * 3 = 144 * 3 \equiv 43 * 3 \equiv 129 \equiv 28.$$

d) $77 = 7 * 11$; $\phi(77) = 6 * 10 = 60$ and $3 \in \mathbb{Z}_{77}^*$, because $\gcd(3, 77) = 1$. Thus Fermat's small theorem says: $3^{60} \equiv 1 \bmod 77$.
Thus it is valid: $3^{123} \equiv (3^{60})^2 * 3^3 \equiv 1^2 * 27 \equiv 27$.

e) Euclid's algorithm:

$$\begin{aligned} 101 &= 4 * 24 + 5; \\ 24 &= 4 * 5 + 4; \\ 5 &= 1 * 4 + 1; \end{aligned}$$

Backwards:

$$\begin{aligned} 1 &= 5 - 1 * 4 && \text{(Now dividing 4 by 5 and substituting 24)} \\ &= 5 - 1 * (24 - 4 * 5) = -24 + 5 * 5 && \text{(now substituting 5 by 24 and 101)} \\ &= -24 + 5 * (101 - 4 * 24) = 5 * 101 - 21 * 24 \\ & && \text{(Probe: } 5 * 101 = 505, -21 * 24 = -504) \end{aligned}$$

Thus: $(24)^{-1} \equiv -21 \equiv 80$.

f) There is no inverse for 21 mod 77 since $\gcd(21, 77) = 7 \neq 1$.

g) First we apply the extended Euclidean algorithm, in order to search u and v with $u * 7 + v * 11 = 1$. (Even if one can see this perhaps at first sight: -21+22=1):

$$\begin{aligned} 11 &= 1 * 7 + 4; \\ 7 &= 1 * 4 + 3; \\ 4 &= 1 * 3 + 1. \end{aligned}$$

Backwards:

$$\begin{aligned} 1 &= 4 - 1 * 3 \\ &= 4 - 1 * (7 - 1 * 4) = -7 + 2 * 4 \\ &= -7 + 2 * (11 - 1 * 7) = 2 * 11 - 3 * 7. \end{aligned}$$

Thus the "base vectors" for $p = 7$ and $q = 11$ are: $up = -21 \equiv 56, v * q = 22$. With the formula $y = up * y_q + vq * y_p$ arises: $y = -21 * 3 + 22 * 2 = (22 - 21) * 2 - 21 = -19 \equiv 58$. (Test: $58 \equiv 2 \bmod 7 \wedge 58 \equiv 3 \pmod{11}$.)

For z we can insert into the same equation (up and vq have to be calculated only once). We simplify before: $6 \equiv -1 \bmod 7$and $10 \equiv -1 \bmod 11$. Thus: $z = -21 * (-1) + 22 * (-1) = -1 \equiv 76 \bmod 77$.
We could have had it simpler: After formula $(*)$ from §3.4.1.3 is $z \equiv -1 \bmod 7 \wedge z \equiv -1 \bmod 11 \Leftrightarrow z \equiv -1 \bmod 77$.

h) Since everything is modulo, prime numbers one can apply Euclid-criteria:

$$\left(\frac{13}{19}\right) \equiv 13^{\frac{19-1}{2}} \equiv (-6)^9 \bmod 19; (-6)^2 \equiv 36 \equiv -2 \Rightarrow (-6)^4 \equiv 4; (-6)^8 \equiv 16 \equiv -3$$

$$(-6)^9 \equiv -3 * (-6) \equiv 18 \equiv -1 : \text{no square.}$$

$$\left(\frac{2}{23}\right) \equiv 2^{11} \equiv 2048 \equiv 1 \bmod 23 : \text{square}$$

$$\left(\frac{-1}{89}\right) \equiv (-1)^{44} \equiv 1 \bmod 89 : \text{square}$$

The simple case $p \equiv 3 \bmod 4$ propounds only with p=23. Thus one is able to extract the root just with our knowledge about the algorithm: A root is

$$w = 2^{\frac{23+1}{4}} = 2^6 = 64 \equiv -5 \equiv 18 \bmod 23 \qquad \text{(Probe: } (-5)^2 \equiv 25 \equiv 2)$$

(Thus the other root is +5.)

i) It is $58 \equiv 2 \bmod 7 \wedge 58 \equiv 3 \bmod 11$. For $p = 7, 11$ $p \equiv 3 \bmod 4$ is valid, and it is

$$2^{\frac{7-1}{2}} \equiv 8 \equiv 1 \bmod 7\text{and}3^{\frac{11-1}{2}} \equiv 9 * 9 * 3 \equiv (-2) * (-2) * 3 \equiv 1 \bmod 11.$$

Thus 2 and 3 are squares mod 7 resp. mod 11. Thus we can use the simple algorithm for extracting roots to receive the roots w_7 mod 7 and w_{11} mod 11:

$$w_7 = 2^{\frac{7+1}{4}} = 2^2 = 4 \quad \bmod 7.$$

$$w_{11} = 3^{\frac{11+1}{4}} = 3^3 = 27 \equiv 5 \bmod 11.$$

With the Chinese Remainder Theorem and the already calculated "base vectors" we receive:

$$w = -21 * w_{11} + 22 * w_7 = -21 * 5 + 22 * 4 = (22 - 21) * 4 - 21 = -17 \equiv 60 \bmod 77.$$

(Probe: $(-17)^2 = 289 = 3 * 77 + 58$.)
If we insert each the "other" root mod 7 or mod 11, we receive the three different roots:
e. g.: $w' = -21 * 5 + 22 * (-4) = -105 - 88 = -193 \equiv 38 \bmod 77$.
(Probe: $38^2 = 1444 = 18 * 77 + 58$.)
Instead of proceeding with such calculations, we now take the inverse of both already known roots:
$w'' = 17$
$w''' = 39$.

j) One knows another, essentially different root of 4, viz 2.
Thus one can receive a factor p as $\gcd(2035 + 2, 10379)$. Euclidean algorithm:

$$10379 = 5 * 2037 + 194$$

$$2037 = 10 * 194 + 97$$

$$194 = 2 * 97 + 0$$

Thus $p = 97$. One receives the complete factorization as $q = 10379\mathrm{div}97 = 107$; and p and q are prime.
(For fans: One starts with the factors, here P=97 and q=107. The roots of 4 mod $p$and q are each ±2. To receive one of two essentially different roots mod p*q, for which +2 mod p and -2 mod q or -2 mod p and +2 mod q is valid, one has to create CRA(+2,-2) or CRA(-2,+2).)

k) Let $w_p := x^{\frac{p+1}{4}}$ and $w_q := x^{\frac{q+1}{4}}$.
Following §3.4.1.8 it is clear that x mod n has 4 roots w, w', w'', w''', that can be calculated using w_p, w_q of x as follows:

$$\begin{aligned} w &= \mathrm{CRA}(w_p, w_q) \\ w' &= \mathrm{CRA}(w_p, -w_q) \\ w'' &= \mathrm{CRA}(-w_p, w_q) \\ w''' &= \mathrm{CRA}(-w_p, -w_q) \end{aligned}$$

Following §3.4.1.6 is $w_p \in QR_p$ and $w_q \in QR_q$, but $-w_p \notin QR_p$ and $-w_q \notin QR_q$. Following §3.4.1.7 it is valid:

$$x \in QR_n \Leftrightarrow x \in QR_p \wedge x \in QR_q.$$

Thus it is valid, that $w \in QR_n$ and $w', w'', w''' \notin QR_n$.

l) If the factorization assumption is wrong an attacker would be able to factorize a not negligible part of the modules $n = p * q$. For such modulos he can apply Euler's criteria and calculate

$$\left(\frac{x}{p}\right) \equiv x^{\frac{p-1}{2}} \bmod p \text{and} \left(\frac{x}{q}\right) \equiv x^{\frac{q-1}{2}} \bmod q$$

for x, for which must be proofed if it is a square or not. x is a square if $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = +1$. Since evaluating of the formulas causes at most cubical effort (in the length of n), thus is especially polynomial in the length of n, an attacker is able to decide for a not eligible part of all modulos if x is a square or not. This contradicts with the quadratic residuosity assumption.
To practice dealing with the formal definitions of the factorization - and quadratic residuosity assumption, the *not negligible part* is now written down formally:
There is a (probabilistic) polynomial algorithm F, so that it applies for a polynomial Q: For each L exists a $l \geq L$, so that it applies: If p, q are chosen as independent, random prime numbers of length l and $n := p * q$

$$W(F(n) = (p; q)) > \frac{1}{Q(l)}.$$

From this algorithm F we construct another likewise (probabilistic) polynomial algorithm R. R calls F. If F factorizes, then R calculates, as just described, $\left(\frac{x}{p}\right)$

and $\left(\frac{x}{q}\right)$. If both are +1, R puts out “true”, otherwise “false”. R is successful if and only if F is successful.

Thus it applies: There is a (probabilistic) polynomial algorithm R, so that it applies for the above mentioned polynomial Q: For each L exists a $l \geq L$, so that it applies: If p, q are chosen as random prime numbers of length l, $N := p * q$, and x is coincidentally beneath the cosets with Jacobi-Symbol +1, then it applies:

$$W(R(n,x) = qr(n,x)) > \frac{1}{2} + \frac{1}{Q(l)}.$$

This contradicts with the quadratic residuosity assumption.

Exercise at page 421.

3-14 Which attacks on which cryptosystems are to be expected?

a) The notary must use a digital signature system. Because he signs arbitrary messages submitted to him, excepting the time stamp (e. g., to appending of date and time to the message), the digital signature system must resist adaptive active attacks. The message format could be as follows:
¡submitted message¿, date, time, s_{notary}(¡submitted message¿, date, time).

b) Either symmetric or asymmetric *encryption systems* can be used.

The first must resist at least a known-plaintext-attack, because the attacker can read the plaintext of the observed article in the newspaper soon. If the attacker does sensational things, he can reach at least particularly a plaintext-ciphertext attack, e. g., he could spread the text that he wants to get encrypted, in the city hall. (Both attacks are irrelevant with the asymmetric system, because the attacker can encrypt arbitrary texts on his own.)

Even if the newspaper printed arbitrary senseless articles (perhaps the correspondent accepts box number notifications, too), the encryption system (symmetric or asymmetric) must resist even a chosen-ciphertext-attack, because the attacker can read the plaintext of his sent ciphertext in the newspaper soon.

Exercise at page 422.

3-15 How long must files be protected? Dimensioning of cryptographic systems; reactions to false dimensioning; Publishing public keys of a signature system or of an asymmetric encryption system?

a) The probability that the cryptographic system will be broken this time must be sufficiently low. If an information-theoretical secure system (with the one-time pad or suitable authentication codes) is used, the following 3 points are irrelevant. Otherwise attention should be paid to:

- The time which is available to the attacker for a cryptoanalysis trivially grows with the length of the time within which the protection should be effective.
- Furthermore, the computation power/EUR probably increases exponentially with a duplication time ca. every year, until technological borders are reached sometimes. This increasing computer power will also be available to the attacker.
- Besides, more efficient algorithms can be discovered in this time. (Because for all information-theoretical secure systems there are no useful lower barriers for the complexity of breaking.)

In both examples one has to oversize the systems so that they are hopefully safe for at least 80 years.

To have a feeling what this means, an example: we assume, that the growing of the computer power would last that long. (However, this might become limited with physical borders already, cp. [Paul_78, DaPr_89, page 236 f., page 41] Then it is so, as if the attacker has approximately 2^{81} years with current computer power at a disposal. If one could not break the system faster than by an entire search in the key space, this would have to be chosen too largely around the factor 2^{81}, thus the keys about 81 bits are too long, if they are coded optimally. This would not even be a lot.

It is already different with systems based on factorization. In order to explain the progress on the one hand with the algorithms of factorization and with the computer power on the other hand, two exemplary calculations are shown:
We assume the complexity is and will be

$$\begin{aligned} L_{1990}(n) &= exp(\sqrt{\ln(n)\ln\ln(n)}), \\ L_{1995}(n) &= exp(1.923 * \sqrt[3]{\ln(n)\ln\ln(n)}), \end{aligned}$$

this is the complexity of the best factorization algorithm known in 1990 and 1995, cp. [Schn_96, page 256].

In 1995 we used n of length 512 bit and this was justified with the fact that those numbers can not be factorized within one year (it is different today, cp. [Riel_99]). Thus we should, from the perspective of 1995, suppose n^* for the future with

$$L(n^*) = 2^{81} * L(n)$$

From the algorithmic point of view of 1990 this means

$$\begin{aligned} & L_{1990}(n^*) = 2^{81} * L_{1990}(n) \\ \Leftrightarrow\ & exp(\sqrt{\ln(n^* \ln\ln(n^*))}) = 2^{81} * exp(\sqrt{\ln(n)\ln\ln(n)}) \\ \leftrightarrow\ & \sqrt{\ln(n^* \ln\ln(n^*))} = 81 * \ln(2) + \sqrt{\ln(n)\ln\ln(n)}. \end{aligned}$$

Furthermore we know that $\ln(n) = 512 * \ln(2) \approx 360$ and $\ln\ln(n) \approx 6$, as well as $\sqrt{360 * 6} = 44$, thus circa

$$\begin{aligned} \Leftrightarrow \sqrt{\ln(n^* \ln\ln(n^*))} &= 56 + 44 \\ \Leftrightarrow ln(n^* \ln\ln(n^*)) &= 100^2 \end{aligned}$$

A little calculations leads to the statement, that this inequality is complied for $\ln(n^*) \approx 1440$. Thus $\log_2(n^*) = 1400/\ln(2) \approx 2020$.

One would have to choose numbers approximately 3.95 times longer than before. The expenditure of calculation increases by the factor $3.95^3 \approx 61$.

