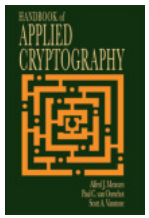
Fonctions de Hachage et Signatures Electroniques

Sébastien VARRETTE

Université du Luxembourg - Laboratoire LACS, LUXEMBOURG
CNRS/INPG/INRIA/UJF - Laboratoire LIG-IMAG
Sebastien.Varrette@imag.fr
http://www-id.imag.fr/~svarrett/

Cours "Cryptographie & Securité Réseau" Master Info
Université de Yaoundé

Quelques références bibliographiques. . . (avant que j'oublie)



Menezes A. J., Vanstone S. A. and Oorschot P. C. V., *Handbook of Applied Cryptography*, Computer Sciences Applied Mathematics Engineering, CRC Press, Inc., 1st edition, 1996,

http://www.cacr.math.uwaterloo.ca/hac/

Schneier B., *"Cryptographie Appliquée"*, Vuibert, Wiley and International Thomson Publishing, NY, 2nd edition, 1997.

http://www.schneier.com/book-applied.html

Stinson D.R, *Cryptography : Theory and Practice*, Chapman & Hall/CRC Press, 2nd edition, 2002.

http://www.cacr.math.uwaterloo.ca/~dstinson/CTAP2/CTAP2.html

Ebrahimi T., Leprévost F. and Warusfeld Ed., *Cryptographie et Securité des systèmes et réseaux*, Hermes/Lavoisier,

http://www-id.imag.fr/~svarrett/book_secu_mult.html

Plan

Notion de fonction de hachage

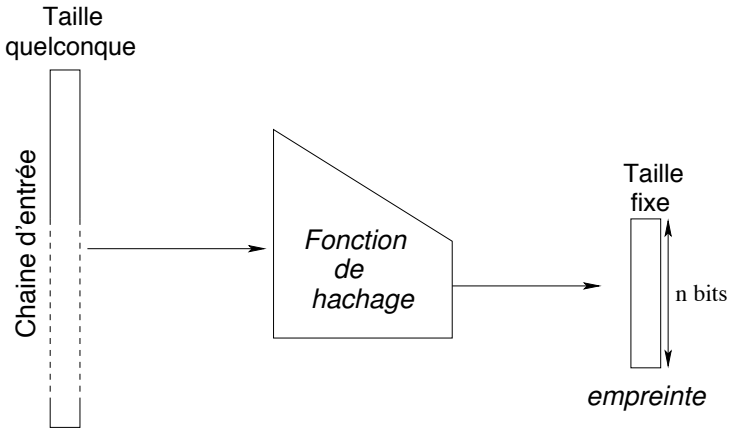
Definition (Fonction de Hachage)

Une fonction de hachage $H : \{0,1\}^* \longrightarrow \{0,1\}^n$

- facilement calculable
- qui transforme une chaîne binaire de taille quelconque t en une chaine binaire de taille fixe n

La chaîne $y = H(x)$ obtenue est appelée *empreinte de hachage*.

- En général, $t > n$: des collisions sont inévitables !



Intérêt cryptographique

- Primitive cryptographique très importante !
  - Schemas de signature (PSS)
  - Schémas de chiffrement (OAEP)
  - Intégrité de documents
  - Message Authentication Code (MACs)
- Facilement calculable
- Compression

Notion de collision

Definition (Collision)

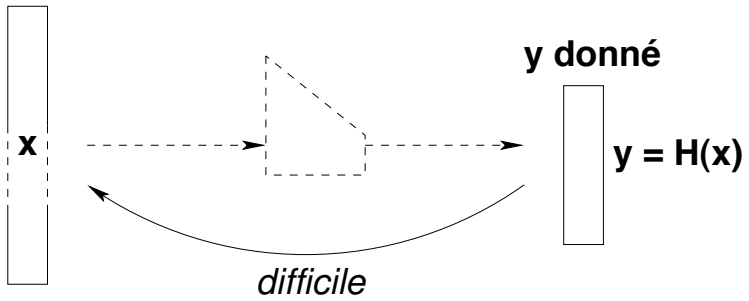
On parle de *collision* entre x et x' lorsque

$$\begin{cases} x \neq x' \\ H(x) = H(x') \end{cases}$$

- Il y a **forcément** des collisions !
- Si $y = H(x)$, alors :
  - x est appelé *préimage* de y
  - Rappel : y est l'empreinte de x

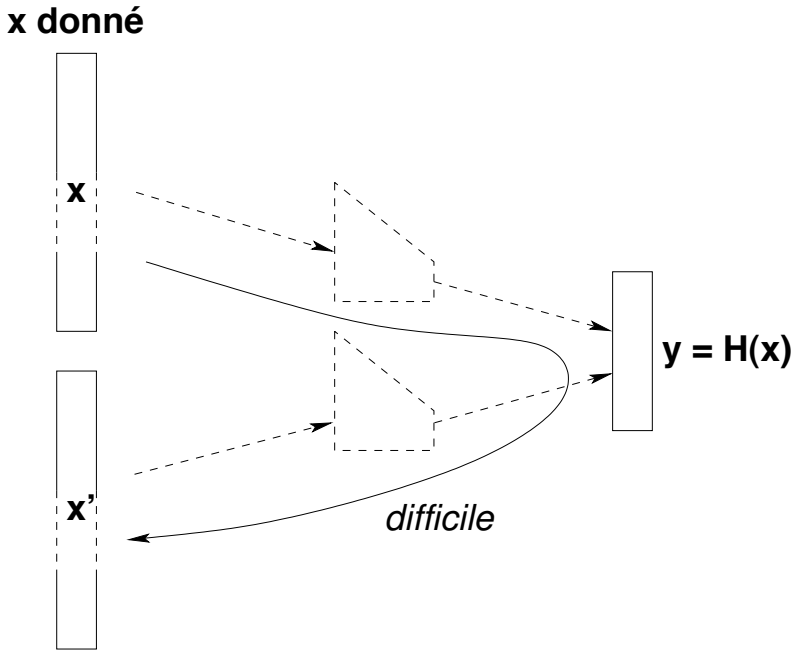
Propriétés des fonction de hachage

- De base : compression et facilité de calcul.
- Résistance à la préimage
  - Etant donné y : difficile de trouver x / $y = H(x)$
  - Etant donné y : 2^n calculs de hash requis pour trouver x



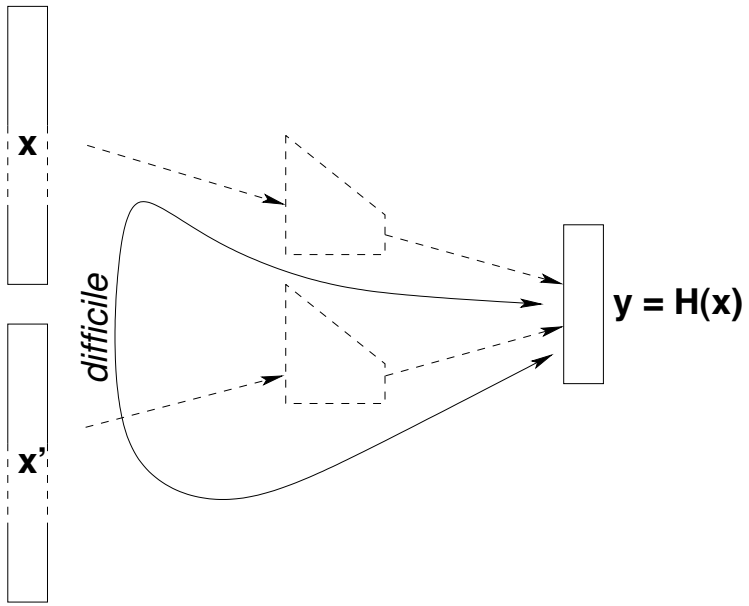
Propriétés des fonction de hachage (2)

- Résistance à la seconde préimage
  - Etant donné x : difficile de trouver $x' \neq x$ / $H(x) = H(x')$
  - Etant donné x : 2^n calculs de hash requis pour trouver x'



Propriétés des fonction de hachage (3)

- Résistance à la collision
  - difficile de trouver x et x' tels que $H(x) = H(x')$.
  - $2^{\frac{n}{2}}$ calculs de hash requis pour trouver un couple (x, x')



Definition (Fonction de Hachage à Sens Unique)

Fonction de hachage à sens unique[a] :

- résistance à la préimage
- résistance à la seconde préimage.

