Introduction to cryptography

1. Historical ciphers and general principles

Gilles Van Assche
Olivier Markowitch

INFO-F-405
Université Libre de Bruxelles
2021-2022

What is cryptography?

Set of techniques to ensure:

- the confidentiality and/or
- the integrity of a message, of a transmission channel

Small part of security

- conceptually advanced
- rarely the weakest link

What is cryptography?

Set of techniques to ensure:

- the confidentiality and/or
- the integrity of a message, of a transmission channel

Small part of security

- conceptually advanced
- rarely the weakest link

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*.

Shift encryption scheme

$$M = C = K = \mathbb{Z}_{26}, 0 \leq k \leq 25 \text{ and } x, y \in \mathbb{Z}_{26}$$

Encryption: $E_k(x) = x + k \mod 26$

Decryption: $D_k(y) = y - k \mod 26$

Example: with $k = 3$, the plaintext CAESAR is ciphered in FDHVDU

Cryptanalysis?

Mono-alphabetic substitution

$M = C = \mathbb{Z}_{26}$, K is the set of permutations on $\{0, \ldots, 25\}$
For each permutation $k \in K$ we have: $\Rightarrow 26!$ Permutations

$$E_k(x) = k(x)$$

$$D_k(y) = k^{-1}(y)$$

where $x, y \in \mathbb{Z}_{26}$ and k^{-1} being the inverse permutation of k

Cryptanalysis? (brute force: $26! > 4 \cdot 10^{26}$ possible keys)

$\Rightarrow$ a key k corresponds to a remap of the alphabet

Mono-alphabetic substitution

a	b	c	d	e	f	g	h	i	j	k	l	m
X	N	Y	A	H	P	O	G	Z	Q	W	B	T

n	o	p	q	r	s	t	u	v	w	x	y	z
S	F	L	R	C	V	M	U	E	K	J	D	I

Remap of the alphabet

Encryption

A	B	C	D	E	F	G	H	I	J	K	L	M
d	l	r	y	v	o	h	e	z	x	w	p	t

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	g	f	j	q	n	m	u	s	k	a	c	i

Decryption

chiffrementparpermutation

$\Downarrow$

YGZPPCHTHSMLXCLHCTUMXMZFS

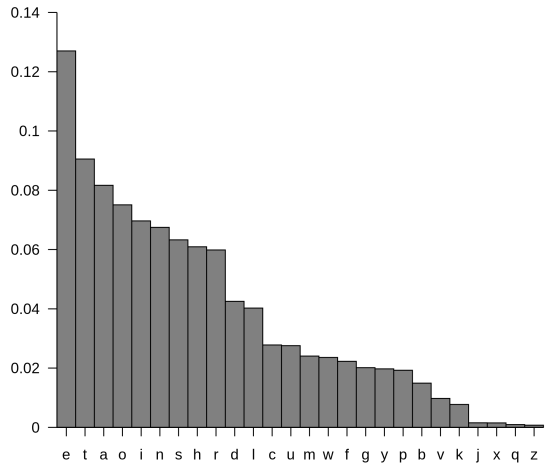
(could appeared difficult to crack (26! attempts for brute-force methods), but tricks can make it easy to crack!!)

Mono-alphabetic substitution

Cryptanalysis:

- letters frequencies in the ciphertexts are the same as in the plaintexts
- use of frequencies tables based on the language of the plaintext (letter, digrams, trigrams, ...)

Mono-alphabetic substitution



Frequency of invidual letters in English

Mono-alphabetic substitution

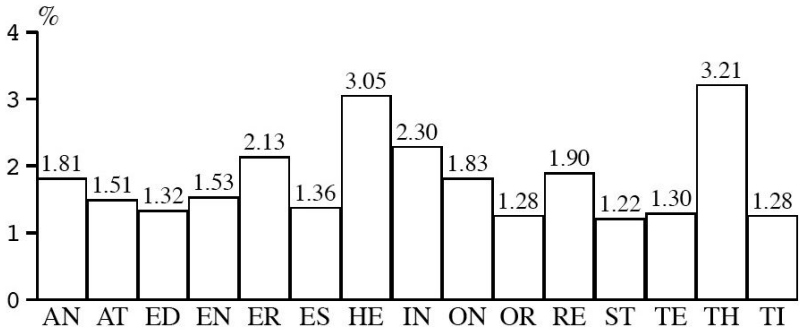


Figure 7.6: *Frequency of 15 common digrams in English text.*

Poly-alphabetic substitution

Encryption of blocs composed of t symbols

- E consists in all the sets of t permutations of the symbols
- each key $k \in K$ define a set of t permutations $(p_1, \ldots, p_t)$
- the plaintext $x = x_1 \ldots x_t$ is encrypted on the basis of the key k:

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

- the decryption key k' define the set of the t corresponding inverse permutations: $(p_1^{-1}, \ldots p_t^{-1})$

Vigenère cipher

Blaise de Vigenère (1523-1596)

Note: The encryption scheme was described by Blaise de Vigenère but actually probably first designed by Giovan Battista Bellaso.

