Mécanismes cryptographiques pour la sécurité

Mohamed Houcine HDHILI
Mohamed.elhdhili@gmail.com

Cryptographie: objectifs

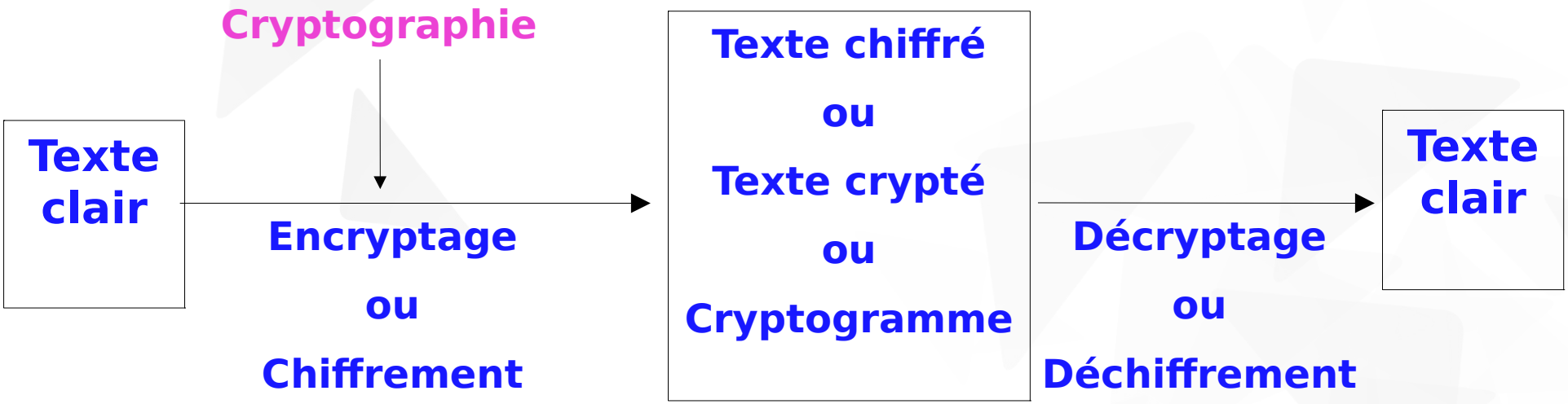
- Objectif principal:
  - Assurer la **sécurité des communications** sur un **canal non sécurisé**
- Sécurité des communications?
  - Confidentialité, Integrité, authenticité, non repudiation
- Canal non sécurisé?
  - Attaques passives: écouter des communications
  - Attaques actives: contrôler la communication (Man in the middle)
    - Injection,
    - Suppression,
    - modification

Terminologie

- Cryptologie= cryptographie+cryptanalyse
  - Science (branche des mathématiques) des communications secrètes.
  - Composée de deux domaines d'études complémentaires :
    - Cryptographie : conception d'algorithmes/protocoles
    - Cryptanalyse : casser des algorithmes/protocoles

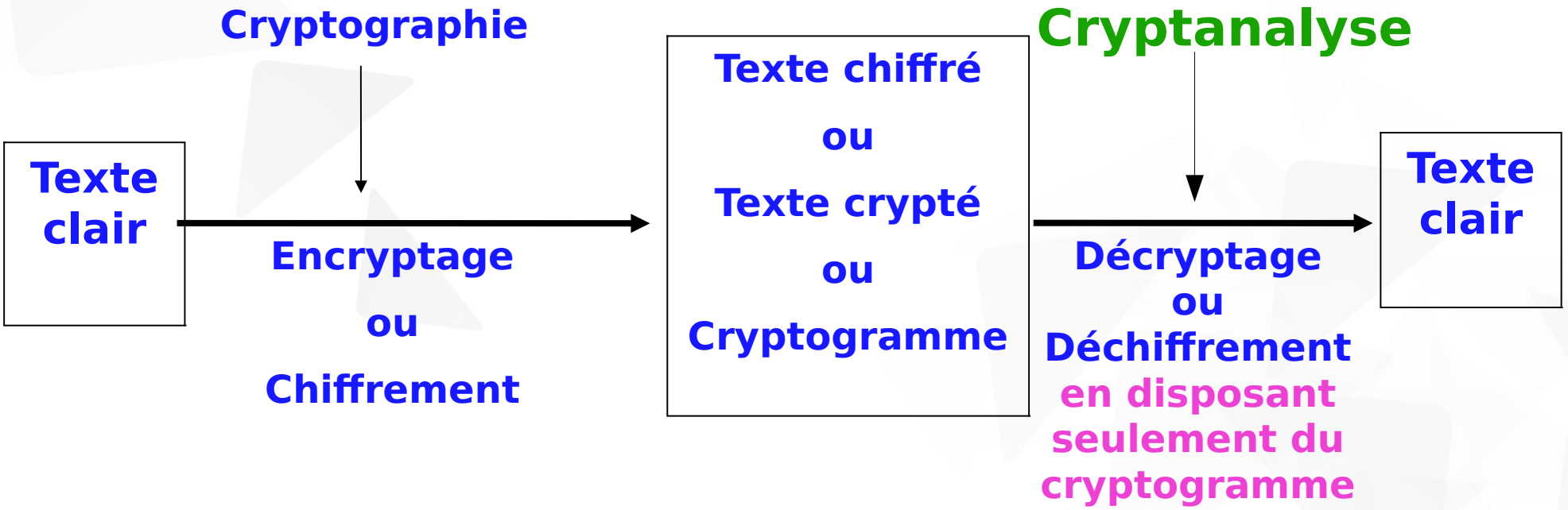
Terminologie

- Cryptographie (cryptography) = Chiffrement=Encryptage
  - Ensemble des méthodes et techniques qui permettent de transformer un message afin de le rendre incompréhensible pour quiconque qui n'est pas doté du moyen de le déchiffrer.
    - On parle d'***encrypter (chiffrer)*** un message,
    - Le code résultant s'appelle ***cryptogramme***.
    - L'action inverse s'appelle ***décryptage (déchiffrement)***.



