Les fonctions de hachage, un domaine à la mode

JSSI 2009

Thomas Peyrin (Ingenico)

17 mars 2009 - Paris

Outline

Qu'est-ce qu'une fonction de hachage

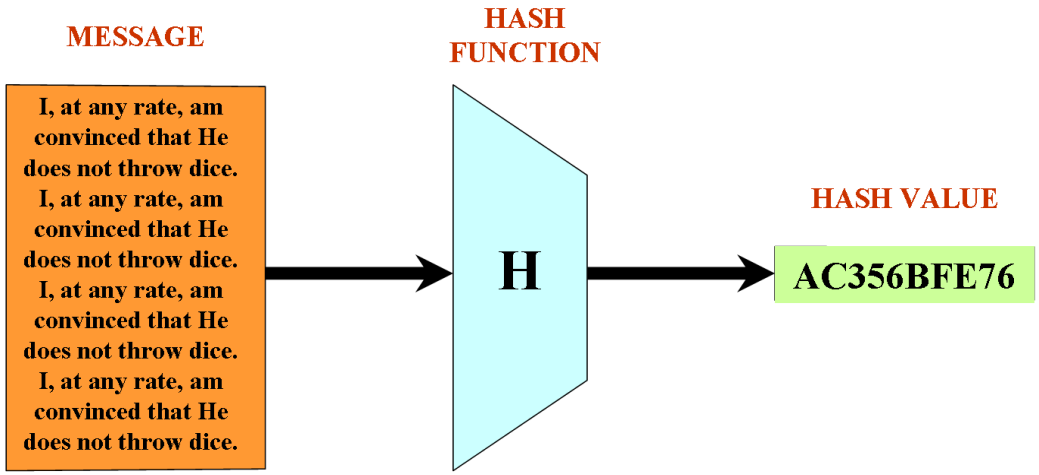
Comment construire une fonction de hachage ?

Les attaques contre la famille MD-SHA

Le concours SHA-3 du NIST

Outline

Qu'est-ce qu'une fonction de hachage cryptographique?



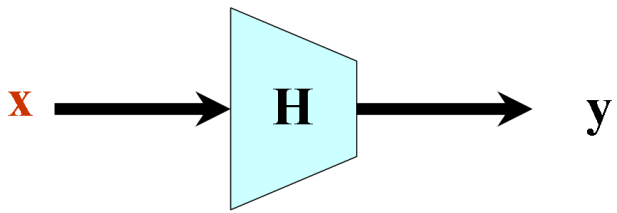
- H fait correspondre **une entrée de taille arbitraire** (un message M) à **une sortie de taille fixe** (typiquement 128, 160 ou 256 bits).
- H doit être rapide et facile à calculer.
- $\neq$ d'une fonction de hachage classique (base de données, ...)
- **aucun secret !**

Les critères de sécurité

- **résistance à la recherche de preimage:** soit un haché y donné, l'attaquant ne doit pas être en mesure de trouver un message x tel que $H(x) = y$, en moins de $\theta(2^n)$ opérations.

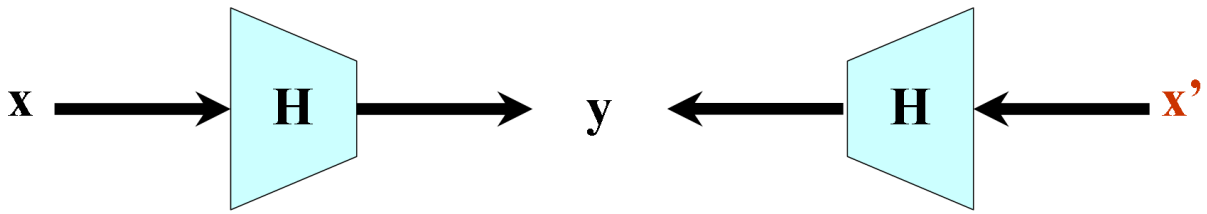
- **résistance à la recherche de 2nd preimage:** soit un message x donné et son haché y correspondant ($H(x) = y$), l'attaquant ne doit pas être en mesure de trouver un nouveau message $x' \neq x$ tel que $H(x') = y$, en moins de $\theta(2^n)$ opérations.

- **résistance à la recherche de collision:** l'attaquant ne doit pas être en mesure de trouver deux messages différents (x, x') tels que $H(x) = H(x')$, en moins de $\theta(2^{n/2})$ opérations (paradoxe des anniversaires).



Les critères de sécurité

- **résistance à la recherche de preimage:** soit un haché y donné, l'attaquant ne doit pas être en mesure de trouver un message x tel que $H(x) = y$, en moins de $\theta(2^n)$ opérations.
- **résistance à la recherche de 2nd preimage:** soit un message x donné et son haché y correspondant ($H(x) = y$), l'attaquant ne doit pas être en mesure de trouver un nouveau message $x' \neq x$ tel que $H(x') = y$, en moins de $\theta(2^n)$ opérations.
- **résistance à la recherche de collision:** l'attaquant ne doit pas être en mesure de trouver deux messages différents (x, x') tels que $H(x) = H(x')$, en moins de $\theta(2^{n/2})$ opérations (paradoxe des anniversaires).

