Introduction à la cryptographie – Chapitre III

Authentification

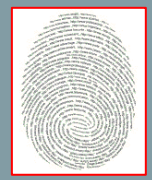
Plan du cours

- **0. Courte introduction.**
- **I. Systèmes à clef publique.**
- **II. Systèmes à clef secrète.**
- **III. Authentification.**
- **IV. Exemples.**

Plan du cours

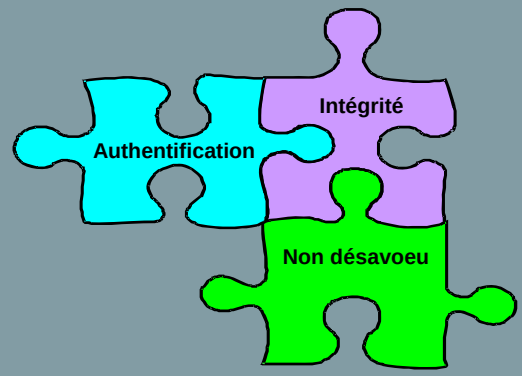
- **0. Courte introduction.**
- **I. Systèmes à clef publique.**
- **II. Systèmes à clef secrète.**
- **III. Authentification.**
- **IV. Exemples.**

- **3.1. Fonctions de hachage.**
- **3.2. Signature à clef publique.**



- Une fonction de hachage est un procédé de vérification d'intégrité d'un document ou d'un mot de passe. Elle doit permettre de l'authentifier.
- La fonction de hachage doit produire, à partir d'un message initial, une séquence de bits très courte (typiquement entre 128 et 512 bits) qui représente la signature du texte.
- On parle également d'**empreinte** numérique, de signature électronique, de condensat ou encore de haché.

- La signature numérique doit réaliser les mêmes fonctionnalités que la signature traditionnelle :
  - Facilité et rapidité de calcul.
  - Résultat court.
  - Non réutilisable.
  - Impossible à falsifier.



- En termes cryptographiques, elle doit assurer l'authentification, l'intégrité et le non-désavoeu.
- En France, la validitité juridique a été votée en mars 2000 et le décret d'application est paru en mars 2001.

Propriétés souhaitées pour une fonction de hachage.

- Une même empreinte ne doit pas pouvoir correspondre à deux messages différents.
- Idéalement, une empreinte doit correspondre à un unique document et vice versa.
- On ne doit pas pouvoir retrouver le message initial à l'aide de l'empreinte.

- Notations.
  - le texte en clair : x
  - la fonction de hachage : h
  - l'empreinte : s

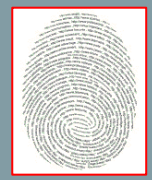
7480640e9e85b3b9de7afe4f0c400d5e

- $s = h(x)$

x

1 Go

h



7480640e9e85b3b9de7afe4f0c400d5e

s

128 bits

- Une même empreinte ne doit pas pouvoir correspondre à deux messages différents.

$$h(x) = h(x') \Rightarrow x = x'$$

h est injective.

C’est impossible car les messages sont plus nombreux que les empreintes.

- Idéalement, une empreinte doit correspondre à un unique document et vice versa.

h est bijective

C’est impossible car l’empreinte a une taille strictement inférieure au texte initial.

- On ne doit pas pouvoir retrouver le message initial à l’aide de l’empreinte.

h est difficile à inverser.

Mathématiquement, elle n’est pas inversible car elle ne peut pas être bijective.

- Une **collision** est une paire de messages ou de mots de passe différents dont l'empreinte est la même.

- Comme il existe beaucoup plus de messages que d'empreintes, les collisions existent obligatoirement, quelque soit l'algorithme de hachage.

- Elles doivent être très difficiles à construire ; on dit que la fonction de hachage doit être résistante aux collisions.

- Trouver deux messages en clair ayant la même empreinte revient à trouver deux personnes dans une classe ayant la même date d'anniversaire.
- On se dit que cela doit être peu probable…
- Considérons un étudiant dont la date d'anniversaire est donnée.
- La probabilité qu'un second étudiant n'ait pas la même date d'anniversaire est donnée par : $1 - \frac{1}{n}$
- La probabilité qu'un ième ait une date d'anniversaire différente des précédents est : $1 - \frac{i}{n}$

- Notons **A** l'évènement « au moins deux étudiants ont la même date d'anniversaire ».