Terminologie

- **Cryptanalyse (cryptanalysis)**
  - **Art de révéler les messages** qui ont fait l'objet d'un encryptage.
  - Lorsqu'on réussie, au moins une fois, à **déchiffrer un cryptogramme**, on dit que l'**algorithme** qui a servi à l'encrypter a été **cassé**.



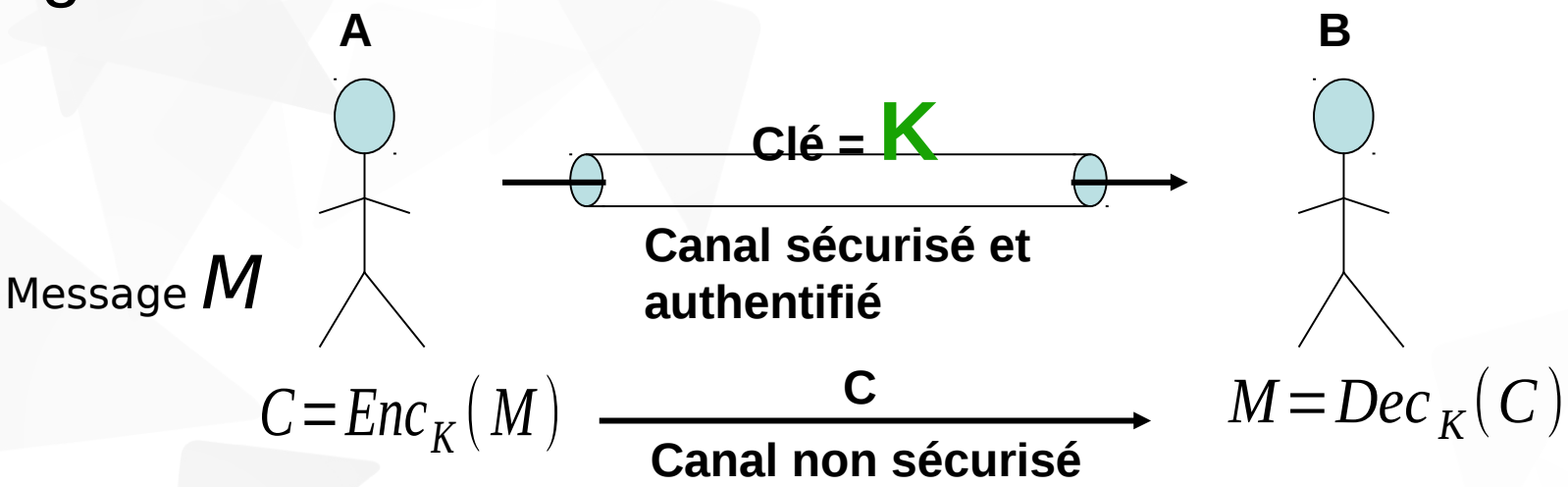
Terminologie

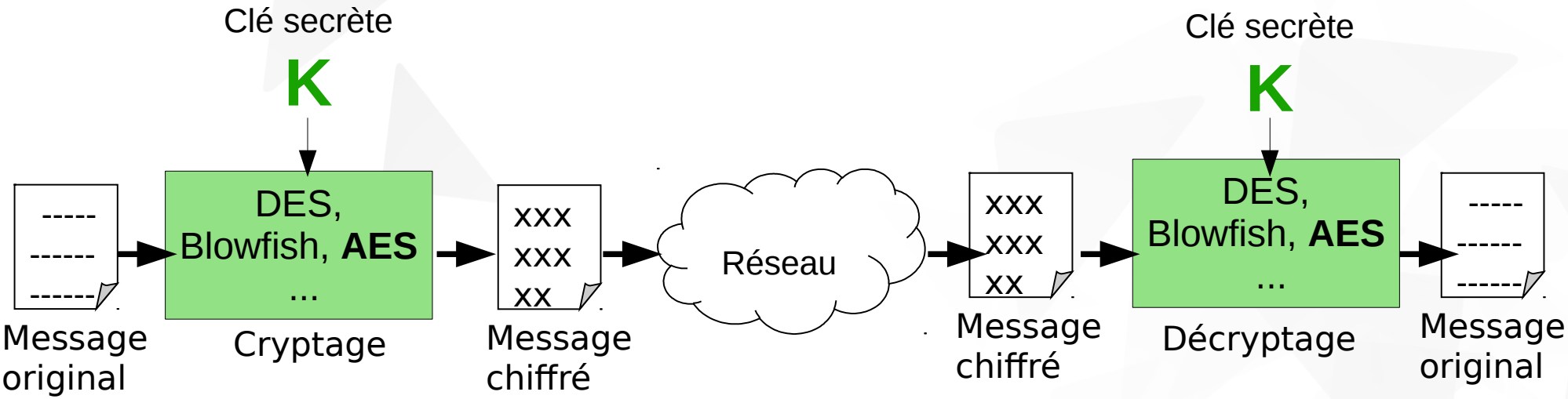
- Clé :
  - Information qui sera utilisée pour encrypter et / ou décrypter un message.

    On peut cependant concevoir un algorithme qui n'utilise pas de clé, dans ce cas c'est lui-même qui constitue le secret et son principe représente la clé
- Crypto système:
  - Ensemble composé
    - d'un algorithme,
    - de tous les textes en clair (**espace des textes clairs**),
    - de tous textes chiffrés (**espace des textes chiffrés**)
    - de toutes les clés possibles (**espace des clés**).