From the algorithmic point of view of 1995 this means:

$$\begin{aligned} L_{1995}(n^*) &= 2^{81} * L_{1995}(n) \\ \Leftrightarrow exp(1.923 * \sqrt[3]{\ln(n^*)(\ln\ln(n^*))^2}) &= 2^{81} * exp(1.923 * \sqrt[3]{\ln(n)(\ln\ln(n^*))^2}) \\ \Leftrightarrow 1.923 * \sqrt[3]{\ln(n^*)(\ln\ln(n^*))^2} &= 81 * \ln(2) + 1.923 * \sqrt[3]{\ln(n)(\ln\ln(n^*))^2} \\ \Leftrightarrow \sqrt[3]{\ln(n^*)(\ln\ln(n^*))^2} &= 42,12 * \ln(2) + \sqrt[3]{\ln(n)(\ln\ln(n^*))^2} \\ \Leftrightarrow \sqrt[3]{\ln(n^*)(\ln\ln(n^*))^2} &= 29 + \sqrt[3]{\ln(n)(\ln\ln(n^*))^2} \end{aligned}$$

Furthermore we know that $\ln(n) = 512 * \ln(2) \approx 360$ and $(\ln\ln(n))^2 \approx 36$, as well as $\sqrt[3]{360 * 36} \approx 23$, thus circa:

$$\begin{aligned} \Leftrightarrow \sqrt[3]{\ln(n^*)(\ln\ln(n^*))^2} &= 52 \\ \Leftrightarrow \ln(n^*)(\ln\ln(n^*))^2 &= 52^3 \end{aligned}$$

A little calculator acrobatics results that this inequality is complied for $\ln(n^*) \approx 2400$. Thus $\log_2(n^*) \approx 2400/\ln(2) \approx 3462$. One would have to choose numbers approximately 6.76 times longer than before. The expenditure of calculation increases by the factor $6.76^3 \approx 309$.

We see: the algorithmical progress within 5 years has “as much” influence on the necessary length of the numbers as the increase of computer power expected within 80 years. Both force an extension of approximately 1.500 bits for the considered period.

(However, fortunately there are many applications where signatures must be valid only shortly, e. g., in a digital payment system if one gets a completion of an account every quarter and against this one has a right of objection for only a few months.)

b) It is clear that one uses only stronger dimensioned systems with both systems types for all data that are to be newly encrypted or to be signed, and that new keys must be exchanged additionally, etc. If one trusts in the fact that the soon breakable digital signature system is not yet broken, one can use it for the exchange of public keys of stronger asymmetric cryptographic systems. Otherwise a new original-key-exchange outside of the computer network, that is to be protected, is necessary - as well as with usage of symmetric cryptographic systems for key exchange. If one wants to regulate something legally obliging with the digital signature system, the public test key should be newly registered. In every case it is beneficial to let at least key registers confirm the exchange of the *public* keys. To avoid that key registers can explain arbitrarily keys for invalid, they should possess a digital signature (if necessary in the old cryptographic system) of the owner for a relevant message (key invalid from: date, time). The proceeding is different for messages that were encrypted / signed using the old, and too weak system in the future: *Secret data* must be encrypted with the stronger cryptographic system. This can happen according to the application, i.e., there is a safe environment for this new decryption with knowledge of keys needed for encryption of the present ciphertexts or not. It can be done by decrypting the present ciphertexts with the encryption system that is soon too weak and then encrypting the plaintext with the stronger dimensioned encryption system or by an additional encryption of the present ciphertexts. If asymmetric systems are considered, the latter might be the most frequent case. Independent from the use of symmetric or asymmetric encryption systems, proceeding after the latter case means that the implementation, of the encryption system which is soon too weak and the applied keys, must remain available. Avoiding the risk of changing the encryption (decrypting the weak, followed by encrypting with a strong system) (the plaintext is present for a short time) has its price. Unfortunately, one can never be sure whether all copies of the ciphertexts, as described, are encrypted stronger. Thus the outlined proceeding helps only against attackers that appear after the stronger encryption and also have, if necessary, difficulties in finding another copy of the only weakly encrypted ciphertext. Thus the outlined proceeding is no substitute for correct dimensioning starting from the beginning, but it is better, of course, than just waiting and wishing to keep the accruing insecurity of the too weakly dimensioned encryption system a secret.

Signed data must be provided with an additional signature in the stronger dimensioned signature system, best from the original signer. Theoretically one could oblige anybody in a transition period: everybody publishes stronger dimensioned test keys (signed for at least occasional authentication in each case with the not yet broken signature system). Then everybody is obliged to resign a message, that was signed by him with the old system, but now using the stronger system. This transition period must end of course, before the breaking of the old signature system is cheap enough. If the original Signer is not able to or not willing to do an additional signing, a time stamp service can be enabled: everyone who has a message signed by another one and wants to prevent the decay of this evidence, submits the message together with the signature (in the old signature system) to a time stamp service. This signs in the new, stronger dimensioned signature system "message, signature in the old signature system, date, time". If this is submitted later, this time-stamped signature is as authentic as it was at "date, time" in the old signature system, as the time stamp service is protected against corruption and as cryptographically secure as the new signature is to the time of submission. The protection of the time stamp service against corruption can be increased by the usual technology: instead of a signature in the new signature system a signature (independent from the others) is appropriated by independently implemented and pursued computers in each case in the new signature system. A message is considered only as correctly time-stamped if enough (independent) signatures are submitted in the new signature system. Because everyone can protect autonomously his own interests in case of resigning, the proceeding outlined here is much more satisfying. This applies particularly if the end of the cryptographic security of digital signatures (and therewith also the end of every transition period) can be determined in certain terms by the use of Fail stop signatures.

c) I transmit the test key of an asymmetric signature system. Because then my partner can prove later with its help whether my public key of an asymmetric encryption system transmitted to him is authentic, by signing this key therefore by myself.

d) Yes, because an "upgrade" of signature systems is securely realizable in an easier way than an upgrade of encryption systems. Furthermore the test key must be registered for legal liability.

Exercise at page 423.

3-16 crypto-policy by choice of a convenient length of key?

This proceeding is hopeless in principle:

Even if governments had clearly more money than big affiliated groups (what is far from clear), affiliated groups (as well as curious neighbors) can focus their cryptanalytic efforts very strongly on separate broadcasting stations and recipients. At the same time the fight against crime (and also espionage) requires of cryptanalysis a certain width, i.e., from a lot of broadcasting stations and recipients. Per key it means that compared to big affiliate groups, broken governments might have rather less than more computer power.

Beside this fundamental argument, from my point of view, there is naturally a row of other arguments, why such a "key length compromise" is unreasonable:

With the increase of available computer power in general, the key length would have to be adapted continually (cp. exercise 3-15a)), which is not possible to be managed meaningfully, however, in terms of secrecy for dated back messages (cp. exercise 3-15b)).

Criminals will not only trust in such weak encryption systems, but encode if necessary with stronger ones and additionally use steganographic systems if needed, cp. §4 and §9.

Exercise at page 423.

3-17 Authentication and encryption – in which order?

Let x be the message:

If asymmetric authentication is desire, so a digital signature, one should sign first and then encrypt, so that the recipient has a signed plaintext message to store after decrypting. The signature must be co-encrypted, because it contains information about the message in general. The sender creates $k(x, s(x))$ or $c(x, s(x))$, depending on a symmetric or an asymmetric encryption system.

If symmetric authentication suffices, the order according to usefulness does not matter: one can do it as above, thus creating $k(x, MAC)$ (In this case one will not have a asymmetric encryption system). Or one can authenticate the encrypted message by creating a MAC for $k(x)$. The latter has 3 advantages according to effort and power:

- It “uses”, through application of one-time pad for secrecy, less keys as well as less arithmetic expenditure, because secrecy is not supposed to increase the length of the message, while authentication does - and the expenditure of the second operation increases with the length of its input, i.e., the output of the first operation.
- If one checks authentication first and this test fails, one can save decrypting (and therewith expenditure) for him.
- Proving authentication and decrypting can take place parallel, so that the answer time can be shortened.

See p. #473
See p. #477
Exercise at page 424.

3-18 $s^2 - \text{mod} - n - generator$

a) For $n = 7 * 11 = 77, s = 2$ the following sequences result:

sequence s_i :	4	16	25	9	4	⋯
sequence b_i :	0	0	1	1	0	⋯

thus for the plaintext 0101 in case of addition modulo 2 the ciphertext 0110 results. The decrypter creates the sequence b_i in the same way and calculates using addition modulo 2 from the ciphertext 0110 the plaintext 0101.
And because it was so much fun, the second example:
For $n = 7 * 11 = 77, s = 13$ the following sequences results:

sequence s_i :	15	71	36	64	15	⋯
sequence b_i :	1	1	0	0	1	⋯

thus for the plaintext 0101 in case of addition modulo 2 results in the ciphertext 1001.

b) First we have to calculate back the sequence of the other s_i using $s_4 = 25$. We need the CRA, so we calculate the base vectors first:
Euclidean algorithm:

$$11 = 3 * 3 + 2; 3 = 1 * 2 + 1;$$

Backwards:

$$1 = 3 - 1 * 2 = 3 - 1 * (11 - 3 * 3) = -11 + 4 * 3$$

Thus $up = 12, vq = -11$ and we have the formula $s = 12 * s_q - 11 * s_p = 11(s_q - s_p) + s_q$. Furthermore we can pre-calculate $(p+1)/4 = 1$ and $(q+1)/4 = 3$.

s_3: (Remark: Here we can not just take 5 (the root that is known from calculating with integers), because we need the root of the 4 roots from 25 mod 33 that itself is a quadratic residue.)

Modulo 3: $s_4 \equiv 1 \Rightarrow s_3 \equiv 1^1 \equiv 1$
Modulo 11: $s_4 \equiv 3 \Rightarrow s_3 \equiv 3^3 \equiv 5$
$\Rightarrow s_3 = 11(5-1) + 5 = 49 \equiv 16 \bmod 33$.

s_2:
Modulo 3: $s_3 \equiv 1 \Rightarrow s_2 \equiv 1^1 \equiv 1$ (one can see that it will stay the same)
Modulo 11: $s_3 \equiv 5 \Rightarrow s_2 \equiv 5^3 \equiv 4$
$\Rightarrow s_2 = 11(4-1) + 4 \equiv 4 \bmod 33$.

s_1:
Modulo 11: $s_2 \equiv 4 \Rightarrow s_1 \equiv 4^3 \equiv 16 * 4 \equiv 9$
$\Rightarrow s_1 = 11(9-1) + 9 \equiv 31 \bmod 33$.

s_0:
Modulo 11: $s_1 \equiv 9 \equiv -2 \Rightarrow s_0 \equiv (-2)^3 \equiv -8 \equiv 3$
$\Rightarrow s_0 = 11(3-1) + 3 \equiv 3 \bmod 33$.
The last bits of those are $b_0 = 1, b_1 = 1, b_2 = 0, b_3 = 0$.
Thus the plaintext is $1001 \oplus 1100 = 0101$.

c) What happens here does not depend just on the $s^2{-}mod{-}n$-Generator but happens always when one adds bitwise a one-time pad (cp exercise 3-10d)) or a more or less secure pseudo-one-time pad (Stream-cipher in deeper meaning, cp. §3.8.1):

If a bit of the ciphertext flips over then just this bit of the plaintext is false after decryption using addition modulo 2. But if a bit gets lost then the whole plaintext from this position is wrong :

Let the $i{-}th$ bit of the plaintext be x_i. The $i{-}th$ ciphertext bit is then $S_i := x_i \oplus b{-}i$. If $S_i^* = S_i \oplus 1$ instead of S_i arrives at the recipient he will decrypt it as $S_i^* \oplus b_i = x_i \oplus 1$..

If S_i falls out, however, he will decrypt each bit from the i-th position with the wrong b_i: He reputes S_{i+1} for S_i and decrypts it with b_i and so on.

d) The $s^2 - mod - n$-Generator guarantees in its normal use as symmetric (and more than ever as asymmetric) encryption system as little authentication as Vernam cipher. (cp. part c)).

Nevertheless the answer for the question is: Yes. One may take an information-theoretical secure authentication code and use the sequence generated by the $s^2 -$

$mod - n$-Generator as key of the authentication system (the key that must be exchanged actually is only the seed for the $s^2 - mod - n - generator$).

The resulting authentication system could be broken if the attacker is able to break the authentication code or the $s^2-mod-n$-Generator. But the authentication code is information-theoretical secure, thus the $s^2 - mod - n$-Generator remains.

For specialists: One ought to prove that the attacker, in order to break the system, has to break the QRA (quadratic residuosity assumption) actually, i.e., that the resulting system is cryptographically secure. Sketch: If one can break the authentication code, if the keys of the $s^2 - mod - n$-Generator can be chosen, while it is unbreakable with random keys, this is a measurable difference between pseudo-random sequences and real random sequences. But it is proven that in the case of the $s^2-mod-n$-Generator the sequences can not be differed by any statistical test from real random sequences below the QRA.

e) The key exchange remains the same. Though both have to memorize not the seed s anymore but the current s_i after the first message. This requires that the recipient decrypts the ciphertexts just once and in the same order as they were encrypted. Otherwise the recipient would have to memorize more. Compare §3.8.2.

f) Following §3.4.1.7 and §3.4.1.8 as well as exercise 3-13k) each square has exactly 4 roots mod n. (with n = p*q, $q \equiv 3 \bmod 4, p \not\equiv q$), from which exactly one is a square again. The root of 1 which itself is a root again is 1. Thus it is valid: if $s_0 \neq 1$ all following inner states are $\neq 1$, since 1 results only from the square 1. The probability of reaching the inner state 1 after i steps is as high as the probability of choosing randomly one of the four roots of 1 as seed s, thus $4/|\mathbb{Z}_n^*|$.

Exercise at page 424.

3-19 GMR

a) $R = 28$ is not within the domain of the permutation pair, because $\gcd(R, n) = \gcd(28, 77) = 7$. Thus one can not sign anything using this. Therefore the example for $R = 17$ follows. (There is $17 \equiv 28 \bmod 11$, so somebody who calculates with 28 can find his calculations modulo 11 again.) Since $\gcd(17, 77) = 1$ and $\left(\frac{17}{77}\right) = \left(\frac{17}{11}\right) * \left(\frac{17}{7}\right) = \left(\frac{6}{11}\right) * \left(\frac{6}{3}\right) = (6^{(11-1)/2} \bmod 11) * (3^{7-1}/2 \bmod 7) = (-1) * (-1) = 1$ $R = 17$ lies in the domain of the permutation pair.

