Introduction to Cryptography

École Polytechnique de Bruxelles

Professeur : *Gilles* Van Assche

Sami Abdul Sater

Année académique : 2021-2022

Table des matières

Chapitre 1

Historical ciphers and general principles

Cryptology is a term merging two similar fields of stuy : cryptograph and cryptanalysis.

- **Cryptography** : the study of secret writing with the goal of **hiding a message**.
- **Cryptanalysis** : breaking cryptosystems.

1.1 Cryptography

In cryptography, to hide a message, there are two things that interest us : hide our content (**Confidentiality**) and authenticating a message (**Authentication**).

1.1.1 Confidentiality

For a generic cryptosystem that ensures confidentiality of a message, we talk about two operations :

- **Encryption** of a plain-text message to get a ciphertext
- **Decryption** of a ciphertext to retrieve a message

We always have to picture two persons communicating with each other, and eventually a third-party intervenant trying to have access to the conversation. Hence, encryption and decryption are meaningless without talking about the intervenants, that we choose to name Ali and Bachar [1].

Ali and Bachar's communicating schema is the following : the first encrypts a message, sends it to the seconds that knows how to decrypt it to find the original content of the message. As they must be the only ones able to encrypt and decrypt the same way, they must have some kind of **key**.

This key generation/sharing/storage is the source of the division of cryptograph in two kinds : a symmetric way or an asymmetric way.

- In **Symmetric crypto**, also called **secret-key** crypto, both parts have an encryption and a decryption method, and they **share the same key that is secret**, kepts out of the sight of any outsider. We also assume that the encryption and decryption algorithms are **publicly known**.

1. Instead of Alice and Bob, let's change continent a bit.

- In **Asymmetric crypto** (since 1976), the two possess both a private and a public key. They share their public key, but never their private key !

So in general, we will talk about encryption as a mechanism that takes a message m, encrypts it with a key k_E to get a ciphertext c, and sends it. As for decryption, it takes a ciphertext c, decyrpts it under a key k_D to obtain m.

- Symmetric : $k_E = k_D$
- Asymmetric : k_E is public, k_D is private.

1.1.2 Authentication

As for authentication, we **are not trying to hide anything**. The message is sent in full plain-text from Ali to Bachar. Our goal here is to **check** the source of our message, assure its authenticity.

Similarly to encryption/decryption, we here have two mechanisms with keys :

- Authentication : Ali generates a tag under a key k_A, and sends the couple (m, tag) to Bachar

$$m \Rightarrow (m, \text{tag})$$

- Verification : Bachar receives (m, tag), and under key k_V, identifies the source.

$$(m, \text{tag}) \Rightarrow \{m, \perp\}$$

In symmetric crypto, $k_A = k_V$ and is **secret**. In this case, the tag is more commonly called **Message Authentication Code** (MAC).

In asymmetric crypto, k_A is private and k_V is public. In this case, the tag is called "**signature**". To do so, with the message, Ali sends a tag.

Confidentiality

Encryption

- plaintext $\Rightarrow$ ciphertext
- Under key $k_E \in K$

Decryption

- ciphertext $\Rightarrow$ plaintext
- Under key $k_D \in K$

Symmetric cryptography: $k_E = k_D$ is the secret key.

Asymmetric cryptography: k_E is public and k_D is private.

Authenticity

Authentication

- message $\Rightarrow$ (message, **tag**)
- Under key $k_A \in K$

Verification

- (message, **tag**) $\Rightarrow$ {message, $\perp$}
- Under key $k_V \in K$

Symmetric cryptography: $k_A = k_V$ is the secret key.
The tag is called a *message authentication code* (MAC).

Asymmetric cryptography: k_A is private and k_V is public.
The tag is called a *signature*.

1.2 Cryptanalysis

Cryptanalysis is the field that studies algorithms and ways of breaking a cryptosystem. This means, recovering the message, or recovering the key. There are several ways to do this, going from "little average mathematician boi that exploits the inner structure of the scheme" to the "chad asking you your password with a gun pointing to the head". All the methods, from the first to the last, are part of **cryptanalysis**. But in this course, we focus on what we call **mathematical cryptanalysis**. We will also place ourselves in the Kerckhoff's principles.