Chiffrement symetrique: principe

- Utiliser une même clé secrète **K** pour chiffrer et déchiffrer
- But : garantir la **confidentialité**



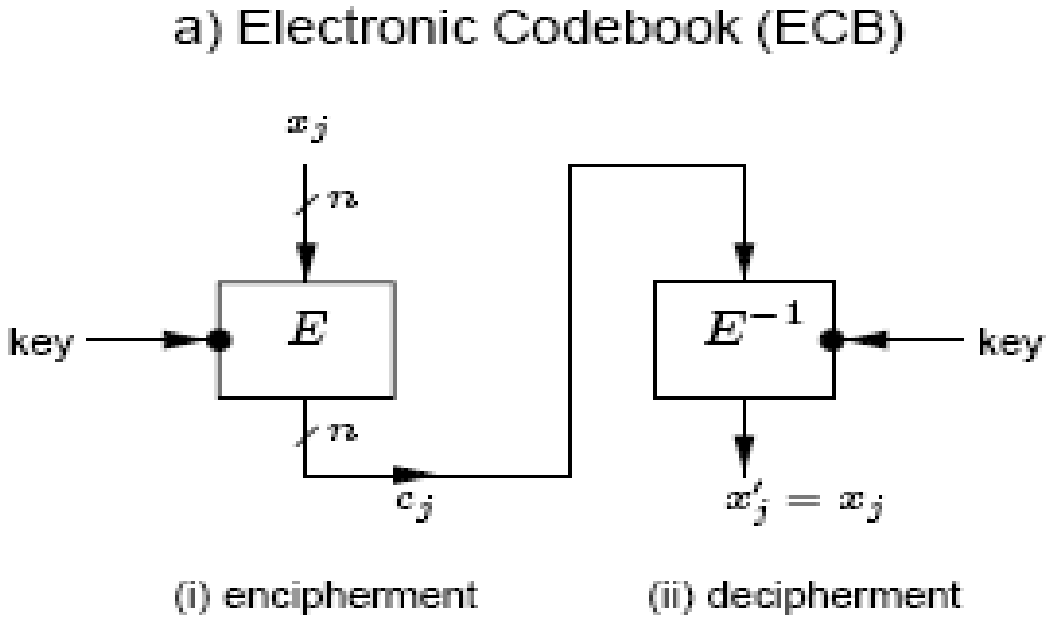


Chiffrement symetrique par blocs

- Méthode:
  - Le message M à chiffrer est scindé en un nombre de bloc de taille fixe: **$M=x_1, x_2, ..x_n$**
  - Cryptage des blocs
  - Le cryptogramme **C** est obtenu en concaténant les cryptogrammes des blocs
-
- Modes de chiffrement par bloc
  - ECB (Electronic CodeBook)
  - CBC(Cipher bloc Chaining)
  - CBF (Cipher FeedBack)
  - OFB (Output FeedBack)

ECB (Electronic CodeBook)

- Chaque bloc est chiffré à part



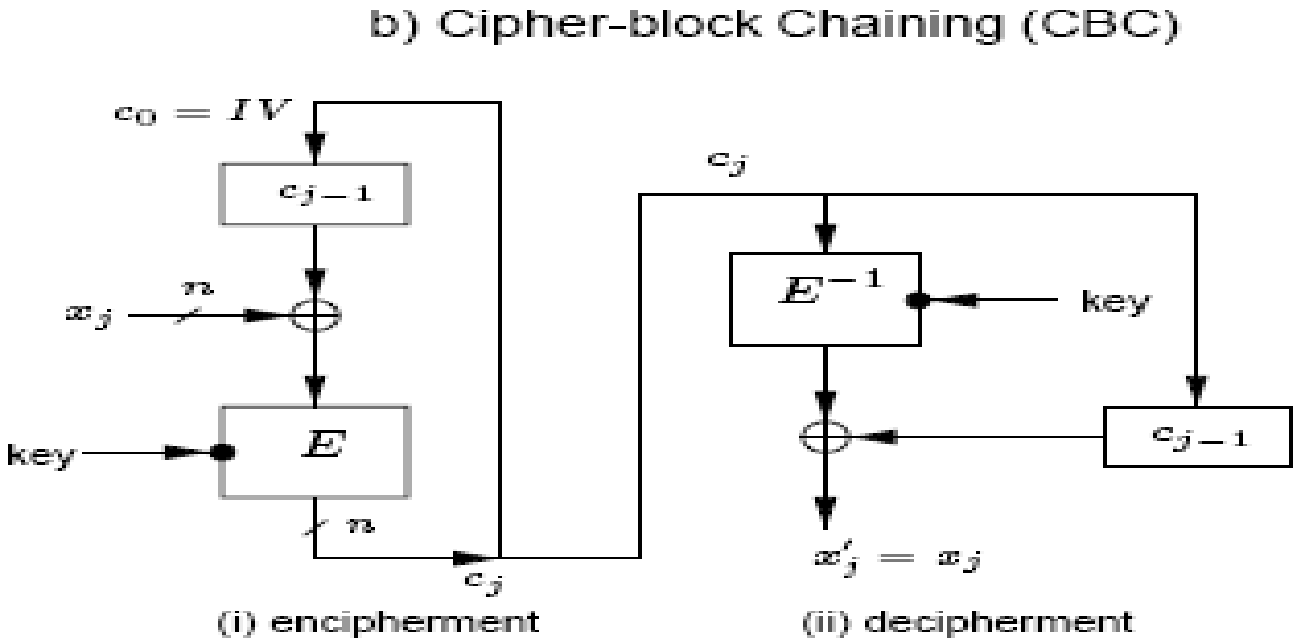
7.11 Algorithm ECB mode of operation

INPUT: k-bit key K; n-bit plaintext blocks $x_1, \ldots, x_t$.
SUMMARY: produce ciphertext blocks $c_1, \ldots, c_t$; decrypt to recover plaintext.