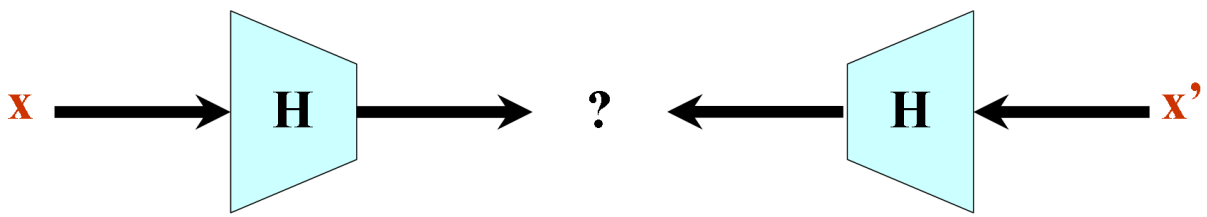


Les critères de sécurité

- **résistance à la recherche de preimage:** soit un haché y donné, l'attaquant ne doit pas être en mesure de trouver un message x tel que $H(x) = y$, en moins de $\theta(2^n)$ opérations.

- **résistance à la recherche de 2nd preimage:** soit un message x donné et son haché y correspondant ($H(x) = y$), l'attaquant ne doit pas être en mesure de trouver un nouveau message $x' \neq x$ tel que $H(x') = y$, en moins de $\theta(2^n)$ opérations.

- **résistance à la recherche de collision:** l'attaquant ne doit pas être en mesure de trouver deux messages différents (x, x') tels que $H(x) = H(x')$, en moins de $\theta(2^{n/2})$ opérations (paradoxe des anniversaires).



Le paradoxe des anniversaires

PROBLÈME: de combien de personne doit on disposer pour avoir une bonne probabilité ($P > 0.5$) que deux personnes aient la même date d'anniversaire ?

- ... 365 possibilités de dates (approximativement uniforme !)
- ... la réponse n'est pas très intuitive!
- ...

Le paradoxe des anniversaires

PROBLÈME: de combien de personne doit on disposer pour avoir une bonne probabilité ($P > 0.5$) que deux personnes aient la même date d'anniversaire ?

- ... 365 possibilités de dates (approximativement uniforme !)
- ... la réponse n'est pas très intuitive!
- ...

RÉPONSE: seulement 23 personnes pour $P > 0.5$!

Applications (1)

Nombreuses applications:

- **signatures**: un schéma de signature possède deux entrées : une clé privée et un message à signer. Cette fonction permet à tous les utilisateurs connaissant la clé publique correspondante à la clé privée de vérifier l'intégrité/l'authenticité du message signé. Les fonctions de hachage permettent dans ce cas d'améliorer la vitesse et la sécurité des schémas de signature.
- **Message Authentication Codes**: un MAC possède deux entrées : une clé secrète (partagée entre deux personnes A et B) et un message à signer. Cette fonction permet pour A de vérifier l'intégrité/l'authenticité du message signé par B. Par exemple, HMAC utilise une fonction de hachage comme composant principal et est utilisé dans SSL/TLS, IPSec, ...

Applications (2)

Nombreuses applications:

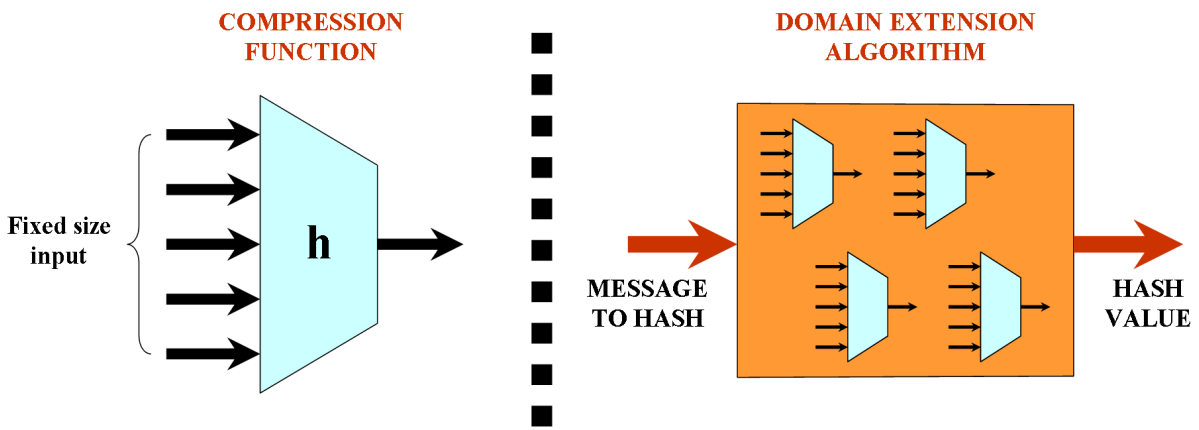
- **protection de mots de passe**: au lieu de stocker tous les mots de passe dans le serveur pour l'authentification d'utilisateurs, il vaut mieux stocker le haché des mots de passe.
- **confirmation de connaissance/engagement sur une valeur**: si quelqu'un veut prouver qu'il connait un secret sans le révéler dans l'immédiat, il peut publier le haché de ce secret. Une fois le secret révélé, il est facile de vérifier ses dires.
- **générateur de suites pseudo-aléatoires/dérivation de clés**: les fonctions de hachage sont utilisées pour détruire toute structure qu'il pourrait exister dans les entrées, tout en préservant leur entropie. Cela permet par exemple de casser la structure algébrique d'un objet.

