

# Summary of “Einführung in die Informationssicherheit”

Martin Winter

**Abstract**—This document should serve as a summary of the necessary information given in the lecture “Einführung in die Informationssicherheit”

## I. ABBREVIATIONS AND STANDARD NOTATION

Abbreviation	Notation
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining mode
CFB	Cipher FeedBack mode
CRL	Certificate Revocation List
DES	Data Encryption Standard
DHP	Diffie-Hellman problem
DLP	Discrete logarithm problem
DN	Distinguished Name
DPA	Differential Power Analysis
DSA	Digital Signature Algorithm
ECB	Electronic Codebook mode
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm
GDLP	Generalized discrete Logarithm Problem
HW	Hamming Weight
IDS	Intrusion Detection System
IFP	Integer Factorization Problem
LSB	Least Significant Bit
MAC	Message Authentication Code
MDC	Modification Detection Code
MSB	Most Significant Bit
OFB	Output Feedback mode
OID	Object Identifier
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
RSA	Rivest-Shamir-Adleman encryption scheme
SCA	Side-Channel Attack
SPA	Simple Power Analysis
SSCA	Simple Side-Channel Attack
DSCA	Differential Side-Channel Attack

## II. BASIC CRYPTOLOGIC PRINCIPLES

Because of the rise of digital communication, the value of information keeps growing, while it is subject to a large number of threats. This brings up the need to protect this information, hence Cryptography is the science of protecting information. A **cryptographic algorithm** is a **mathematical function**, that uses a **key** to encipher information, without this key, deciphering wouldn't be possible.

### A. CIA and Non-repudiation

Typically two entities want to **exchange messages securely** over an **insecure channel**, the **adversary** can now try to do several things, including eavesdropping the communication or altering it. **Eavesdropping** (Abhören) is a threat to the **confidentiality** of the message, a third member could possibly read the content of the message. On the other hand, **altering** (Verändern) is a threat to the **integrity** of the message, therefore it is required to **authenticate** the communicating entities. The last of the security service deals is called **non-repudiation** (Nicht-Anerkennung), which means, that a sender cannot deny a message was sent, this property cannot be achieved by secret key cryptography alone unfortunately.

### B. Secret Key Cryptography

The most important requirement for such algorithms, besides their resistance against attacks, is that the reduction of performance is minimal. There are three types, which are discussed here

- **Block Ciphers** : Used to Encrypt Data
- **Stream Ciphers** : Used as alternative
- **MDCs**: Ensure integrity of data
- **MACs**: For authentication

1) **Block Ciphers**: are defined as a set of boolean permutations on a  $n$ -bit vector, this set contains a boolean permutation for each value of a key  $k$ . It takes an element (from plaintext) and transforms it into an element from ciphertext. Such a block cipher usually consists of several transformations, which form the encryption algorithm, those transformations happen several times, the transformation itself is called *round function*, for each round a *round key*  $K_i$  is generated from the cipher key. There are mainly two different types of design, the *Feistel ciphers* and the *substitution permutation network (SPN)*.

There are also several modes of operation, whenever a message is longer than the block size, four modes are standardized. The *electronic code book* (ECB) corresponds

to the usual use case, the message is split into blocks and each block is encrypted separately with the same key.

In *cipher block chaining* (CBC) each ciphertext block is x-ored with the next plaintext block before encryption.

In the *output feedback* (OFB) mode and the *cipher feedback* (CFB) mode, a keystream is generated and x-ored with the plaintext.

2) **Stream Ciphers**: encrypt individual characters, usually bits, of a plaintext one at a time and use an encryption transformation, which varies with time. In contrast to block ciphers, this encryption not only depends on the key and plaintext, but also on the current state.

**Synchronous** stream ciphers generate a keystream independently of the plaintext, sender and receiver must therefore be synchronized (same key and same state within that key).

**Asynchronous** stream cipher is a stream cipher, in which the keystream is generated as a function of the key and a fixed number of previous ciphertext bits. Because the keystream is dependent on only a few previous ciphertext bits, self-synchronization is possible even if some of the transmitted ciphertext bits are corrupted.

3) **MDCs**: take an input of arbitrary length and compress it to an output of fixed length, which is called the hash value. It satisfies the following properties

- *preimage resistance*: computationally infeasible to find preimage to given hash value
- *2nd preimage resistance*: computationally infeasible to find 2nd preimage
- *collision resistance*: computationally infeasible to find two different inputs with same hash value

Hash functions are used to ensure the integrity of data, data is used as input to the hash function and the output is stored. It is possible to check the input for alteration by simply redoing the computation and comparing it with the original hash value.