1. Encryption: for $1 \leq j \leq t$, $c_j \leftarrow E_K(x_j)$.
2. Decryption: for $1 \leq j \leq t$, $x_j \leftarrow E_K^{-1}(c_j)$.

CBC(Cipher bloc Chaining)

- Chaque bloc du cryptogramme dépend du bloc de texte en clair et de tous les blocs précédents



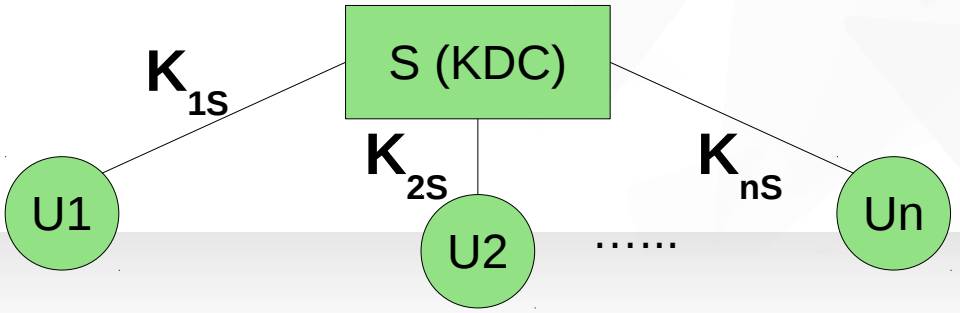
7.13 Algorithm CBC mode of operation

INPUT: k-bit key K; n-bit IV; n-bit plaintext blocks $x_1, \ldots, x_t$.
SUMMARY: produce ciphertext blocks $c_1, \ldots, c_t$; decrypt to recover plaintext.

1. Encryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $c_j \leftarrow E_K(c_{j-1} \oplus x_j)$.
2. Decryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j)$.

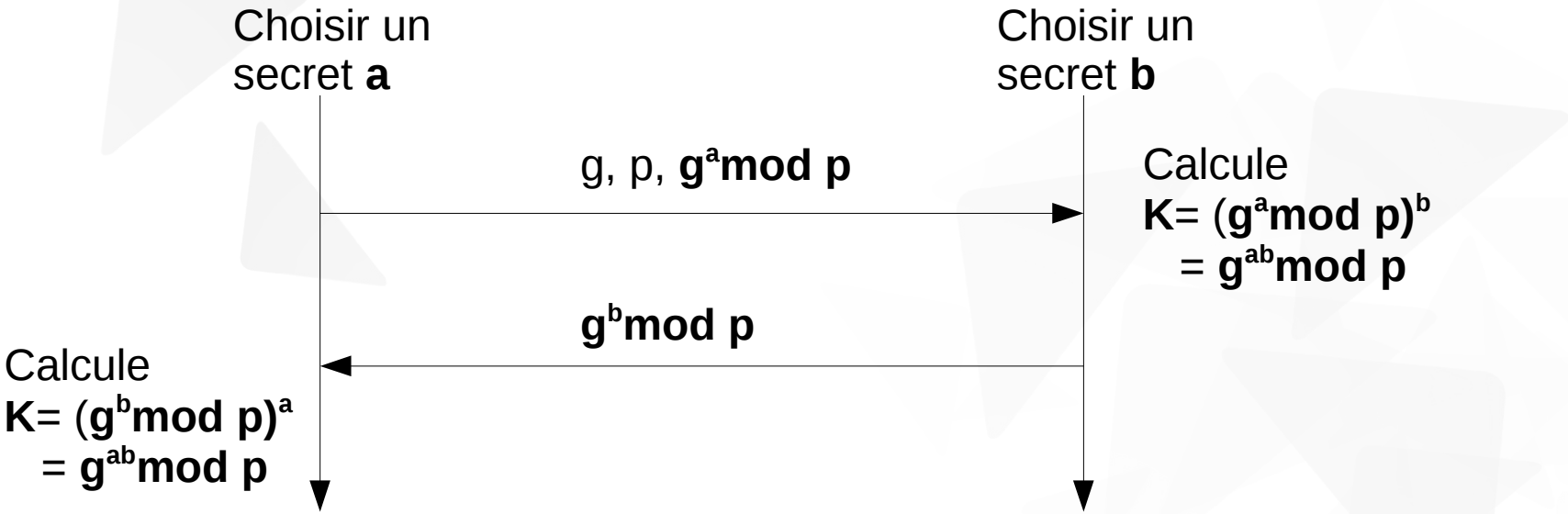
Chiffrement symétrique: critique

- Comment établir la clé **K** d'une manière sécurisée (confidentialité)**?**
  - Solution 1 : **Key transport**
    - l'un crée la clé et la transmet à l'autre (**chiffrement asymétrique**)
  - Solution 2 : **Key agreement**
    - une clé partagée est dérivée par 2 entités ou + (**protocole de DH**)
- Comment mettre à jour la clé K **?**
  - Clé de session : temporaire
  - Clé à long terme
- Besoin de **N(N-1)/2 clés** pour N utilisateurs => trop de clés
- ==> Sol : utiliser un centre de distribution de clé (KDC)
  - Chaque entité partage une clé avec le centre (**qui connaîtra toutes les clés!!!!**)
  - Exemple : Télécom