- La probabilité qu'aucun étudiant n'ait de date d'anniversaire commune est :

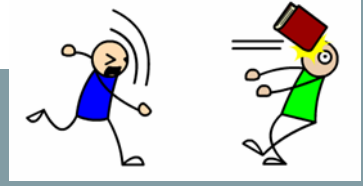
$$\mathbb{P}(\bar{A}) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right)$$

- On a l'inégalité (utile) suivante : $\forall x/\ 0 \leq x \leq 1 \quad 1 - x \leq e^{-x}$

- Appliquée à la formule précédente, on en déduit :

$$\mathbb{P}(\bar{A}) \leq \prod_{i=1}^{k-1} e^{-i/n} = e^{-\frac{k(k-1)}{2n}}$$

- En passant à l'évènement contraire : $\mathbb{P}(A) \geq 1 - e^{-\frac{k(k-1)}{2n}}$
- Et donc : $\mathbb{P}(A) \geq \frac{1}{2} \iff k = \sqrt{2 \ln 2 \times n}$ $\mathbb{P}(A) \simeq \sqrt{n}$
- Pour $n = 365$ et $k = 23$ on a déjà $\mathbb{P}(A) \geq 1/2$
- Maintenant si n représente la taille en bits d'une empreinte, on peut trouver deux messages ayant la même empreinte en testant en moyenne un nombre de messages égal à : $2^{\frac{n}{2}}$
- Pour une empreinte de 128 bits, cela fait : 2^{64}
- Pour une empreinte de 160 bits, cela fait : 2^{80}



- On crée une très longue liste de mots de passe dont on calcule l’empreinte.
- On stocke ceux qui ont même empreinte.
- On pourra les utiliser pour falsifier des documents en envoyant un autre document ou un autre mot de passe que celui qui a été signé.
- Pour éviter cela, on ajoute à la fin du mot de passe une chaîne de caractères aléatoire que l’on appelle le **sel**, et l’on calcule l’empreinte du mot de passe seulement après salage.

L’algorithme MD5. (1)

- Message Digest a été inventé par Rivest en 1991 comme amélioration de l’algorithme MD4.
- Il produit des empreinte de 128 bits.
- Des faiblesses ont été détectées dès 1996.
- Il est pourtant resté l’un des algorithmes de hachage les plus utilisé au monde. Il permet en particulier de vérifier l’intégrité de fichiers téléchargés sur internet.

- Exemple : La cryptographie c'est formidable.

  5948dd20a2a8c0f1982a35069e158dbc

- Autre exemple : l’empreinte de « l’odyssée » d’Homère est :

  0dabea84864654390e1e6578d58a8846

- Pour calculer une empreinte, on découpe le message en blocs de 512 bits. Chaque bloc est ensuite découpé en 16 blocs de 32 bits.
- L’algorithme entre alors dans une boucle comportant 4 rondes durant lesquelles une opération de base non linéaire sera effectuée 16 fois.
- On dispose de 4 variables a, b, c, d qui sont initialisées à des valeurs précises.

- On dispose également des fonctions suivantes :

$$\begin{cases} f(x,y,z) = (x \wedge y) \vee (\neg x \wedge z) \\ g(x,y,z) = (x \wedge z) \vee (y \wedge \neg z) \\ h(x,y,z) = x \oplus y \oplus z \\ i(x,y,z) = y \oplus (x \vee \neg z) \end{cases}$$

- Chaque opération de base calcule l’image d’une de ces fonctions pour trois des quatre variables. On ajoute au résultat la quatrième variable, ainsi qu’un constante et un sous bloc. On décale le résultat vers la gauche d’un nombre variable de bits, on l’ajoute encore à l’une des variables et on stocke le tout à la place de a,b,c ou d !!