4) **MACs**: Hash functions, which involve a secret key, are called MACs (message authentication codes). The output of such a *keyed hash function* is also called MAC, in contrast to MDCs they can also guarantee **data origin authentication** and **data integrity**. In order to ensure authenticity of data, an entity computes a MAC on the data using a private key, to verify the authenticity of the data later on, the MAC can be recomputed using the private key.

### C. Public Key Cryptography

In public key cryptography, secret keys are **replaced by keypairs**, consisting of a **private key**, which must be kept confidential, and a **public key**, which is openly accessible. A message, encrypted by the public key, can be decrypted by the private key, this works, because the keys are linked in a mathematical way, such that knowing the public key does not allow to recover the private key, but enable independent encryption/decryption.

1) **RSA**: The Rivest-Shamir-Adleman(RSA) algorithm was the first public key encryption algorithm invented. It

is based on IFP (Integer factorization problem) and keys are generated as follows:

One selects two large, **secret prime numbers**  $p$  and  $q$  and computes the public RSA modules

$$n = p \cdot q$$

Then one chooses a public encryption exponent  $e$  which satisfies

$$\gcd(e, (p-1)(q-1)) = 1$$

$\gcd$  stands for *greatest common divisor*. The private key  $d$  can be calculated by solving

$$e \cdot d = 1 \bmod (p-1)(q-1)$$

Hence the **public key** is the **pair**  $(e, n)$  and the **private key** is the **triple**  $(d, p, q)$ .

For encryption, the message needs to be represented as a number  $m < n$ , the ciphertext itself is computed by raising  $m$  to the power of  $e$

$$c = m^e \bmod n$$

The message can be decrypted by exponentiation like

$$m = c^d \bmod n \quad (1)$$

#### Example Key Generation:

In real applications,  $p$  and  $q$  should at least be 512 bits!

$$p = 5 \quad q = 11$$

$$n = p \cdot q = 55$$

$$(p-1)(q-1) = 40$$

$$\text{Random Number } e = 7$$

$$\gcd(e, (p-1)(q-1)) = 1$$

$$d = e^{-1} \equiv 23 \bmod 40$$

The public key is now  $(e, n) = (7, 55)$  and the private key is  $(d, n) = (23, 55)$ .

#### Example Encryption:

Encrypt the message  $M$ , assuming  $M = 25$  with public key  $(e, n) = (7, 55)$ .

$$c = 25^7 \equiv 20 \bmod n$$

#### Example Decryption:

Received message is  $c = 20 \bmod 55$  and private key is  $(d, n) = (23, 55)$ . This leads to

$$m = 20^{23} \equiv 25 \bmod 55$$

2) **El Gamal**: The El Gamal encryption algorithm is based on the DLP, and some public parameters can be shared by a group of users, called *domain parameters*. This parameter is a **large prime**  $p$  (such that  $p-1$  is divisible by another prime  $q$ ) and an element  $g \in \mathbb{Z}_p^*$  of order  $p$  (or that the order is divisible by  $q$ ). The **private key**  $e$  can then be computed by solving

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

### Example Key Generation:

In real applications,  $p$  should at least be 1024 bits!

$$p = 41 \quad g = 6 \quad d = 17$$

$$e := g^d = 6^{17} \equiv 26 \pmod{41}$$

The private key  $d$  is a random number between 2 and  $p - 1$ .

### Example Encryption:

Encrypt the message  $M$ , assuming  $M = 55$  with public key  $(e, p, g) = (26, 41, 6)$ . Then, use random number  $k = 11$  and compute

$$c_1 = g^k = 6^{11} \equiv 28 \pmod{41}$$

$$c_2 = m \cdot e^k = 55 \cdot 26^{11} \equiv 19 \pmod{41}$$

The encrypted message  $(c_1, c_2) = (28, 19)$  is now sent.

### Example Decryption:

Received message is  $(c_1, c_2) = (28, 19)$  and private key is  $(d, p, g) = (17, 41, 6)$ . This leads to

$$\frac{c_2}{c_1^d} = \frac{19}{28^{17}} \equiv 5 \pmod{41}$$

3) **ECC**: ECC (Elliptic Curve) is a smooth curve in the so called *Weierstrassform*. Similar to the El Gamal cryptosystem, several public parameters can be shared among a group of users. These parameters are the elliptic curve group itself and the base point, the **private key** is a chosen integer  $d$  and the public key is given by  $e = dB$ .