Outline

Squelette général

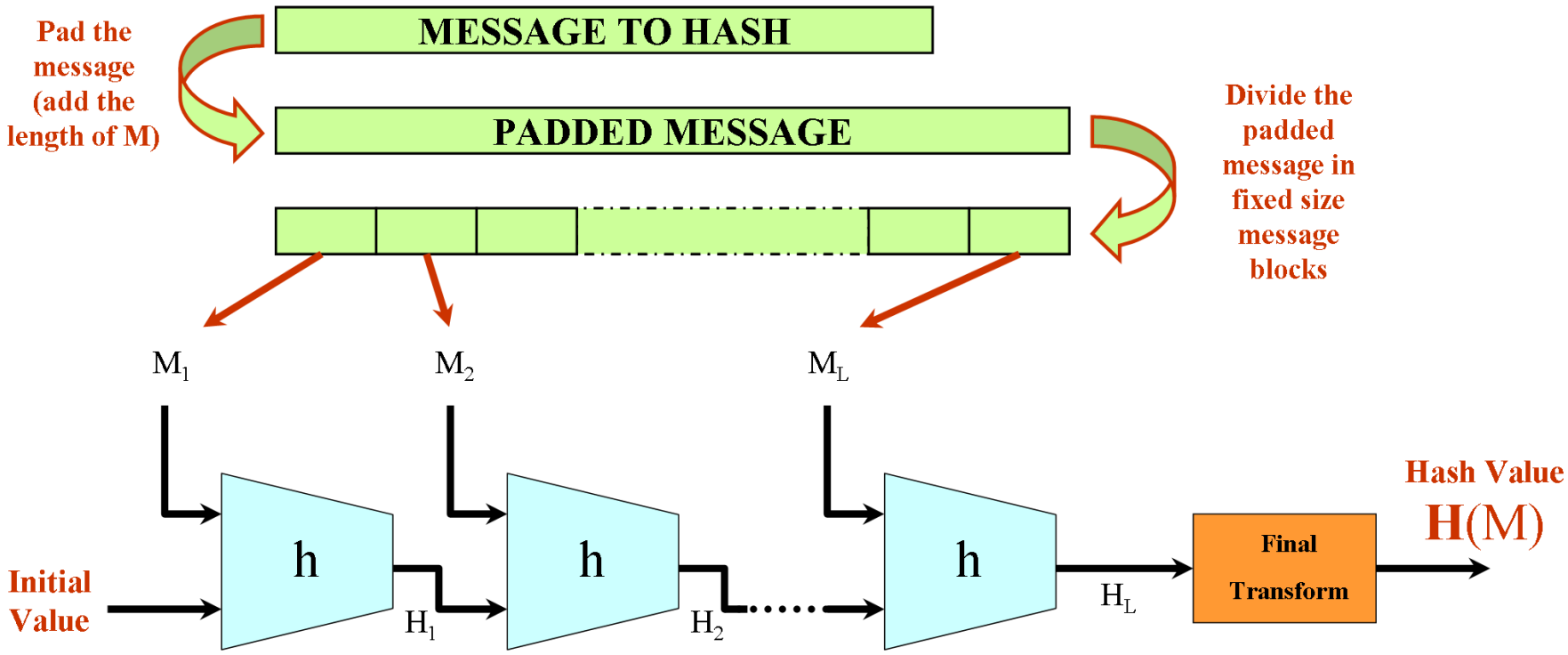
Pour des raisons historiques, quasiment toutes les fonctions de hachage sont composées de deux éléments:

- **une fonction de compression** *h***:** une fonction dont la taille d'entrée et de sortie est fixe.
- **un algorithme d'extension de domaine:** un processus (généralement itératif) utilisant la fonction de compression *h* pour que la fonction de hachage *H* puisse hacher des messages de taille arbitraire.