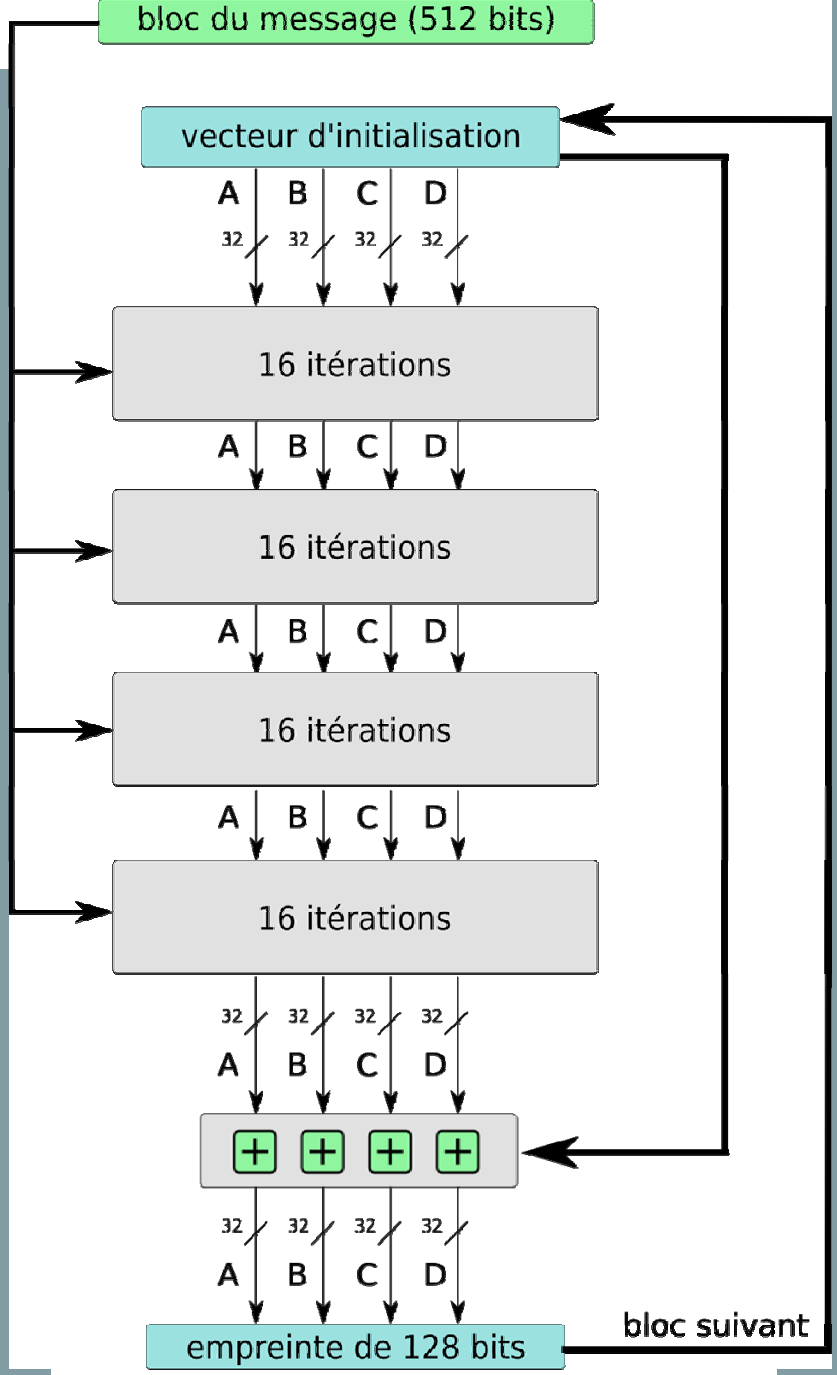
- Exemple :

$$\begin{cases} a = b + ((a + f(b,c,d) + x_i + \alpha_i) \lll s \\ d = a + ((d + g(a,b,c) + x_i + \alpha_i) \lll s \\ c = d + ((c + h(d,a,b) + x_i + \alpha_i) \lll s \\ a = b + ((a + i(b,c,d) + x_i + \alpha_i) \lll s \end{cases}$$

