

Implémentation de fonctions en éponge

Amélie Guémon
Ida Tucker

<amelie.guemon@etu.u-bordeaux.fr>

<ida.tucker@etu.u-bordeaux.fr>

Master CSI, Université de Bordeaux, France

25 avril 2016



- 1 Merkle-Damgård et ses applications
- 2 Attaques contre Merkle-Damgård
- 3 Fonctions de hachage en éponge

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

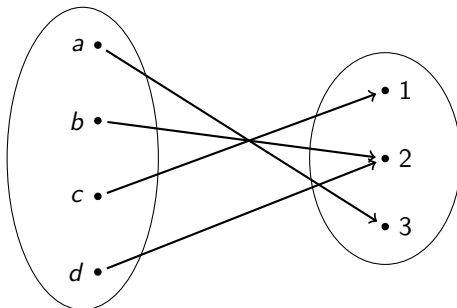
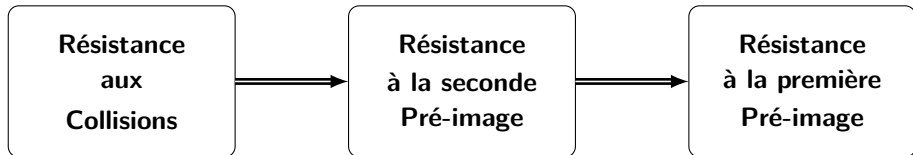


Figure: Surjectivité d'une fonction de Hachage

- **Résistance à la Pré-image** : Pour une empreinte y donné, il est dur de trouver une pré-image $x \in f^{-1}(H)$ tel que $y = H(x)$.
- **Résistance à la Seconde Pré-image** : Pour un clair x , il est dur de trouver un autre clair x' , $x' \neq x$ tel que $H(x) = H(x')$.
- **Résistance aux Collisions** : Il est dur de trouver 2 messages clairs x et x' avec $x \neq x'$ tel que $H(x) = H(x')$.



- **Merkle-Damgård Padding** : Représenté par $10 * |M|$, il faut rajouter un 1, puis un nombre fini de 0, de telle sorte que la longueur du résultat soit congru à $448 \bmod 512$. Ensuite, on y ajoute la longueur du message, sur 64 bits.

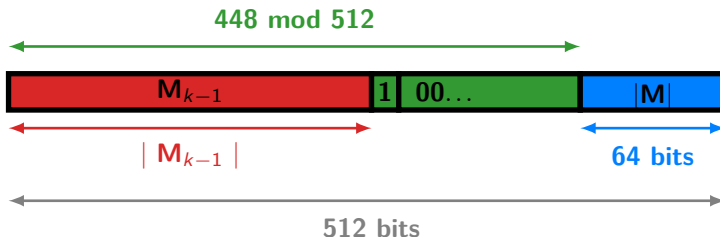


Figure: Merkle-Damgård padding.

- 1 Merkle-Damgård et ses applications
- 2 Attaques contre Merkle-Damgård
- 3 Fonctions de hachage en éponge

Merkle-Damgård définit des fonctions de hachage en **itérant des fonctions de compression**.

- **Fonction de compression** : part d'un ensemble fini vers un ensemble fini réduit.
- **Fonction de hachage** : part d'un ensemble infini vers un ensemble fini.

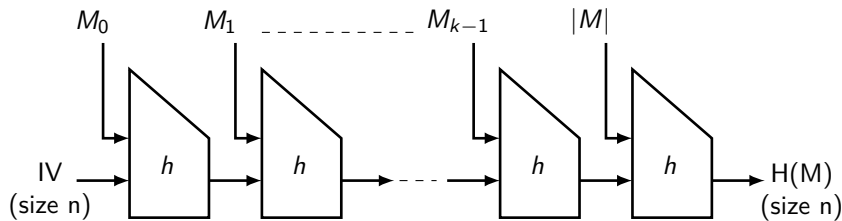


Figure: Merkle-Damgård construction.