Kerckhoff's principles for cryptographic systems

The security of a cryptosystem must only rely on the **secrecy of its key**. We hence assume, when evaluating the security of a system, that everything is known : length of the messages, encryption and decryption scheme.

1.2.1 Mathematical cryptanalysis

Some definitions

- Key space : set of all possible keys
- Brute-force attack : attack that tries all the keys of the key space.

This branch studies brute-force attacks and analytical attacks. Analytical attacks can be of several types : exploiting some statistical patterns, length extension attack, ...

1.2.2 Key length

This is an informative section on key length, just to develop an intuition on the impact of the length of a key in a cryptographic system.

First, it is important to mention that the key length in a **symmetric** crypto system is relevant only if the brute-force attack is the best-known attack.

Excluding this case, then the guaranteed security of a cryptosystem according in function with the key length is very different depending on the kind of crypto : a 80-bit key in symmetric crypto can ensure the same security as a 1024-bits key asymmetric scheme (such as RSA).

1.2.3 Time to break

Here is an indicator of the meaning of the "time-to-break" (TTB) of cryptosystems in function of the key length for a **symmetric scheme**.

Key length	Security estimation
56–64 bits	short term: a few hours or days
112–128 bits	long term: several decades in the absence of quantum computers
256 bits	long term: several decades, even with quantum computers that run the currently known quantum computing algorithms

1.3 Some ciphers

1.3.1 Shift encryption scheme

Chose a key k between 0 and 26, message m also between 0 and 26. The shift encryption scheme is all about XORing the message with the key :

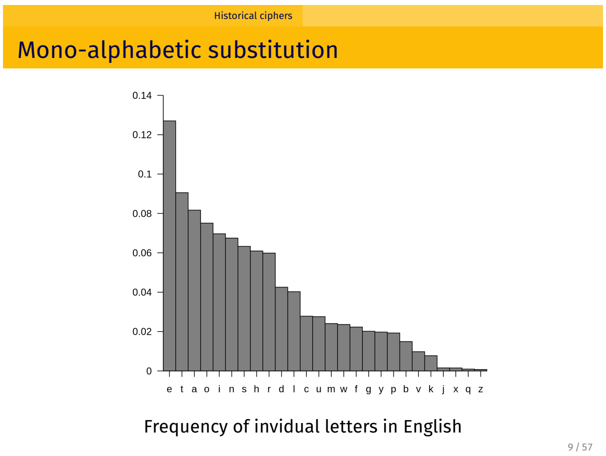
$$\begin{aligned} E_k(m) &= m + k \mod 26 &= c \\ D_k(c) &= c - k \mod 26 &= m \end{aligned}$$

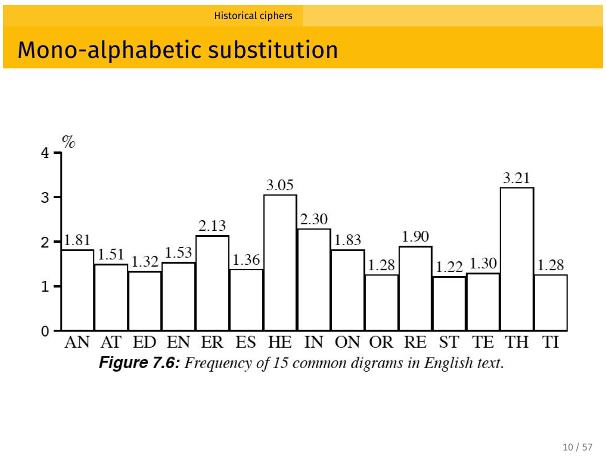
It is really dumb : as the key space is very limited (26), it is quickly subject to brute-force methods.

1.3.2 Mono-alphabetic substitution

The mono-alphabetic cipher consists in replacing each letter of the message by a corresponding letter in a mixed alphabet chosen randomly. So we define a substitution table, and we apply our mapping. Let's break it down a little bit.