4) **Key Agreement**: A major disadvantage of symmetric key cryptography is, that secret keys need to be distributed securely to the users. This problem can be solved using public key cryptography.

5) **Digital Signatures**: A digital signature is the equivalent of a handwritten signature, it binds someone's identity to a piece of information. A digital signature scheme consists of a **signature creation algorithm** and an associated **signature verification algorithm**.

Digital signatures *with appendix* require the original message as input to the verification algorithm, digital signatures *with message recovery* **do not** require the original message as input, it is recovered from the signature itself.

The **DSS** is a standardized signatures scheme with appendix, which describes an algorithm called **DSA** (Digital Signature Algorithm) based on DLP, it has recently been extended and is now based on elliptic curves.

In order to sign a message, the owner first computes the hash value of the message, this value and the private key are then fed to the signing algorithm. The output is a signature of the input message, anyone who wishes to verify the signature can simply take the message, the public key and the signatures and compute the verification algorithm.

### D. Security and Attacks

There exist many different types of attacks on cryptographic primitives, usually the goal is to deduce the secret key.

A **passive attack** is one, where the adversary only monitors the communication channel or the side-channels of the communication devices, they target the implementation, not the algorithm itself.

An **active attack** is one, where the adversary attempts to alter the transmission on the channel or the computation inside the device (*fault attacks*). Side-channel attacks and fault attacks are so called **implementation attacks**, which pose a serious threat to cryptographic systems. To classify attacks, an assumption is made, that the adversary knows all details, except for the secret key.

- **ciphertext-only**: deduce secret key by only observing ciphertext
- **known-plaintext**: adversary has number of plaintext and corresponding ciphertexts
- **chosen-plaintext**: adversary chooses plaintext and is given corresponding ciphertext
- **adaptive chosen-plaintext**: choice of plaintext may depend on previous ciphertexts
- **chosen-ciphertext**: adversary chooses the ciphertext and is given the corresponding plaintext
- **adaptive chosen-ciphertext**: choice of ciphertext may depend on previous plaintexts

Encryption schemes, which are vulnerable to ciphertext-only attacks, are considered completely insecure.

1) **Security Models**: There are several different models, under which the security of a cryptographic primitive can be evaluated.

- **Unconditional security**: In this model, the adversary is assumed to have unlimited computational power, so this model is also called *perfect secrecy*. A necessary condition is, that the secret key is at least as long as the message, because of that, systems like that are impractical.
- **Computational security**: Adversaries are assumed to have polynomial computational power, an algorithm is considered to be secure, if the best algorithm for breaking requires superpolynomial or exponential number of steps.
- **Provable security**: A cipher has provable security, if the difficulty of breaking can be shown to be equivalent to solving a well known, hard mathematical problem like DLP.
- **Practical security**: It is also related to the computational power of the adversary, a cipher is considered practically secure, if the best *known* attack requires at least  $N$  operations, where  $N$  is a sufficiently large number.

### E. Review Exercises

1) *Name and explain the four security services (properties) cryptography can provide!:*

- confidentiality: "no unintended third party can understand the content"
- integrity: "no one can change the message while it is in transit"
- authentication: "get assurance of identity"
- non-repudiation: "sender cannot deny a message was sent"

2) *What is Kerckhoffs' principle?:* The adversary knows all details about the used algorithms and protocols, only the secret key is unknown.

3) *What are the basic components of a block cipher?:*

The block cipher is a mathematical function, which takes plaintext as input, separates it into blocks, encrypts those with the secret key and has as output the ciphertext. Examples are DES (Data Encryption Standard) and AES (Advanced Encryption Standard).

4) *What is the difference between a block cipher and a stream cipher?:* A block cipher encrypts blocks of data using a function dependent on the key, while a stream cipher generates a bitstream from the key and XORs it with the data.

5) *What is the difference between a MAC and an MDC?:* MAC (Message Authentication Code)

- keyed hash-function (hash-value is signed)
- ensures data integrity and data authentication

MDC (Modification Detection Code)

- unkeyed hash-function
- ensure data integrity

6) *What is the meaning of "pre-image resistance"? For what cryptographic primitive is it relevant?:* It should be computationally unfeasible, to find a pre-image to a given hash-value, this is relevant for MDCs (Modification Detection Codes). So it should be efficiently possible, to find the pre-image of a hash-value, meaning, finding a value that is mapped to a certain hash value.

7) *Explain how RSA encryption and decryption work!:* Shown in (II-C.1).

