1. Historical ciphers and general principles

#chap1
Slides.

1.1 What is cryptography?

#cryptography #definition
It is a set of techniques to ensure the **confidentiality** and/or the **integrity** of a message, of a transmission channel.
It has also a small part in security: it is conceptually advanced and is rarely the weakest link.

1.2 Confidentiality

Encryption #encryption

- *plaintext* $\implies$ ciphertext
- under key $k_\epsilon \in K$

Decryption #decryption

- ciphertext $\implies$ *plaintext*
- Under key $k_d \in K$

In #symmetric cryptography: $k_E = k_D$ is the **secret key**.
In #asymmetric cryptography: k_E is public and k_D is private.

1.3 Authenticity

Authentication #authentication

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

Verification #verification

- (message, **tag**) $=>$ message
- 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*. #signature

1.4 Historical ciphers

1.4.1 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

Code Caesar.

1.4.2 Mono-alphabetic substitution

#alphabetic #encryption #method

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

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

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

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

The default of this is that with probabilities we can recognize the code. n fact, the letter frequencies in the ciphertext are the same as in the plaintexts. The use of frequencies tables based on the language of the plaintext makes the decryption very easy. For example the e is the most used letter in the language.

1.4.3 Poly-alphabetic substitution

Encryption of blocs composed of t symbols

- E consists in all the sets of t permutations of the symbols
- each key $k \in K$ defines 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 $(p_1^{-1}, \ldots, p_t^{-1})$

1.4.4 Vigenère cipher

#vigenere #encryption #method

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

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

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

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

Use number = letter and shift (add) the second element of the number to have the encryption.

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

$\Downarrow$

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

#brute-force

First we suppose the key length t is known.

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

To find t, use the **lazy approach** #lazy-approach : test with $t = 1, t = 2, \ldots$ until the attack succeeds.

#probability #cryptanalysis

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.
$\rightarrow$ this remains valid if the letters are transposed.

So when 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

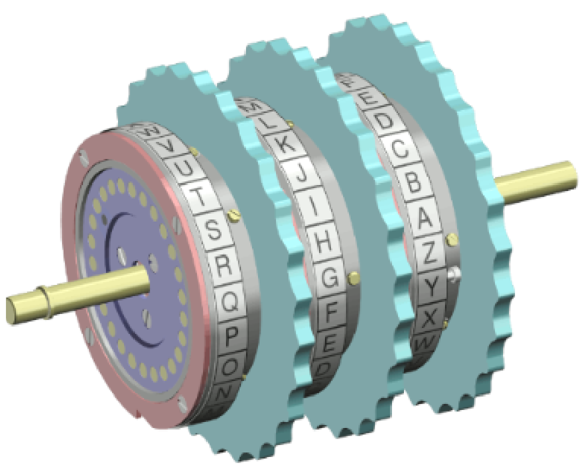
#binary #vigenere #method

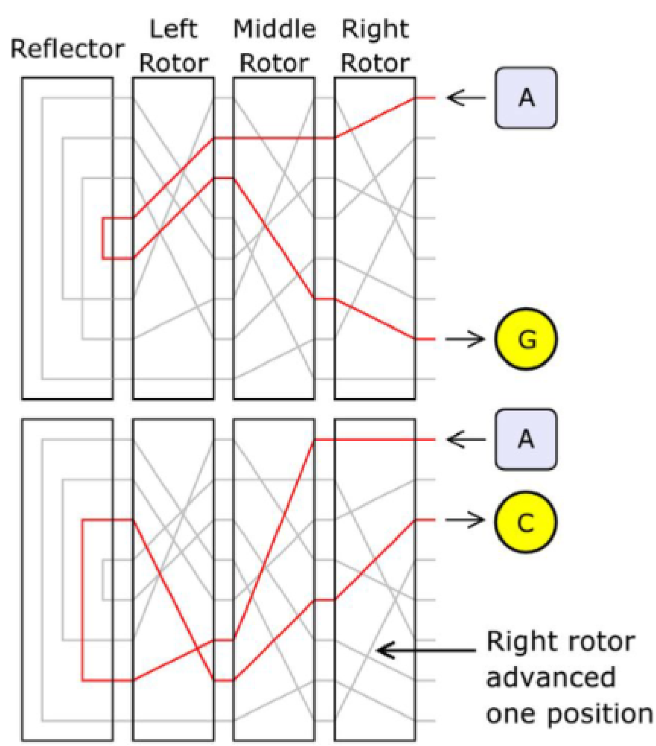
> To work on any binary string, we can rewrite the Vigenère cipher as follows. Let $M = C = (\mathbb{Z}_2)^*$ and $K = (\mathbb{Z}_2)^t$ for some $t > 0$.
> Given a randomly-chosen key $k = (k_0, \dots, k_{t-1})$:
>
> $$E_k(m) = E_k(m_0, \dots, m_{|m|-1}) = (m_i + k_{i \bmod t})_{0 \leq i \leq |m|-1}$$
>
> $$D_k(c) = D_k(c_0, \dots, c_{|c|-1}) = (c_i + k_{i \bmod t})_{0 \leq i \leq |c|-1}$$
>
> with $m_i, c_i \in \mathbb{Z}_2$ and all the operations are computed **modulo 2**.