Because the messages are always 2 bit long, one does not need a prefix–free transformation. $S := f_m^{-1}(R)$ can be created directly. This is (cp. §3.5.3):

$$S = f_{01}^{-1}(17) = f_1^{-1}(f_0^{-1}(17)).$$

Step 1: First $x := f_0^{-1}(17)$ is needed. Following §3.5.2 one has to test at the beginning whether 17 or -17 is a square. (cp. §3.4.1.7)

Modulo p=11: $17 \equiv 6$; Euler criteria: $6^{(11-1)/2} = 6^5$;
$6^2 \equiv 3; 6^4 \equiv 3^2 \equiv 9; 6^5 \equiv 9 * 6 \equiv -1 \Rightarrow 17 \notin QR_p \Rightarrow 17 \notin QR_n$.

The root w of -17 is to be extracted. Modulo p=11: $60 \equiv 5; 5^{(11+1)/4} = 5^3 \equiv 4$;
Modulo q=7: $60 \equiv 4; 4^{(7+1)/4} = 4^2 \equiv 2$;
(One can shorten the following, but it is mentioned for completeness: Applying QRA and the "base vectors" from exercise 3-13g) results in $w = -21 * 4 + 22 * 2 = -40 \equiv 37$. This w is the root from which to square again. x is supposed to be the one with Jacobi symbol +1 (thus $\pm w$), which is $< 77/2$. This applies for $x = 37$.)

Step 2: Now $S = f_1^{-1}(x) = f_1^{-1}(37)$ is calculated, thus $S \in D_n with (2S)^2 \equiv \pm 37$.
(First we would have to test whether x or $-x$ is a square. But we know this from the first step: We calculated the root w which was a square in any case. One can proceed immediately with w. Furthermore we have to now calculate again with mod p and q solely for extracting a root of w. We did not need to compound w using QRA, i.e., we could leave out all parenthesized parts.)

Thus: Root w^* from the result of step 1:

Modulo 11: $4^{(11+1)/4} = 4^3 \equiv 9$;
Modulo 7: $2^{(7+1)/4} = 2^2 \equiv$;

Next one has to divide by 2. It can happen solely with p and q as well: (Extended Euclidean Algorithm for determination of the multiplicative inverse of 2 and subsequent multiplication with it, or halving in $\mathbb{Z}$ of w^*, $w^* + p$ or $w^* + q$.)

Modulo 11: $w^*/2 \equiv 10$;
Modulo 7: $w^*/2 \equiv 2$;

The demanded signature S is one of the 4 numbers $CRA(\pm 10, \pm 2)$. Before QRA we have to ensure Jacobi symbol +1. Instead of the Euler criteria we can use this one (§3.5.1)

$$\left(\frac{2}{p}\right) = -1 \text{ and} \left(\frac{2}{q}\right) \equiv +1.$$

According to that it follows, because the Jacobi Symbols of $w^* mod p$ and q were 1 each, that

$$\left(\frac{w^*/2}{p}\right) = -1 and \left(\frac{w^*/2}{q}\right) \equiv +1.$$

We create $CRA(-10, 2)$ (One could also take $CRA(10, -2)$).
$CRA(-10, 2) = CRA(1, 2) = -21 * 1 + 22 * 2 = 23$.
And again we have just caught that of the numbers $\pm S$ that is $< 77/2 \Rightarrow S = 23$.
Annotations:

1. It does not matter, whether one divides by 2 first and then applies CRA, or the other way round.

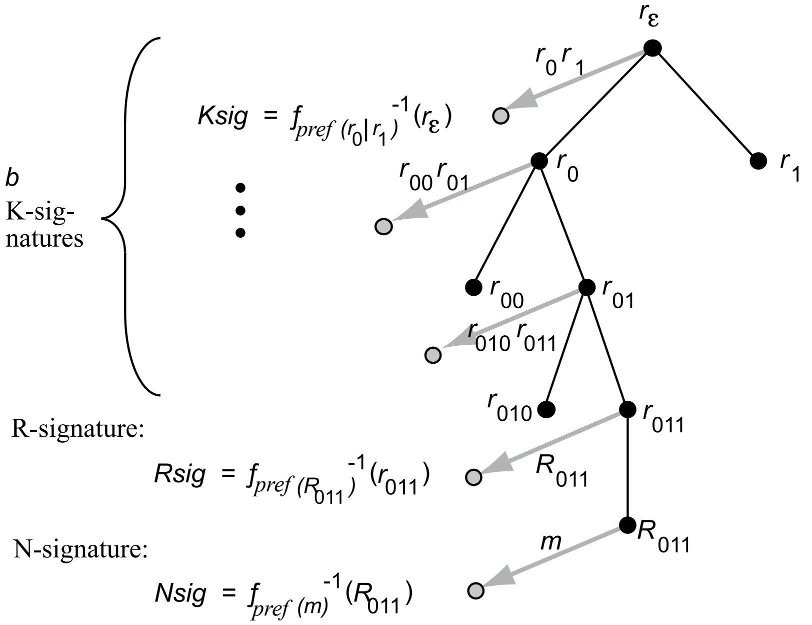
2. If you receive 12 as signature, then you have forgotten to pay attention to the Jacobi symbols. 12 has the Jacobi symbol

$$\begin{aligned}\left(\frac{12}{77}\right) &= \left(\frac{12}{11}\right) * \left(\frac{12}{7}\right)\\ &= 1^5 \bmod 11 * 12^3 \bmod 7 = 1^5 \bmod 11 * (-2)^3 \bmod 7 = -1.\end{aligned}$$

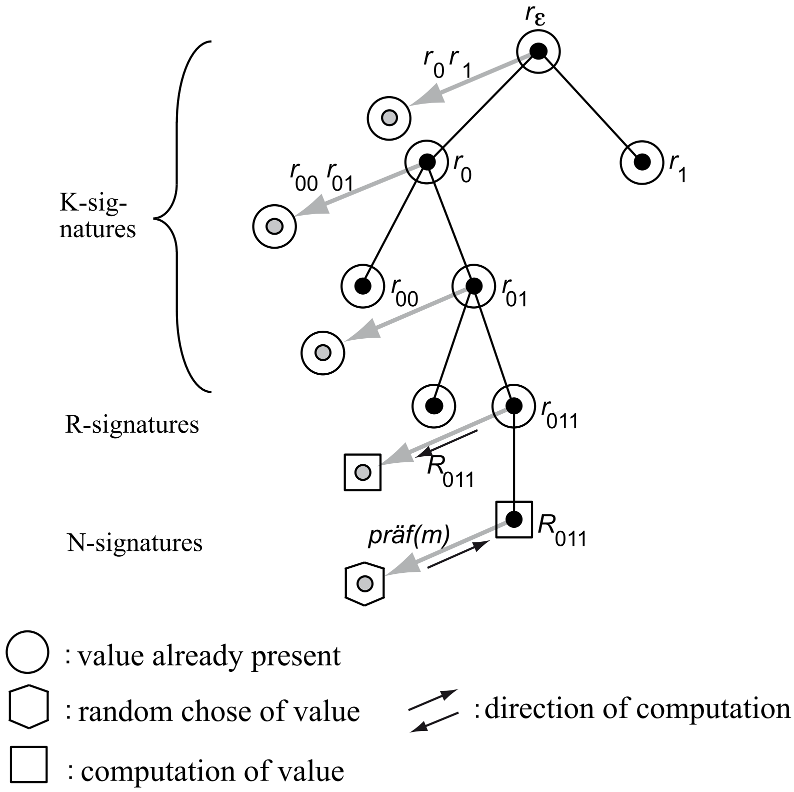
It means that 12 is not in the domain of the permutations. The tricky thing is that the rest of the signature appears to work though: $f_0(f_1(12)) = 17$, because $f_1(12) = 4 * 12^2 \bmod 77 = 4 * 144 \bmod 77 = 4 * -10 \bmod 77 = 37. f_0(37) = -(37^2) \bmod 77 = 17$. The test for whether the signature lies in the domain must be a part of the signature test.

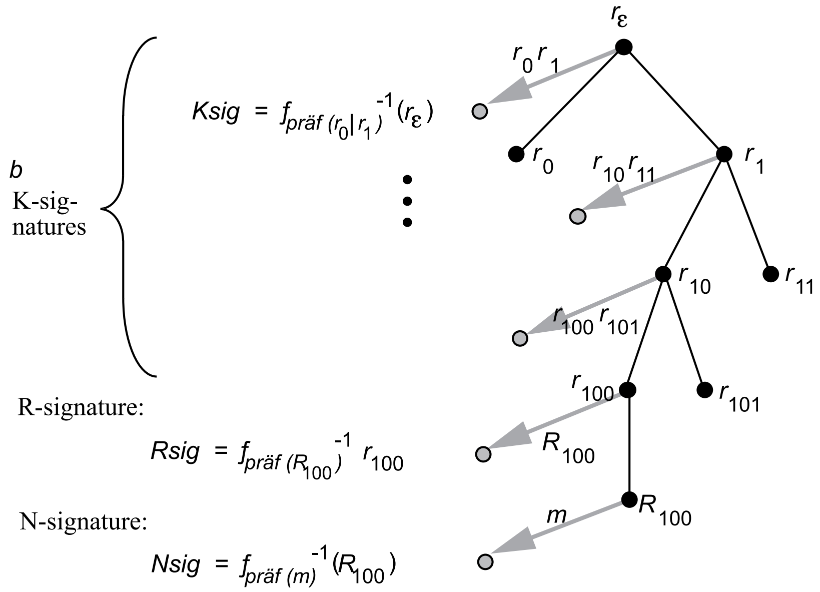
b) Because 8 messages are to be signed the reference tree has a depth about 3, as in the figures in §3.5.4.

Analogous to Figure 3.26, the fourth figure, i.e., the one attached to R_{011}, consists of the following parts: (“parts” are those parts here: The black references and the gray signatures.)



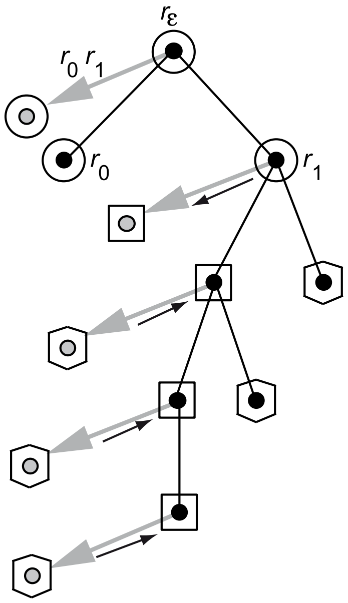
Through a comparison with Figure 3.26, one can see what must be newly chosen or calculated: the only new reference is R_{011}. New signatures are those of R_{011} in terms of r_{011} and (of course) of the message m in terms of R_{011}. If one considers, as in Figure 3.27, in which order one calculates at best, it results:





By comparing it with the fourth signature above one can see that only the toggling three references (r_ϵ, r_0, r_1) are already there as well as the toggling K-signature.

(This is just the change from the left to the right half of the tree, after the first signature it is the most time-consuming operation.) For the order for calculation it results:



Of course one can send everything that was signed, but this is only necessary for parallel testing. If testing is proceeded sequentially one does not need to send the inner nodes of the signatures: The recipient of the signature has to recalculate all grey arrows in the opposite direction and only compare whether the root of the tree r_ϵ that is known to him results.

c) It is to show: if there would be an algorithm that finds collisions of functions $f_{pref(m)}$ efficiently, then there is an algorithm that finds collisions for pairs (f_0, f_1).

Therefore it is shown how to directly calculate from each collision $f_{pref(m)}$ a collision of (f_0, f_1). The following collision is given m, m^*, S, S^* with $m \neq m^*$ and $f_{pref(m)}(S) = f_{pref(m^*)}(S^*)$. Set $v := pref(m)$ and $w := pref(m^*)$.

Let us call the bits of $v = v_0 v_1 \cdots v_k$ or $w = w_0 w_1 \cdots w_{k'}$. v_i and w_i for the first bits from the left where v and w differ. Since v and w are prefix free codings of different messages m, m^* applies $i \leq min(k, k')$ and of course $0 \leq i$. Then:

$$f_v(S) = f_{v_0} \cdot f_{v_1} \cdot \cdots \cdot f_{v_i} \cdot \cdots \cdot f_{v_k}(S) =$$
$$f_{w_0} \cdot f_{w_1} \cdot \cdots \cdot f_{w_i} \cdot \cdots \cdot f_{w_{k'}}(S^*) = f_w(S^*)$$

Since in both lines the same permutations are shown (for all j with $0 \leq j \leq i$ it applies $V_j = w_j$), from the equality of the figures follows the equality of the original figures. Thus it is

$$f_{v_i} \cdot \cdots \cdot f_{v_k}(S) = f_{w_i} \cdot \cdots \cdot f_{w_{k'}}(S^*).$$

Consequently

$$f_{v_i}(f_{v_{i+1}} \cdot \dots \cdot f_{v_k}(S)) = f_{w_i}(f_{w_{i+1}} \cdot \dots \cdot f_{w_{k'}}(S^*))$$

is a collision of f_0 and f_1.

Prefix free coding is needed to ensure that v and w differ on each bit position, which *exists* for both (otherwise there would be either no v_i or no w_i.)

Exercise at page 425.

3-20 RSA

a) Key generation:
Let $p = 3, q = 11$ therefore $n = 33$

Therefore:
$\Phi(n) = (p-1) \bullet (q-1)$

Let $c = 3$. Test if $\gcd(3, 20) = 1$. (Euclidean Algorithm) ⇒ Yes!