8) *What is a digital signature with appendix?:* A digital signature binds someones identity to a piece of information, those with appendix require the original message as input to the verification algorithm.

9) *Describe and explain the Diffie-Hellman Problem? What is it used for?:* It is used for RSA, generally used for computing a common secret key. Two entities A and B agree to use a prime number  $p$  and a base  $g$ , now A generates a secret random number  $a$  and B  $b$ . Then A sends  $g^a \bmod p$  and B sends  $g^b \bmod p$  and both can compute  $g^{ab} \bmod p$ , which is their common secret key.

10) *Name and explain at least three different models of security:* Shown in (II-D.1).

11) *What does the term "exhaustive key search" mean?:* All keys in the key space are being tried out, until the correct one is found.

## III. ELECTRONIC SIGNATURES AND PUBLIC KEY INFRASTRUCTURES

### A. Digital Signatures

A **digital signature** is a data string, which associates a message in digital form with some originating entity.

**DS with appendix** require the original message as input

**DS with message recovery** do not require the original message as input, the message is recovered from the signature itself.

**Randomized DS** has several valid signatures for a given message.

Most of the signatures schemes which are in use today, belong to the class of (randomized) digital signatures with appendix, this is due to the fact, that in most schemes, the hash of a message is signed instead of the message. This has advantages, the signatures are much shorter, when the hash is signed and several attacks, which apply for signatures schemes with message recovery can be thwarted (vereitelt) by signing the hash. Breaks can be classified as

- **Total break:** An adversary is either able to compute the private key or find an efficient signing algorithm, which is functionally equivalent to the valid signing algorithm
- **Selective forgery:** An adversary is able to create a valid signature for a particular message or a class of messages a priori, creating the signature does not involve directly the legitimate signer.
- **Existential forgery:** An adversary is able to forge a signature for at least one message, there is little or no control over the message and the legitimate user may be involved in the deception.

Attacks can be classified to the capabilities of the attacker

- **Key-only attack:** Adversary knows signer's public key
- **Known-message attacks:** Adversary has signatures for a set of messages, which are known to him.
- **Chosen-message attacks:** Adversary obtains valid signatures from a set of messages, chosen by him.
- **Adaptive chosen-message attack:** Adversary may request signatures which depend on the signer's public key and he may request signatures, which depend on previously obtained signatures.

1) *RSA Signature Scheme:* Key setup is the same as in RSA encryption,  $m$  denotes the input message,  $n$  the modulus,  $d$  the private key and  $e$  the public key.

---

#### Algorithm 1 RSA Signature Creation

---

**Input:**  $m, n, d$

**Output:**  $Sig(m)$

1.)  $M = Encode(m)$

2.)  $s = M^d \bmod n$

3.) return  $s$

---

---

**Algorithm 2** RSA Verification Creation

---

**Input:**  $m, s, e$ **Output:** Accept/Reject

- 1.) Check length of  $s$ , if invalid then Return *Reject*
  - 2.)  $M = s^e \bmod n$
  - 3.) If  $M = \text{Encode}(m)$ , then Return *Accept*, else *Reject*
- 

2) **DSA Signature Standard (DSS):** It requires the usage of a hash function, originally SHA-1 was required, but now other hash functions are also allowed.

The numbers created by the key generations algorithm are the two primes  $p$  and  $q$ , the element  $\alpha$  with order  $q$ , the private key  $x$  and the corresponding public key  $y$ .

---

**Algorithm 3** DSA Key Generation

---

**Input:****Output:**  $p, q, \alpha, x, y, g$ 

- 1.) Choose prime  $q$  such that  $2^{N-1} < q < 2^N$
  - 2.) Select another prime  $p$  such that  $2^{L-1} < q < 2^L$  and with the property that  $q|(p-1)$
  - 3.) Choose  $\alpha$  of order  $q$ , it holds then that  $\alpha = g^{(p-1)/q}$
  - 4.) Choose a random integer  $x$  such that  $1 \leq x \leq q-1$
  - 5.) Compute  $y = \alpha^x \bmod p$
  - 6.) Return public key  $(p, q, \alpha, y)$  and private key  $x$
- 

---

**Algorithm 4** DSA Signature Creation

---

**Input:**  $m, x$ **Output:** Sig( $m$ )

- 1.) Choose a random  $k$ ,  $0 < k < q$
  - 2.) Compute  $r = (a^k \bmod p) \bmod q$
  - 3.) Compute  $s = k^{-1}(h(m) + xr) \bmod q$
  - 4.) Return the signature for  $m$  which is  $(r, s)$
- 