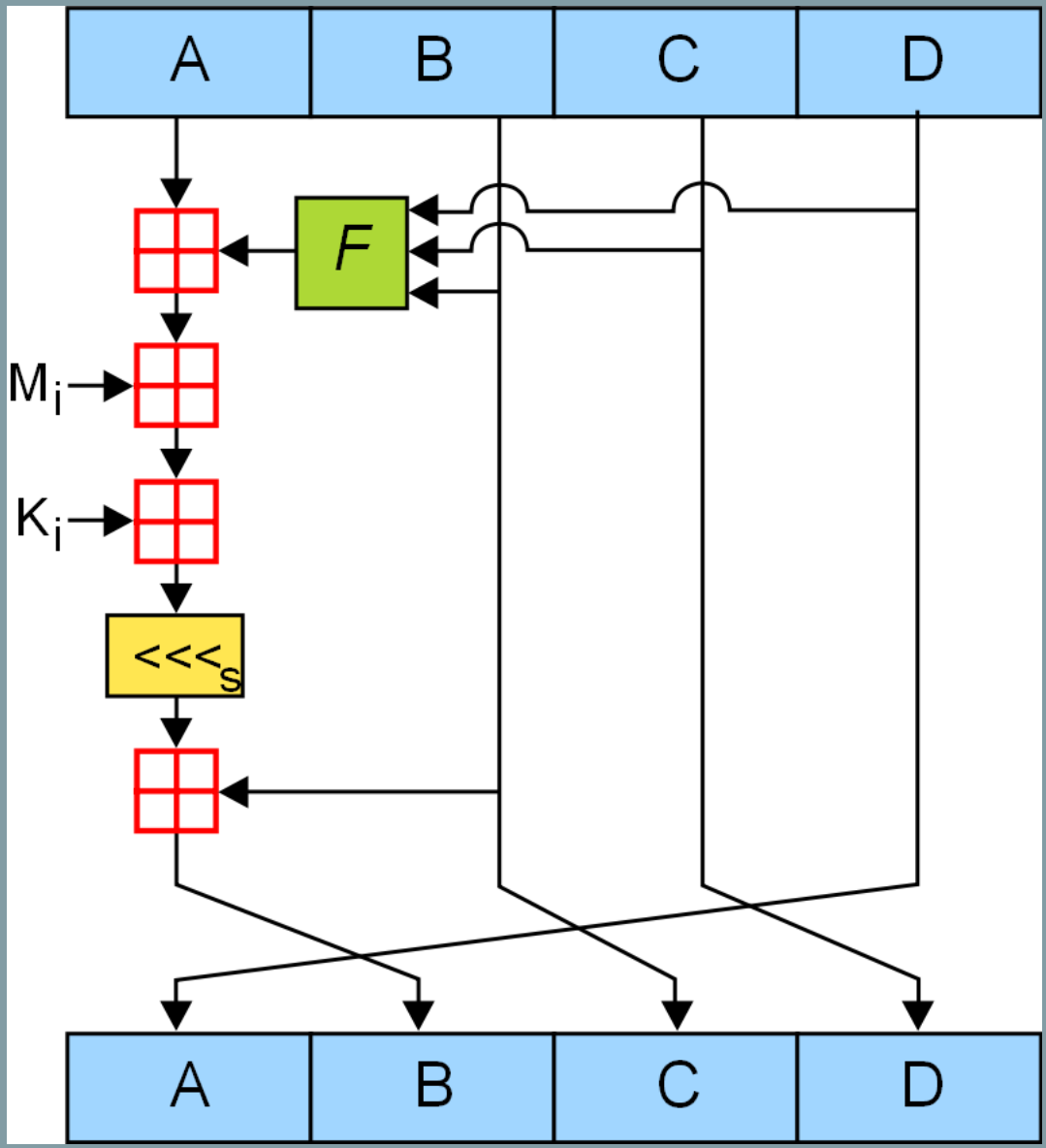
- x_i ième sous bloc.
- α_i ième constante. $\alpha_i = 2^{32}|\sin i|$

- Au final, les constantes modifiées au fur et à mesure des boucles sont concaténées pour donner l’empreinte.

L’algorithme MD5. (6)

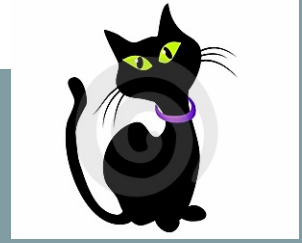


L’algorithme MD5. (7)