Vigenère is not used anymore.

Enigma

By Alan Turing.
Used during WWII to decrypt the messages from the German army.
It worked with 2 motors and a reflector. There was an electrical impulse corresponding to the letter.





1.5 Perfect secrecy vs computational security

Slides.

1.5.1 Perfect secrecy

We have c = **ciphertext** and m = **message (plaintext)**, the #perfect #secrecy is satisfied if the *hacker that has the ciphertext and the plaintext can not find the encryption method.*

> **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
>
> $$\Pr[\text{Enc}_k(m_1) = c] = \Pr[\text{Enc}_k(m_2) = c],$$
>
> where both probabilities are taken over the choice of k in the key space K.

#definition
The objective of perfect secrecy is to ensure that the hacker has the same probability to decrypt any message.

$$Pr[Enc_k(m_1) = c] = Pr[Enc_k(m_2) = c]$$

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

Claude Shannon showed that in order to achieve perfect secrecy, we need a non practical algorithm. The **entropy** of the key is at least the entropy of the plaintext.

$$H(K) \geq H(M)$$

This means that the secret key must be at least as long as the plaintext and it may not be reused! The **key must be used only once**! If the key is reused, we then have information for both messages m_1 and m_2.

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

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

We can achieve this. The answer is the **one-time pad**.

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 $k = (k_0, \ldots, k_{t-1})$:

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

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

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 $(\mathbb{Z}_2)^t$.

If I'm given the plaintext and the cipher text, there is only one key possible.
Here t is the length of the message. #OTP #method

One-time pad vs Vigenère

#otp #vigenere #vs
In what do the one-time pad ad 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**!
- In the OTP, the **key is secret and uniformly distributed**

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 general *easy to break*.

1.5.2 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 $(t, \epsilon)-$**secure** if any adversary running for time at most t, succeeds in breaking the scheme with probability at most ϵ.

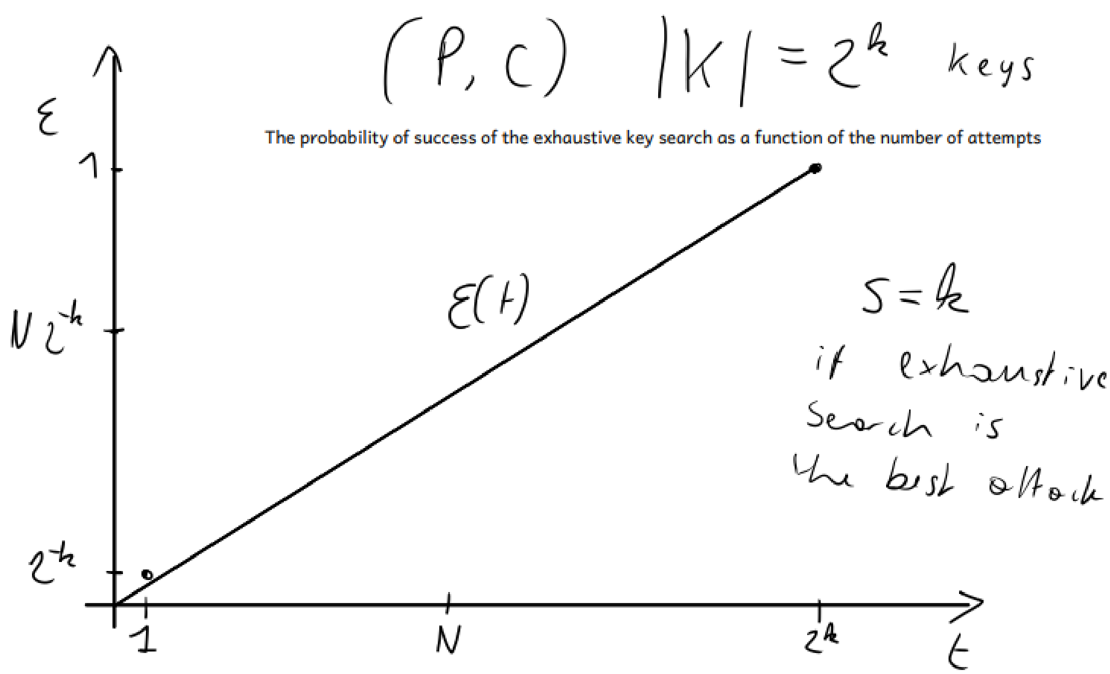
Security strength #security-strength
We say that a scheme is **s-bit secure** if, for all t, the scheme is $(t, \epsilon)-$secure and $\log_2 t - \log_2 \epsilon \geq s$.