L'algorithme d'extension de domaine Merkle-Damgård

L'algorithme d'extension de domaine de très loin le plus connu et le plus utilisé jusqu'à maintenant est **l'algorithme itératif de Merkle-Damgård** [Merkle Damgård-89]



Avantages et défauts de l'algorithme de Merkle-Damgård

- **Avantage:** l'algorithme de Merkle-Damgård permet de réduire le problème de la construction d'une fonction de hachage résistante à la recherche de collision/preimage à celui de la construction d'une fonction de compression résistante à la recherche de collision/preimage :
  si l'on ne trouve pas de collision/preimage pour h, alors on ne trouve pas de collision/preimage pour H.
- **Défaut:** la résistance à la recherche de collision/preimage n'est pas tout. Récemment (2004-2005) des chercheurs ont montré que ce processus n'est pas une candidat d'extension de domaine idéal : multi-collision, 2nd preimages longues, ...

Par conséquent, **plusieurs nouveaux candidats ont vu le jour depuis**, fournissant de meilleures preuves de sécurité mais au prix d'une complexité légèrement accrue.

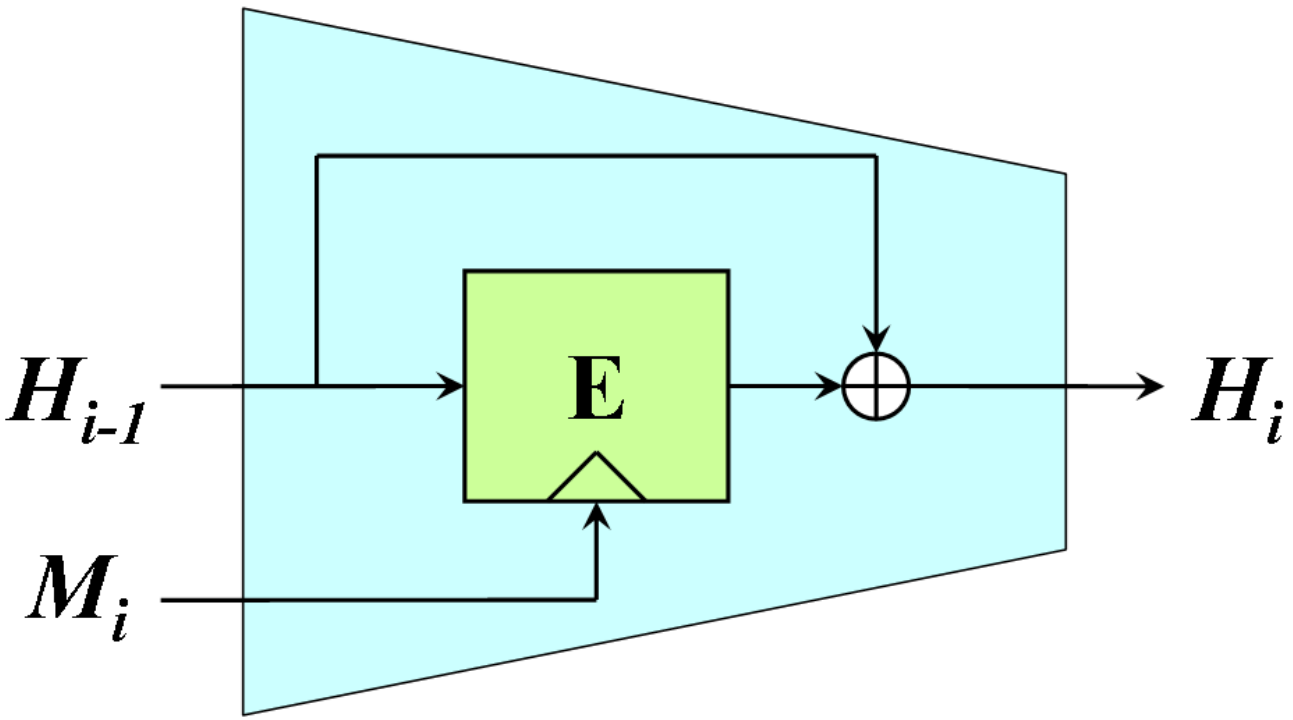
Trois grands groupes de fonctions de compression

Trois grands groupes de fonctions de compression:

- **ad-hoc:** grande majorité des fonctions utilisées et standardisées (MD, SHA, ...). Très rapides mais on ne peut avoir confiance en leur sécurité qu'après une longue analyse par la communauté des cryptographes.
- **fondées sur un chiffrement par bloc:** un peu plus lentes que les fonctions ad-hoc, elles permettent souvent de prouver leur sécurité en supposant le chiffrement par bloc utilisé comme idéal.
- **fondées sur un problème difficile:** généralement très lentes, leur sécurité repose totalement sur la difficulté de résoudre un problème difficile (factorisation, résolution de systèmes algébriques, etc.).