In order to find the exponent for decoding, one can either use the conventional algorithm or calculate it for *mod* p and q separately:

Conventional Algorithm: We are looking for
$d = c^{-1} \ mod \ \Phi(n)$
here
$d = 3^{-1} \ mod \ 20$

The extended Euclidean Algorithm gives us $d = 7$.

More efficient Algorithm:

$$\begin{array}{lllll} d_p & := & c^{-1} & mod \ (p-1) & \text{in our case} & d_p & := 3^{-1} \equiv 1 & mod \ 2 \\ d_q & := & c^{-1} & mod \ (q-1) & \text{—} & d_q & := 3^{-1} & mod \ 10 \end{array}$$

From the extended Euclidean Algorithm we get $d_q = 7$.

Encryption:
Take $m = 31$ for the plaintext. The corresponding ciphertext S is
$S = 31^3 \equiv (-2)^3 \equiv -8 \equiv 25 \ mod \ 33$

Decryption:
Conventional algorithm:
$S^d = 25^7 \equiv (-8)^7 \equiv 64^3 \bullet (-8) \equiv (-2)^3 \bullet (-8) \equiv 64 \equiv 31 \ mod \ 33$

33 is indeed the plaintext.

More efficient algorithm:
Mod 3: $S^{d_p} = 1^1 = 1$
Mod 11: $S^{d_q} \equiv 3^7 = 9^3 \bullet 3 \equiv (-2)^3 \bullet 3 \equiv 3 \bullet 3 = 9$

Using the CRA (as usual) we get $m = 31$.

b) To make things simple we can just modify the example used in a): Let key generation be identical (only c is now called the test key and d the signing key).

If the plaintext just so happens to be $m = 25$, then the signature will be the 3^{rd} root of m which gives us $Sig = 31$ (see above) and testing will result in $Sig^3 = 31^3 \equiv 25$.

Note:

1. If you want to use encryption for some messages and authentication for others, you should choose different key pairs. Otherwise, in order to get the plaintext for an encrypted RSA block, one could just get the owner of the key to sign that block. This risk can be circumvented, though, using a one-way hash function before signing a message or by designing the cryptographic protocol in a way that prevents signing of arbitrary messages. But why take this risk in the first place just to save that little bit of effort in key generation and storage in RSA?

2. Neither redundancy nor hash functions were used in our example; this system is still weak against Davida attacks and should not be used unmodified.

c) To 0 and 1 respectively – independently of the chosen key pairs. We see that these plaintexts should be avoided by using an appropriate encoding as input to RSA.

If blanks (the characters are by far most likely to result in long sequences) are not encoded as 0. . . 0, this will be the case automatically with reasonable probability. You can also take care of this by means of redundancy and hash functions which should be used anyway.

d) One approach would be to insert 100 random bits at fixed positions in each RSA block.

We need enough random bits so it becomes impossible for an attacker who sees the ciphertext S to just guess a likely plaintext m, and do a complete search to find out if S contains m. In order to do this, he would just encrypt m using the public key c and iterating through all combinations of random bits. With just 8 random bits this would only take 256 iterations.

Timestamps, even if encoded in more than 8 bits, are not random enough. An attacker might just try out all probable timestamp values.

e) If the randomly chosen bits are part of the output of the encryption system, the proposed in-deterministic flavor of RSA is not safe against active attacks: an attacker could just use the same approach as in deterministic RSA.

The answer becomes more complex if the random bits are not part of the output. But, as was pointed out in §5.4.6.8 and more elaborately in [PfPf_90], it will still be "no" — at least if they are the first or the last 100 bits.

Therefore, in a concrete implementation of RSA, for example, 100 random bits should be chosen in combination with a redundancy rating. The message format could then be $m^* = (m, z, h(m, z))$ (where ordering is irrelevant), m being the message itself, z being the random bits and h being a one-way hash function. The encrypted message will be m^{*c}.

If the range of h is 160 bits (see §3.8.3), we need to subtract 260 bits off the overall RSA block size in order to get the net RSA block size. 740 bits remain if the module is 1000 bits long which is 74 %.

f) Random bits can either be inserted into the to-be-signed message itself or into the block that is exponentiated for RSA (or both).

If they are inserted into the to-be-signed message, they will be hashed with it. It is crucial for the security of the signature that the hash function is collision resistant for every choice of random bits. This should be the case for a good hash function — but why take the additional risk and the extra effort of hashing more bits?

If random bits are inserted into the block signed by RSA, the risk of multiplicative attacks is increased. Why take that risk?

All in all, I would always sign deterministically using RSA.

g) According to §3.4.1.1 the complexity of signing an RSA block of length $2l$ with an exponent of length $|c|$ is proportional to $(2l)^2 \bullet |c|$. The expense per bit when signing extremely short messages does not differ, because only one block has to be encrypted.

When encrypting very long messages it is noted that the number of bits that can be put into a single RSA block is increased with l. Neglecting the random number and the hash value that take up space also, we can encrypt $2l$ bits in each RSA block. Therefore the complexity for very long messages is proportional to $\frac{(2l)^2 \bullet |c|}{2l} = 2l \bullet |c|$. Under the best of circumstances the complexity only grows proportionally to l. (In reality it is even less, at least for very large values of l, since the fast algorithms mentioned in §3.4.1.1 really pay off. On the other hand, for extremely long messages most of the time hybrid encryption (§3.1.4) is used instead of plain RSA.)

h) Let l be the security parameter, i. e., the length of p,q.

Mod n: What we need is $x^d \; mod \; n$ ($d = c^{-1} mod \; \Phi(n)$ has already been calculated while generating the key). The length of x and d each are about $2l$.

Exponentiation using "square-and-multiply" will therefore be broken down to $\frac{3}{2} \bullet 2l = 3l$ modular multiplications of numbers of length 2l (counting squaring as a multiplication).

Mod p,q: We need $x^{d_p} \; mod \; p$, $x^{d_q} \; mod \; q$ and the CRA for those values (d_p, d_q, and the "base vectors" for the CRA have already been calculated during key generation). So we are left with two more exponentiations, the arguments of which only have a length o l (x, because it can be reduced $mod \; p$ and q resp., and d_p and d_q because they were calculated $mod(p-1)$ and $mod(q-1)$, respectively) and the

CRA. The latter operation is made up of only two multiplications and is therefore negligible compared to the exponentiations.

That makes about $2 \bullet \frac{3}{2} \bullet l \;=\; 3l$ modular multiplications of numbers of length l.

So the ratio between the overall complexity of the approaches is just the same as between modular multiplications of numbers of size l and $2l$ respectively. (Standard) modular multiplication is made up of multiplication and division. These operations both have a complexity of $O(l^2)$. Therefore, the expense quadruples when doubling l.

i) A random exponent will have roughly size $2l-1$. Instead of $2l-2$ squaring operations and $\frac{(2l-2)}{2}$ multiplications when using a random exponent, only 16 squaring operations and one multiplication will have to be performed. Therefore we will only have to do 17 operations instead of $3l-3$. With this, the public operation will be sped up by $\frac{3l-3}{17}$. If $l = 512$, which is more realistic, the factor will be $\frac{3 \bullet 512-3}{17} \sim 90$.

j) Great idea: since $scd(p-1,q-1) \;<\; \Phi(n) \;=\; (p-1)(q-1)$, because $p-1$ and $q-1$ are at least both divisible by 2, we can substitute $\Phi(n)$ by $scd((p-1),(q-1))$ and get:

c will be chosen from a slightly smaller interval in step 3 this way, although the same numbers in that interval can be chosen. In step 4 the multiplicative inverse to c, d, will be smaller on average. We now get
$c \bullet d \;\equiv 1 \; mod \; scd(p-1,q-1)$.

This yields
$c \bullet d \equiv \; 1 \; mod \; p-1$
as well as
$c \bullet d \equiv \; 1 \; mod \; q-1$.
The validation of encryption and decryption in §3.6.5.1 still holds.

Does this modification interfere with the security of RSA? I believe that is very, very improbable, because only the choice of c, which has to be published anyway, changes slightly — and in practice c is very often chosen to be small anyway, to make encryption less costly (see §3.6.1 and also part i) of these exercises).

Those who wish to be absolutely puristic about it may want to use this hybrid approach:

Step 3 is done as in §3.6.1 using $\Phi(n)$, and in step 4 first $\Phi(n)$ is used and then $gcd(p-1,q-1)$ where the smaller result will be selected.

This way nothing changes as seen from the outside — the security of RSA will stay completely unchanged. We only save a bit of effort while encrypting since d is a little smaller. Not much though, because $p-1$ and $q-1$ will not have too many common prime factors if p and q are chosen randomly and stochastically independently.

k) We recycle the example from a), so $n_1 = 33$

We choose

$$\begin{aligned} n_2 &= 5 \bullet 17 = 85 \\ n_3 &= 23 \bullet 29 = 667 \end{aligned}$$

We get

$$\begin{aligned} \Phi(n_2) &= 4 \bullet 16 = 64 \\ \Phi(n_3) &= 22 \bullet 28 = 616 \\ gcd(3,64) &= 1 \\ gcd(3,616) &= 1 \end{aligned}$$

(In case you chose 7 or 13 as prime factors of n, you probably found $\Phi(n)$ to be divisible by 3)

For the plaintext $m = 31$ we get

$$\begin{aligned} S_1 &= 31^3 \equiv (-2)^3 \equiv -8 \equiv 25 && mod\ 33 \\ S_2 &= 31^3 \equiv 29791 \equiv 41 && mod\ 85 \\ S_3 &= 31^3 \equiv 29791 \equiv 443 && mod\ 667 \end{aligned}$$

The extended Euclidean Algorithm gives us
$-18 \bullet 33 + 7 \bullet 85 = 1$

From the CRA we get
$-18 \bullet 33 \bullet 41 + 7 \bullet 85 \bullet 25 = -9479\ mod\ 33 \bullet 85 = 1741$
(Check: $1741\ mod\ 33 = 25, 1741\ mod\ 85 = 41$)

Once more, this time with $33 \bullet 85 = 2805$ and 667:

The extended Euclidean Algorithm[4]yields:
$778 \bullet 667 - 185 \bullet 2805 = 1$

The CRA gives us:
$778 \bullet 667 \bullet 1741 - 185 \bullet 2805 \bullet 443 = -673566391\ mod\ (667 \bullet 2805) = 29791$
(Check: $29791\ mod\ 667 = 443, 29791\ mod\ 2805 = 1748$)

The third root of 29791 is 31 — which is exactly the plaintext m.

[4] Since computation gets difficult with these numbers, I include it here.

Euclidean Algorithm:

$$\begin{aligned} 2805 &= 4 \bullet 667 + 137 \\ 667 &= 4 \bullet 137 + 119 \\ 137 &= 1 \bullet 119 + 18 \\ 119 &= 6 \bullet 18 + 11 \\ 18 &= 1 \bullet 11 + 7 \\ 11 &= 1 \bullet 7 + 4 \\ 7 &= 1 \bullet 4 + 3 \\ 4 &= 1 \bullet 3 + 1 \end{aligned}$$

l) Breaking RSA completely means being able to calculate d for any given n and c. The proof mentioned in the exercise shows that knowing n, c, and d you can factor n — breaking RSA completely therefore is just as hard as factoring. Unfortunately this proof does not even prove the impossibility of breaking RSA universally without knowledge of n — i. e., to find a procedure that is equivalent to the key without being able to factorize n.

Exercise at page 425.

3-21 DES

a) **Encryption:**

$$\begin{aligned}
K_1 &= 011, & K_2 &= 010\\
L_0 &= 000, & R_0 &= 101\\
L_1 &= R_0 = 101, & R_1 &= L_0 \oplus f(R_0, K_1) = 000 \oplus (R_0 \bullet K_1 \ mod\ 8)\\
 & & &= 101 \bullet 011 \ mod\ 8 = 5 \bullet 3 \ mod\ 8 = 111\\
L_2 &= R_1 = 111, & R_2 &= L_1 \oplus f(R_1, K_2) = 101 \oplus (111 \bullet 010 \ mod\ 8)\\
 & & &= 101 \oplus (7 \bullet 2 \ mod\ 8) = 101 \oplus 110 = 011
\end{aligned}$$

Exchanging both halves yields the ciphertext 011111.

Decryption

We are using the variable names from the pictures in §3.7.3. Go there for explanation.

$$\begin{aligned}
R_2 &= 011, & L_2 &= 111 \text{ are given}\\
R_1 &= L_2 = 111, & L_1 &= R_2 \oplus f(R_1, K_2) = 011 \oplus (111 \bullet 010 \ mod\ 8)\\
 & & &= 011 \oplus (7 \bullet 2 \ mod\ 8) = 011 \oplus 110 = 101\\
R_0 &= L_1 = 101, & L_0 &= R_1 \oplus f(R_0, K_2) = 111 \oplus (R_0 \bullet K_1 \ mod\ 8)\\
 & & &= 111 \oplus (101 \bullet 011 \ mod\ 8) = 111 \oplus (5 \bullet 3 \ mod\ 8)\\
 & & &= 111 \oplus 111 = 000
\end{aligned}$$

Since our choice of variable names includes an implicit exchange of values from left to right, we do not need to exchange both halves explicitly. The plaintext therefore reads 000101, which is exactly the one that was encrypted.

Going back:

$$\begin{aligned}
1 =& 4 - 3\\
=& 4 - (7 - 4)\\
=& -7 + 2(11 - 7)\\
=& 2 \bullet 11 - 3(18 - 11)\\
=& -3 \bullet 18 + 5(119 - 6 \bullet 18)\\
=& 5 \bullet 119 - 33(137 - 119)\\
=& -33 \bullet 137 + 38(667 - 4 \bullet 137)\\
=& 38 \bullet 667 - 185(2805 - 4 \bullet 667)\\
=& 778 \bullet 667 - 185 \bullet 2805
\end{aligned}$$

b) Tables are created explicitly so every result can be looked up immediately. Where possible, multiple s-boxes are integrated into one (in DES at least two: array of 2^{12} entries).

c) The trick in fast software implementations of these systems is to store and look up permutations that would take up many bit operations otherwise in tables. A table with 2^{32} entries would be too large though. As an example we will demonstrate, how to get along with only 4 tables of 2^8 entries each:

Entries will be 32 bits long and the i^{th} table tells us where to permute the i^{th} set of 8 bits.