[a] One-Way Hash Function

Definition (Fonction de Hachage résistante aux collisions)

Fonction de hachage résistante aux collisions[a] :

- résistance à la seconde préimage
- résistance aux collisions

[a] Collision Resistant Hash Function

Remarque :
résistance aux collisions $\Longrightarrow$ résistance à la seconde préimage

Paradoxe des anniversaires

Paradoxe des anniversaires

Dans une assemblée de 23 personnes, la probabilité qu'au-moins 2 d'entre-elles aient leur anniversaire le même jour[a] est égale à $\frac{1}{2}$.

[a] en ne tenant pas compte de l'année de naissance

Problème sous jacent :

- Fonction de hachage $H : \{0,1\}^t \longrightarrow \{0,1\}^m$ avec $t > m$.
- **On pose :** $n = 2^m = |\{0,1\}^m|$
- k messages $\{x_i\}_{1 \leq i \leq k}$ aléatoires $(k << n)$
- Question : quelles est la probabilité pour obtenir une collisions à partir des messages x_i ?

Paradoxe des anniversaires (2)

- Hypothèse : les $z_i = H(x_i)$ sont aléatoires
- But : calculer p_k = prob que les $\{z_i\}_{1 \leq i \leq k}$ soient tous $\neq$
  - $k = 2 : p_2 = \frac{n-1}{n}$ ($n-1$ possiblilités dans le choix de z_2)
  - $k = 3 : p_3 = \left(\frac{n-1}{n}\right)\left(\frac{n-2}{n}\right)$
  - $\vdots$
  - $p_k = \left(\frac{n-1}{n}\right)\left(\frac{n-3}{n}\right)\cdots\left(\frac{n-k+1}{n}\right)$

$$p_k = \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right)$$

$\Longrightarrow$ prob qu'il y ait au moins une collision :
$1 - p_k = 1 - \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right)$

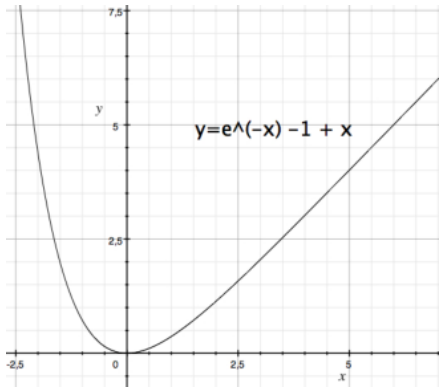
- Comment simplifier pour déterminer k tel que $1 - p_k > \frac{1}{2}$?

L'astuce sioux :-)

$$e^{-x} = 1 - x + \sum_{j=2}^{\infty} (-1)^j \frac{x^j}{j!}$$

$$\forall x \in \mathbb{R}, e^{-x} \geq 1 - x$$

$$\Longrightarrow \forall i \in [1, k-1], e^{-\frac{i}{n}} \geq 1 - \frac{i}{n}$$



$$\prod_{i=1}^{k-1} \left(e^{-\frac{i}{n}} \right) \geq \prod_{i=1}^{k-1} \left(1 - \frac{i}{n} \right) \iff e^{-\frac{(k-1)k}{2n}} \geq p_k$$

$$\iff 1 - p_k \geq 1 - e^{-\frac{(k-1)k}{2n}}$$

Bilan : pour avoir $1 - p_k > \alpha$, il suffit d'avoir $1 - e^{-\frac{(k-1)k}{2n}} > \alpha$!

Retour en terminale

Pour $0 < \alpha < 1$ et $k > 1$:

$$
\begin{aligned}
1 - e^{-\frac{(k-1)k}{2n}} > \alpha &\iff e^{-\frac{(k-1)k}{2n}} < 1 - \alpha \\
&\iff -\frac{(k-1)k}{2n} < \ln(1-\alpha) \\
&\iff \frac{(k-1)k}{2n} > \ln\left(\frac{1}{1-\alpha}\right) \\
&\implies k^2 > (k-1)k > 2n \ln\left(\frac{1}{1-\alpha}\right)
\end{aligned}
$$

Bilan : $1 - p_k > \alpha \implies k > \sqrt{2n \ln\left(\frac{1}{1-\alpha}\right)} = \mathcal{O}(\sqrt{n})$

- $n = 365,\ \alpha = \frac{1}{2} \implies k > 22,49 : k = 23$
- $n = 365,\ \alpha = 99\% \implies k > 57,98$: k=58

Attaque de Yuval (attaque anniversaire)

Soit $H : \{0,1\}^* \longrightarrow \{0,1\}^m$ une fonction de hachage.

- **Input** : M message initial ; $\tilde{M}$ message corrompu
- **Output** : $M' \approx M$ et $\tilde{M}' \approx \tilde{M}$ tels que $H(M') = H(\tilde{M}')$
  1. générer $2^{\frac{m}{2}}$ modifications mineures de M notées M'
     - pour chacune, calculer $H(M')$ et les stocker dans une table T
  2. générer des $\tilde{M}'$, modifications mineures de $\tilde{M}$ jusqu'à ce qu'un $H(\tilde{M}')$ soit dans T (collision)

- Application : répudiation
  - envoyer M et soutenir avoir envoyer $\tilde{M}$

Ce qu'il faut retenir du paradoxe des anniversaires

Concernant les fonctions de hachage

- Soit H la fonction de hachage $H : \{0,1\}^* \longrightarrow \{0,1\}^m$
- La recherche brutale de collisions à plus d'une chance sur 2 d'aboutir après seulement $\mathcal{O}\left(2^{\frac{m}{2}}\right)$ essais !
  - *Remarque* : on est sûr d'aboutir après $\mathcal{O}\left(2^m\right)$ essais

Concernant les algorithmes de chiffrement

- Soit E un algorithme de chiffrement et $K \in \{0,1\}^m$ une clé
- La recherche brutale de la clé dans le chiffrement $E_K(M)$ par une attaque à texte clair connu a plus d'une chance sur 2 d'aboutir après seulement $\mathcal{O}\left(2^{\frac{m}{2}}\right)$ essais !
  - *Remarque* : on est sûr d'aboutir après $\mathcal{O}\left(2^m\right)$ essais