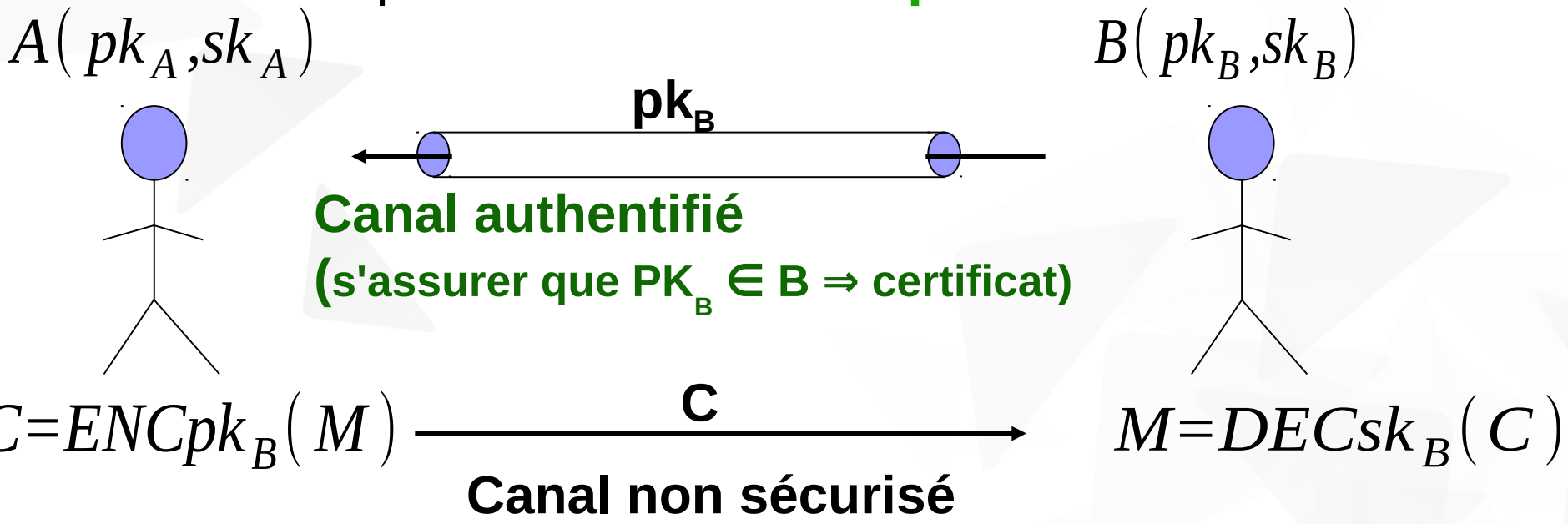
Diffie-Hellman key agreement protocol (logarithmique)

- Etablir une clé commune via un canal publique
  - p : nombre premier, g : générateur de Zp* ==> publiques
- Basé sur deux problèmes difficiles
  - Discrete Log Problem(DLP) : étant donné **(g, h, p)** calculer **a** tel que $\mathbf{g^a = h \bmod p}$
  - Computational Diffie Hellman (CDH):étant donné $\mathbf{g^a \bmod p}$ et $\mathbf{g^b \bmod p}$ calculer $\mathbf{g^{ab} \bmod p}$



Chiffrement asymétrique (à clé publique)

- But : garantir la **confidentialité**
- Chaque entité possède **deux clés**
  - **pk**: public key ==> clé disponible pour tout le monde
  - **sk**: secret (private) key ==> clé privée de l'entité
- Il est imposible de calculer **sk** en connaissant **pk**
- **sk** déchiffre ce qui a été chiffré avec **pk** **et vice versa**



- **Exemples d'algorithmes: RSA, ElGamel, Rabin**

Cryptosystème RSA (Ron Rivest, Adi Shamir and Leonard Adleman 1978): génération des clés

Algorithm Key generation for RSA public-key encryption

SUMMARY: each entity creates an RSA public key and a corresponding private key. Each entity A should do the following:

1. Generate two large random (and distinct) primes p and q, each roughly the same size.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$. (See Note 8.5.)
3. Select a random integer e, $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.
4. Use the extended Euclidean algorithm (Algorithm 2.107) to compute the unique integer d, $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$.
5. A's public key is (n, e); A's private key is d.

2.107 Algorithm Extended Euclidean algorithm

INPUT: two non-negative integers a and b with $a \geq b$.
OUTPUT: $d = \gcd(a, b)$ and integers x, y satisfying $ax + by = d$.

1. If $b = 0$ then set $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$, and return(d,x,y).
2. Set $x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$, $y_1 \leftarrow 1$.
3. While $b > 0$ do the following:
   3.1 $q \leftarrow \lfloor a/b \rfloor$, $r \leftarrow a - qb$, $x \leftarrow x_2 - qx_1$, $y \leftarrow y_2 - qy_1$.
   3.2 $a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$, and $y_1 \leftarrow y$.
4. Set $d \leftarrow a$, $x \leftarrow x_2$, $y \leftarrow y_2$, and return(d,x,y).

Cryptosystème RSA: chiffrement

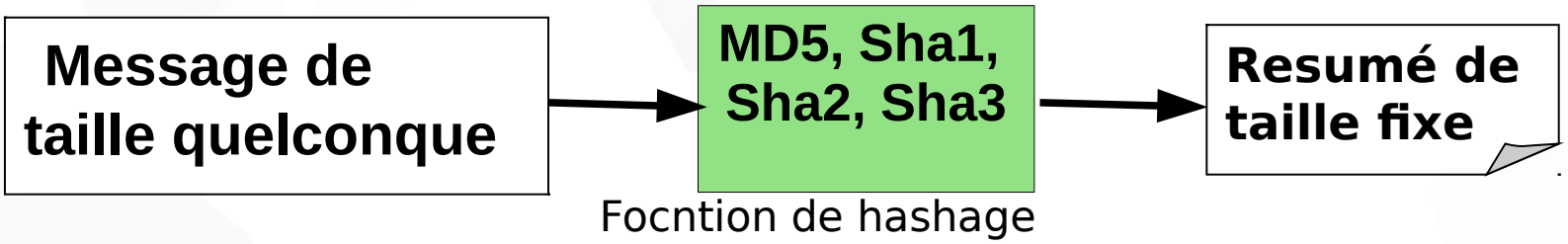
Algorithm RSA public-key encryption

SUMMARY: B encrypts a message m for A, which A decrypts.