---

**Algorithm 5** DSA Signature Verification

---

**Input:**  $m, (r, s), y$ **Output:** Accept/Reject

- 1.) Verify that  $0 < r < q$  and  $0 < s < q$
  - 2.) Compute  $u_1 = wh(m) \bmod q$  and  $u_2 = rw \bmod q$  with  $w = s^{-1} \bmod q$
  - 3.) Compute  $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$
  - 4.) Return the signature for  $m$  which is  $(r, s)$
  - 5.) Return *Accept*, if  $v = r$ , otherwise return *Reject*.
- 

3) **ECDSA Signature Scheme:** The elliptic curve digital signature algorithm (ECDSA) is the analogue to the DSA, the major difference is, that the DLP, which gives the algorithm its security, is hereby defined over an elliptic curve. Two entities choose a finite field  $\mathbb{F}_q$ , an elliptic curve  $E$ , defined over that field and a base point  $G$  with order  $n$ . One key pair is  $(d, Q)$ , where  $d$  is her private and  $Q$  is her public key.

---

**Algorithm 6** ECDSA Signature Creation

---

**Input:**  $m, d$ , elliptic curve with base point  $G$  of order  $n$ **Output:** Accept/Reject

- 1.) Chosse a random number  $k$  with  $k : 1 \leq k \leq n-1$
  - 2.) Compute  $kG = (x_1, y_1)$  and  $r = x_1 \bmod n$ . If  $r = 0$ , then go to step 1.
  - 3.) Compute  $k^{-1} \bmod n$
  - 4.)  $e = h(M)$
  - 5.) Compute  $s = k^{-1}(e + dr) \bmod n$ . If  $s = 0$ , then go to step 1.
  - 6.) Return the signature which is  $(r, s)$
- 

---

**Algorithm 7** ECDSA Signature Verification

---

**Input:**  $m, (r, s), Q$ , elliptic curve**Output:** Accept/Reject

- 1.) Verify, that  $r, s$  are integers in the intervall  $[1, n-1]$
  - 2.) compute  $e = h(M)$
  - 3.) Compute  $w = s^{-1} \bmod n$
  - 4.) Compute  $u_1 = ew \bmod n$  and  $u_2 = rw \bmod n$
  - 5.) Compute  $X = u_1G + u_2Q$
  - 6.) **if**  $X = 0$  **then**
  - 7.) Return *Reject*
  - 8.) **else**
  - 9.) Compute  $v = x_1 \bmod n$  where  $X = (x_1, y_1)$
  - 10.) Return *Accept* if  $v = r$ , else return *Reject*!
- 

## B. Public Key Infrastructures

Even public keys need protection, because adversaries could try to modify published public keys. So there needs to be some mechanism to check their authenticity, meaning to check, if a certain key belongs indeed to a certain person. A public key infrastructure (PKI) is the basis of a security infrastructure, whose services are implemented and delivered by using public-key concepts and techniques.

1) **Public Key Distribution:** Different possibilities for distributing public keys include the *face-to-face approach*, isn't practical, the person would need to meet with everyone who wishes to obtain her public key in person. The second option is, that the person publishes her key on a trusted web server and anyone, who wishes to obtain the key simply looks there. Is still impractical, because there is

no way to verify the identity and integrity of the key.

The third option is, that the certified public key is stored on a trusted web server, meaning that the trusted web server provides a service, which adds additional information to a key, the so-called **certificate**. A certificate basically consists of a public key and some user information, which has been digitally signed by the issuer of the certificate.

2) **PGP**: In a PGP based PKI, each user acts as a third party, who can sign keys. Each user also manages keys herself, hence, there is no need for a central controlling authority, it is based on the so-called **user-centric trust model**. Each user trusts herself and a limited amount of other users. For example, Alice trusts Bob, hence she trusts all the keys, which have been signed by Bob, so if Bob trusts Clark, then also Alice trusts Clark.