- It is a symmetric scheme.
- The key space is $s = 26! > 4 \cdot 10^{26}$: a brute-force attack would take some time.
- It is breakable using a statistical approach.





1.3.3 Poly-alphabetic substitution

Instead of encrypting a entire message with the same mapping, we here divide the message into t blocks.

$$x = x_1 \| x_2 \| \dots \| x_t$$

Then, we define a mapping for each block. This will be encoded in the key k of the scheme. Indeed, each $k \in K$ will define a **set of permutations**

$$k \Rightarrow (p_1, p_2, \dots, p_t) \; .$$

Hence, $E_k(x)$ will be given by

$$E_k(x) = p_1(x_1) \| p_2(x_2) \| \ldots \| p_t(x_t) \|$$

As for the decryption key k', it needs to define the set of the t corresponding inverse permutations :

$$k' \Rightarrow (p_1^{-1}, p_2^{-1}, \ldots, p_t^{-1}) \ .$$

1.3.4 Vigenère cipher

- Message m of length $|m|$, chosen in $(\mathbb{Z}_{26})^\star$
- Key space $K \subset (\mathbb{Z}_{26})^t$. Size : logically 26^t
- Key k taken randomly in K, so

$$k = (k_0, k_1, \ldots, k_{t-1}) \in K$$

Then, the encryption of m will result in the concatenation of the XORing of each bit m_i with a part of the key that. Remember that the key has only t parts, so we will repeat the same parts if the message is very long ! The same with a modular difference for the decryption

$$\begin{aligned} E_k(m) \equiv E_k(m_0 \| m_1 \| \ldots \| m_{|m|-1}) &= \big\|_{0 \le i \le |m|-1} (m_i + k_i \mod t) &= c \\ D_k(c) \equiv E_k(c_0 \| c_1 \| \ldots \| c_{|c|-1}) &= \big\|_{0 \le i \le |c|-1} (c_i - k_i \mod t) &= m \end{aligned}$$

In practice, the key can actually be a t-long string. During the process, each character is converted to a number.

Historical ciphers

Vigenère cipher – example

plaintext: `rendezvousahuitheure`
key: `hello` (7 4 11 11 14)

17 04 13 03 04 25 21 14 20 18 00 07 20 08 19 07 04 20 17 04
07 04 11 11 14 07 04 11 11 14 07 04 11 11 14 07 04 11 11 14
⇓
24 08 24 14 18 06 25 25 05 06 07 11 05 19 07 14 08 05 02 18

ciphertext: `YIYOSGZZFGHLFTHOIFCS`

14 / 57

Cryptanalysis of Vigenère cipher

We stick to Kerckhoff's principles : so we know how every message is encrypted, and the only thing that is kept secret is the key k. We don't even know its length t. And as it has been seen before, some bits of m are encoded with the same key chunk because of the modulo in the XOR ($k_{i \mod t}$). So we can group the bits of m that are encoded with the same chunks.

$$m_i, m_{i+t}, m_{i+2t} \longleftarrow k_{i \mod t}$$

For each chunk, we see that we actually have a simple shift encryption scheme that can be easily broken. So, if we know the length t of the key, we can break Vigenère cipher easily.

How to find the length of the key ? We can try brute-force. It will work. But there is a better method, using the **index of coincidence**.

Now that we have introduced the fundamental ciphers, we can move on some more cryptanalysis definitions : how do we quantify the security of cryptosystems ?