Example: 2^{nd} table – e. g., bits 9-16. Let $W(9), \ldots, W(16) = 28, 13, 1, 5, 30, 14, 2, 19.$. In every entry bit no. 9 will be at position 28, bit no. ten will be at position 13,. . . , bit no. 16 will be at position 19 and all other bits will be 0.

The final result of the permutation will be achieved by indexing the i^{th} table with the i^{th} set of 8 bits from the input and get the binary OR or XOR of all 4 results.

d) For one pair of plaintext blocks and corresponding ciphertext blocks we try out different keys until we find one that matches. This will, on average, take 2^{55} attempts. After that, we check whether the key we found also is correct for other pairs of plaintext and ciphertext blocks, which is very likely. Thus, the average expense will be 2^{55} DES-encryptions.

e) As was laid out in §3.7.6 we choose the complement of one of the given plaintext block / ciphertext block – pairs, i. e., we take the compliment of the plaintext block and ask for the corresponding ciphertext. With this approach it takes 2^{54} attempts on average to obtain the key, which is half as many.

More formally:

The plaintext block/ciphertext block – pair (x, DES(k,x)) is given.

We get $\overline{x}$ and ask for DES(k,$\overline{x}$).

Because of the complementarity property we have DES(k,$\overline{x}$) = $\overline{\text{DES}(\overline{k}, x)}$. We calculate DES($\overline{k}, x$) and have the ciphertexts for x encrypted with k as well as with $\overline{k}$ at our hands for the attack.

In order to find the key k we try out different keys until encryption yields either DES(k,$\overline{x}$) or DES(k, x). k will be k' or $\overline{k'}$ respectively.

If we make sure that each of the possible 2^{56} values of k' and $\overline{k'}$ is tried out at most once, the search terminates after at most 2^{55}, on average after 2^{54} steps, where every step is made up of one DES-encryption and two comparison operations between the plaintext and the ciphertexts. Since the comparison operations are much less costly than the encryption, the expense for this chosen plaintext/ciphertext attack is only half as big as for the plaintext/ciphertext attack described earlier.

Same way as above we have to test whether k is the right key for other plaintexts blocks as well.

f) There is a linear equation for every one of the 64 ciphertext bits that is applied to plaintext bits according to the permutations and to key bits according to permutations and partial key generation — with known coefficients because the permutations and the partial key generation are public and fixed. Due to the fact that plaintext and ciphertext are known, we get 64 linear equations with 56 coefficients. Every software package for equation solving can figure those out.

   It obviously does not matter if the block length is 64 or 128 bits or whether the key length is 56 or 256 bits. The algorithm evidently breaks every Feistel cipher with known, fixed permutations. It probably can be modified to work on arbitrary linear block ciphers.

Exercise at page 426.

3-22 Modes

a) There is no redundancy. More precisely: A given ciphertext can be decrypted into any arbitrary plaintext. When re-encrypted, the resulting ciphertext will of course be identical to the input — in particular its last block. If the messages do not contain any internal redundancy, e. g., in files that only contain numbers, the attacker might just send arbitrary ciphertexts.

b) (This scheme is a bit better: In general the last plaintext block to some random ciphertext will not be all zeros, so the attack will be spotted. But:)

   If the ciphertext is at least three blocks long, the attacker can modify all but the last two. Since the mode CBC synchronizes every two blocks, the last block will still be decrypted correctly to a block of zeros and the attack will not be detected automatically.

c) Now, if some ciphertext block is altered, encryption will result in something completely different. Thus there will be a wrong block in the plaintext block buffer when encrypting the next block. In general, the resulting error will not be eliminated by synchronization because a function of the last block will be added to the next block after decryption, so decryption will fail again and result in a wrong plaintext buffer all over again.

   {Arguing PCBC that is a synchronous cipher which would render the attack ineffective, would be wrong though. For this the definition of "synchronous" in §3.8.1 is too weak, because even dependency on the bit position in the message satisfies it. Regard the following example for clarification: OFB is a synchronous stream cipher, but the ciphertext can be modified at will, as long as no ciphertext bits of the authenticator are changed and no ciphertext is omitted.}

d) **ECB:** The attacker can not achieve much, because the block cipher is assumed to be secure and it is used for nothing more than a block cipher. So there are only the disadvantages left that were mentioned along the definition of block ciphers and their respective attacks: Deterministic block ciphers can be attacked actively by asking the victim for the encryption of likely plaintexts. The resulting ciphertexts can be compared then to intercepted ciphertexts. With a bit of luck some of the intercepted ciphertexts can be “decrypted” and even without luck certain plaintexts can be ruled out.

CBC for encryption: The attacker can attack the system actively by inserting the values in the buffers. If these values are simply recycled, i. e., unless every message begins with a value that is chosen by the sender, after an active attack on the first block the same attack as on ECB can be used with the same probability of success. If the values in the buffers are reset after the active attacks, the attacker can still scan the ciphertext for blocks he knows (i. e., from an earlier attack on ECB or CBC). For these blocks he can then proceed to calculate the respective plaintext in CBC: he just needs to subtract the last key. But this kind of search is harder, because the attacker can not predict the encrypted values as well (and have them encrypted during his attack). His chances to succeed are slimmer than in ECB if the buffers are reset appropriately.

CBC for authentication: If single blocks are authenticated, everything that was said for encryption holds. If many blocks are authenticated at a time even if the buffers are not reset he has to guess their precise values, which is unfeasible if reasonable block lengths are chosen. This is said assuming that the attacker can not just “ask for” the authenticated values already during his attack, because no cryptographic system is secure against that. It is the embedding protocol’s responsibility to take care of this. The simplest approach is to have the partner create part of the message that is to be authenticated.

CFB for encryption: The attacker succeeds for the next output block precisely if he knows the encryption of the value in the shift register. If the shift register is not reset for every message, he can in an active attack ask for the encryption for a chosen plaintext (works for ECB, CBC and CFB), leave that plaintext in the shift register, have a cup of coffee, and wait for the next output unit to subtract the known value and celebrate the successful attack...

CFB for authentication: ditto.

OFB: The attacker either needs to predict the value of the shift register or insert it himself. The latter can not be done by manipulating the plaintext stream, because it is not fed back. Success of an active attack depends solely on the initialization of the shift register.

PCBC: For the advantages mentioned for CBC to be achieved, i. e., same probability of success as for ECB, the attacker can try to set the values in the two buffers actively to insert a value of h that suits his purposes. Apart from initializations he faces more obstacles than with CBC, which is illustrated by the following attempt to set the buffer values in PCBC: Send plaintext P_1. Now we know the contents of the plaintext block buffer (P_1) and we can find out the contents of the ciphertext block buffer by examining the output (C_1). Thus $h(P_1, C_1)$ can be calculated.

Still it is hard to set both values. A plaintext block P_2 could be chosen to set $P_2 + h(P_1, C_1)$ to some arbitrary value and this way produce the ciphertext C_2. But now P_2 has to be modified and the plaintext block buffer changes. The same is true for choosing the plaintext block P_2, because adding $h(P_1, C_1)$ produces unwanted results in C_2.

For a known function h it proves difficult to produce a certain output value from a small set of input values. Thus even a known and irregular distribution of plaintexts probably can not be used in favor of the attacker, provided initialization of the buffers is done in a wise manner. See "CBC for authentication'.

OCFB: As opposed to PCBC, in OCFB the shift register value can be set by the attacker even without initialization as long as:

- h is not a one-way function
- he has the chance to see the result *Res* of the encryption before the corresponding unit in the plaintext stream has to be handed over.

With *Res* the attacker has the value of the first input parameter of h and, unless h is a one-way function, is able to calculate an appropriate second value for the second input parameter for many output values of h. From this value he subtracts the desired plaintext and receives the plaintext unit that he has to insert to get his choice of result of h. If this scheme is repeated $\frac{b}{r}$ times, there will be a chosen value in the shift register. Now what was said in "CFB for encryption" applies.

e) Coding will be done like this: append a bit of given value (say 1) to the plaintext. If the text length is not a multiple of the block length, pad with bits of another value (say 0).

1. is satisfied, because we pad up to the next multiple of the block length.

2. is satisfied, because all bits from the last 1 to the end of the text can be cut off to decode the text.

3. needs a definition for "minimal". Sensible definitions might be:

- Expansion over all plaintexts is minimal.
- Expansion over all plaintexts weighted their respective probability is minimal. This will be the same as above in case of standard distribution.
- Worst case expansion (expansion of the worst-case plaintext) is minimal.

The given encoding satisfies the last definition, because every encoding has to add at least one bit to at least one text (in order to distinguish between "padded" and "not padded", and because this encoding adds exactly one bit if possible. The encoding probably also satisfies the first definition (why not try to prove that?). You can find a distribution of plaintexts for every given encoding that makes length expansion not minimal according to the second definition.

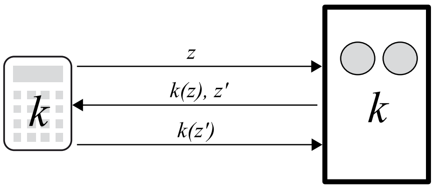
f) Shift registers and feedback are substitute by a counter that starts at some initialization value and is increased for every output unit. In order to access the i^{th} unit,

> $i-1$ is added to the initialization value — off we go. This mode is called **Counter Mode** for obvious reasons. [Schn_96, page 205].

See p. #473
Exercise at page 426.

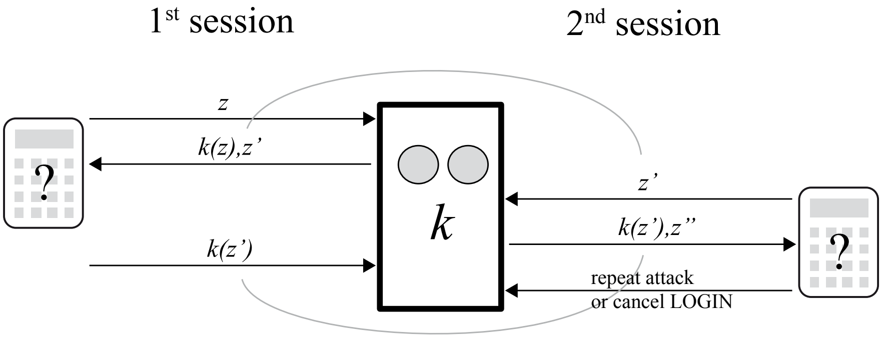
3-23 Mirror attack

Let the main frame implement the same secure block cipher (amongst other helpful features). Also assume that independently chosen keys have been exchanged between the main frame and each user (more accurately: between the main frame and the "calculator"). Login will now be done according to the following protocol: send random number, expect encryption, decrypt, compare. This will be done in both directions, i. e., client authenticates to main frame and main frame authenticates to client (for simplicity the user id, which has to be submitted, too, so the main frame knows which key to use, is not included in the following picture).



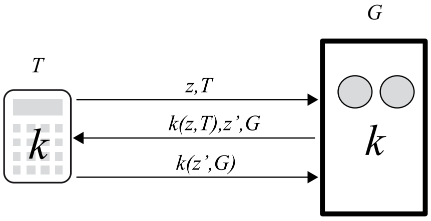
If main frame and "calculator" behave completely symmetrically (as suggested by the picture and by our discussion so far) the protocol is insecure regardless of the strength of the block cipher. If the main frame supports more than one session at a time, the second random number it generated in the first session may be used as the first random number in the second session. If the main frame does not detect this (for example by enforcing one login per user at any one time) and responds correctly in session two, his correct response may be fed back to it in session one. The main frame does not authenticate the "calculator", but its own "reflection". We are out of luck!

Minor mistake — major security hole (same goes for the "calculator", if it accepts the role of the second partner during authorization, so the roles are symmetrical, too. This could be the case, if clients can log into many main frames at a time).

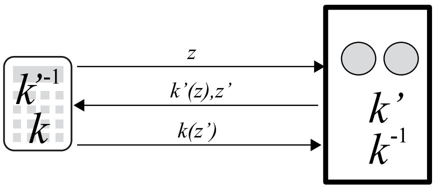


Correct processes can be designed by following one out of two schemes: either instances have some memory and accept only random numbers that satisfy certain conditions (which is kind of contrary to the idea of a random number), or they behave asymmetrically. This could be done by either

- attaching their identity (e. g., ‘M’ for ‘main frame’ and ‘C’ for ‘calculator’) to the random number.



- having only one of the parties decrypt and one encrypt.
- having two keys instead of one for one pair of main frame and client; one for the main frame to encrypt and the client to decrypt and vice versa.



It would also be secure – though not very elegant – to only react to a random number after having received replies to all random numbers that were sent, as in 3-6 part e. This way the main frame could not handle logins in parallel.

With precise enough clocks in place in both main frame and “calculator”, copying from the terminal to the calculator can be omitted once by using time instead of the random number (note the remark in 2-3 to make weakest possible assumptions on random number generation).

This will still not put up with attackers hijacking sessions after login. To account for this scenario identity must be authenticated continuously.

(It is noted that we need a block cipher with strong security properties in this exercise, because it is used more in the way of an authentication system than a encryption system and encryption systems do not generally make good authentication systems. See exercise 3-10 part d. If time stamps are used, for example, an attacker should not be able to guess from an encrypted message that he has seen the encryption of similar, forged messages.)

Exercise at page 427.

3-24 End-to-end encryption

No matter if symmetric or asymmetric cryptography is used, every user needs at least one secret key. Thus, wherever a computer is used by multiple users, the main problem is where to keep secret keys out of reach of other legitimate users of the system.

If the system is trusted enough, so that we can safely assume that it is able to protect one user's data from other users and if its hardware security shields off physical attacks, then the problem is solved inherently.