1. *Encryption.* B should do the following:
   (a) Obtain A's authentic public key (n, e).
   (b) Represent the message as an integer m in the interval $[0, n-1]$.
   (c) Compute $c = m^e \bmod n$ (e.g., using Algorithm 2.143).
   (d) Send the ciphertext c to A.
2. *Decryption.* To recover plaintext m from c, A should do the following:
   (a) Use the private key d to recover $m = c^d \bmod n$.

Fonction de hashage

- But: assurer l'**integrité**
- Génère à partir d'un message de **longueur quelconque** un résumé de **taille fixe**



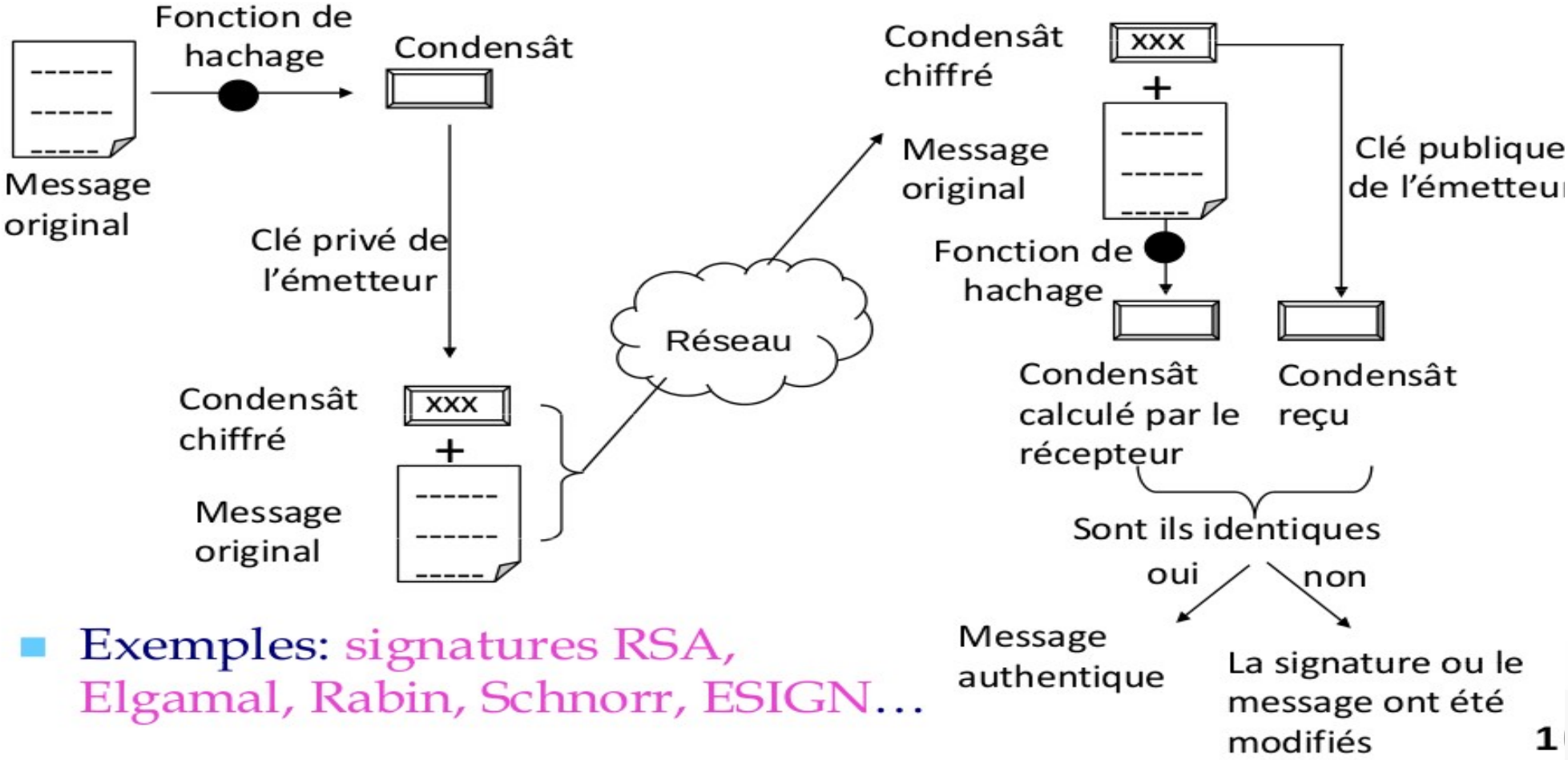
- Unicité (pas de collision):
  - deux messages différents ne donnent pas la même empreinte
- Irréversible (one way):
  - à partir du resumé, on ne devrait pas trouver le message
- Sensible à l'attaque Man in The Middle (MITM) ==> **solution: signature**