An example is the **exhaustive key search**.

Exhaustive key search

After t attempts, the probability 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 $s = \log_2 t \geq s$-bit secure.

The most you try, the most the probability to find grows in a **linear** way.



The probability of success of the exhaustive key search as a function of the number of attempts

Computational power

If you dont know, you will wait a long long time.

Cryptanalysis and peer review

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 Kerckhoff's principles.

1.6 Security principles

Kerckhoff's principles

All the #Kerckhoff 's 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

1.7 Security definitions

Offline and online complexities

#computational-security of a scheme:
A scheme is $(t, d, \epsilon)-$secure if any adversary running for time at most t and having access to d data, succeeds in breaking the scheme with a probability at most ϵ.

#Offline-complexity : t computational effort or resources required by an attacker before interacting with the system they are attempting to compromise.
#Online-complexity : d It involves computations performed in real-time or on-the-fly.

#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

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

with s the number of bits wanted for security.

1.7.1 Encryption scheme

An #encryption-scheme must resist to the attacks of an **adversary**.
The #adversary can have the following **goals**:

- The recovery of the (secret/private) key
- The 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
- Chosen plaintexts
- 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 crypt-analyst was able to show a disitinguisher even with chosen plaintexts and ciphertexts.

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)*.

#Exhaustive-key-search example:
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** $\epsilon = 1$.
$\rightarrow$ Depending on the relative size of the plaintext and the key, this may require more than one pair to avoid multiple key candidates.

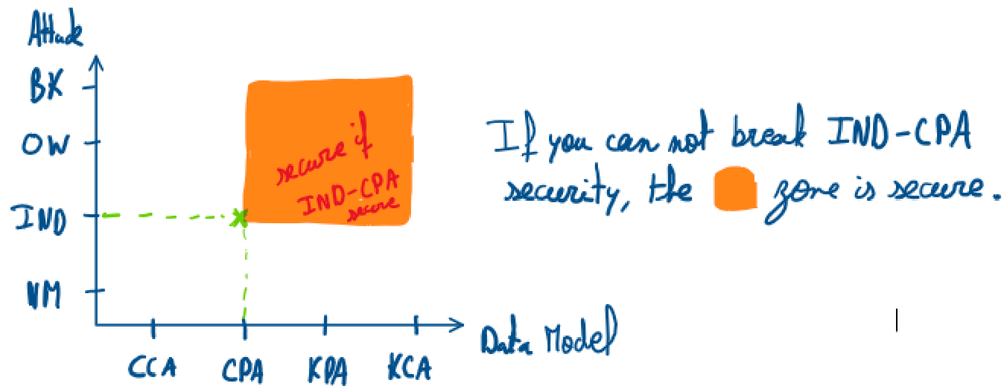
Formal definition: encryption scheme

#formal-definition
An #encryption-scheme is a **triple of algorithm** $E = (Gen, Enc, Dec)$ and a plaintext space M.