If the system is completely untrusted, then not only storing the key, but also encryption, decryption, testing signatures and most of all signing needs to take place on a different, personal computer. It should have at least some kind of a small key pad and display to communicate with its owner directly. Incarnations of this concept include so-called 'super smart cards' 'smart floppy disks' (computers resembling floppy disks that connect to others via the floppy disk drive, which do not need an extra reader and provide more space than a wallet-sized chip card with it is standardized magnetic-strip-dimensions) up to notebooks.

As a compromise between effort and security every user can enter a password of length (actually entropy) great enough to create a key for a symmetrical encryption system. Using this key the actual keys for end-to-end encryption are decrypted at the beginning of the session and encrypted at its end. This procedure may of course be broken either if the passwords do not contain enough entropy after alll or the symmetric encryption system can be broken, or if other users can get their hands on the password by simulating a password prompt, so the user thinks he is communicating with the operating system itself. The latter could be prevented by always checking the computer for hardware manipulations and booting the authentic operating system before logging in. The two things that are needed for this approach are intrusion detecting casings, see §2.1, and authentic booting see [Cles_88, ClPf_91, Groß_91].

Exercise at page 427.

3-25 End-to-end encryption without exchanging keys before

a) To send out a message to others, a set of one-time pads is generated. Those will be sent out separately on separate routes (e. g., via a friend and, where possible, from separate places like home, office, public phones, cellular phones) and media (e. g., phone, fax, e-mail) and possibly at different times (provided the service supports this, i. e., it does not have real-time requirements). The sum of all those pads is used to en-/decrypt and/or authenticate the message. An attacker would have to be at many different places at many different times in order to eavesdrop on all the pads and read or manipulate the encrypted/authenticated message. This method trades of security in availability for security in confidentiality and integrity: only one pad (or, of course, the message itself) needs to be intercepted in order to prevent the message from being read.

   The one-time pad and the information-theoretically secure authentication codes may serve as cryptographic systems, the former being as simple as can be and the latter certainly feasible with reasonable effort without a computer.

b) Once they get back their crypto devices they can use asymmetric crypto systems. For this the crypto devices that have deleted all their secret keys need to create new key pairs. In addition to a) Bob tells Alice his public key (or vice versa), for example via telephone — Alice's crypto device of course has also deleted Bob's public keys because it could not tell whether only its secret keys were going to be extracted, or whether the public keys were going to be manipulated, too. (These days telephone conversations are authenticated by the speaker's voice. As soon as speech synthesizers work sufficiently well, that will be history. We should not rely on this authentication alone.). After that the public keys are used to exchange one or more pads or authenticate the message directly.

If Bob and Alice took precautions in case they lost their keys by agreeing on a common secret they memorized (and thus always carry around with them) they can, of course, use that as a starting value for symmetric key generation. This applies most to b). Using a mnemonic pass phrase[5] both will be able to remember and a cryptographic hash function[6] a shorter value can be generated, that contains more entropy per character. This value can now be used as a starting value for key generation or even directly as a symmetric key.
Exercise at page 428.

3-26 Identification and authentication by an asymmetric encryption system

a) U creates a random number and encrypts it, so only T can read it: $c_T(z)$. U sends $c_T(z)$ to the instance that is supposed to be T and expects T, depending on the protocol, to either send back z, or to use z in some other cryptographic operation and send back the result. If that happens, that instance is in fact T (with a certain probability that is determined by the length of z and given that c_T is authentic, that the asymmetric encryption system is secure and z really was random). Now U has authenticated T. But beware: this naïve approach is prone to the changing attack laid out in 1. If some attacker A wants to be authenticated as some user T he just sends on $c_T(z)$ to T and relays the reply to U. We see that this protocol does not authenticate T as a direct communication partner, but only makes sure T is involved in communication somewhere and, more specifically, T receives z. So, if U inserts its message to T into $z - z = c_T(\text{message})$, this protocol nevertheless does its job. When receiving z, U knows that the message has reached T.

(Warning: If you came up with an essentially different solution, check whether you made T use his private key d_T on something that had not been encrypted using c_T. This kind of operation — using d_T first, for signing, then c_T, for testing the signature — is only possible in certain encryption systems, like RSA, but not in every encryption system. Where this is possible, the encryption system can be used as an authentication system as well, and the solution of the exercise is trivial.)

[5] a fairly long string with little entropy per character
[6] that may be publicly known

b) The asymmetric encryption system needs to be secure against adaptive chosen-cipher-text attacks. Otherwise an entity's responses during authorization may be used to break their encryption system.

c) Construction of a symmetric authentication protocol from an asymmetric concealment protocol:

1) T sends message M_i to U.
2) U generates a random number z, encrypts N_i and z, and sends $c_T(N_i, z)$ to T.
3) T decrypts using d_T and checks whether M_i from step 1 is identical to the one from step 2. If so, T sends z to U - otherwise T never returns z.
4) If U receives the same value for z that he created, M_i is an authentic message from T.

This authentication system is only symmetric, because z is useless for everyone else and therefore does not prove anything, since it was not created by T, but by U. Therefore U could send z to himself arbitrarily.

The disadvantage of this protocal compared to other authentication protocols is that it needs three messages instead of one. It therefore consumes more time and network resources.

Tabular representation of a symmetric authentication system on top of an asymm. encryption system.
Given: U knows c_t, T's public key.

$$\begin{array}{ccc} T & \rightarrow^{M_i}\rightarrow & U \\ T & \leftarrow^{c_T(M_i,z)}\leftarrow & U \\ T & \rightarrow^{z}\rightarrow & U \end{array}$$

Result: If — in this asymmetric authentication system — every effectively possible modification of $c_T(M_i, z)$ changes z, which in general should be the case in a block cipher, then U knows after the third message that M_i was really sent by T.

Correspondingly, the tabular representation of a symmetric authentication system through a symmetric authentication system could look like this:
Given: U and T both know k, k being a key for a symmetric authentication system.

$$T \quad \rightarrow^{M_i, k(N_i)}\rightarrow \quad U$$

In the first approach T produces N_i and U only supplies z. Also, T does not react to U's reply to his message if N_i is not part of it. Thus, the asymmetric encryption system does not have to withstand an adaptive active attack in its nastiest form.

Exercise at page 428.

3-27 Secrecy by symmetric authentication systems

Instead of every single bit of the message, only an appropriate MAC is sent. The recipient then figures out which bit corresponds to the MAC.

Sender	**recipient**
Knows k_{AB}	Knows k_{AB}
Let the message M be $b_1, \dots, b_n,\ b_i \in \{0,1\}$	
Calculates $MAC_1 := \mathit{code}(k_{AB}, b_1)$, $\dots,$ $MAC_n := \mathit{code}(k_{AB}, b_n)$	
Sends $MAC_1, \dots, MAC_n$	Checks whether $MAC_1 = \mathit{code}(k_{AB}, 1)$ or $MAC_1 = \mathit{code}(k_{AB}, 0)$ Thus receives the matching value for b_1 $\dots$ Checks whether $MAC_n = \mathit{code}(k_{AB}, 1)$ or $MAC_n = \mathit{code}(k_{AB}, 0)$ Thus receives the matching value for b_n

The MACs need to be long enough, so the recipient can check which bit matches the MAC. The small authentication code in Figure 3.15, for example, is insufficient: with a key value of 00, H and T are authenticated with 0. Therefore the recipient is unable to decide whether H or T are supposed to be transmitted. The same holds for a key value of 11 and a MAC of 1. Designing the MAC long enough makes the probability for this to be insignificant.

The message is concealed perfectly using the above protocol, since no one but the sender and recipient can test which bits match the MACs. If someone could guess those bits with a probability greater than 0.5, this would be an approach for breaking the authentication system.

With long enough MACs the message is actually authenticated: after changing a MAC it will, with overwhelming probability, either not match 0 *or* 1, or it will match the same value as before. Otherwise the authentication system could be broken with a probability too great to neglect: the attacker could change the MAC and flip the corresponding message bit that is sent in the regular authentication system.

Instead of authenticating a single bit it can be more efficient to authenticate more than one bit using a single MAC. This saves bandwith (more bits per MAC), and also makes it necessary for the recipient with a given number l of authenticated bits per

MAC to do a maximum of 2^l and an average of 2^{l-1} steps to decrypt. If l bits are sent separately, then $2 \times l$ steps are necessary in a complete search, and only l steps if 0 is assumed whenever the test fails for 1. Comparing the difference in expense between l bits being sent together and separately we find that for $l = 2$ sending l bits together has only advantages, because $2^l = l \times 2$, but the expense for greater l grows very fast. $l = 8$ might be a sensible compromise for many applications.
See p. #369
Exercise at page 428.

3-28 Diffie–Hellman key exchange

a) *Basic idea:*

We are using the fact that exponentiation is commutative, i. e., it does not matter in what order exponentiation is done.

This, put into a formula:

$$(g^x)^y = g^{xy} = g^{yx} = (g^y)^x$$

And, graphically:

$$\begin{tikzcd}
& g^{xy} & \\
g^x \arrow[ur, dash, "\uparrow y"] & & g^y \arrow[ul, dash, "\uparrow x"'] \\
& g \arrow[ul, dash, "\uparrow x"] \arrow[ur, dash, "\uparrow y"'] &
\end{tikzcd}$$

b) *Anonymity:*

Solution on the organizational level: there is no reason why public keys should have to be linked to real identities — they could belong to pseudonyms just as well.

Solution on the protocol level: instead of using his commonly known public key the sender chooses a new key pair for every encrypted message that is sent out. He uses the new private key to calculate a common secret key for communication with the recipient and encrypts accordingly. After that, he sends out the message along with the new public key[7] to his partner. At reception the public key can be used together with the recipient's private key to get the common secret key and decrypt the message. (If you have ever heard of asymmetric crypto using the

[7] that is why this modification of Diffie-Hellman key exchange can not be used for stenography with public keys, see §4.1.2

ElGamal-style encryption system: that is a special case of what we just discussed. Elegiacal uses modular multiplication as the symmetric encryption routine in the hybrid encryption system laid out here. [ElGa_85][Schn_96, p 478]):

Now, what if the sender wants an anonymous reply? Simple: the common secret key can just be re-used.

So asymmetric encryption systems and Diffie-Hellmann key exchange are on par with respect to anonymity, too.

c) *Individual choice of parameters*

Every participant chooses their own p and g to publish as part of their public key.

Every one can exchange secret keys with everybody else by using their p and g and then proceed as was laid out in §3.9.1. After that the new, pair-specific public keys can be told to the other participants.

Because of that last step, this modification of the Diffie-Hellman protocol is not adequate for steganography with public keys, see §4.1.2. This modification is used in PGP 5 and over, unless “Faster Key Generation” is selected. This is de facto the protocol variant from exercise part b) except that every instance also chooses and publishes their own p and g.

Even though this way public keys are chosen on a per participant pair basis, it is important to make sure public keys are authentic. This can be ensured by using certificates (see §3.1.1.2) and testing of certificates by the recipient. Wherever the sender can create signatures which the recipient can validate, it is best to have the sender sign his pair-specific public keys himself. (see also §9.2).

d) *DSA (DSS) as a basis*

On this basis Diffie-Hellman key exchange works flawlessly, because all parameters provide sufficient security. Depending on whether p and g are not the same for participants, things will have to be done according to b). DSA (DSS) can — contrary to its original purpose — be used as a basis for encryption and — if p and g are the same for all participants — even for steganography with public keys §4.1.2.

It is much harder to argue whether Diffie-Hellman key exchange when carried out this way is as secure as the method in §3.9.1 and b), because there is a subtle difference in distribution of g and a great difference in the length of x. To my knowledge though, this procedure can be regarded as secure.

Exercise at page 429.

3-29 Blind Signatures with RSA

Let 2 be the text (plus the result from the collision-resistant hash function, see §3.6.4.2) that is to be signed blindly, and 17 the masking factor. Thus we get for the masked text

$$2 \times 17^t \; mod \; 33 = 2 \times 17^3 = 2 \times 29 \; mod \; 33 = 58 \; mod \; 33 \; = 25$$

(When making up such an example, of course, you go the other way around: 25 is given for the to-be-signed message from exercise 3-20b). Therefore we are looking for two numbers, whose product (mod 33) is 25. After choosing 2 and 29 the third root of 29 (mod 33) has to be calculated. We are using the inverse exponent:

$$\begin{aligned} 29^7 &\equiv 29^2 \times 29^2 \times 29^2 \times 29 && \mathit{mod}\ 33 \\ &\equiv (-4)^2 \times (-4)^2 \times (-4)^2 \times (-4) && \mathit{mod}\ 33 \\ &\equiv 16 \times 16 \times 16 \times (-4) && \mathit{mod}\ 33 \\ &\equiv 256 \times (-64) && \mathit{mod}\ 33 \\ &\equiv 25 \times 2 && \mathit{mod}\ 33 \\ &\equiv 17 && \mathit{mod}\ 33 \end{aligned}$$

From exercise 3-20b) we now use the masked text: 25, 31
(since $31^3 \equiv (-2)^2 \equiv (-8) \equiv 25\ \mathit{mod}\ 33$)
The recipient now divides the second component, the blind signature, by the masking factor 17.
Euclidean Algorithm: calculate $17^{-1}\ \mathit{mod}\ 33$:

$$\begin{aligned} 33 &= 1 \times 17\ +\ 16 \\ 17 &= 1 \times 16\ +\ 1 \end{aligned}$$

And back again:

$$\begin{aligned} 1 &= 17 - 1 \times 16 \\ &= 17 - 1 \times (33 - 1 \times 17) \\ &= 2 \times 17 - 33 \end{aligned}$$

Therefore $17^{-1}\ \mathit{mod}\ 33 = 2$.
So the signature is:

$$31 \times 17^{-1}\ \mathit{mod}\ 33 = 31 \times 2\ \mathit{mod}\ 33\ = 29$$

Double-checking our result for correctness — how the recipient of the blind signature can test whether the signer is trying to shortchange him:
$\mathit{Signature}^t \equiv (-4)^3 \equiv -64 \equiv 2\ \mathit{mod}\ 33 = 2$
With relief we note that $\mathit{Signature}^t\ \mathit{mod}\ 33 = 2 = \mathit{Text}$ and thus our calculation probably is correct, too.