Fonctions de hashage importantes

- MD5 (128 bits)
  - Resistance aux collisions
    - cassé (Prof. Xiaoyun Wang, china 2004)
- SHA1 (160 bits)
  - “One way” est encore valide
  - Resistance aux collisions
    - Considérée non sécurisée (pas de collisions déterminées mais des méthodes existent en 2^80)
- SHA2 (SHA224, SHA-256, sha-384, SHA-512)
  - Donnent 224, 256, 384, 512 bits respectivement
  - Sécurisée
- SHA3(SHA3-224, SHA3-256, SHA3-384, SHA3-512)
  - Sécurisée

Signatures digitales

- Permet l'authentification, l'intégrité et la non répudiation

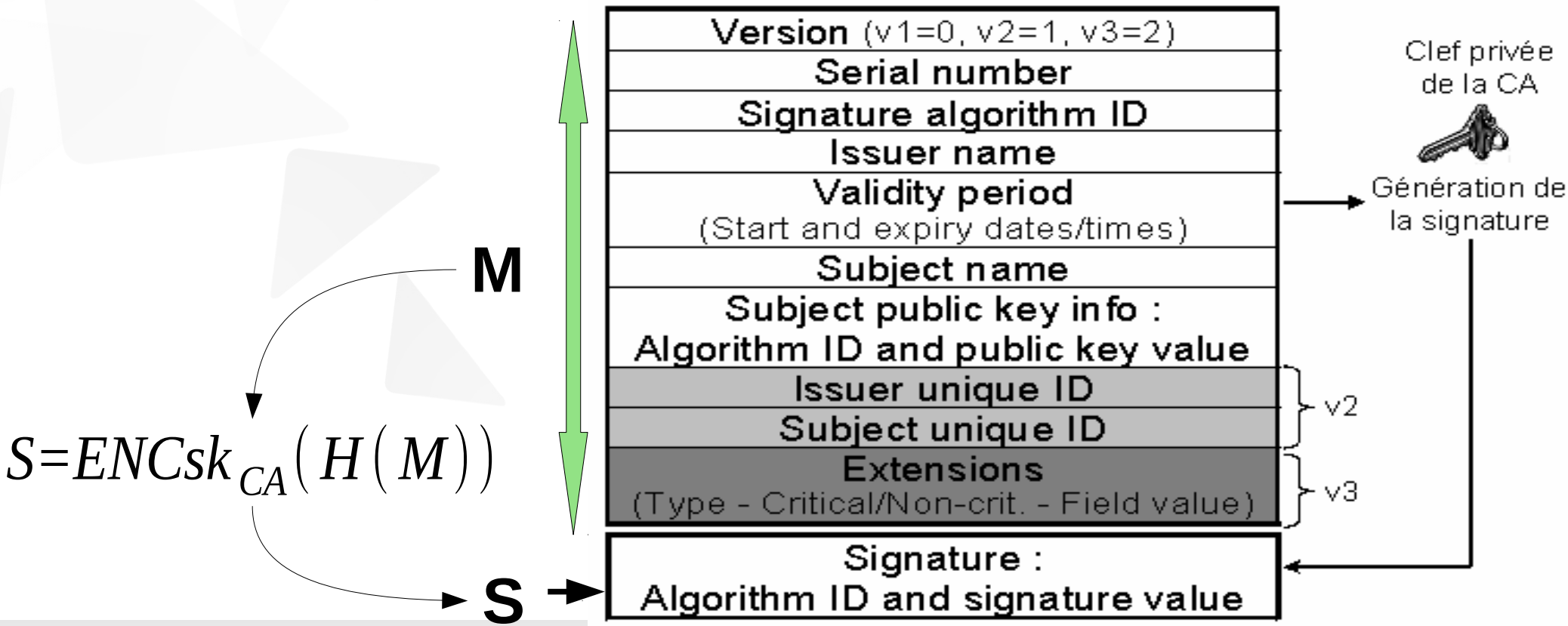


- Exemples: signatures RSA, Elgamal, Rabin, Schnorr, ESIGN…

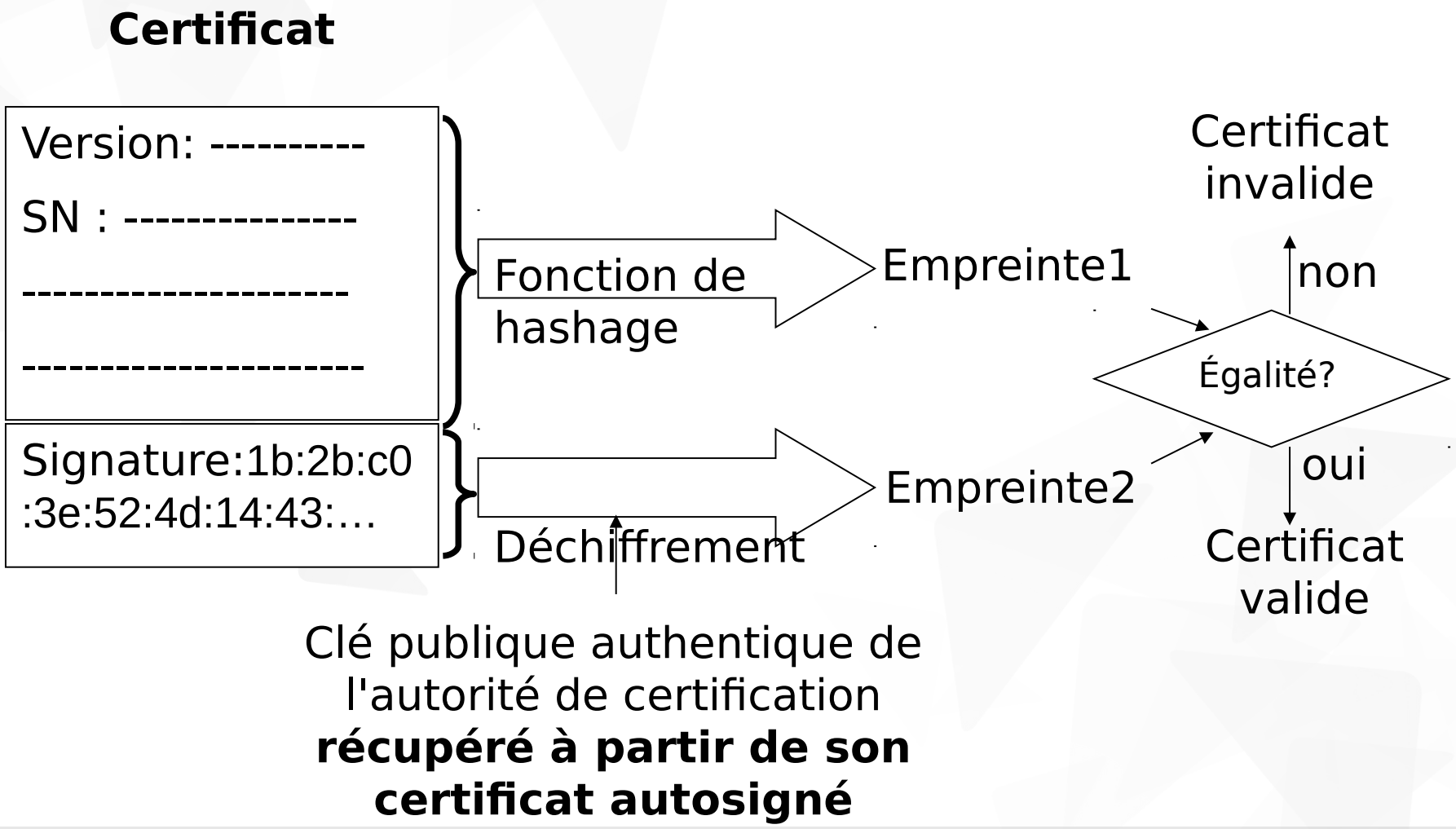
- On peut signer un message sans appliquer une fonction de hashage
- **Sensible à l'attaque MIM** sauf si la clé publique de l'émetteur est authentifié **==> sol:** récupérer la clé publique de l'émetteur à partir de son **certificat**

Certificat electronique

- Assure l'authentification
  - **Garantit l'appartenance d'une clé publique à une entité**