1.4 Perfect secrecy (PS – unconditional security)

A first definition that comes into the hand when talking about security of cryptosystems is **perfect secrecy**. It is an **ideal property** that a cryptosystem can achieve. In English, it states that it must not leak any information, even to an adversary with unlimited computational power. In mathematical terms (not real mathematics, but mmmhh), it states the following

Perfect secrecy

An encryption scheme satisfies perfect secrecy an adversary can not distinguish two random encryptions :

- For any two messages $m_1, m_2 \in M$
- For every ciphertext $c \in C$
- Choosing a key $k \in K$

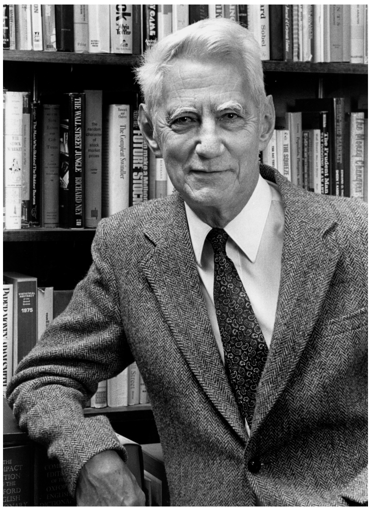
$$\Pr\left[\text{Enc}_k(m_1) = c\right] = \Pr\left[\text{Enc}_k(m_2) = c\right]$$

1.4.1 Perfect secrecy and length of keys

Claude Shannon showed that for a system to achieve perfect secrecy, the length of the key must be at least the length of the message. Note that it is possible to have a scheme with a key much longer than the message but for it to be not secure at all... It is an implication : it must, but it is not sufficient.

Perfect secrecy vs computational security

Perfect secrecy implies long keys



Claude Shannon (1916-2001)

Claude Shannon showed that, for perfect secrecy, the entropy of the key is at least the entropy of the plaintext, i.e.,

$$\text{perfect secrecy} \Rightarrow H(K) \geq H(M).$$

So the secret key must be at least as long as the plaintext and it may not be reused!

22 / 57

Note that in practical, this is a property that annoys us a lot because for long messages, we must have a longer key. We will see later some *lighter* security definitions.

1.4.2 One-time pad (OTP)

The one-time pad encryption scheme is quite easy : it is close to the Vigenère cipher, except for the fact that **the key is as long as the message**. It can thus ensure PS. But, does it really ?

We are given a message space
$$M = (\mathbb{Z}_2)^t$$
and $C = K = M$. This means we play with binary strings. The key will be written as
$$k = (k_0, k_1, \ldots, k_{t-1}) \ ,$$
and the messages, ciphertexts, can also be similarly written. The scheme is the following :
$$\begin{aligned} E_k(m) &\equiv E_k(m_0, m_1, \ldots, m_{t-1}) &&= (m_i + k_i)_{0 \leq i \leq t-1} &&= c \\ D_k(c) &\equiv D_k(c_0, c_1, \ldots, c_{t-1}) &&= (c_i - k_i)_{0 \leq i \leq t-1} &&= m \end{aligned}$$
It looks indeed similar to the Vigenère cipher. But here, let's compute the probability of knowing a plaintext – ciphertext pair.

$$\begin{aligned} \Pr[E_k(m) = c] &= \Pr[m \oplus k = c] && \text{Bit-wise operator} \\ &= \Pr[k = m \oplus c] && \text{Because } m \oplus m = 0 \\ &= 2^{-t} && \text{Because } k \text{ is chosen randomly in } (\mathbb{Z}_2)^t \end{aligned}$$

We thus prove that the OTP achieves perfect secrecy. It may look like Vigenère's, but in security ways, it is way different. OTP is the ideal scheme, Vigenère's is easily broken as we saw.

Why "one-time" pad? Because if we use twice the same key for a different message, there is information that is leaked from the system. In particular, we can find a link between the ciphertexts and the messages. Indeed :

If	$c_1 = m_1 \oplus k$	
and	$c_2 = m_2 \oplus k$	
Then	$c_1 \oplus c_2 = m_1 \oplus m_2$	Because $k \oplus k = 0$

1.5 Computational security

As we already saw, perfect secrecy requires the key being at least as long as the message. This annoys us very much. Some systems can be very secure **without achieveing perfect secrecy**. This allows us to leak some information. This introduces us the notion of **computational security**.

Perfect secrecy vs computational security

Computational security

Computational security

A scheme is (t, ϵ)-secure if any adversary running for time at most t, succeeds in breaking the scheme with probability at most ϵ.