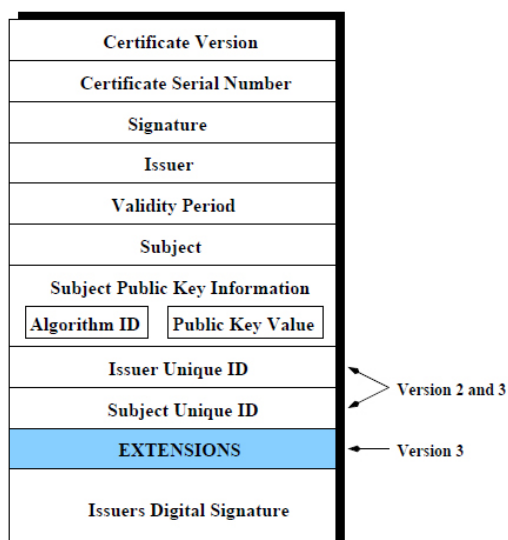
The problem here is, that it is implicitly assumed, that **trust is transitive**, also the **revocation** (Widerruf) is difficult.

3) **X.509**: The trust model, which is the basis for X.509-based PKIs is the hierarchical trust model, there exists a so-called **root CA**, which acts as root or anchor of trust for the entire domain of entities below it, it owns a root-certificate, which is the basis of trust for all entities, that belong to the domain. The root CA certifies CAs immediately below it, who certify again some CAs below them and so on.

At the second to last level, the CAs certify end entities. **Each entity in the hierarchy must be supplied with the certificate of the root CA.**

The advantage is, that because of the hierarchical structure, revocation checking can be reliably implemented, disadvantages are, that they tend to be big and complex.

4) **X.509 Certificates**: A certificate is a data structure, which essentially **consists of the public key** of an entity **plus some information about the entity**, and it is **digitally signed** by the certificate issuer.



An **OID** is a unique representation for a given object, for example an algorithm or a standard, it is represented as a sequence of integers separated by decimal points. They are hierarchical and they are registered with authorities to ensure uniqueness.

A **DN** is a hierarchical naming convention, they are supposed to ensure the unique name of an entity and are expressed as a concatenation of relative DNs from the top-level node down to the last node of DN.

#### 5) X.509 PKI Components:

- **Registration Authority (RA)**: Often, a separate registration authority is used to verify the identity of entities. For highly trusted certificates, the verification of the identity requires physical presence of the entity.
- **Certification Authority (CA)**: The certification authority issues certificates to entities, which have already been identified.
- **Certificate Repository**: The certificates are stored in the certificate repository, those can be implemented using LDAP (lightweight directory access protocol).
- **Certificate Revocation**: The revocation of issued certificates is hardly implemented in practice, instead, most mechanisms are focused on the distribution of revocation information.

### C. Legal Aspects of Electronic Signatures

The legal definition of an electronic signature is given as follows

*Data in electronic form, which are attached to or logically associated with other electronic data and which serve as a method of authentication.*

According to the above mentioned directive, the validity of this electronic signature cannot be denied solely based on the fact, that it is in electronic form.

1) **Advanced Electronic Signatures**: An *advanced electronic signature* must be uniquely linked to the signatory (Unterzeichner). It also must be capable of identifying the signatory and must be created using means, that the signatory can maintain under his sole control. It also must be linked to the data, to which it relates, in such a manner, that any subsequent change of the data is detectable.

This type of electronic signatures are not considered equal to handwritten signatures from a legal point of view.

2) **Qualified (Secure) Electronic Signatures**: A *qualified (secure) electronic signature* is an advanced electronic signature, which was produced on a secure signature creation device and which is based on a qualified certificate. It must be assured, that the signature creation data (private key) exists only once and is protected by the user.

The Austrian Signature Law (SigG) specifies that, apart from a few exceptions, a qualified digital signature is legally equivalent to a handwritten signature.

#### D. Electronic Signatures in Practice

Applications, that profit from electronic signatures, are primarily available in the e-government sector. The Austrian initiative is known as the concept "Bürgerkarte", in this concept, the so-called "Bürgerkartenumgebung" is the combination of the security-relevant components with respect to electronic signatures, applications communicate with the Bürgerkartenumgebung over the so-called **security layer**, which is an interface, that provides a logical view.

#### E. Review Exercises

1) *What is a digital signature?*: In the language of cryptographers, a *digital signature* is a data string, which associates a message in digital form with some originating entity.

2) *What classes of digital signatures can you distinguish?*:

- DS with appendix
- DS with message recovery
- Randomized DS

3) *What types of breaks of digital signatures schemes can you distinguish?*:

- **Total break**: An adversary is either able to compute the private key or find an efficient signing algorithm, which is functionally equivalent to the valid signing algorithm
- **Selective forgery**: An adversary is able to create a valid signature for a particular message or a class of messages a priori, creating the signature does not involve directly the legitimate signer.
- **Existential forgery**: An adversary is able to forge a signature for at least one message, there is little or no control over the message and the legitimate user may be involved in the deception.