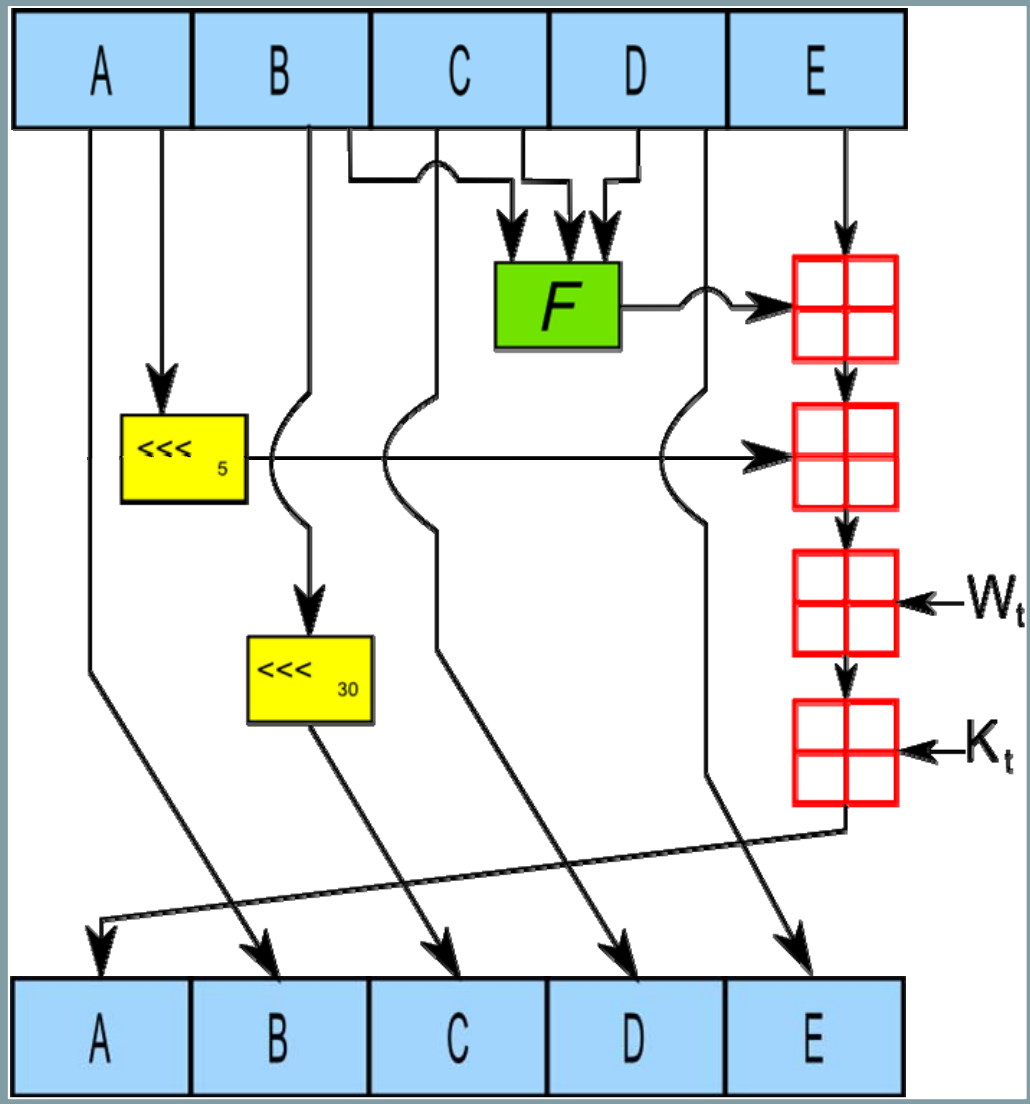
MD5 n’est plus sûr.

- En décembre 2008, une équipe d’informaticien (HashClash) démontre que le protocole SSL / TLS contient une fausse autorité de certification.
- En fait, la faille provient d’une faiblesse de conception de l’algorithme MD5.
- A l’aide de 300 « Playstation 3 » l’équipe construit deux certificats différents ayant exactement la même empreinte.
- On sait maintenant construire en quelques minutes des collisions avec MD5 (cf. logiciels « John the Ripper » ou bien BarWF).



- Le procédé Secure Hash Algorithm est le standard américain de fonction de hachage. Il a été conçu par la NSA.
- Il est utilisé dans le protocole DSA de signature numérique.
- Il produit des empreintes numériques de 160 bits.
- Le principe de fonctionnement ressemble à celui de MD5.

Principe de fonctionnement de SHA 1.