Exercise at page 430.

3-30 Threshold scheme

The smallest adequate prime number p is 17, because the numbers between 13 and 17 are not prime.

The polynome $q(x)$ is $q(x) = 2x^2 + 10x + 13\ \mathit{mod}\ 17$.
Substitution yields $(i, q(i))$:

$$\begin{aligned} q(1) &\equiv\ \ 2 + 10 + 13 \equiv\ \ 25\ \ \mathrm{mod}\ \ 17\ =\ \ 8 \\ q(2) &\equiv\ \ 8 + 20 + 13 \equiv\ \ 41\ \ \mathrm{mod}\ \ 17\ =\ \ 7 \\ q(3) &\equiv 18 + 30 + 13 \equiv\ \ 61\ \ \mathrm{mod}\ \ 17\ = 10 \\ q(4) &\equiv 32 + 40 + 13 \equiv\ \ 85\ \ \mathrm{mod}\ \ 17\ =\ \ 0 \\ q(5) &\equiv 50 + 50 + 13 \equiv 113\ \ \mathrm{mod}\ \ 17\ = 11 \end{aligned}$$

The formula from §3.9.6 gives us:

$$\begin{aligned}
q(x) &= 8 \times \tfrac{x-3}{1-3} \times \tfrac{x-5}{1-5} + 10 \times \tfrac{x-1}{3-1} \times \tfrac{x-5}{3-5} + 11 \times \tfrac{x-1}{5-1} \times \tfrac{x-3}{5-3} \quad mod\ 17 \\
&= \tfrac{8}{8} \times (x-3) \times (x-5) + \tfrac{10}{-4} \times (x-1) \times (x-5) + \tfrac{11}{8} \times (x-1) \times (x-3) \quad mod\ 17 \\
&= (x-3) \times (x-5) + \tfrac{10}{-4} \times (x-1) \times (x-5) + \tfrac{11}{8} \times (x-1) \times (x-3) \quad mod\ 17 \\
&= (x-3) \times (x-5) + 10 \times 4 \times (x-1) \times (x-5) + 11 \times 15 \times (x-1) \times (x-3) \quad mod\ 17 \\
&= (x-3) \times (x-5) + 6 \times (x-1) \times (x-5) + 12 \times (x-1) \times (x-3) \quad mod\ 17
\end{aligned}$$

$$= 2x^2 + 10x + 13$$

$q(0)$ of course values to the constant term, which in this case is 13.

Alternatively it might me more convenient substituting 0 for x from the start instead of figuring out the polynome $q(x)$.

$8 \times \frac{-3}{1-3} \times \frac{-5}{1-5} + 10 \times \frac{-1}{3-1} \times \frac{-5}{3-5} + 11 \times \frac{-1}{5-1} \times \frac{-3}{5-3}\ mod\ 17 =$
$\frac{120}{8} + \frac{50}{-4} + \frac{33}{8}\ mod\ 17 =$
$15 + 50 \times 4 + 33 \times 15\ mod\ 17 =$
$15 + (-1) \times 4 + (-1) \times 15\ mod\ 17 =$
13

Exercise at page 430.

3-31 Provably secure usage of several encryption systems?

Having n encryption systems we divide up our plaintext into n pieces using an information theoretically secure threshold scheme in such a way that all n pieces are necessary to reconstruct the plaintext. We then encrypt every single piece using a different encryption system.

We get additional expense from the application of the threshold scheme as well as from having to submit all n pieces to the recipient (instead of just the result from the product cipher). The overall expense we get is about n times as big.

None of the two combinations beats the other with regard to security:

- It is true that it is impossible to prove anything in a product cipher about the effectiveness of the ciphers 2 to n. But since the attacker never gets to know the intermediate results, a product cipher generally is much more secure than the most secure cipher in the chain, and even more secure than the sum of all used encryption systems. What this says is that the effort necessary to break the product cipher is substantially higher than the sum of efforts needed to break each individual cipher.

- In order to be able to prove that this combination is as hard to break as the most securely used cipher (and even as the sum of all encryption systems), we need to make use of the strong assumptions that were given in the exercise. In general they should hold. But as in the discussion of the security of the product cipher we are now assuming things that are true 'in general'. That is why, from my point of view, the product cipher out-performs this combination not only in terms of efficiency, but also in terms of security.

A very simple implementation of an information theoretically secure threshold scheme in which all n parts are needed in order to decrypt the secret would be applying the Vernam cipher n-1 times: the n-1 keys together with the ciphertext make up the n parts necessary.

See p. #464

Exercise at page 430.

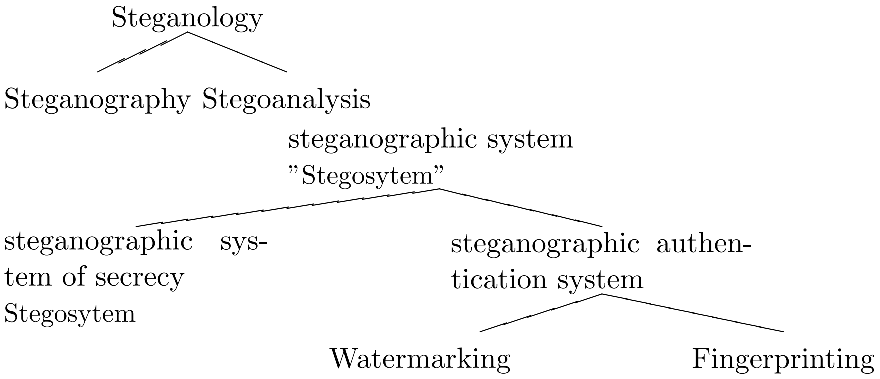
B.4 Answers for “Basics of Steganography”

4-1 Terms

Steganology integrates steganography (the knowledge of the “good people”’s algorithms) and steganalysis (the knowledge of the attacker’s algorithms on a steganographic system). It is crucial to know about steganalysis to be able to estimate the security of steganographic systems.

Steganographic systems are either encryption systems (security aim: hiding confidential communication) or authentication systems (security aim: embedding protected author information). The next criterion for differentiation is whether only author information is embedded (watermarking) or also user related information (fingerprinting).

The advantage of introducing the term “stegosystem” as the root of the right concept tree is that now we have a short, popular term to our disposal. The disadvantage, however, lies in the loss of differentiation between secrecy and authentication. So I use this term for steganographic systems of secrecy only — when using it in a broader sense, I will always denote that with quotation marks. My concept tree looks like this.



Exercise at page 430.

4-2 Properties of in- and output of cryptographic and steganographic systems

a) Below you will find figures copied from §3.1.1.1, which are precisely block diagrams for cryptographic systems.

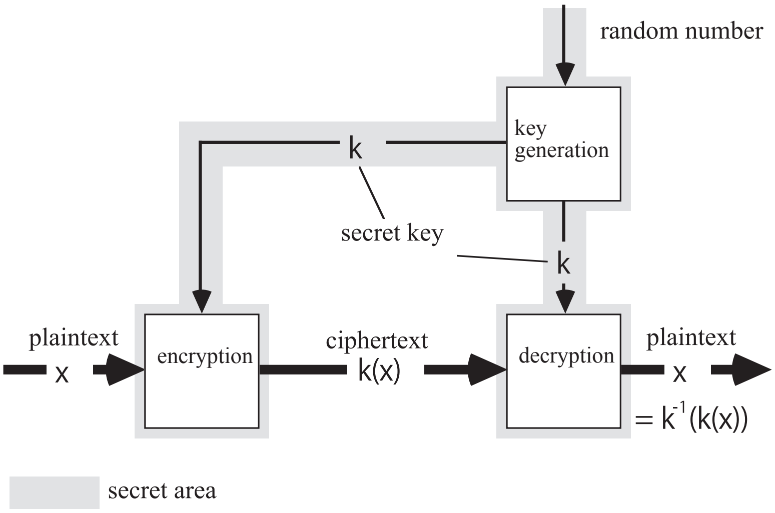


Figure 3.2: symmetric encryption system.

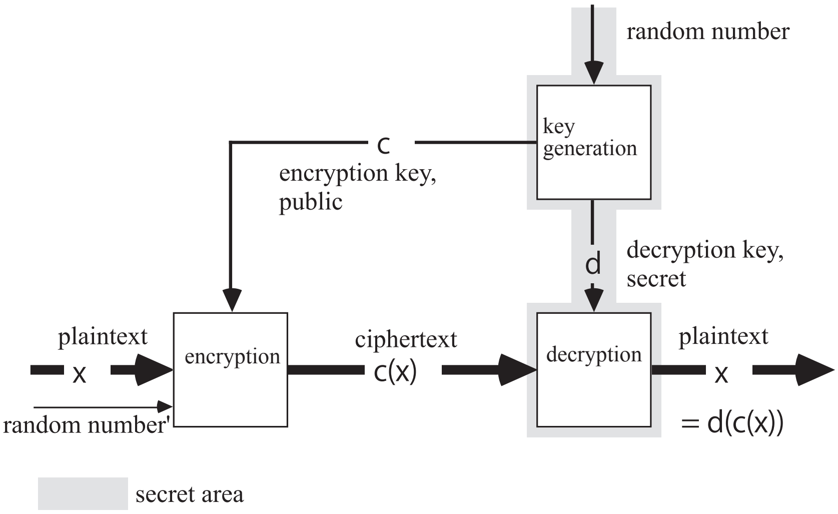


Figure 3.4: asymmetric encryption system.

When considering, for each diagram, all three algorithms as one system, the difference between in- and output symmetric and asymmetric encryption systems is only whether or not encryption needs an extra random number. Another difference lies in the requirements to the channel for key distribution. When looking at the interfaces of the three algorithms, the difference between symmetric and asymmetric encryption is whether the same key is used for de- and encryption (so it has to be kept secret) or keys for de- and encryption differ (so they are public).

A cryptographic system has (and is allowed) to assume that random numbers it gets as input are really random — where necessary random number generation should be part of the system that uses it. A good cryptographic system should not make any assumptions about the plaintext — that is input from outside the system (and therefore a given). Therefore, when publishing the key, encryption has to be carried out indeterministically, so 'simple' plaintexts can not be guessed, encrypted, and compared to the ciphertext. On the other hand though, there are no restrictions to the resulting ciphertext — except that it perhaps should not be excessively long.

b) Below you will find a figure that is copied from §4.1.1.1, which is precisely a block diagram for steganographic secrecy according to the embedding model.

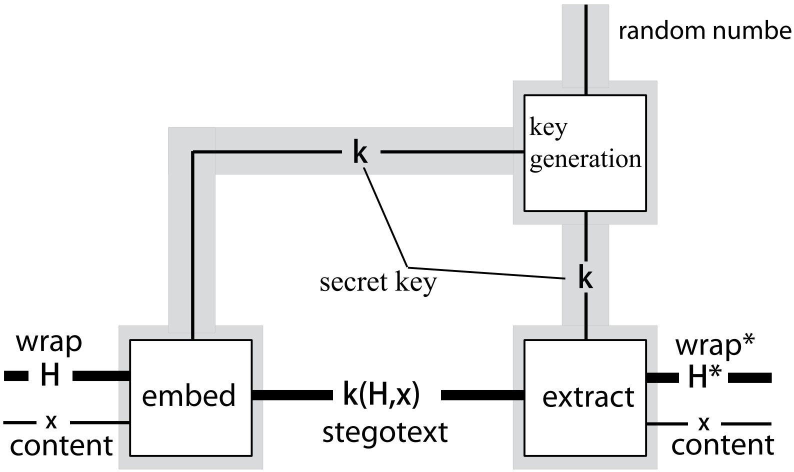


Figure 4.4: steganographic system of secrecy according to the embedding model.

There is only one block diagram, because there are no known asymmetric steganographic systems of secrecy.

Ideally a steganographic system of secrecy would not have to make any assumptions about the wrap, because in this model this is input (and therefore a given). (If we created a wrap in our steganographic system — synthetic steganographic secrecy — we would have a plausibility issue in our application context.) Unfortunately, no stegosystem will be able to work with arbitrary wraps, so certain assumptions have to be made and therefore every stegosystem has to rely on them to be adhered to. (The stegosystem should check adherence to those assumptions to a certain extent and refuse to embed the content into an inappropriate wrap.)

For all other pieces of input the same is true as for cryptography. Additionally, the output has to be exactly as plausible as the input wrap.

c) In encryption systems there are no assumptions the implementation is unable assure. This is different in steganographic secrecy (according to the embedding model — not according to the synthetic model): Assumptions about the wrap can not be assured by the implementation.

Exercise at page 431.

4-3 Is encryption superfluous when good steganography is used?

If it is impossible for an attacker to tell whether a secret message is embedded, he can not possibly understand it.

I would still always use encryption in practice. The first reason is to make sure that at least the secret message stays confidential even if the steganographic system of secrecy is not as secure as was hoped for. The second — and at least equally important — reason though, is to make the content indistinguishable from white noise. This makes it possible for the steganographic system of secrecy — more accurately: to its embedding function — to make strong, yet realistic assumptions about the input content.

Exercise at page 431.

4-4 Examples for steganographic systems

1. From my collection of about 90 recent holiday pictures I choose the one that by coincidence contains the four bits I would like to send. Security would be excellent — provided there is such a picture, which is very likely.

2. I choose one picture and scan it over and over again until the result contained the four bits I want to send. Security would be excellent.

3. In order to send 40 bits I choose 10 pictures as I did in 1. (which will still work out with fair probability). Every once in a while I will have to send the same picture twice. This will only provide reasonable security if the sender is known to be a bit absent-minded at times. So this crypto system would be primarily for notoriously abstracted professors.

