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	Generlized 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 possible 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 preformance 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 ciphtertext 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 wich satisfies

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

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

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

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 represented as a number $m < n$, the ciphtertext itself is computed by raising m to the power of e

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

The message can be decrypted by exponentiation like

$$m = c^d \ mod \ n \tag{1}$$

Example Key Generation:
In real applications, p and q should at least be 512 bits!

$$\begin{aligned} p = 5 \quad q = 11 \\ n = p \cdot q = 55 \\ (p-1)(q-1) = 40 \\ \text{Random Number } e = 7 \\ gcd(e, (p-1)(q-1)) = 1 \\ d = e^{-1} \equiv 23 \ mod \ 40 \end{aligned}$$

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 $(e, n) = (7, 55)$.

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

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

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

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 $q \in \mathbb{Z}_p^*$ of order p (or that the order is divisible by q. The **private key** e can then be computed by solving

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

Example Key Generation:
In real applications, p should at least be 1024 bits!

$$p = 41 \quad g = 6 \quad d = 17$$
$$e := g^d = 6^{17} \equiv 26 \; mod \; 41$$

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

$$c_1 = g^k ? 6^{11} \equiv 28 \; mod \; 41$$
$$c_2 = m \cdot e^k = 526^{11} \equiv 19 \; mod \; 41$$

The encrypted message $(c_1, c_2) = (28, 41)$ is now sent.
Example Decryption:
Received message is $(c_1, c_2) = (28, 41)$ and private key is $(d, p, g) = (17, 41, 6)$. This leads to

$$\frac{c_2}{c_1^d} = \frac{41}{28^{17}} \equiv 5 \; mod \; 41$$

III. Electronic Signatures and Public Key Infrastructures

IV. Electronic Commerce

V. Network Security

VI. Implementation Security

VII. Operating System Security

VIII. Privacy

Appendix