Vigenère cipher

Let $M = C = (\mathbb{Z}_{26})^*$ and $K = (\mathbb{Z}_{26})^t$ for some $t > 0$.
Given a randomly-chosen key $k = (k_0, \ldots, k_{t-1})$:

$$E_k(m) = E_k(m_0, \ldots, m_{|m|-1}) = (m_i + k_{i \bmod t})_{0 \le i \le |m|-1}$$

$$D_k(c) = D_k(c_0, \ldots, c_{|c|-1}) = (c_i - k_{i \bmod t})_{0 \le i \le |c|-1}$$

with $m_i, c_i \in \mathbb{Z}_{26}$ and all the operations are computed in $\mathbb{Z}_{26}$.

In practice the key can be a string of t characters (converted in numbers within the ciphering process). There are 26^t possible keys.

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

Cryptanalysis of the Vigenère cipher

First, suppose the key length t is known.

- Group the ciphertext letters according to their position mod t. We now have t independent shift ciphers.
- For each group, brute-force the corresponding key letter using the single-letter distribution.

Then, how to find t?

- Lazy approach: test with $t = 1$ then $t = 2$, ..., until the attack succeeds.
- Use the index of coincidence.

Cryptanalysis of the Vigenère cipher

First, suppose the key length t is known.

- Group the ciphertext letters according to their position mod t. We now have t independent shift ciphers.
- For each group, brute-force the corresponding key letter using the single-letter distribution.

Simple shift encryption scheme

Then, how to find t?

- Lazy approach: test with $t = 1$ then $t = 2$, ..., until the attack succeeds.
- Use the index of coincidence.

Cryptanalysis of the Vigenère cipher

First, suppose the key length t is known.

- Group the ciphertext letters according to their position mod t. We now have t independent shift ciphers.
- For each group, brute-force the corresponding key letter using the single-letter distribution.

Then, how to find t?

- Lazy approach: test with $t = 1$ then $t = 2$, ..., until the attack succeeds.
- Use the index of coincidence.

Cryptanalysis of the Vigenère cipher

If we draw two random letters from a text, say x and x', there is a collision if $x = x'$. In English, the estimated probability of collision is:

$$\Pr[x = x'] = \sum_x p_x^2 \approx 0.065 > \frac{1}{26},$$

where p_x is the frequency of the x-th letter.

This remains valid if the letters are transposed.

So, we compute the cross-correlation:

$$C_s = \Pr[y_i = y_{i+s}].$$

If s is a multiple of t, then C_s should be about 0.065. Otherwise, it should be about $\frac{1}{26}$.

Cryptanalysis of the Vigenère cipher

If we draw two random letters from a text, say x and x', there is a collision if $x = x'$. In English, the estimated probability of collision is:

$$\Pr[x = x'] = \sum_x p_x^2 \approx 0.065 > \frac{1}{26},$$

where p_x is the frequency of the x-th letter.

This remains valid if the letters are transposed.

So, we compute the cross-correlation:

$$C_s = \Pr[y_i = y_{i+s}].$$

If s is a multiple of t, then C_s should be about 0.065. Otherwise, it should be about $\frac{1}{26}$.

Cryptanalysis of the Vigenère cipher

If we draw two random letters from a text, say x and x', there is a collision if $x = x'$. In English, the estimated probability of collision is:

$$\Pr[x = x'] = \sum_x p_x^2 \approx 0.065 > \frac{1}{26},$$

where p_x is the frequency of the x-th letter.

This remains valid if the letters are transposed.

So, we compute the cross-correlation:

$$C_s = \Pr[y_i = y_{i+s}].$$

If s is a multiple of t, then C_s should be about 0.065. Otherwise, it should be about $\frac{1}{26}$.