4) *Describe the working principle of a digital signature scheme with appendix!*: Signatures with appendix require the message as input for the verification.

5) *What is a PKI used for?*: PKI stands for Public Key Infrastructure, when using asymmetric encryption, the public keys have to be exchanged which can happen over a trusted web server. Besides the key there is also a certificate available, that guarantees, that the key hasn't been altered.

6) *What is the difference between a PGP-based and a X.509-based PKI?*: Difference lies in the trust model.

**PGP**: Each user acts as CA and signs keys from other users, a user centric trust, if A trusts B and B trusts C, A also trusts C.

**X.509**: Hierarchical structure, root CA creates root certificate, can define some CAs below him, which also can certify CAs below them and so on, the second to last defines the end user.

7) *What is a CRL?*: A CRL (Certificate Revocation List) is a list of certificates, that have been revoked (für ungültig erklärt worden).

8) *What is an advanced electronic signature?*:

- It must be uniquely linked to the signatory
- It must be capable of identifying the signatory
- It must be created using means, that the signatory can maintain under his sole control
- It must be linked to the data, which it relates to, in such a manner, that any subsequent change of the data is detectable.

9) *What type of electronic signature is equivalent (from a legal point of view) to a handwritten signature?*:

Qualified electronic signatures are equivalent to handwritten signatures.

- advanced electronic signatures based on
- secure signature-producing-process and based on qualified certificate
- mandatory requirements like a private key must only exist once
- the requirements are protected from harmful access of others

#### IV. ELECTRONIC COMMERCE

Electronic commerce, is the business of buying and selling products, information or services electronically, those are mostly conducted using a credit or debit card. Typically, e-commerce merchants store cardholder information in databases to streamline the process, hence, containing thousands of payment card accounts.

##### A. E-shopping

E-shopping is a *multi-phase process*. In the first phase, the customer browses through products, selected goods are typically put into a shopping cart and at the end of this process, the customer is usually asked to proceed to check out. In the second phase, the customer selects the payment and delivery options and in the third phase the goods are delivered.

One of the most important factors for the success of e-business is having a secure and reliable system, this depends on who's asking the question.

##### B. Types of Payment

**Cash** is a floating claim on a bank or other financial institution, that is not linked to any particular account, it is a *direct payment method*, it is well suited for occasional purchasing and for buying physical products but less suited for buying services, that are occur repeatedly.

Other types of payment systems are **checks** and **bank cards**. Checks are suitable for not too small payments, a disadvantage is the costly rejection of a check.

Bank cards are also suitable for any not too small value, the payee can be remote from the payer, many of those systems require only connections for verification nowadays.

**Credit systems** work differently, the customer has to register with a bank, if goods are purchased, they get debited on his account and the customer is only billed after a certain amount of time has passed or only once a month.

In a **debit system**, the customer pays before the purchase, such systems are suitable for subscriptions or any regular purchases.

##### C. E-payments

1) **E-money**: E-money or **digital currency** is simply an encoded string of digits and essentially consists of a serial number and the amount. It is a floating claim on a bank or other financial institution, that is not linked to any particular account. Expected Advantages are, that it is anonymous, and involved entities do not have to trust each other, and there follows the direct disadvantage, following the anonymity aspect, it is **not traceable**.

It is possible to distinguish **micropayments** and **macropayments**.

**Micropayments** are tiny value transactions below \$1, they are supposed to be cheap and a little cheating can be tolerated because of the minuscule amount.

**Macropayments** are medium value transactions (between \$1 and \$1000) and large value transactions (over \$1000).

- **Security**: It is mandatory to prevent forgery, double spending and collusion of other parties (multi-party security)
- **Acceptability**: A wide range of parties needs to accept the payments
- **Convenience**: Factors that increase the convenience are speed, reliability, divisibility and transferability.
- **Cost**: Virtually no cost should arise while dealing with digital currency, as it is with physical currency.
- **Privacy**: Cash is the most anonymous payment form, e-money should be as well.
- **Durability**: E-money shouldn't be easily lost, like during a system crash
- **Independence**: E-money should be easily storable on different devices
- **Transferability**: Cash can be easily exchanged between entities, should be the same for e-money.
- **Divisibility**: Customer gets return money, if price is lower than the given amount in the physical world, should be the same for e-money.
- **Immediate Control**: In case of a security breach, such as forgery of money, it must be possible to identify the security breach immediately.
- **Traceability**: Especially with large transactions, anonymous money can be a problem, some sort of tracing mechanism would be an advantage.
- **Spread of Encryption Mechanisms**: E-money systems, which involve encryption heavily, might be difficult to export to other countries.