- **Théorème :** Si la fonction de compression h utilisée par la fonction de hachage H est résistante aux collisions, alors H l'est aussi.

- MD5 : $\{0,1\}^* \rightarrow \{0,1\}^{128}$
- SHA1 : $\{0,1\}^* \rightarrow \{0,1\}^{160}$

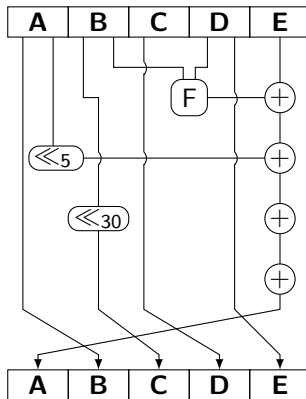


Figure: Le $i^{\text{ème}}$ tour de la fonction de compression de SHA-1 ($0 \leq i \leq 79$).

1 Merkle-Damgård et ses applications

2 **Attaques contre Merkle-Damgård**

3 Fonctions de hachage en éponge

Objectif : Trouver des collisions pour une fonction de hachage

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

Algorithme :

- Choisir de façon aléatoire un ensemble \mathcal{E} de messages dans $\{0,1\}^*$ de cardinal K .
- Tester pour tous les couples de messages $(M, M') \in \mathcal{E}^2$ tels que $M \neq M'$ si $H(M) = H(M')$.

Si :

- \mathcal{E} est choisi de façon aléatoire et uniforme parmi tous les messages possibles.
- $N = \#\{0,1\}^n = 2^n$ est le nombre total de condensés possibles.

Alors :

La probabilité de trouver des collisions au bout de $K = \#\mathcal{E}$ essais ne dépend que de la taille du condensé.

Pour une probabilité de trouver des collisions $> 50\%$:

$$K \approx 1.18 \times \sqrt{N} \quad (1)$$

Application à MD5 et SHA1 :

- taille du condensé de MD5 : 128 bits.

$P_{Success} > 50\%$ pour 2^{64} calculs de condensés.

- taille du condensé de SHA1 : 160 bits.

$P_{Success} > 50\%$ pour 2^{80} calculs de condensés.

Recommandations : $n \geq 128$, voir $n \geq 160$

Définition :

Une fonction de hachage est dite *cassée* lorsqu'il existe une attaque connue permettant de trouver des collisions ayant une complexité moindre que l'attaque par force brute.

État actuel :

- La *cryptanalyse différentielle* a cassé de nombreuses fonctions de hachage itérées, basées sur Merkle-Damgård (dont SHA0, SHA1, MD5).
- Des attaques génériques contre Merkle-Damgård existent.

- 1 Merkle-Damgård et ses applications
- 2 Attaques contre Merkle-Damgård
- 3 Fonctions de hachage en éponge**

- $R||C$: État de la fonction.
- r : tailles des blocs M_i traités.
- c : capacité.
- R : partie externe de l'état, les premier r bits de l'état.
- C : partie interne de l'état, de taille $c = b - r$ bits.

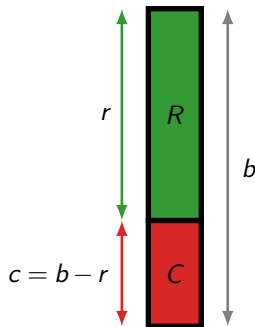


Figure: État d'une fonction en éponge.