Binary Vigenère cipher

To work on any binary string, we can rewrite the Vigenère cipher as follows. Let $M = C = (\mathbb{Z}_2)^*$ and $K = (\mathbb{Z}_2)^t$ for some $t > 0$.
Given a randomly-chosen key $k = (k_0, \ldots, k_{t-1})$:

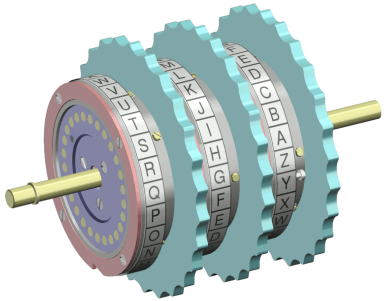
$$E_k(m) = E_k(m_0, \ldots, m_{|m|-1}) = (m_i + k_{i \bmod t})_{0 \le i \le |m|-1}$$

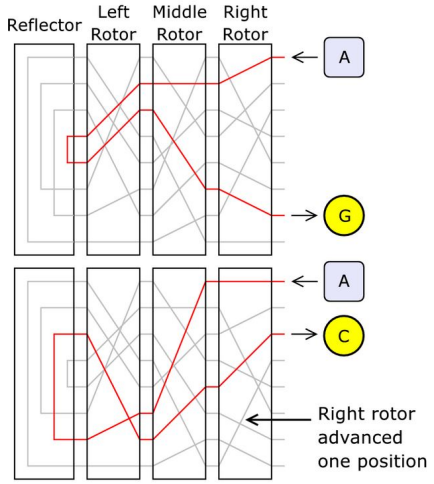
$$D_k(c) = D_k(c_0, \ldots, c_{|c|-1}) = (c_i + k_{i \bmod t})_{0 \le i \le |c|-1}$$

with $m_i, c_i \in \mathbb{Z}_2$ and all the operations are computed **modulo 2**.

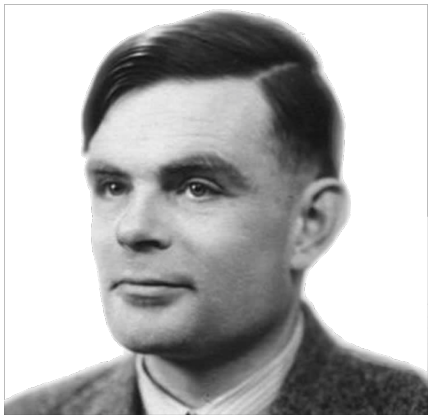
Enigma

Enigma

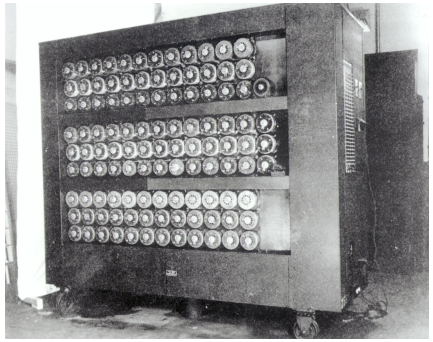




Enigma



Alan Turing (1912-1954)



Bombe

Perfect secrecy

Suppose an adversary that knows everything expect the key k exchanged by Alice and Bob to encrypt messages.

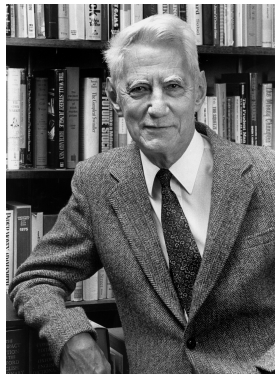
Perfect secrecy = unconditional security

An encryption schemes satisfies *perfect secrecy* if the ciphertexts reveal nothing about the corresponding plaintexts, even if the adversary has *unlimited computational power*.
Formally, for any two messages m_1, m_2 in the message space M and every ciphertext $c \in C$, the scheme must ensure

$$\Pr[\mathrm{Enc}_k(m_1) = c] = \Pr[\mathrm{Enc}_k(m_2) = c],$$

where both probabilities are taken over the choice of k in the key space K.

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!

The one-time pad

One-time pad (OTP) also known as Vernam scheme (1917)

Gilbert Vernam
(1890-1960)

Joseph Mauborgne
(1881-1971)

Perhaps already initially designed around 1880 by Frank Miller

The one-time pad

Let $M = C = (\mathbb{Z}_2)^t$ and $K = (\mathbb{Z}_2)^t$ for some $t > 0$.
Given a randomly-chosen key $k = (k_0, \ldots, k_{t-1})$:

$$E_k(m) = E_k(m_0, \ldots, m_{t-1}) = (m_i + k_i)_{0 \le i \le t-1}$$

$$D_k(c) = D_k(c_0, \ldots, c_{t-1}) = (c_i + k_i)_{0 \le i \le t-1}$$

with $m_i, c_i \in \mathbb{Z}_2$ and all the operations are computed **modulo 2**.