  - Signé par un organisme de confiance (ANCE, Verisign, Digicert...
- **Exemple: certificat X.509**

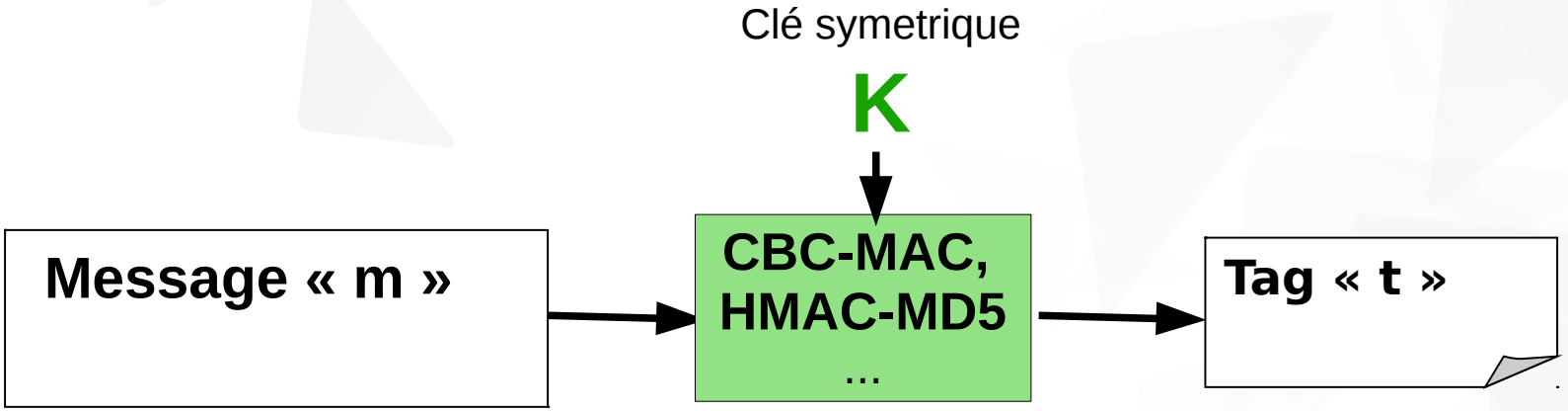


Vérification d'un certificat electronique



Message Authentication Code (MAC) (cas du chiffrement **symétrique**)

- **But: Intégrité** des données et **authentification** de la source
- **La source et le destinataire partage une clé K** qui est utilié pour l'authentification des messages
  - Key generation: **$k=GEN(1^n)$**
  - Tag generation: **$t=MAC_k(m)$**
  - Verification algorithm: **$b=verify_k(m,t)$** (b=1 sig verification valide)
- Exemple CBC-MAC, HMAC-MD5



CBC MAC (fixed length MAC)

- Soit une fonction pseudo aléatoire PRF, et L(n) une fonction longueur
- PRF $F:\{0,1\}^n x \{0,1\}^n \rightarrow \{0,1\}^*$
- CBC MAC est:
  - $mac_K(m)$, m est de longueur L(n).n
    - Diviser m en $m_1, m_2, \ldots m_L$ chacune de taille n
    - $t_0 = 0^n$
    - **Pour i de 1 à L, $t_i = F_K(t_{i-1} \oplus m_i)$**
    - **Output: t_L**
  - Verify (K, m, t)? ==> vérifier si **$mac_K(m) = t_L$**
- Proved to be secure