Classification fonctionnelle

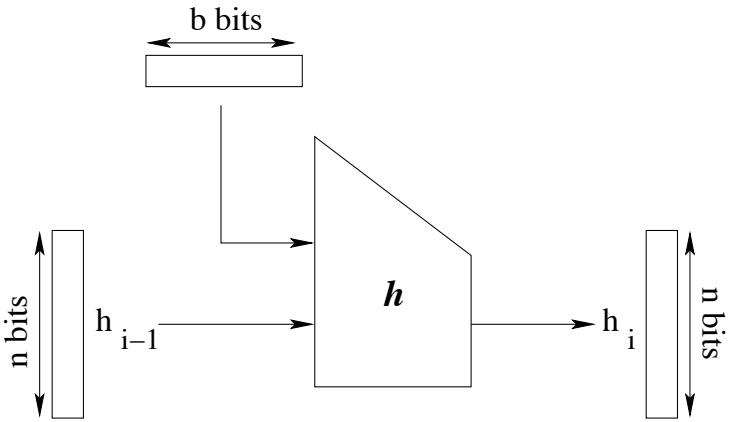
1. Codes de détection de modifications[1] (MDC)
   - utilisent des fonctions de hachage "*sans clé*" publiques
     - Unkeyed Hash Functions
   - vérifient l'intégrite d'une chaîne binaire transmise sur un canal
2. Codes d'authentification de messages[2] (MAC)
   - utilisent des fonctions de hachage "*avec clé*"
     - Keyed Hash Function
   - vérifient l'intégrité d'une chaîne binaire transmise sur un canal
   - authentifient la source d'une donnée

[1] Modification Detection Code
[2] Message Authentification Code

Construction de fonctions de hachage

Construction de Merkle-Damgård

- Se base sur une fonction de compression h
  - $h : \{0,1\}^b \times \{0,1\}^n \longrightarrow \{0,1\}^n$

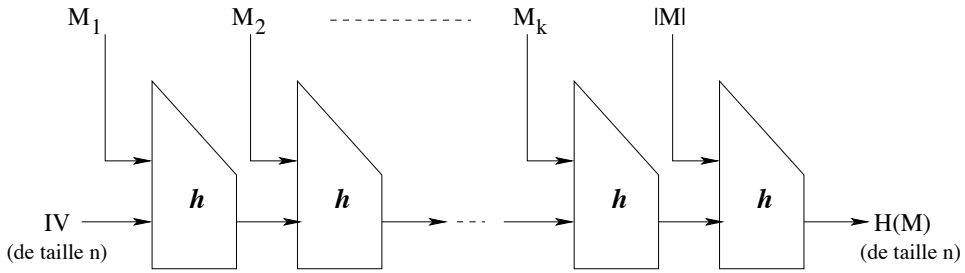


Construction de Merkle-Damgård (2)

- Application au calcul de $H(M)$:
  - Application d'un *padding* à M pour avoir $|M| = k.b$ bits
  - Découpage du message M en blocs de taille b

$$M = M_1 M_2 \dots M_{k-1} M_k \text{ avec } |M_i| = b \quad \forall i \in [1, k]$$

  - Itération de la fonction h (IV : Initial Value) :



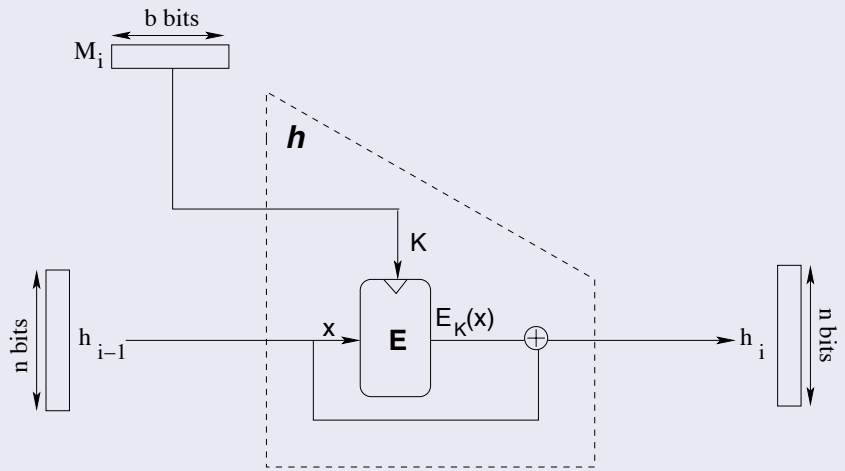
Theorem (Merkle-Damgård)

Si h est résistante aux collisions, alors H l'est aussi !

Construction de la fonction de compression

- "From scratch" : cf MD5,SHA-1 etc...
- A base d'un chiffrement à clé secrète par bloc E :

Construction de Davies-Meyer



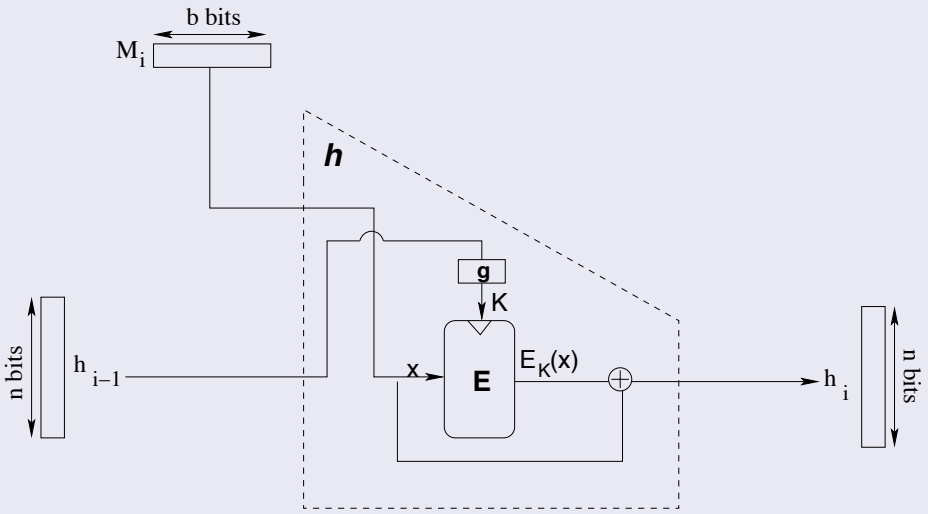
$$h_i = E_{M_i}(h_{i-1}) \oplus h_{i-1}$$

- Résistance à la préimage : attaque de Drew Dean en 1999 exploitant la définition de point fixe sur cette construction

  $\Longrightarrow$ fonctions de hachage moins robustes.

Construction de la fonction de compression (2)

Construction de Matyas-Meyer-Oseas



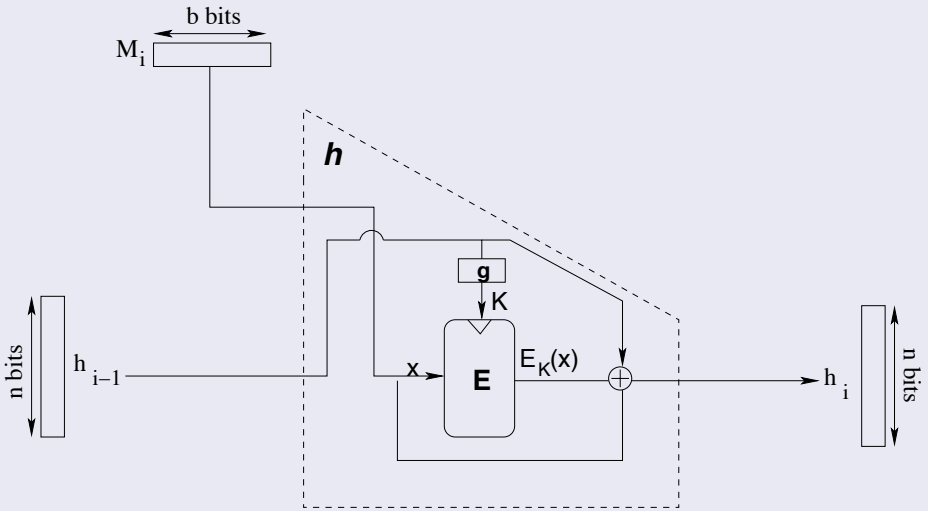
g : fonction d'adaptation à la taille de clé

$$h_i = E_{g(h_{i-1})}(M_i) \oplus M_i$$

- améliorée par Miyaguchi-Preneel.