For any $m, c \in (\mathbb{Z}_2)^t$, we see that:

$$\begin{aligned} \Pr[E_k(m) = c] &= \Pr[m \oplus k = c] \\ &= \Pr[k = m \oplus c] \\ &= 2^{-t}, \end{aligned}$$

where $\oplus$ denotes the element-wise addition in $(\mathbb{Z}_2)^t$.

The one-time pad

Let $M = C = (\mathbb{Z}_2)^t$ and $K = (\mathbb{Z}_2)^t$ for some $t > 0$.
Given a randomly-chosen key $k = (k_0, \ldots, k_{t-1})$:

$$E_k(m) = E_k(m_0, \ldots, m_{t-1}) = (m_i + k_i)_{0 \le i \le t-1}$$

$$D_k(c) = D_k(c_0, \ldots, c_{t-1}) = (c_i + k_i)_{0 \le i \le t-1}$$

with $m_i, c_i \in \mathbb{Z}_2$ and all the operations are computed **modulo 2**.

For any $m, c \in (\mathbb{Z}_2)^t$, we see that:

Bit wise operation

$$\begin{aligned} \Pr[E_k(m) = c] &= \Pr[m \oplus k = c] \\ &= \Pr[k = m \oplus c] \\ &= 2^{-t}, \end{aligned}$$

$m \oplus m \oplus k = m \oplus c$, $0 \quad k = m \oplus c$

$\hookrightarrow \left(\frac{1}{2}\right)^t$

⇒ with the m and c, we find the corresp. bit of the key

Message ⊕ key = cipher. Message ⊕ Cipher = key

Nb of bits used for ..

where $\oplus$ denotes the element-wise addition in $(\mathbb{Z}_2)^t$.

What if the key is reused?

If $c_1 = m_1 \oplus k$ and $c_2 = m_2 \oplus k$, then

This gives us information about the key

$$c_1 \oplus c_2 = m_1 \oplus m_2.$$

For every bit position, if we add the 2 ciphert., it is the same as $m_1 + m_2$.

If we reuse the secret key, it breaks the cycle, no secrecy !

One-time pad vs Vigenère

- Vigenère is easy to break
- OTP is secrecy

In what do the one-time pad and the binary Vigenère cipher differ?

- In the OTP, the key size is equal to the plaintext size.
- In the OTP, the key may not be reused ("one-time").

(In Vigenère, we can use a small key and repeat)

Despite their similarity, they stand at two extremes:

- The OTP is secure, even against an adversary that has unlimited computational power.
- The Vigenère cipher is in general easy to break.

One-time pad vs Vigenère

In what do the one-time pad and the binary Vigenère cipher differ?

- In the OTP, the key size is equal to the plaintext size.
- In the OTP, the key may not be reused ("one-time").

Despite their similarity, they stand at two extremes:

- The OTP is secure, even against an adversary that has unlimited computational power.
- The Vigenère cipher is in general easy to break.

Computational security

- Perfect secrecy requires that absolutely no information about an encrypted message is leaked, even to an eavesdropper with unlimited computational power.
- Perfect secrecy requires secret keys as long as the messages, which is not convenient.
- In practice, an encryption scheme is still secure if it leaks only a tiny amount of information to eavesdroppers with bounded computational power.
- When the security takes into account the computational limits of the attack and allows a very small probability of failure, we talk about 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.

Computational security

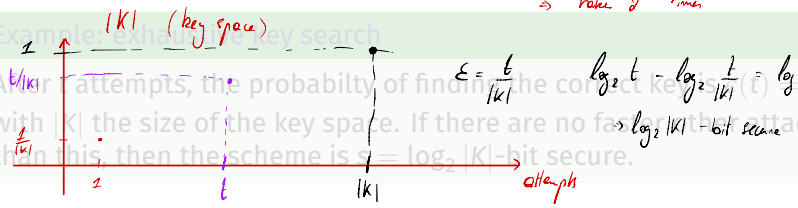
Computational security

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 $\log_2 t - \log_2 \epsilon(t) \geq s$.

128 bits of security ⇒ take 2^{128} times



$\epsilon = \frac{t}{|K|}$

$\log_2 t - \log_2 \frac{t}{|K|} = \log_2 |K|$

$\Rightarrow \log_2 |K|$-bit secure

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.

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.