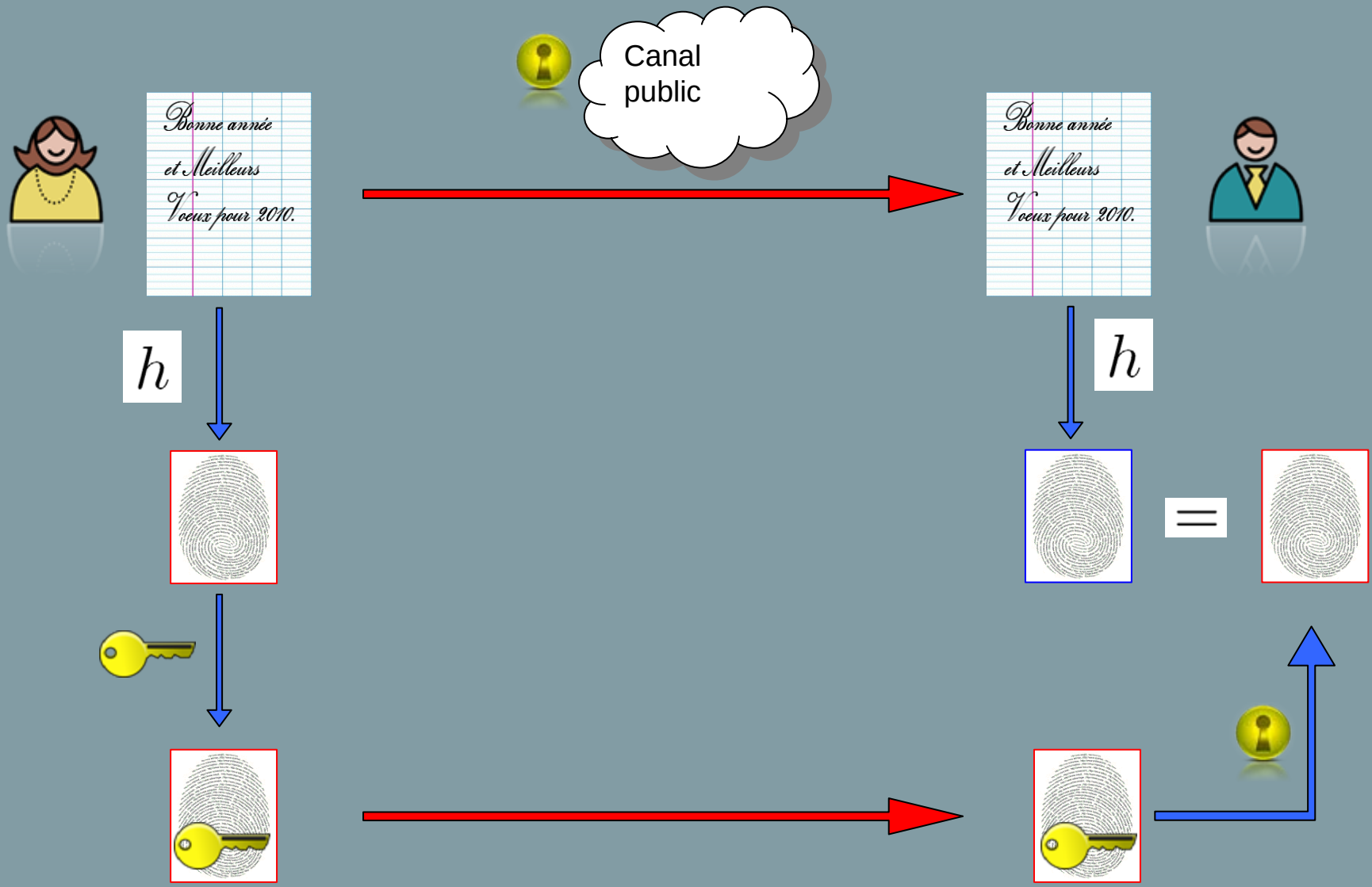
- Plusieurs attaques existent qui permettent de diminuer le nombre d’essais à effectuer pour trouver des collisions.

- SHA-1 a été retiré progressivement de la plupart des applications pour être remplacé par SHA-2, qui permet des empreintes de 256, 384 ou 512 bits.

- Inventé en 2000 par l'un des créateurs de l'AES, sur un modèle semblable. Il est libre de droit.
- Il calcule des empreintes sur 512 bits, ce qui est relativement long.
- Pas d'attaques connues pour l'instant.

- Importance de combiner les algorithmes de chiffrement et d'authentification pour consolider leur sécurité.
- Ceci est possible à l'aide de fonctions de hachage avec clef : les CAM (Codes d'Authentification de Messages).

Signature à clef publique avec RSA.



- Comment Bernard peut-il être sûr que la clef d'Alice est bien celle qui lui a été envoyée ? Attaque « man in the middle »…
- Nécessité d'une autorité de certification qui délivre des certificats de confiance (PKI, SSL / TLS, etc.).

- Algorithme de signature à clef publique utilisant SHA.
- A la place du protocole RSA, il utilise une version modifiée d’El Gamal pour le chiffrement.

Des questions ?

Fin du chapitre 3. Crédits photos & copyright :

- Photo de couverture: © Adi Shamir.
- Dessin du puzzle: © Haykel Mejri.
- Photo nombres binaires: © Westwood Schools.
- Autres sources d’inspiration:
  - Cryptographie appliquée de Bruce Schneier, éditions Vuibert.
  - L’art du secret, in « dossier pour la science ».
  - Histoire des codes secrets, de Simon Singh, éditions livre de poche.