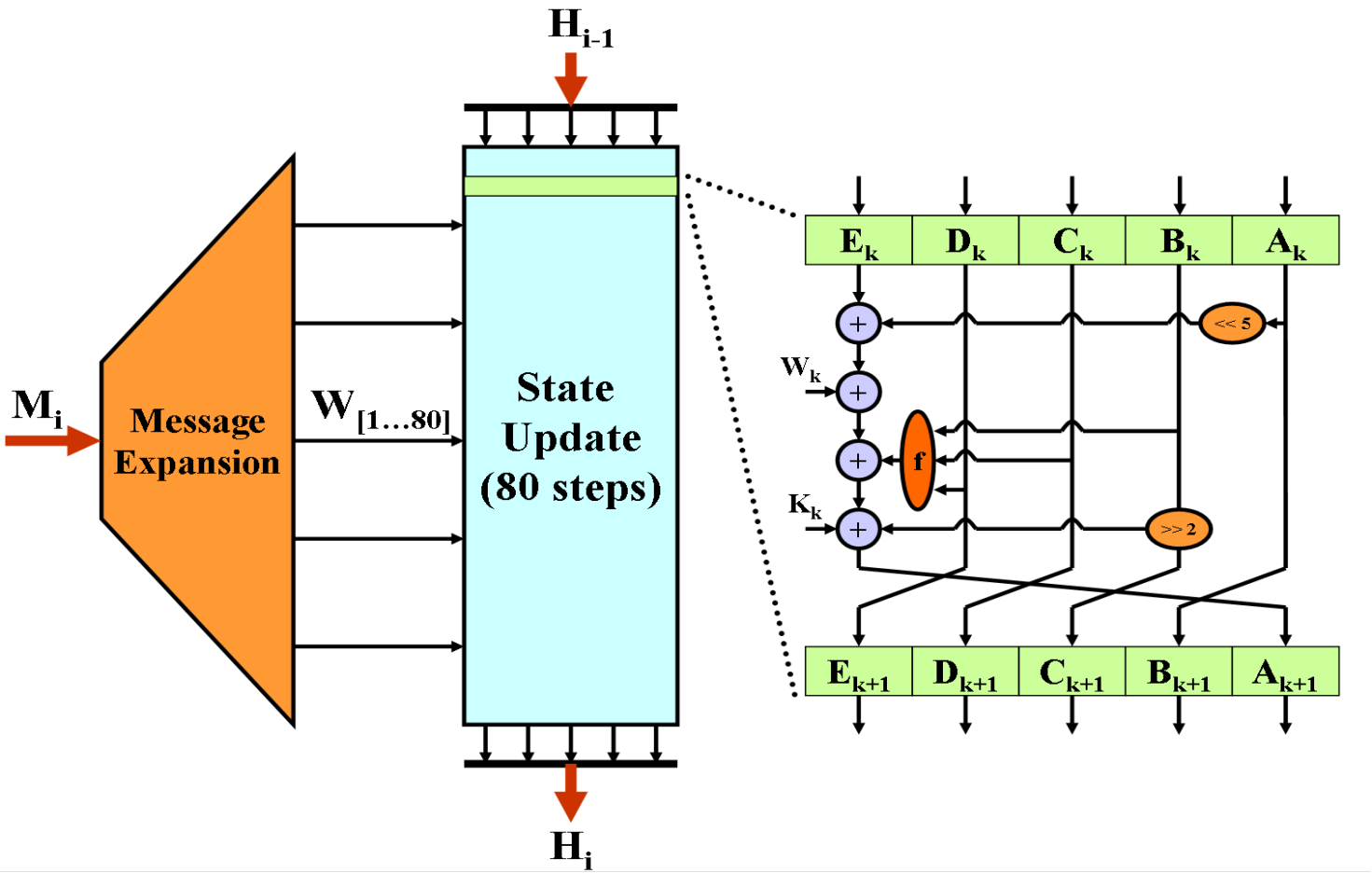
Fonctions de compressions ad-hoc

La famille MD/SHA (MD4, MD5, SHA-0, SHA-1, SHA-2, ...)



Fonctions de compressions ad-hoc

La famille MD/SHA (MD4, MD5, SHA-0, SHA-1, SHA-2, ...)