Computational power

If one operation is computed in one cycle, on a 4 GHz processor (that executes $4 \cdot 10^9$ cycles per second):

- 2^{60} operations are computed in $\frac{2^{60}}{4 \cdot 10^9}$ seconds $\approx$ 9 years
- 2^{64} operations are computed in $\frac{2^{64}}{4 \cdot 10^9}$ seconds $\approx$ 146 years
- 2^{80} operations are computed in $\frac{2^{80}}{4 \cdot 10^9}$ seconds $\approx 9.5 \times 10^9$ years

More computational power (and energy!)

Now assume now a huge bot of 10^9 computers, each capable of 10^9 operations per second.

2^{100}	40 000 years
2^{120}	42×10^9 years, or about 3 times the age of the universe
2^{140}	About 3 million times the age of the universe
2^{160}	...

Considering a computer consuming only the minimum energy required by the laws of thermodynamics, at the average temperature of the universe, just counting from 1 to 2^{160} requires more energy than the entire world's electrical consumption during 670 000 years.

Cryptanalysis and peer review

Big gap between cryptography and practical cryptosystems

Important warning: Except for the one-time pad, none of the schemes we will see offer perfect secrecy, nor can their security be mathematically proven. Cryptanalysis and peer review are the only ways one can gain confidence in the security of a scheme.

Security by obscurity is usually a bad idea because it is difficult to evaluate the scheme's intrinsic security. In other words, the algorithm should be public, and only the key remains secret. This is according to Kerckhoffs' principles.

Kerckhoffs' principles

Auguste Kerckhoffs (1835-1903)

Kerckhoffs' principles

1. The system must be substantially, if not mathematically, undecipherable;
2. The system must not require secrecy and can be stolen by the enemy without causing trouble;
3. It must be easy to communicate and retain the key without the aid of written notes, it must also be easy to change or modify the key at the discretion of the correspondents;
4. The system ought to be compatible with telegraph communication;
5. The system must be portable, and its use must not require more than one person;
6. Finally, given the circumstances in which such system is applied, it must be easy to use and must neither stress the mind or require the knowledge of a long series of rules.

Auguste KERCKHOFFS: *La cryptographie militaire* du Journal des sciences militaires, 1883

`https://www.petitcolas.net/kerckhoffs/index.html`

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

↳ That's completely about design

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.

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 → Can feed to the system with the plaintext he wants
- known plaintexts (and corresponding ciphertexts)
- chosen plaintexts (and corresponding ciphertexts)

} But we don't even want him to have that!

- 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.

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.

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 $\log_2(t + d) - \log_2 \epsilon(t, d) \geq s$.

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 $\log_2(t + d) - \log_2 \epsilon(t, d) \geq s$.

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.

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.

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 $c = \mathsf{Enc}_{k_\mathsf{E}}(m)$. 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 $m' = \mathsf{Dec}_{k_\mathsf{D}}(c)$.

Semantic security

An encryption scheme is semantically secure if whatever a passive adversary can compute in expected polynomial time about the plaintext given the ciphertext, it can also compute in expected polynomial time without the ciphertext.

Implications of semantic security

Problem with deterministic encryption: Suppose that Alice and Bob exchange many encrypted plaintexts by using the same key k. Suppose that among all the exchanged encrypted plaintexts, two of them are the encryption of the same plaintext m. From the previous definition it appears that if the used encryption scheme is such that the encryption of m with the same key k results in the same ciphertext c, the encryption scheme cannot be considered as semantically secure. Therefore, if the encryption scheme is deterministic, it cannot be semantically secure

A possible solution: Encryption scheme must be randomized. In this case, we talk about **probabilistic encryption**.

Another possible solution: Use a nonce! (Symmetric crypto only.)

In both cases, when the same plaintext is encrypted multiple times, different ciphertexts are produced.

Implications of semantic security

Problem with deterministic encryption: Suppose that Alice and Bob exchange many encrypted plaintexts by using the same key k. Suppose that among all the exchanged encrypted plaintexts, two of them are the encryption of the same plaintext m. From the previous definition it appears that if the used encryption scheme is such that the encryption of m with the same key k results in the same ciphertext c, the encryption scheme cannot be considered as semantically secure. Therefore, if the encryption scheme is deterministic, it cannot be semantically secure

A possible solution: Encryption scheme must be randomized. In this case, we talk about **probabilistic encryption**.

Another possible solution: Use a nonce! (Symmetric crypto only.)

In both cases, when the same plaintext is encrypted multiple times, different ciphertexts are produced.