Construction de la fonction de compression (3)

Construction de Miyaguchi-Preneel

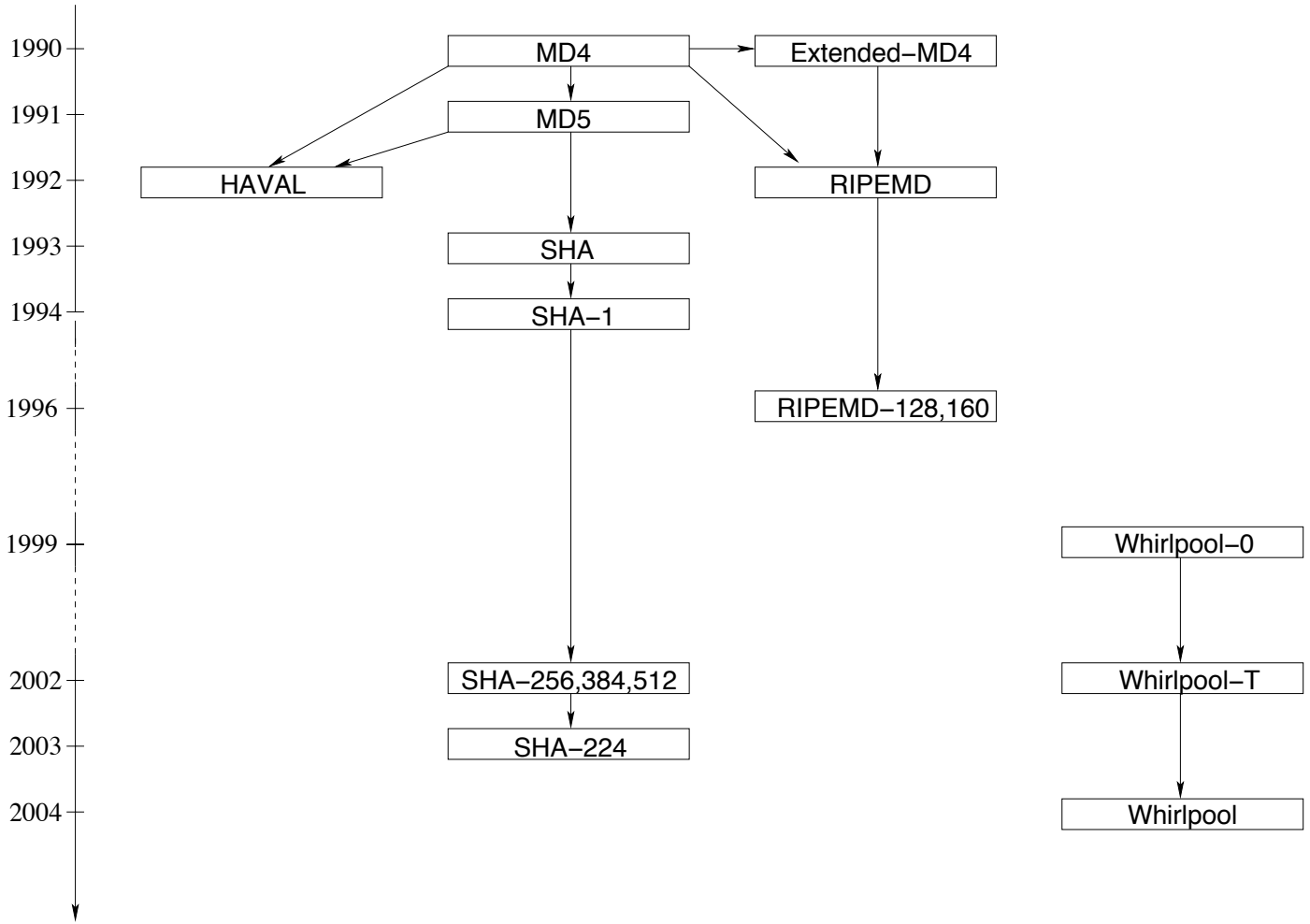


g : fonction d'adaptation à la taille de clé

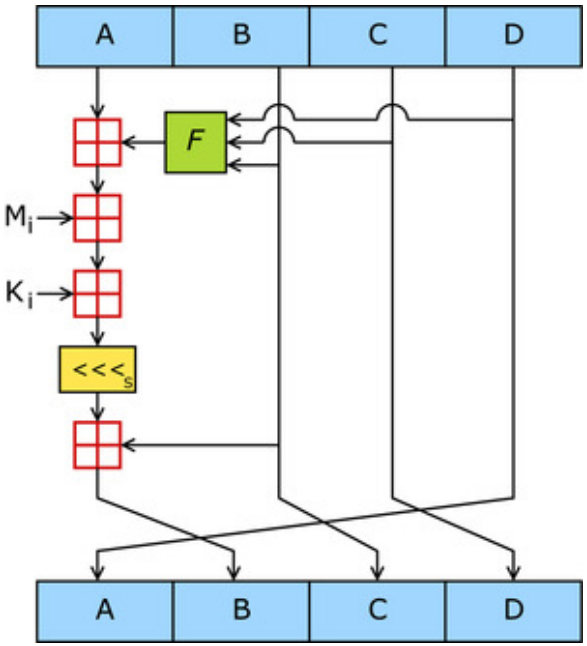
$$h_i = E_{g(h_{i-1})}(M_i) \oplus h_{i-1} \oplus M_i$$

- Variante renforcée de la construction de Matyas-Meyer-Oseas
- Particulièrement robuste d'un point de vue cryptographique [Black, Rogaway et Shrimpton]
- Utilisé dans la fonction de hachage Whirlpool