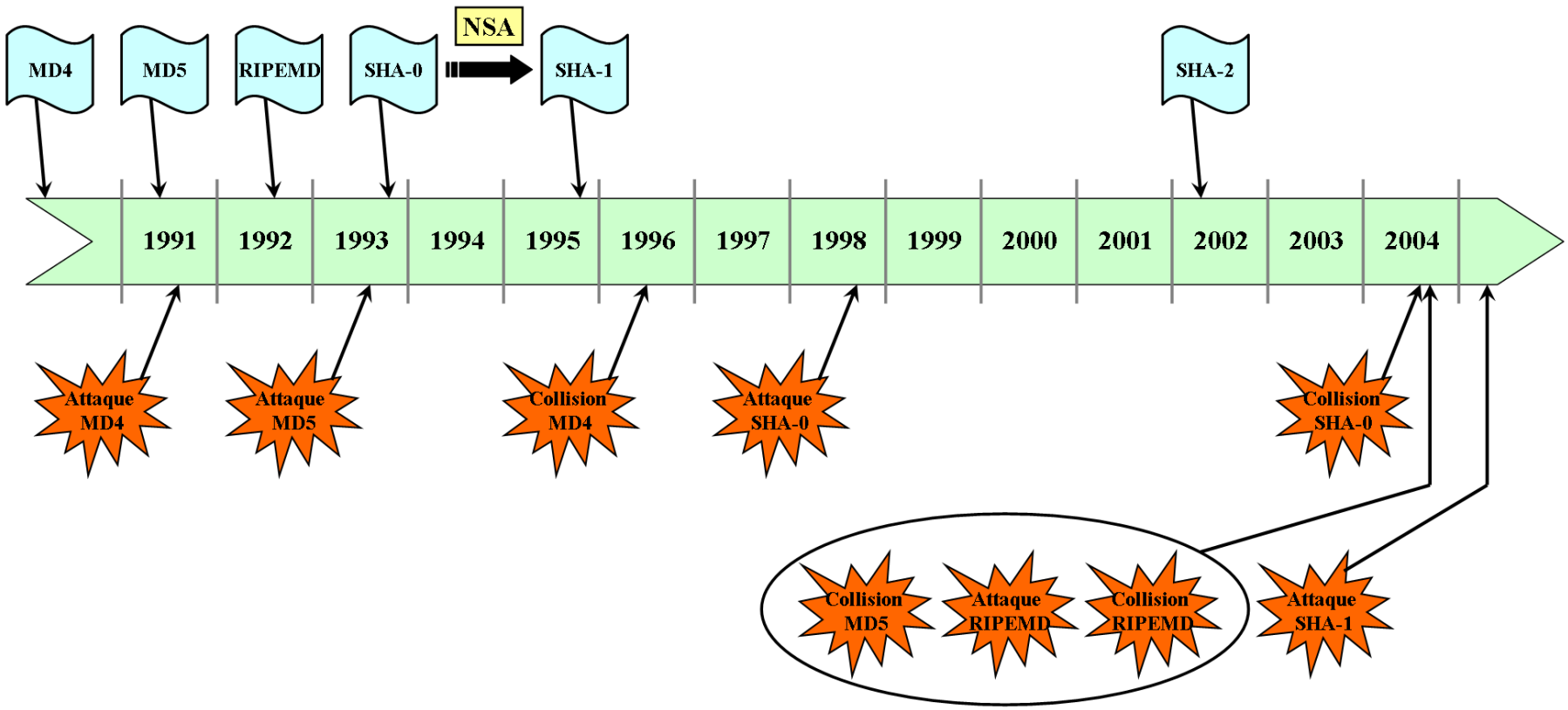


Efficacité des fonctions de hachage

Type	Algorithme	Vitesse (MiB/seconde)
Block Cipher	DES	34
Block Cipher	IDEA	36
Block Cipher	AES	90
Hash Function	CRC-32	256
Hash Function	MD5	258
Hash Function	SHA-1	155
Hash Function	SHA-2	81
MAC	HMAC(SHA-1)	152
MAC	CBC-MAC(AES)	86

Outline

Premières attaques par collision



Les attaques par collision de Wang et al.

Coup de tonnerre dans le petit monde de la cryptographie à la conférence CRYPTO 2004 (août 2004): **une collision a été calculée pour MD4, MD5 et RIPE-MD !**

Quelques mois plus tard (février 2005), **SHA-1 est cassée** ! ... certes uniquement en théorie mais personne ne s'y attendait !

- Aboutissement d'une dizaine d'années de travail de l'équipe de Wang ...
- ... une grande partie de l'attaque a été trouvée à la main !
- la même méthode s'applique à MD4, MD5, SHA-0, SHA-1, RIPE-MD (aie !)
- mais tout restera longtemps flou : attaque très complexe, très mal expliquée, des erreurs, ...

Les nouvelles améliorations

Depuis, de nombreuses améliorations ont vu le jour :

- **mieux comprendre** ce qui se cache derrière les attaques, donner un socle plus théorique
- **automatiser** les attaques (pour ne pas avoir à tout refaire à la main)
- **améliorer** les résultats : pousser ces méthodes jusqu'à leur limite
- étudier d'autres cas que la collision : attaquer les 2nd-preimages, les preimages, HMAC, des protocoles, collision sur des certificats, etc.

La sécurité actuelle des fonctions de hachage de la famille MD-SHA

Algorithme	**Cas idéal**	**Complexité de l'attaque**
MD4 (1990)	2^{64}	2^{1}
MD5 (1992)	2^{64}	2^{16}
SHA-0 (1993)	2^{80}	2^{33}
SHA-1 (1995)	2^{80}	2^{60}
SHA-2 (2002)	2^{128}	pas d'attaque (pour l'instant !)

Attaque certificats X.509 signés avec MD5

”Outdated Security Threatens Web Commerce.”
The New York Times, 30 décembre 2008.

”SSL broken! Hackers create rogue CA certificate using MD5 collisions !”
ZDNET, 30 décembre 2008.

”L’algorithme MD5, utilisé par de nombreux sites, n’est pas fiable.”
Le monde, 2 janvier 2009.