Implications of semantic security

Problem with deterministic encryption: Suppose that Alice and Bob exchange many encrypted plaintexts by using the same key k. Suppose that among all the exchanged encrypted plaintexts, two of them are the encryption of the same plaintext m. From the previous definition it appears that if the used encryption scheme is such that the encryption of m with the same key k results in the same ciphertext c, the encryption scheme cannot be considered as semantically secure. Therefore, if the encryption scheme is deterministic, it cannot be semantically secure

A possible solution: Encryption scheme must be randomized. In this case, we talk about **probabilistic encryption**.

Another possible solution: Use a nonce! (Symmetric crypto only.)

In both cases, when the same plaintext is encrypted multiple times, different ciphertexts are produced.

Implications of semantic security

Problem with deterministic encryption: Suppose that Alice and Bob exchange many encrypted plaintexts by using the same key k. Suppose that among all the exchanged encrypted plaintexts, two of them are the encryption of the same plaintext m. From the previous definition it appears that if the used encryption scheme is such that the encryption of m with the same key k results in the same ciphertext c, the encryption scheme cannot be considered as semantically secure. Therefore, if the encryption scheme is deterministic, it cannot be semantically secure

A possible solution: Encryption scheme must be randomized. In this case, we talk about **probabilistic encryption**.

Another possible solution: Use a nonce! (Symmetric crypto only.)

In both cases, when the same plaintext is encrypted multiple times, different ciphertexts are produced.

INDistinguishability

A scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is **IND-secure** if no adversary can win the following game for more than a negligeable advantage.

- Challenger generates a key (pair) $k \leftarrow \mathsf{Gen}()$
- Adversary chooses two plaintexts $m_0, m_1 \in M$ with $|m_0| = |m_1|$
- Challenger randomly chooses $b \leftarrow_R \{0, 1\}$, encrypts m_b and sends $c = \mathsf{Enc}_k(m_b)$ to the adversary
- Adversary guesses b' which plaintext was encrypted
- Adversary wins if $b' = b$ (Advantage: $\epsilon = |\Pr[\text{win}] - \frac{1}{2}|$.)

Advantage = P_r[win | real attack] − P_r[win | random attack] $\frac{1}{2}$

Advantage of the adversary

Problem (in symmetric crypto):

This only addresses the security of a single encryption.

It needs the challenger to encrypt, no public key

Adv = diff of chance of winning in a random attack vs Random att.

⇒ if Adv. = 0, no advantage really // we want it to be as small as possible

IND-CPA (chosen plaintexts)

Chosen-Plain. Text attacks

A scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is **IND-CPA-secure** if no adversary can win the following game for more than a negligeable advantage.

- Challenger generates a key (pair) $k \leftarrow \mathsf{Gen}()$
- Adversary queries Enc_k with plaintexts of his choice
- Adversary chooses two plaintexts $m_0, m_1 \in M$ with $|m_0| = |m_1|$
- Challenger randomly chooses $b \leftarrow_R \{0,1\}$, encrypts m_b and sends $c = \mathsf{Enc}_k(m_b)$ to the adversary
- Adversary queries Enc_k with plaintexts of his choice
- Adversary guesses b' which plaintext was encrypted
- Adversary wins if $b' = b$ (Advantage: $\epsilon = \left|\Pr[\text{win}] - \frac{1}{2}\right|$.)

not limited! # of parameter d

For public-key crypto: IND and IND-CPA can both be done offline because attacker has access to the public key.

IND-CCA (chosen plaintexts and chiphertexts)

A scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is **IND-CCA-secure** if no adversary can win the following game for more than a negligeable advantage.

- Challenger generates a key (pair) $k \leftarrow \mathsf{Gen}()$
- Adversary queries Enc_k with plaintexts of his choice and Dec_k with ciphertexts of his choice
- Adversary chooses two plaintexts $m_0, m_1 \in M$ with $|m_0| = |m_1|$
- Challenger randomly chooses $b \leftarrow_R \{0,1\}$, encrypts m_b and sends $c = \mathsf{Enc}_k(m_b)$ to the adversary
- Adversary queries Enc_k with plaintexts of his choice and Dec_k with ciphertexts of his choice except c
- Adversary guesses b' which plaintext was encrypted
- Adversary wins if $b' = b$ (Advantage: $\epsilon = \left|\Pr[\text{win}] - \frac{1}{2}\right|$.)

IND-CPA or IND-CCA security strength

Security strength for IND-CPA (resp. IND-CCA)

A scheme is (t, d, ϵ)-IND-CPA (resp. IND-CCA) secure if any adversary running for time at most t and having access to d data, succeeds in winning the IND-CPA (resp. IND-CCA) game with advantage at most ϵ.