Historique des principales fonctions de hachages



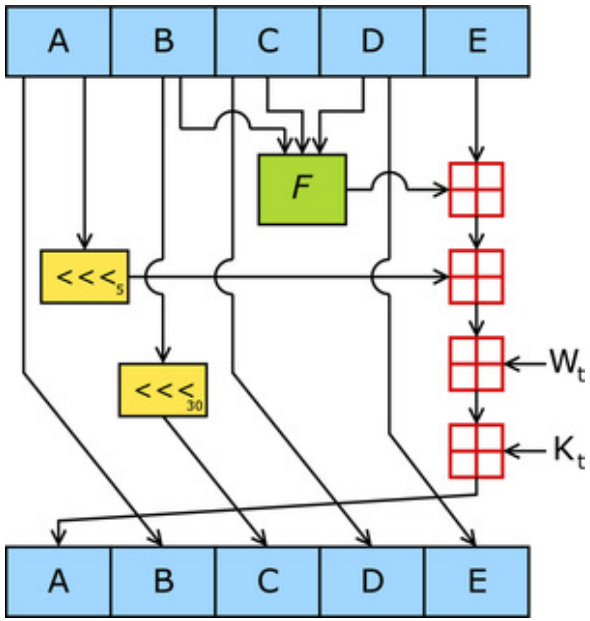
La fonction de compression de MD5



- Proposé par Rivest en 1991
- Blocs de b=512 bits, sortie de n=128 bits
  - 16 sous-blocs M_i de 32 bits
  - 64 constantes fixées K_i
- 64 rondes sur 4 sous-blocs de 32 bits A,B,C,D
  - 4×16 sous-rondes où F prend les valeurs :
    1. $F = (B \text{ AND } C) \text{ OR } (\bar{B} \text{ AND } D)$
    2. $F = (D \text{ AND } B) \text{ OR } (\bar{D} \text{ AND } C)$
    3. $F = B \oplus C \oplus D$
    4. $F = C \oplus (B \text{ OR } \bar{D}))$

- Sécurité de MD5 face aux collisions
  - Force brute : $\mathcal{O}(2^{64})$ (attaque anniversaire)
  - MAIS collisions en $\mathcal{O}(2^{42})$ [Wang04] puis $\mathcal{O}(2^{30})$[Sasaki05]
  - **MD5 n'est plus considéré comme sûr aujourd'hui**

La fonction de compression de SHA-1



- Publié dans FIPS PUB 180-1 par NIST en 1995
- Blocs de b=512 bits, sortie de n=160 bits
  - 16 sous-blocs M_i de 32 bits
  - Etendus en 80 nouveaux blocs W_t
  - 4 constantes fixées K_t
- 80 rondes sur 5 sous-blocs de 32 bits A,B,C,D,E
  - 4×20 sous-rondes où F prend les valeurs :
    1. $F = (B \text{ AND } C) \text{ OR } (\bar{B} \text{ AND } D)$
    2. $F = B \oplus C \oplus D$
    3. $F = (B \text{ AND } C) \oplus (B \text{ AND } D) \oplus (C \text{ AND } D)$
    4. $F = B \oplus C \oplus D$
- Sécurité de SHA-1 face aux collisions
  - Force brute : $\mathcal{O}(2^{80})$ (attaque anniversaire)
  - MAIS collisions possibles en $\mathcal{O}(2^{63})$ [Wang05]
  - Encore largement utilisé (signature, prot. anti-copie XBOX)

La fonction de compression de SHA-256

- Publié dans FIPS PUB 180-2 en 2000
- Basé sur le chiffrement à clé secrète par bloc SHACAL-2
- Blocs de b=512 bits, sortie de n=256 bits
  - 16 sous-blocs M_i de 32 bits
  - Etendus en 64 nouveaux blocs W_t
  - 64 constantes fixées K_t
- 64 rondes sur 8 sous-blocs de 32 bits
- Sécurité de SHA-256
  - Force brute : $\mathcal{O}(2^{128})$ (attaque anniversaire)
  - **Pas d'attaque connues pour le moment**
  - (to be continued)

Whirlpool

- Fonction de hachage recommandée par NESSIE (2004)
- Basé sur la construction de Miyaguchi-Preneel
  - avec $E = W \approx$ Rijndael (à la base de AES)

	Rijndael	**W**
Taille Bloc	128, 160, 192, 224 ou 256	512
Nb rondes	10, 11, 12, 13 ou 14	10
Polynome de réduction sur $\mathbb{F}_{256}$	$x^8+x^4+x^3+x+1$ (**0x11B**)	$x^8+x^4+x^3+x^2+1$ (**0x11D**)
Origine de la S-Box	$t : a \longrightarrow a^{-1}$ sur $\mathbb{F}_{256}$	structure récursive
Origine des constantes de ronde	polynômes x^i sur $\mathbb{F}_{256}$	entrées successives de la S-Box

- Blocs de b=512 bits, sortie de n=512 bits
- 10 rondes
- Sécurité de Whirlpool
  - Force brute : $\mathcal{O}(2^{256})$ (attaque anniversaire)
  - **Pas d'attaque connues pour le moment**
  - (to be continued)

Bilan sur les fonctions de hachage connues

Fonction	Empreinte	Complexité requise	Résistance aux collisions	Compléxité de l'attaque
MD5	128 bits	$\mathcal{O}(2^{64})$	Cassé [Sasaki&al.05]	$\mathcal{O}(2^{30})$
SHA-1	160 bits	$\mathcal{O}(2^{80})$	Cassé (Crypto05 - Wang&al.])	$\mathcal{O}(2^{63})$
HAVAL	256 bits	$\mathcal{O}(2^{128})$	Cassé (Asiacrypt'04)	$\mathcal{O}(2^{10})$
SHA-256	256 bits	$\mathcal{O}(2^{128})$	**Sûr**	
Whirlpool	512 bits	$\mathcal{O}(2^{256})$	**Sûr**	

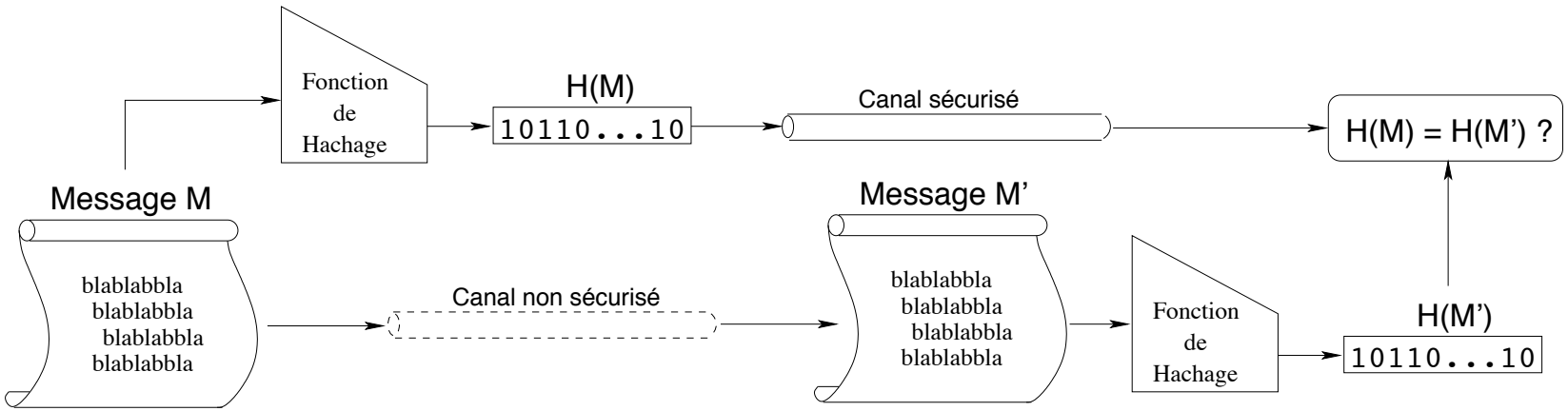
- La résistance aux collision n'est pas toujours suffisante !
- D'autres propriétés de sécurité peuvent être requises !
  - Résistance aux collisions proches
    $\hookrightarrow$ Résistance aux collisions (x, x') ou $d(H(x), H(x'))$ faible
  - Résistance aux pseudo-collisions
    $\hookrightarrow$ Résistance aux collisions lorsque des IV peuvent différer
  - Pseudo-randomness
    $\hookrightarrow$ rendre difficile distinction entre H et une fonction aléatoire