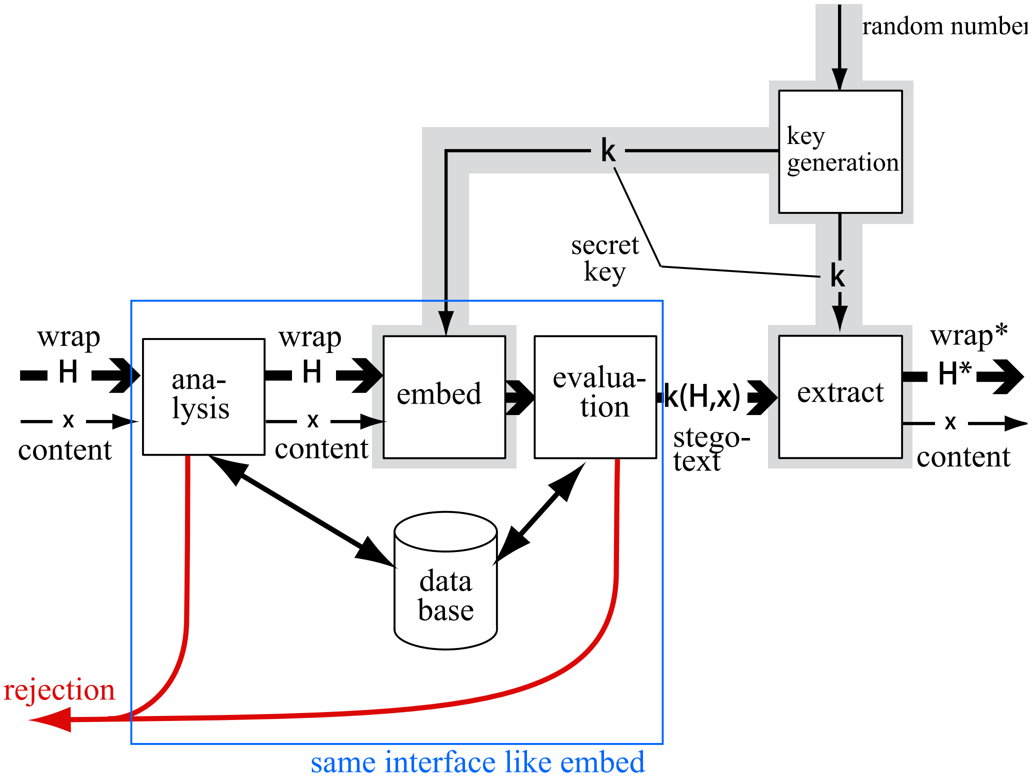
4. In order to send 40 bits I choose 10 of my favorite pictures and scan them as in 2. Security would be good, although the time that scanning takes could be a concern.

Exercise at page 432.

4-5 Modeling steganographic systems

Making up models is more an art than a deterministic process — there are no right (or wrong) models but models that are more (or less) appropriate for a given purpose than others. A smart person once said, you are not done constructing a machine whenever there is nothing you could add, but when there is nothing you could take out of it. This certainly is a good creed to work by in the art of model building.

This, applied to the context of steganographic systems, means that Anja, Berta, and Caecilie all do make good points about certain steganographic systems. I.e. if you, for example, can be sure that wrap material never repeats bit by bit, you do not need a data base etc. So, since analysis, data base, and rating are not part of every steganographic system, and also since they can be thought of as part of the implementation of the model given by Figure 4.4 (see the following figure), I tend to agree with Doerthe... to a certain extent. This is because Anja's idea, that at least in some steganographic systems there should be some error output that rejects inappropriate wrap material, is actually a new one. There could be discussion in a gentleman's round some time about whether this should be included in the model explicitly, or whether it is implied because every operation may fail and there needs to be error output that the embedding system has to react to. Anyway: Our ladies' round's decision is that models of steganographic systems should include error output.



Exercise at page 432.

4-6 Steganographic secrecy with public keys?

To start with, I would not call it steganographic secrecy with public keys, because embedding and extraction are always done using the same information. I would say this is bootstrapping a steganographic system with key using a keyless steganographic system. It can be useful for example where keyless steganography provides less net bandwith, requires more computing power, or is less secure.
Exercise at page 433.

4-7 Enhancement of security by using several steganographic systems

There is not much sense in combining several steganographic systems of secrecy, because the existence of a confidential message in a chain of steganographic systems (see exercise 3-5) is hidden only by the outmost system. Also a great data overhead would be created.

When talking about steganographic authentication systems, the answer will be less simple: it might be useful to use a chain of just a few systems, since they would all have to be broken. On the other hand every application of a system will cause a certain — though very small — disturbance of the artwork. Hence not too many systems should be used.

Exercise at page 433.

4-8 Crypto- and steganographic secrecy combined?

Combining those two is a good idea, because that way the confidelity of the message is at least as strong as the weaker one of the two systems (as long as stego and crypto keys are chosen independently of each other!). To make sure the existence of the message is hidden as well as when using steganography only, it needs to be encrypted first with a cryptographic system (or two, or more — see exercise 3-5) and then embedded steganographically. The recipient, of course, first has to extract and then to decrypt the message.

Exercise at page 433.

4-9 Crypto- and steganographic authentication in which order?

Steganographic authentication first (i. e., watermarking or fingerprinting) then digital signature of the steganographically modified artwork (otherwise steganographic authentication would invalidate the digital signature).

Exercise at page 433.

4-10 Stego-capacity of compressed and non-compressed signals

When using lossless compression all the entropy of the signal (i. e., its stego-capacity) is preserved. If there is any redundancy in the signal, it is just encoded in a shorter way. Hence per-bit stego-capacity is increased by the use of Gzip, Stuffit et al.

Lossy compression, however, reduces the overall entropy of the signal and therefore the overall stego-capacity. What this means for the per-bit stego-capacity depends on the ratio of reduction of redundancy to reduction of entropy.

Exercise at page 433.

B.5 Answers for chapter “Security in communication networks”

5-1 Quantification of unobservability, anonymity, and unlinkability

Exercise at page 434.

5-2 End-to-End or link encryption

Exercise at page 434.

5-3 First encrypting and then encoding fault-tolerance or the other way round?

Exercise at page 434.

5-4 Treatment of address- and error-recognizing fields in case of encryption

Exercise at page 434.

5-5 Encryption in case of connection-oriented and connection-less communication services

Exercise at page 435.

5-6 Requesting and Overlaying

Exercise at page 435.

5-7 Comparison of “Distribution” and “Requesting and Overlaying”

Exercise at page 436.

5-8 Overlayed Sending: an example

Exercise at page 436.

5-9 Overlayed sending: Determining a fitting key combination for a alternative message combination

Exercise at page 438.

5-10 Several-access-methods for additive channels, e. g., overlayed sending

Exercise at page 438.

5-11 Key-, overlaying-, and transmission topology in the case of overlayed sending

Exercise at page 439.

5-12 Coordination of overlaying- and transmissionalphabets in case of overlayed sending

Exercise at page 439.

5-13 Comparison of “Unobservability of adjacent wires and stations as well as digital signal regeneration” and “overlayed sending”

Exercise at page 439.

5-14 Comparison of "overlayed sending" and "MIXes that change the encoding"

Exercise at page 440.

5-15 Order of basic function of MIXes

Exercise at page 440.

5-16 Does the condition suffice that output messages of MIXes must be the same lengths?

Exercise at page 440.

5-17 No random numbers → MIXes are bridgeable

Exercise at page 440.

5-18 Individual choice of a symmetric encryption system at MIXes?

Exercise at page 440.

5-19 Less memory effort for the generator of untraceable return addresses

Exercise at page 440.

5-20 Why changing the encoding without changing the length of a message

Exercise at page 441.

5-21 Minimal expanding of length in a scheme for changing the encoding without changing length of a message for MIXes

Exercise at page 441.

5-22 Breaking the direct RSA-implementation of MIXes

Exercise at page 441.

5-23 Use of MIX-channels?

Exercise at page 441.

5-24 Free order of MIXes in case of fixed number of used MIXes?

Exercise at page 441.

5-25 How to ensure that MIXes receive messages from enough different senders?

Exercise at page 442.

5-26 Coordination protocol for MIXes: Wherefore and where?

Exercise at page 442.

5-27 Limitation of responsibility areas – Functionality of network terminals

Exercise at page 442.

5-28 Multilaterally secure Web access

Exercise at page 443.

5-29 Firewalls

Exercise at page 443.

B.6 Answers for “Value exchange and payment systems

6-1 Electronic Banking – comfort version

a) On the one hand, this is ultra-comfortable for the customer only as long as his trust in the bank is ultimate — because in case of a lawsuit there are no documents that could serve as proof. On the other hand, the bank has to trust its customers — or trust that they will always win in court.

b) Unfortunately not; in today’s electronic banking systems the bank (its banking machine) gets nothing (no digital messages) from the customer it could not forge itself.

c) This does not change a thing, because — provided the bank has a copy of the keys — it could still create all the messages it could receive from the customer. This is true for all symmetric authentication systems as was laid out in §3.1. In digital signature systems this would be different if the keys were created by the customer himself.

Exercise at page 443.

6-2 Personal relation and linkability of pseudonyms

Role-pseudonyms grant more anonymity than personal pseudonyms, because actions of a user that were done using different pseudonyms can not be linked. That way no information about a pseudonym can be collected which could make deanonymization possible. In the same way transaction-pseudonyms provide more anonymity than business-relation-pseudonyms because even actions that were committed in the same business relation can not be linked. Thus no information accumulates for a given pseudonym that excesses what is necessary for the transaction.
Exercise at page 443.

6-3 Self-authentication vs. foreign-authentication

Self–authentication means authorization by the one that is making a statement, e. g., by referring to an earlier statement he made. Authentication in this case means proving to be the same person who acquired a certain right back then. This can be done by signing a message according to a pseudonym (=test key of a digital signature system) that was used then. A typical example would be spending digital money (=transfer of an ownership right) that was given to s.o. using that pseudonym.

Foreign authentication means authorization by a third party, which for example signs an authorization statement containing the pseudonym (=test key of a digital signature system) in use. Grade sheets, driver's licenses, passports, ratings of creditworthiness, debit and credit cards are typical examples of authorization statements.

The reason why the difference is crucial is that very often the one giving the authorization accepts responsibility in case of abuse. Therefore, with foreign authentication, two parties are liable instead of one with self–authentication. This is important especially when designing protocols in which some principal can act anonymously (more precisely: pseudonymously).

Exercise at page 444.

6-4 Discussion about security properties of digital payment systems

In my opinion it is necessary that banks and vendors can not create consumer profiles of their clients. Additionally it would be desirable that they can not do so even with combined forces.

It would be great if media of exchange that were lost in hard-disk crashes or together with the hardware could be returned to the respective owners (so-called loss tolerance). This can be seen as part of the "he only loses a claim if he intends to" property.

Multilaterally secure digital payment system

- **Security aim confidentiality**
  - Payment data should be kept secret from everyone except transaction partners (payer, payee, where app. bank, witness).
  - Payer and/or payee should stay anonymous to each other and should not be observable by third parties (incl. payment system provider, bank, witness, ...).
- **Security aim integrity**
  - Claims are only given up when intended.
  - When a recipient for a claim has been specified by its owner, precisely that recipient receives the claim.
  - The payer is able to provide evidence for a successful transaction to third parties when necessary.
  - Even when cooperating, users can not increase the sum of their monetary claims.

- **Security aim availability**
  - Users can transfer any claims they received.

Exercise at page 444.

6-5 How far do concrete digital payment systems fulfill your protection goals?

- **Telephone Banking:** Security aims are not met at all with regard to confidentiality. Security aims integrity and availability are satisfied only if the bank is not suspected to be potentially malicious i. e., if there is no need for provability towards the bank. We also have to assume that phone lines can not be tapped, otherwise attackers can use passwords, TANs, PINs they overheard, or whatever else is used to identify customers and check their authorization to violate integrity by transferring other people's claims and thus keep their legitimate owners from accessing them — at least temporarily.

- **HBCI:** Using encryption, confidentiality can be achieved towards third parties, although of course not towards the bank or payer/payee. No anonymization provisions are provided for. When using the interim solution triple-DES as a hardware based signature substitute, integrity (and thus availability) is only guaranteed, if the bank is not seen as a potential attacker. When using RSA with key generation on the client's side, integrity and availability are obtained only as long as their computer works correctly.

Exercise at page 444.

B.9 Answers for "Regulation of security technologies"

9-1 Digital signatures at "Symmetric authentication allows concealment"

Classically the answer would have been 'no', because everybody can test a signature and thus, whether authentication is valid, which enables them to detect the appropriate bits (or message fragments) and combine them.

To make the answer 'yes', precisely that needs to be prevented: everybody being able to test the signatures. These are ways to accomplish this:

- A digital signature system is used, but the test key is made available to the recipient only. This corresponds to the inclusion relation between digital signature systems and symmetric authentication systems depicted in fig. 3.12. The question whether we can still talk about digital signatures in this context can of course be argued.

- A special digital signature system is used, where digital signatures can only be tested by certain specified partners [Chau_95] and only one partner is specified [Rive_98].

Exercise at page 444.

9-2 "Symmetric authentication allows concealment" vs. Steganography

The procedure proposed by Shamir is somewhat similar to steganography in that the confidential message is embedded into an envelope bitstream and that confidentiality is ensured, because the message is not detected in the bitstream. The difference from good steganography is that it is possible for the observer to take notice of the anomaly of the bitstream: it contains all possible bit combinations. The aim of steganography — hiding the very existence of the secret message — therefore is not met by Rivest's scheme.

Exercise at page 444.

9-3 Crypto-regulation by prohibition of freely programmable computers?

Even this does not do the trick:

At the very least people could come together to exchange one-time pads and for example encrypt e-mails by hand. Modular addition and subtraction is commonly known and not hard to do — at least the XOR operation is simpler than writing and could even be learned in kindergarten.

Also people could use Key-Escrow without retroactive decryption at a time when total observation is not yet allowed to exchange long-enough keys so they do not even have to meet in person.

Exercise at page 444.