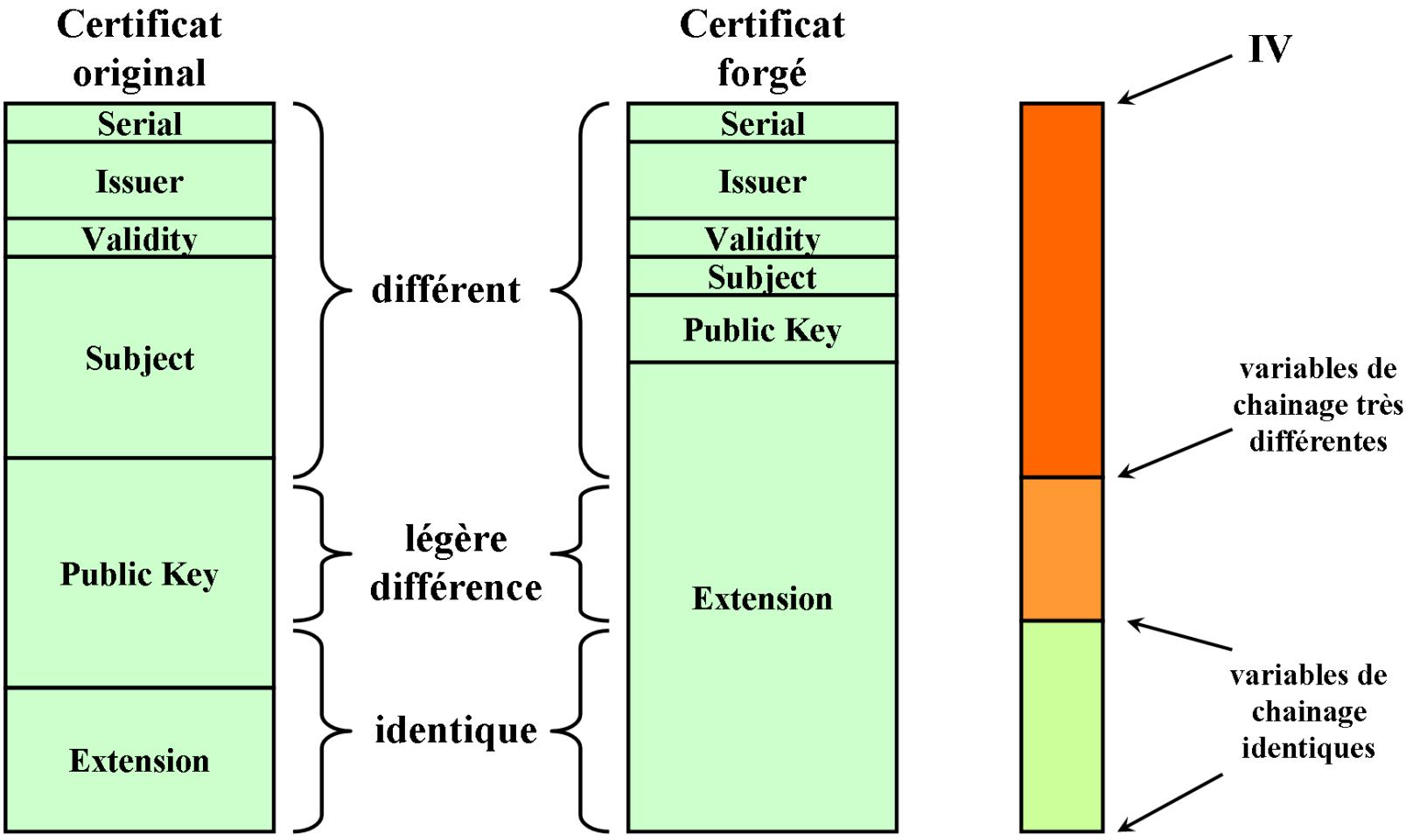
Que se cache t’il derrière cette attaque ?

Attaque certificats X.509 signés avec MD5

Ce qui était déjà connu:

- deux messages qui collisionent avec MD5 (Wang 2004).
- deux documents PDF qui collisionnent avec MD5.
- deux fichiers exécutables qui collisionnent avec MD5.
- deux certificats X.509 avec deux clés RSA différentes qui collisionnent avec MD5 (Lenstra et al. 2005).
- deux certificats X.509 avec deux clés RSA différentes et deux identités différentes qui collisionnent avec MD5 (Stevens et al. 2006). Mais le prefixe du certificat doit pouvoir être contrôlé par l'attaquant (période de validité, numéro de série, ...).

Attaque certificats X.509 signés avec MD5



Attaque certificats X.509 signés avec MD5

Ce qui est nouveau:

- les détails pour que ça marche avec une vraie implémentation du CA !
- utilisation d'un cluster de 200 PS3 pour rendre l'attaque pratique !
- mais **PAS GRAND CHOSE** de neuf au niveau crypto !