Contrôle d'intégrité et MAC

Contrôle d'intégrité

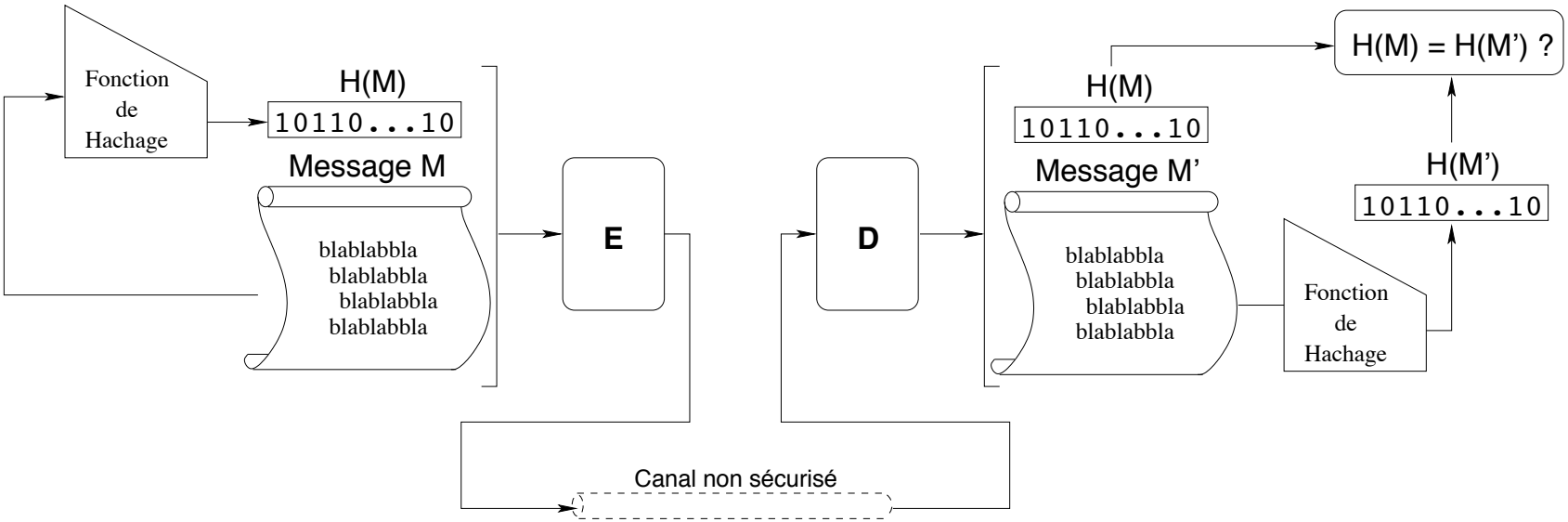
Comment garantir l'intégrité des données sur un canal non sûr ?

- Méthode 1 : en utilisant un canal sécurisé :
  - Transmettre le message sur le canal non fiable
  - Transmettre le checksum sur le canal sûr
  - Vérifier que le hachage du message recu correspond



Contrôle d'intégrité (2)

- Méthode 2 : en utilisant une fonction de chiffrement E
  - chiffrer le message et le checksum avec E
  - transmettre le contenu chiffré



- Méthode 3 : utiliser un MAC
  - transmettre le message et le checksum

Definition (Message Authentification Code - MAC)

- Fonction de hachage **à sens unique** paramétrée par une clé K
- $\forall K$ inconnu et $\begin{cases} (x_1, H_K(x_1)) \\ (x_2, H_K(x_2)) \\ \vdots \\ (x_i, H_K(x_i)) \end{cases}$, il est "impossible"[a] de calculer une paire $(x, H_K(x))$ avec $x \notin \{x_1, x_2, ..., x_i\}$

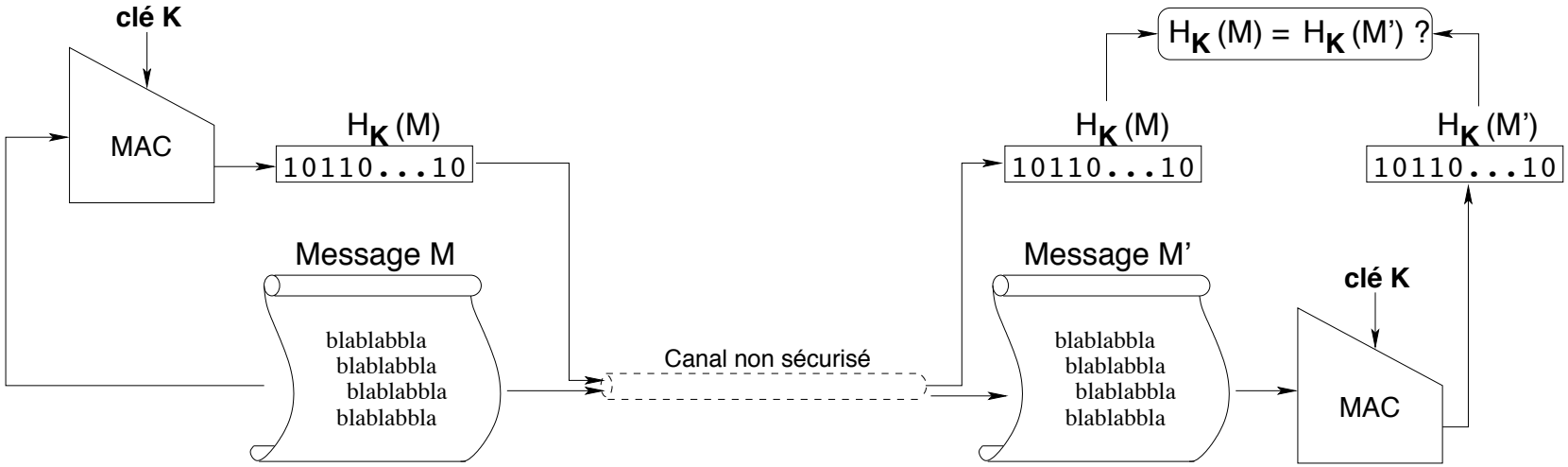
On note $H_K(x)$ l'empreinte de x.

[a] en temps raisonnable

- Seul celui qui possède K peut calculer/vérifier l'empreinte :
  1. Alice et Bob partagent une clé K
  2. Alice envoit à Bob un message M et $r = H_K(M)$
  3. Bob vérifie que le r reçu est bien égal à $H_K(\tilde{M})$ pour $\tilde{M}$ reçu

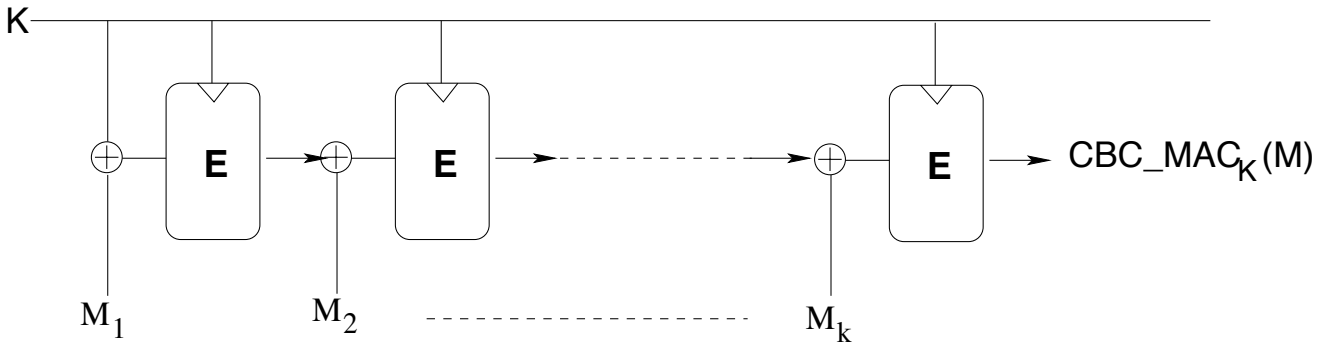
A quoi sert un MAC ?