- #Gen is a *probabilistic algorithm* that *outputs a secret key* (or private) 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 $c = Enc_{k_E}(m)$*. 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* $m' = Dec_{k_D}(c)$.

INDistinguishability

Here is a little summary of all the attacks and security steps that can have a data model.



A scheme $E = (Gen, Enc, Dec)$ is #IND-secure if no adversary can win the following game for more than a negligible advantage.

- The *challenger generates a key* (pair) $k \leftarrow Gen()$
- *Adversary chooses two plaintexts*: $m_0, m_1 \in M$ with $|m_0| = |m_1|$
- The *challenger randomly chooses* $b \leftarrow_R \{0,1\}$, *encrypts* m_b and *sends* $c = Enc_k(m_b)$ *to the adversary*
- The adversary guesses b' which plaintext was encrypted
- The adversary wins if $b' = b$
  $\rightarrow$ In other words, the challenger chooses one of the two plaintexts and encrypt it before sending both to the adversary that guesses which is the encrypted plaintext.

The #problem in symmetric crypto is that this only addresses **the security of a single encryption!**

#IND-CPA-secure (**chosen plaintext attack**)
A scheme $E = (Gen, Enc, Dec)$ is **IND-CPA-secure** if no adversary can win the following game for more than a negligible advantage.

- The *challenger generates a key* (pair) $k \leftarrow Gen()$
- *Adversary queries* Enc_k *with plaintexts of his choice*
- *Adversary chooses two plaintexts*: $m_0, m_1 \in M$ with $|m_0| = |m_1|$
- The *challenger randomly chooses* $b \leftarrow_R \{0,1\}$, *encrypts* m_b and *sends* $c = Enc_k(m_b)$ *to the adversary*
- *Adversary queries* Enc_k *with plaintexts of his choice*
- The adversary guesses b' which plaintext was encrypted
- The adversary wins if $b' = b$

#IND-CCA-secure (**chosen ciphertexts attack**)
A scheme $E = (Gen, Enc, Dec)$ is **IND-CCA-secure** if no adversary can win the following game for more than a negligible advantage.

- The *challenger generates a key* (pair) $k \leftarrow Gen()$
- *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 $|m_0| = |m_1|$
- The *challenger randomly chooses* $b \leftarrow_R \{0,1\}$, *encrypts* m_b and *sends* $c = Enc_k(m_b)$ *to the adversary*
- *Adversary queries* Enc_k *with plaintexts of his choice* and Dec_k *with ciphertexts of his choice* excepts c.
- The adversary guesses b' which plaintext was encrypted
- The adversary wins if $b' = b$

Security strength for IND-CPA and IND-CCA

A scheme is $(t, d, \epsilon)-$**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.

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

Symmetric encryption with diversification

A #symmetric-key #encryption-scheme with diversification is a triple of algorithms $E = (Gen, Enc, Dec)$, 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* $c = Enc_k(m)$. The range of Enc is the ciphertext space C.
- #Dec takes as *input a secret key* k, a *diversifier* $d \in D$ *and message* $m \in M$. And *outputs ciphertext* $c = Dec_k(m)$.