What is counted towards the online complexity d?

	Symmetric	Asymmetric
IND-CPA	Enc_k	-
IND-CCA	Enc_k + Dec_k	Dec_k

Diversification using a nonce

In symmetric crypto, encryption is diversified at each use:

- Encryption takes an extra parameter d
  - diversifier d is **public**
  - it must be a *nonce*, i.e., **unique** at each encryption (for a given key)
- Decryption needs d as input to be able to decrypt

Often, d is implicit (e.g., packet number). Otherwise, it must be explicitly communicated.

Symmetric encryption with diversification

A symmetric-key encryption scheme with diversification is a triple of algorithms $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, a diversifier space D and a plaintext space M.

- Gen is a probabilistic algorithm that outputs a secret key k from the key space K.
- Enc takes as input a secret key k, a diversifier $d \in D$ and message $m \in M$, and outputs ciphertext $c = \mathsf{Enc}_k(d, m)$. The range of Enc is the ciphertext space C.
- Dec takes as input a secret key k, a diversifier $d \in D$ and ciphertext $c \in C$ and output a plaintext $m' = \mathsf{Dec}_k(d, c)$.

Correctness: $\forall k \in K, d \in D, m \in M$, we must have $\mathsf{Dec}_k(d, \mathsf{Enc}_k(d, m)) = m$.

IND-CPA (chosen plaintext, chosen diversifier)

A scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is **IND-CPA-secure** if no adversary can win the following game for more than a negligeable advantage.

1. Challenger generates a key (pair) $k \leftarrow \mathsf{Gen}()$
2. Adversary queries Enc_k with (d, m) of his choice
3. Adversary chooses d and two plaintexts $m_0, m_1 \in M$ with $|m_0| = |m_1|$
4. Challenger randomly chooses $b \leftarrow_R \{0, 1\}$, encrypts m_b and sends $c = \mathsf{Enc}_k(d, m_b)$ to the adversary
5. Adversary queries Enc_k with (d, m) of his choice
6. Adversary guesses b' which plaintext was encrypted
7. Adversary wins if $b' = b$ (Advantage: $\epsilon = \left|\Pr[\text{win}] - \frac{1}{2}\right|$.)

The adversary must respect that d **is a nonce**! The values of d used in steps 2, 3 and 5 must all be different.

To what does an authentication scheme must resist?

An adversary can have the following **goals**: (all things to protect)

- recovery of the (secret or private) key
- a forgery, i.e., (message, tag) not from the legitimate party
  - universal forgery
  - selective forgery
  - existential forgery
- a property that distinguishes the scheme from ideal (we expect MAC to be random)

(universal forgery → able to submit a message that don't come from the authentic party and it is still verified)

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

- known messages (and corresponding tags)
- chosen messages (and corresponding tags)

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 as few as possible known messages. A designer will be happy if no cryptanalyst was able to show a disitinguisher even with chosen messages.

To what does an authentication scheme must resist?

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

- recovery of the (secret or private) key
- a forgery, i.e., (message, tag) not from the legitimate party
  - universal forgery
  - selective forgery
  - existential forgery
- a property that distinguishes the scheme from ideal

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

- known messages (and corresponding tags) (make a new message or modify messages)
- chosen messages (and corresponding tags) (→ have the device send messages of his choice)

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 as few as possible known messages. A designer will be happy if no cryptanalyst was able to show a disitinguisher even with chosen messages.

To what does an authentication scheme must resist?

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

- recovery of the (secret or private) key
- a forgery, i.e., (message, tag) not from the legitimate party
  - universal forgery
  - selective forgery
  - existential forgery
- a property that distinguishes the scheme from ideal

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

- known messages (and corresponding tags)
- chosen messages (and corresponding tags)

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 as few as possible known messages. A designer will be happy if no cryptanalyst was able to show a disitinguisher even with chosen messages.

Types of forgeries

- Universal forgery: the attack must be able to work for any message, possibly chosen by a challenger
- Selective forgery: the adversary chooses the message beforehand
- Existential forgery: the message content is irrelevant, the adversary can choose it adaptively just to make the attack work

Weakest goal

⇒ No meaningful messages !

Taxonomy of attacks

As for encryption, 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).

Chosen...

Amount of data needed

Example: random tag guessing

The random tag guessing is a **universal forgery attack** that submits d random (message, tag) pairs. It requires **no known message** and takes **online complexity** d for a **success probability** $\frac{d}{2^n}$, assuming that the tag length is n bits. (Here t is negligible.)