- Intégrité : Tous les possesseurs de K peuvent vérifier que M n'a pas été modifié puisqu'il correspond à r
- Authentification si la clef n'est distribuée qu'à deux personnes
  - Seuls Alice et Bob peuvent calculer $H_K(M)$



Construction d'un MAC

1. A partir d'un chiffrement à clé secrète par blocs E
   - utilisé en mode CBC ou CFB
   - l'empreinte est le dernier bloc chiffré
   - ex : CBC-MAC-AES



   - AES : 275 Mb/s sur PIII 500 MHz
   - CBC-MAC-AES : 234 Mb/s sur PIII 500 MHz

2. A partir d'une fonction de hachage à sens unique H.

Construction à partir d'une fonction de hachage

- **NE PAS UTILISER** (sur construction de Merkle-Damgard) :
  - $H_K(M) = H(K \bullet M)$
  - $H_K(M) = H(M)$ avec $IV = K$
  - Pourquoi ?

Construction à partir d'une fonction de hachage

- **NE PAS UTILISER** (sur construction de Merkle-Damgard) :
  - $H_K(M) = H(K \bullet M)$
  - $H_K(M) = H(M)$ avec $IV = K$
  - Pourquoi ?
    $\hookrightarrow$ On peut ajouter ce que l'on veut à la fin !

Construction à partir d'une fonction de hachage

- **NE PAS UTILISER** (sur construction de Merkle-Damgard) :
  - $H_K(M) = H(K \bullet M)$
  - $H_K(M) = H(M)$ avec $IV = K$
  - Pourquoi ?
    $\hookrightarrow$ On peut ajouter ce que l'on veut à la fin !
  - $H_K(M) = H(M \bullet K)$ Pourquoi ?

Construction à partir d'une fonction de hachage

- **NE PAS UTILISER** (sur construction de Merkle-Damgard) :
  - $H_K(M) = H(K \bullet M)$
  - $H_K(M) = H(M)$ avec $IV = K$
  - Pourquoi ?
    $\hookrightarrow$ On peut ajouter ce que l'on veut à la fin !
  - $H_K(M) = H(M \bullet K)$ Pourquoi ?
    $\hookrightarrow$ Une collision sur H fournit un MAC valide !
    - $H(x) = H(x') \Longrightarrow H(x \bullet K) = H(x' \bullet K)$

Construction à partir d'une fonction de hachage

- **NE PAS UTILISER** (sur construction de Merkle-Damgard) :
  - $H_K(M) = H(K \bullet M)$
  - $H_K(M) = H(M)$ avec $IV = K$
  - Pourquoi ?
    $\hookrightarrow$ On peut ajouter ce que l'on veut à la fin !
  - $H_K(M) = H(M \bullet K)$ Pourquoi ?
    $\hookrightarrow$ Une collision sur H fournit un MAC valide !
    - $H(x) = H(x') \Longrightarrow H(x \bullet K) = H(x' \bullet K)$
- **Préférer les constructions suivantes** :
  - Enveloppe : $H(K_1 \bullet M \cdot K_2)$
  - NMAC : $H_{K_1}(H_{K_2}(M))$
  - HMAC : $H[K \oplus pad_1 \bullet H((K \oplus pad_2) \bullet M)]$
  - Hybride : $H_{K_1}(x \bullet K_2)$

Signatures électroniques

Notion de signature électronique

But des signatures manuscrites :

- prouver l'identité de leur auteur **et/ou**
- l'accord du signataire avec le contenu du document

Notion de signature électronique

But des signatures manuscrites :

- prouver l'identité de leur auteur **et/ou**
- l'accord du signataire avec le contenu du document

La signature éléctronique dépend du signataire **et** du document !

Propriétés d'une signature éléctronique

- *Authentique* : convainc le destinataire que le signataire a délibérément signé un document
- *Infalsifiable*
- *Non réutilisable* : attachée à un document donné
- *Inaltérable* : toute modification du document est détectable
- *Non reniable* : le signataire ne peut répudier le document

Notion de signature électronique (2)

Intégrité + Authentification + Non-répudiation

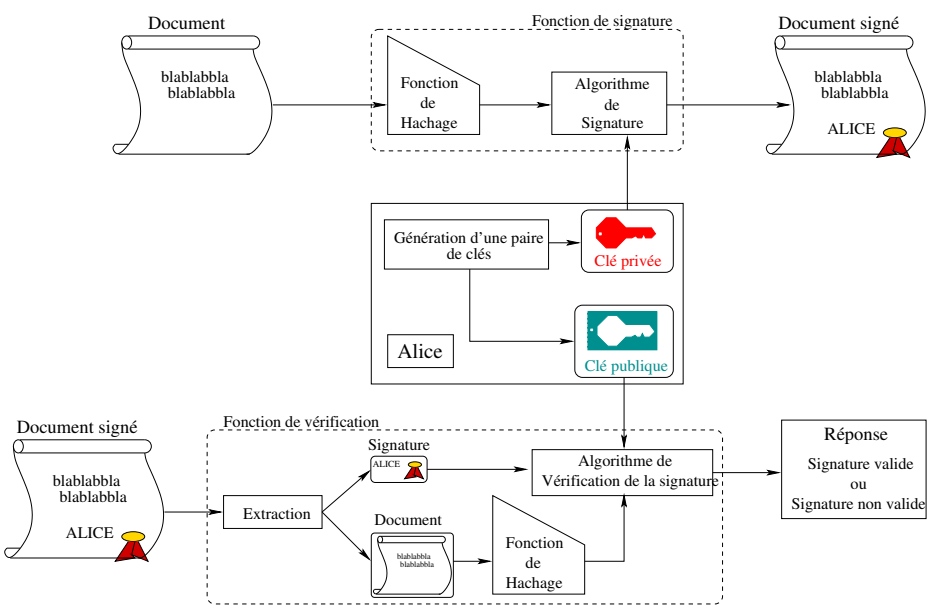
- Parallèle avec les classes d'algorithmes de chiffrement :
  - les MAC sont symétriques
  - les signatures sont asymétriques
- Réalisation pratique :
  - Cryptosystèmes à clef secrète (et arbitre)
  - **Cryptosystèmes à clef publique + fonction de hachage**
- On préfère en effet signer le hachage d'un document !
  - Taille fixe suffisamment petite pour être utilisée efficacement par un cryptosystème à clé publique

Notion de signature électronique (3)

- Protocole de signature électronique sûr :
  - Impossible de falsifier la signature $s(M)$ d'un document M
    - sans connaitre la clef secrète K (resp. K_d)
    - même en disposant de signatures d'autres documents.
    - Attention : impossibilité *pratique*
- Il existe d'autres conditions nécessaires de sécurité !
  - relève davantage des architectures de sécurité des cryptosystèmes à clef publiques (PKI) ou du secret entourant la clé secrète.