To ensure #correctness : $\forall k \in K, d \in D, m \in M$, we **must have**

$$Dec_k(d, Enc_k(d, m)) = m$$

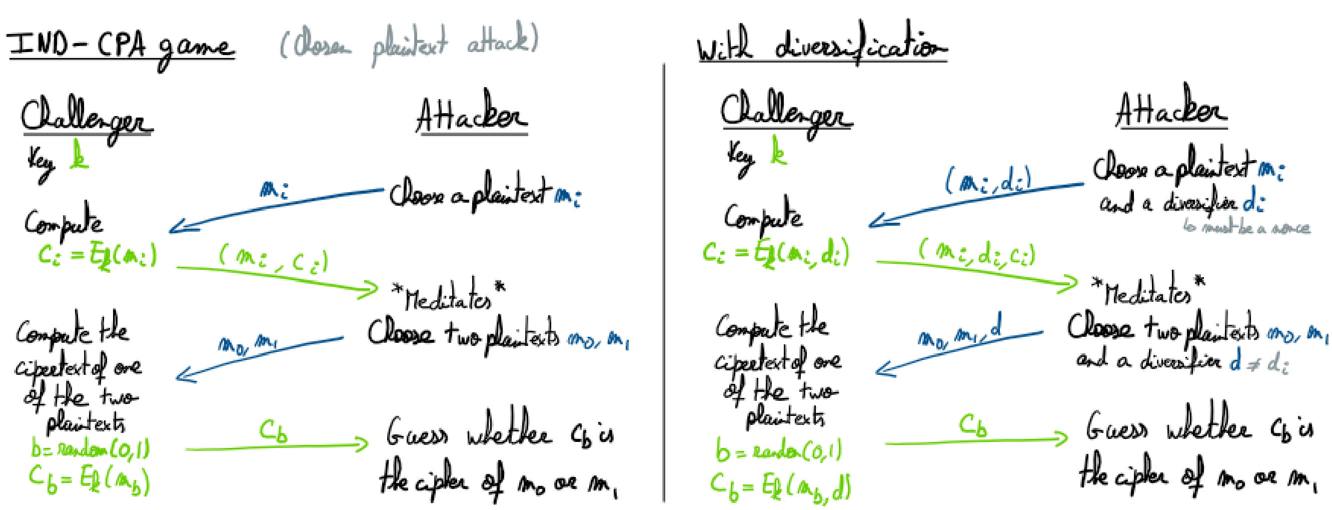
#IND-CPA-secure (chosen plaintext attack, chosen #diversifier)
A scheme $E = (Gen, Enc, Dec)$ is **IND-CPA-secure** if no adversary can win the following game for more than a negligible advantage.

1. The *challenger generates a key* (pair) $k \leftarrow Gen()$
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. The *challenger randomly chooses* $b \leftarrow_R \{0, 1\}$, *encrypts* m_b and *sends* $c = Enc_k(m_b, d)$ *to the adversary*
5. *Adversary queries* Enc_k *with (d, m) of his choice*
6. The adversary guesses b' which plaintext was encrypted
7. The adversary wins if $b' = b$

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

Schematic IND-CPA and diversification

A scheme E is #IND-CPA-secure if no polynomial time algorithm can win the following game with non-negligible advantage.



1.7.2 Authentication scheme

An #authentication-scheme must resist to an adversary that can have different **goals**:

- The *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)

Types of forgeries

A #forgery is a sort of algorithm that is used to get information over the encryption method, or a scheme used to decrypt (se faire passer pour qqn d'autre par exemple).

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

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). Same as [Encryption scheme](1.%20Historical%20ciphers%20and%20general%20principles.md#1.7.1 Encryption scheme).

#random-tag-guessing #example
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.)

#Exhaustive-key-search #example
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** ϵ = 1.

Formal definition: authentication scheme

#formal-definition
An #authentication-scheme is a **triple of algorithms** $T = (Gen, Tag, Ver)$ 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* $t = Tag_{k_A}(m)$. 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

#EU-CMA-secure #EU-CMA #TP3-ex11
A scheme $T = (Gen, Tag, Ver)$ is **EU-CMA-secure** if no adversary can win the following game for more than a negligeable probability.

- *Challenger generates a key* (pair) $k \leftarrow Gen()$

- *Adversary queries Tag_k with messages of his choice*
- *Adversary produces a (message, tag) pair*, with a *message not yet queried*
- Adversary *wins if $Ver_k(message, tag) \neq \perp$* (Advantage: ϵ = Pr [win])

1.7.3 Properties

A cipher with $\mathcal{M} = \mathcal{C}$ for certain key's $k \in \mathcal{K}$ is an #involution if its encryption and decryption procedures become identical.

$$\text{For all } m \in \mathcal{M} \implies E_k(m) = D_k(m)$$

1.8 Taking a step back

1.8.1 Covered in cryptography

#Generic-attacks
Attacks that work independently of the underlying primitives (mostly predictable) Examples: exhaustive key search attacks on mode of operation (see #chap2).

#Shortcut-attacks
Attacks that break a primitive more easily than claimed (mostly unpredictable) Examples: RC4, DES (see next chapter) advances in factoring or discrete log (see #chap4).

1.8.2 Not covered in cryptography

#Attacks-outside-the-model
Examples: no secrecy without encryption, forgery when there is no authentication, traffic analysis, length of messages.

#Implementation-attacks
Attacks that exploit flaws in implementations such as: bugs, assumptions not satisfied, side-channel attacks or fault attacks.