Le déroulement:

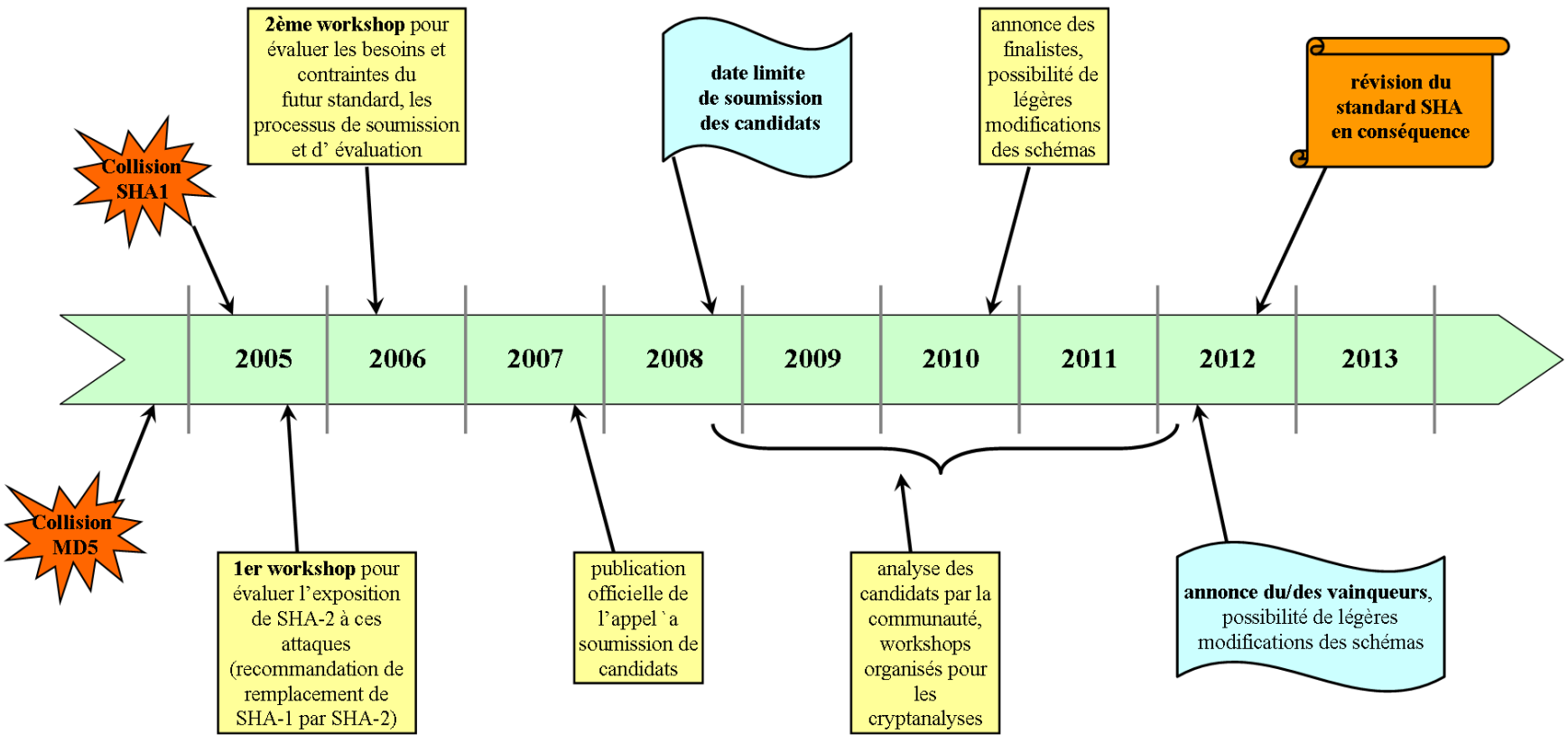
- prédire le serial number qui sera utilisé dans x jours par le CA.
- calculer en moins de x jours deux certificats qui collisionnent avec ce serial number.
- utiliser des autres participants pour forcer le bon serial number.

Outline

Problématiques du concours

- **pourquoi ?**
  - doit on remplacer SHA-2 ou juste lui succéder ?
  - doit on éviter de continuer dans la lignée MD-SHA ?
- **quand ?**
  - doit on attendre que la connaissance dans le domaine s'améliore avant de débuter, au risque de se faire surprendre par l'évolution rapide des attaques ?
  - combien de temps doit durer le concours ?
- **quoi ?**
  - qu'est ce que l'on attend du futur standard ?
  - doit on beaucoup contraindre les candidats ?
  - faire un concours d'algorithmes d'extension de domaine et un concours de fonctions de compression séparés ?
  - un concours par critère de sécurité ?
  - sur quels critères de sécurité doit on se focaliser ?
  - qu'est ce que l'on considère comme étant une attaque ?

Roadmap du concours SHA-3



Où en est-on aujourd'hui ?

- **61 candidats soumis** en octobre 2008. **51 dossiers acceptés** (les autres dossiers étant incomplets).
- pour l'instant **10 candidats déjà cassés**, et une dizaine d'autres "blessés".
- il reste **une trentaine de candidats potentiellement finalistes**.
- 3 ou 4 candidats "stars" (équipe renommée, algorithme médiatisé, etc.)

Des questions ?