Idée générale des signatures électroniques (4)

- Signature utilisant un cryptosystème à clef publique :



- Alice signe M en utilisant :
  - $h_M = H(M)$ le hachage de M
  - sa clé secrète K_d.
  - la fonction de déchiffrement D.
  - Résultat : $s(M) = D_{K_d}(h_M)$
- Document signé : $[M, s(M)]$
- Vérification de $[M, s(M)]$:
  - utilise la clé publique K_e d'Alice et la fonction de chiffrement E
  - $E_{K_e}(s(M)) = h_M =? H(M)$
  - Seule Alice a pu générer $s(M)$

Signature RSA

Génération des paramètres

Identique à la génération des clefs de RSA !

- Alice choisit au hasard deux nombres premiers p et q.
  - Alice calcule $n = p.q$
  - Indicatrice d'Euler : $\varphi(n) = (p-1)(q-1)$
- Alice choisit au hasard un entier e (impair) tel que $1 < e < \varphi(n)$ et $\text{pgcd}(e, \varphi(n)) = 1$
- Alice calcule alors l'entier d tel que $e.d \equiv 1 \mod \varphi(n)$.

Clef publique : (n, e)
Clef secrète : d

On suppose disposer d'une fonction de hachage à sens unique H connue publiquement.

Signature RSA (2)

Génération d'une signature RSA

Alice souhaite signer un document M

- Alice calcule $h_M = H(M)$ (on suppose $0 \leq h_M < n$)
- Signature de M : $s(M) = (h_M)^d \mod n$
- Le document signé est alors $[M, s(M)]$.

Signature RSA (3)

Vérification d'une signature RSA

- Bob reçoit un document signé $[\tilde{M}, s(M)]$ d'Alice.
  - Ce document est potentiellement altéré/illégitime
- Il récupère la clé publique d'Alice (n, e)
- Il calcule $\tilde{h_M} = H(\tilde{M})$
- Il vérifie l'identité : $s(M)^e \equiv \tilde{h_M} \mod n$

En effet : $s(M)^e \equiv (h_M)^{e.d} \mod n \equiv h_M \mod n = h_M$ et si le document est authentique : $h_M = \tilde{h_M}$.

- La sécurité est donc celle du cryptosystème RSA.
- Présentation simpliste et en l'état sujette à des attaques

Signature El Gamal

Génération des paramètres

- Alice choisit :
  - un nombre premier p
  - g une racine primitive modulo p.
  - un entier $a \in \{1, \ldots, p-2\}$ au hasard
- Elle calcule alors $A = g^a \mod p$.

Clef publique : (p, g, A).
Clef secrète : a.

On suppose disposer d'une fonction de hachage à sens unique H connue publiquement.

Signature El Gamal (2)

Génération d'une signature El Gamal

Alice souhaite signer un document M

- Alice calcule $h_M = H(M)$ (on suppose $0 \leq h_M < p$)
- Elle choisit au hasard un entier $k \in [1, p-2]$ tel que $\text{pgcd}(k, p-1) = 1$ ($\Longrightarrow k^{-1} \in \mathbb{Z}_{p-1}$ existe).
- Signature de M : $s(M) = (r, s)$ avec

$$\begin{cases} r = g^k \mod p \\ s = k^{-1}(h_M - a.r) \mod (p-1) \end{cases}$$

- Le document signé est alors $[M, s(M)]$.

Signature El Gamal (3)

Vérification d'une signature El Gamal

- Bob reçoit un document signé $[\tilde{M}, s(M)]$ d'Alice.
  - Rappel : $s(M) = (r, s)$
  - Ce document est potentiellement altéré/illégitime
- Il récupère la clé publique d'Alice (p, g, A)
- Il calcule $\tilde{h_M} = H(\tilde{M})$
- Il vérifie l'identité : $A^r r^s \equiv g^{\tilde{h_M}} \mod p$

En effet,

$$\begin{aligned} A^r r^s &\equiv g^{a.r}.g^{kk^{-1}(h_M - a.r)} \mod p \\ &\equiv g^{h_M} \mod p \end{aligned}$$

Si le document est authentique : $h_M = \tilde{h_M} \Rightarrow g^{h_M} \equiv g^{\tilde{h_M}} \mod p$

Sécurité des signatures El Gamal

- Sécurité intimement liée à DLP dans $\mathbb{F}_p^*$
  - Résolution de DLP dans $\mathbb{F}_p^*$
    $\Longrightarrow$ possibilité de calculer a à partir de A
    $\Longrightarrow$ possibilité d'impersonaliser Alice
- Attention au choix des paramètres.

Le standard DSA

Génération des paramètres

- Alice génère un nb premier **q** de 160 bits ($2^{159} \leq q < 2^{160}$)
- Elle génère un nb premier **p** de 512 à 1024 bits vérifiant :

$$\begin{cases} \exists t \in [0,8] \ / \ 2^{511+64t} < p < 2^{512+64t} \\ q | (p-1) \end{cases} \quad (1)$$

- Soit $\tilde{g}$ une racine primitive modulo p
- Un générateur du sous-groupe de $\mathbb{F}_p^*$ d'ordre q est alors

$$\mathbf{g} = \tilde{g}^{\frac{p-1}{q}} \mod p$$

(1) assure que $\mathbb{F}_p^*$ possède un sous-groupe d'ordre q

Le standard DSA (2)

Génération des paramètres (suite)

Une fois choisis (p, q, g) :

- Alice choisit $a \in \{1, \ldots, q-1\}$
- Elle calcule $A = g^a \mod p$

Clef publique : (p, q, g, A).
Clef secrète : a

Le problème du logarithme discret sous-jacent se passe dans le groupe d'ordre q.

Le standard DSA (3)

Génération d'une signature DSA

Alice souhaite signer un document M :

- Alice calcule $h_M = H(M)$ (on suppose $1 \leq h_M < q-1$)
- Elle choisit un entier $k \in \{1, \ldots, q-1\}$
- Signature de M : $s(M) = (r, s)$ avec

$$\begin{cases} r = (g^k \mod p) \mod q \\ s = k^{-1}(h_M + a.r) \mod q \end{cases}$$

- Le document signé est alors $[M, s(M)]$.

Le standard DSA (4)

Vérification d'une signature DSA

- Bob reçoit un document signé $[\tilde{M}, s(M) = (r, s)]$ d'Alice.
- Il récupère la clé publique d'Alice (p, q, g, A)
- Il vérifie que les formats sont respectés : $1 \leq r, s \leq q - 1$
- Il calcule $\tilde{h_M} = H(\tilde{M})$
- Il vérifie l'identité :

$$r \equiv \left[\left(g^{s^{-1}\tilde{h_M}} \mod q \right) . \left(A^{rs^{-1}} \mod q \right) \mod p \right] \mod q.$$

Ce qu'il reste à voir...

- Comment récupérer et être sûr d'une clé publique ?
  - En effet, Oscar peut se faire passer pour le destinataire !
  - Il faut un moyen de **prouver** la correspondance entre une clé publique et une personne !
  - Solution : Architecture PKI !
- Utilisation de la cryptographie pour sécuriser les infrastructures et leurs accès
  - Kerberos
  - Systèmes hybrides : PGP/SSH
- Legislation française sur la cryptologie
- Programmation sécurisée