Example: exhaustive key search

The exhaustive key search is a **key recovery attack** that requires $d = 1$ **known** (**message**, **tag**) **pair** and takes **offline complexity** $t = |K|$ for a **success probability** $\epsilon = 1$.

Taxonomy of attacks

As for encryption, 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: random tag guessing

The random tag guessing is a **universal forgery attack** that submits d random (message, tag) pairs. It requires **no known message** and takes **online complexity** d for a **success probability** $\frac{d}{2^n}$, assuming that the tag length is n bits. (Here t is negligible.)

Example: exhaustive key search

The exhaustive key search is a **key recovery attack** that requires $d = 1$ **known** (**message**, **tag**) **pair** and takes **offline complexity** $t = |K|$ for a **success probability** $\epsilon = 1$.

Taxonomy of attacks

As for encryption, 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: random tag guessing

The random tag guessing is a **universal forgery attack** that submits d random (message, tag) pairs. It requires **no known message** and takes **online complexity** d for a **success probability** $\frac{d}{2^n}$, assuming that the tag length is n bits. (Here t is negligible.)

Example: exhaustive key search

The exhaustive key search is a **key recovery attack** that requires $d = 1$ **known** (**message**, **tag**) **pair** and takes **offline complexity** $t = |K|$ for a **success probability** $\epsilon = 1$.

Formal definition of an authentication scheme

An authentication scheme is a triple of algorithms $\mathcal{T} = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ and a message space M.

Gen	is a probabilistic algorithm that outputs a secret (or private) key k_A from the key space K. In asymmetric cryptography, it publishes the corresponding public key k_V.
Tag	takes as input a secret/private key k_A and message $m \in M$, and outputs tag $t = \mathsf{Tag}_{k_A}(m)$. The range of Tag is the tag space T.
Ver	is a deterministic algorithm that takes as input a secret/public key k_V, a message $m \in M$ and a tag $t \in T$ and output either m (if the tag is valid) or $\perp$ (otherwise).

EU-CMA: Existential unforgeability, chosen messages

A scheme $\mathcal{T} = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ is **EU-CMA-secure** if no adversary can win the following game for more than a negligeable probability.

- Challenger generates a key (pair) $k \leftarrow \mathsf{Gen}()$
- Adversary queries Tag_k with messages of his choice
- Adversary produces a (message, tag) pair, with a message not yet queried
- Adversary wins if $\mathsf{Ver}_k(\text{message}, \text{tag}) \neq \perp$
  (Advantage: $\epsilon = \Pr[\text{win}]$.)

What cryptography does (not) cover

- Attacks outside the model
- Generic attacks
- Shortcut attacks
- Implementation attacks
- Human attacks

Not covered: attacks outside the model

Examples:

- no secrecy without encryption
- forgery when there is no authentication
- traffic analysis → Crypto protects the content, but does not hide the existence
- length of messages

Covered: generic attacks

2 big categories: generic & shortcut

Attacks that work independently of the underlying primitives
(mostly predictable)

Examples:

- exhaustive key search
- attacks on mode of operation (see next chapter)

Covered: shortcut attacks

Attacks that break a primitive more easily than claimed
(mostly unpredictable)

Examples:

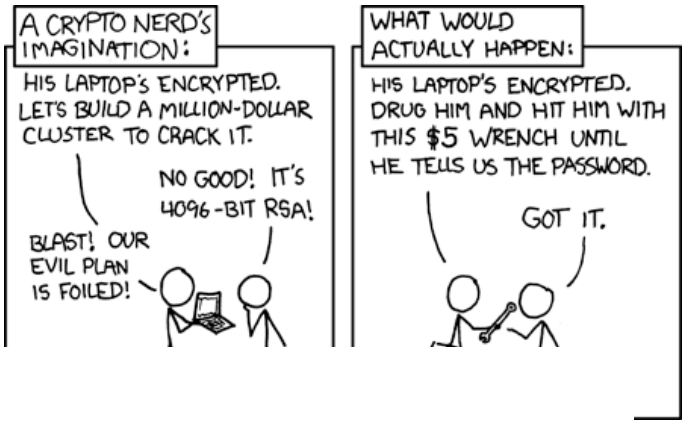
- RC4, DES (see next chapter)
- advances in factorting or discrete log (see Chapter 4)

Not covered: implementation attacks

Attacks that exploit flaws in implementations

- bugs
- assumptions not satisfied
- side-channel attacks
- fault attacks

Not covered: human attacks



Symmetric crypto

msg; they both know the secret

- layers
- Primitives
- Operational mode

ive anything on it's own: on