Trade-offs include

- **Privacy vs Traceability**: Tiny value payments deserve to be anonymous, while for large value payments traceability is probably favourable.
- **On-line vs Off-line**: On-line payments allow to prevent double spending easily and simplify tracing transactions. Offline payments often use special purpose hardware to prevent double spending.
- **Hardware vs Software**: Hardware solutions can also prevent double spending, but often hardware decrease the customer acceptance.
- **Transparency vs Explicitness**: User want to have control over all their transactions but they also expect not to be bothered with any details.

Discussed will be the famous DigiCash system, coins there are anonymous, so the bank doesn't know, which coin belongs to which customer. Despite the anonymity, the system prevents multiple spending of coins, it allows even to determine with a high probability, who tried to double-spend money in case such a fraud occurred. The protocol,



describing the DigiCash system works as follows:

- **Money Generation Phase:**

- 1) The customer prepares  $n$  anonymous money orders for a given amount, each order consists of a random unique string  $X$  (the serial number) and  $m$  pairs of identity bit strings, who have three important properties. They cannot be changed later by the customer, each pair identifies the customer and they are sealed by default!
- 2) Customer puts all orders in envelopes, seals them and sends them to bank
- 3) Bank asks customer to unseal and open  $n - 1$  envelopes of his choice, it checks correctness and asks customer to reveal all identity strings.
- 4) If the bank is satisfied (customer didn't try to cheat), the bank signs the one remaining seals money order, gives it back to the customer and deducts the proper amount from the customers bank account
- 5) Customer unseals the order to extract the coin.

- **Money Spending Phase**

- 1) Customer unseals order and spends coin with merchant
- 2) Merchant checks validity by verifying the banks signature.
- 3) Merchant asks Customer to reveal either left or right half of each identity string, he does that by giving the customer a random selector string, that tells, which half to reveal.

- **Double-spending Checking Phase:**

- 1) Merchant takes the coin to the bank
- 2) Bank verifies signature and checks database for uniqueness of string  $X$ . If unique, bank credits the amount to the merchant's account and records the unique string in the database.
- 3) If on the other hand the string isn't unique (already in DB), the bank doesn't accept coin. It compares the identity string with the one in the DB, if they match, the bank knows, that the merchant cheated. If the strings do not match, then Alice must have copied the coin. The second merchant, who accepted the coin, gave the customer different random selector strings, meaning, the bank can find a position, where one merchant has opened the left half and the other the right half, together they reveal the identity of the customer.

So in short, customer produces  $n$  money orders with  $n$  serial numbers and  $n$  identity strings, then the customer "blinds" the money orders, so the bank doesn't see the serial numbers, so it is anonymous. The bank now checks  $n - 1$  money orders for their consistency, if valid, it signs the last remaining blinded money order with its private key, without seeing what's inside, customer now possesses an anonymous coin

signed by the bank. In the spending phase, the merchant checks the validity of the coin by requesting halves of the identity strings, randomly either the left or right one and checks validity.

When merchant now deposits coin at the bank, signature is once again validated and the database is checked for the serial number. If the string is already present, bank checks stored identity strings, if merchant wanted to collect money twice, those strings will match. Otherwise there will be a pair of identity string halves, that will produce a full identity string and reveal the cheating customer.

2) **Bank cards:**

3) **SET and its Successors:**

#### **D. Alternative E-payment Methods**

1) **PayPal:**

2) **PaySafe Card:**

3) **First Gate Click&Buy:**

4) **Google Wallet and NFC:**

#### **E. Review Exercises**

1) *What types of payment systems can you distinguish?:*

2) *What is the difference between micropayments and macropayments?:*

3) *Discuss 4 aspects of digital currency!:*

4) *Discuss the trade-offs that are typically made in electronic payment systems!:*

5) *Explain in short, how money is generated in the DigiCash system!:*

6) *What does SET mean and what was/is the SET system used for?:*

7) *Explain the purpose of the dual signature in the SET system!:*

### **V. NETWORK SECURITY**

#### **A. Review Exercises**

### **VI. IMPLEMENTATION SECURITY**

#### **A. Review Exercises**

### **VII. OPERATING SYSTEM SECURITY**

#### **A. Review Exercises**

### **VIII. PRIVACY**

#### **A. Review Exercises**

### **APPENDIX**