Security strength

We say that a scheme is s-bit secure if, for all t, the scheme is $(t, \epsilon(t))$-secure and $\log_2 t - \log_2 \epsilon(t) \geq s$.

Example: exhaustive key search

After t attempts, the probabilty of finding the correct key is $\epsilon(t) = \frac{t}{|K|}$ with $|K|$ the size of the key space. If there are no faster other attacks than this, then the scheme is $s = \log_2 |K|$-bit secure.

28 / 57

We really need to interpret this with the example : if the key space of a system is of size 2^{128} and that for the best attack, the probability of finding k after t attempts is $\varepsilon(t) = t/2^{128}$, then the scheme is $s = 128$-bit secure.

1.6 Security principles

There is a big gap between cryptography and practical cryptosystems. In practical, none of the cryptosystems will see offer perfect secrecy. The only ways to gain confidence in the security of a scheme is by *peer-reviewing*. Particularly, the **algorithm must be public**, first for people to help proving its security/breaking it, but most importantly to stick to Kerckhoff's principles.

1.6.1 Goals of an adversary

Security definitions | For encryption schemes

To what does an encryption scheme must resist?

An adversary can have the following **goals**:

- recovery of the (secret or private) key
- recovery of even some partial information about the plaintext
- a property that distinguishes the scheme from ideal

The adversary is allowed to get (**data model**):

- ciphertexts only
- known plaintexts (and corresponding ciphertexts)
- chosen plaintexts (and corresponding ciphertexts)
- chosen plaintexts and ciphertexts

Attacks continue to work when going down in the two lists above. The best for the attacker is to be able to recover the key with ciphertexts only. A designer will be happy if no cryptanalyst was able to show a disitinguisher even with chosen plaintexts and ciphertexts.

34 / 57

1.6.2 Ability to query the scheme

Security definitions For encryption schemes

Offline and online complexities

Computational security

A scheme is (t, d, ϵ)-secure if any adversary running for time at most t and having access to d data, succeeds in breaking the scheme with probability at most ϵ.

- t: offline complexity
- d: online complexity

Security strength

We say that a scheme is s-bit secure if, for all (t, d), the scheme is $(t, d, \epsilon(t, d))$-secure and $\log_2(t + d) - \log_2 \epsilon(t, d) \geq s$.

1.6.3 Taxonomy of attacks

Security definitions For encryption schemes

Taxonomy of attacks

To describe an attack, one should specify:

- the goal;
- the data model and the online complexity (d);
- the offline complexity (t) and success probability (ϵ).

Example: exhaustive key search

The exhaustive key search is a **key recovery attack** that requires $d = 1$ **pair*** **of known plaintext / ciphertext** and takes **offline complexity** $t = |K|$ for a **success probability** $\epsilon = 1$.

* Depending on the relative size of the plaintext and the key, this may require more than one pair to avoid multiple key candidates.

1.6.4 Formal definition of an encryption scheme

Security definitions | For encryption schemes

Formal definition of an encryption scheme

An encryption scheme is a triple of algorithms $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a plaintext space M.

- **Gen** is a probabilistic algorithm that outputs a secret (or private) key k_D from the key space K. In asymmetric cryptography, it publishes the corresponding public key k_E.
- **Enc** takes as input a secret/public key k_E and message $m \in M$, and outputs ciphertext $c = \mathsf{Enc}_{k_E}(m)$. The range of Enc is the ciphertext space C.
- **Dec** is a deterministic algorithm that takes as input a secret/private key k_D and ciphertext $c \in C$ and output a plaintext $m' = \mathsf{Dec}_{k_D}(c)$.

37 / 57

1.6.5 Semantic security

An encryption scheme is **semantically secure** if : when the adversary can compute with the ciphertext something about the plaintext in a polynomial time, it can also do it without the ciphertext (in a polynomial time).

This immediately implies that **deterministic encryption** is not semantically secure. To make it secure, we must go **probabilistic**. Using a nonce for the key generation.

Nonce

A *nonce* is a number used once. Most likely randomly generated.

1.6.6 INDististinguishability

Let's play a bit.