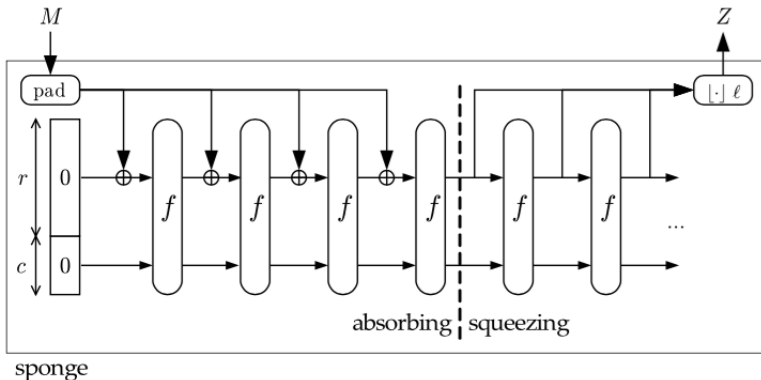


Figure: Construction de fonctions en éponge $Z = \text{SPONGE}[f, \text{pad}, r](M, l)$

Paramètres de la fonction Keccak $[b, n_r]$:

- w : la longueur fixée des chaînes de bits permutés.
 $w = 2^l$ bits, avec $0 \leq l \leq 6$.
- n_r : le nombre d'itérations effectuées des routines internes.
 $n_r = 12 + 2 \times l$.

État de la fonction Keccak

- État constitué de $b = 5 \times 5 \times w$ bits.
- État représenté sous forme matricielle :
 - $5 \times w$ lignes indexées horizontalement par x , $0 \leq x < 5$.
 - $5 \times w$ colonnes indexées verticalement par y , $0 \leq y < 5$.
 - 25 mots (ou rangées) indexées en profondeur par z , $0 \leq z < w$.
- $A[x, y, z]$ permet d'accéder à tous les bits de l'état.

État de la fonction Keccak :

Représentation matricielle

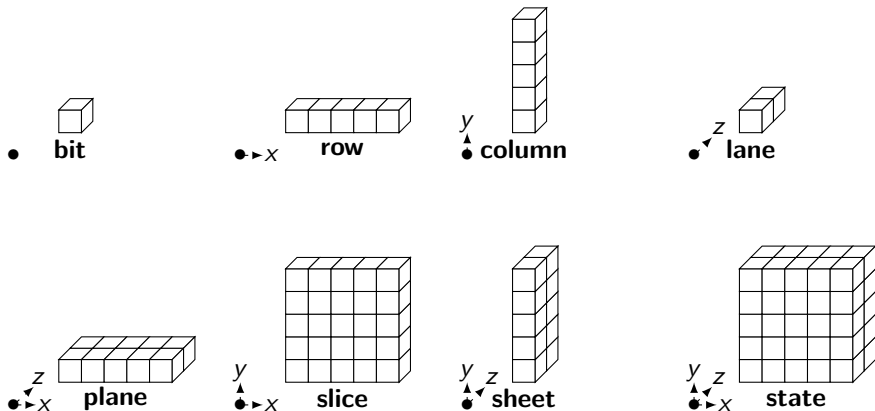


Figure: État manipulé sous forme matricielle A avec $x, y, z \in \llbracket 0, 5 \rrbracket \times \llbracket 0, 5 \rrbracket \times \llbracket 0, w \rrbracket$

KECCAK : Pour i de 1 à 24 faire :

$$\text{RND}(A, i_r) = \iota(\chi(\pi(\rho(\Theta(A))))), i_r)$$

Les routines d'une ronde **Rnd** :

- La routine Θ
- La routine ρ
- La routine π
- La routine χ
- La routine ι

SHA-3 implémente Keccak[1600,24]

w	l	n_r	b
2^6	6	24	1600

Table: Paramètres de la fonction KECCAK pour SHA-3

Taille des condensés :

- Fonctions de hachage : 224, 256, 384 et 512 bits
- Fonctions à sortie de taille variable : SHAKE128(M, l) et SHAKE256(M, l) pour des capacités de 128 et 256 bits, et une sortie de longueur l

- Doutes sur la sécurité de Merkle-Damgård
- Apparition de SHA-3, basé sur la construction de l'éponge



BUT THEY COULD NOT UNDERSTAND ITS ALIEN LANGUAGE



(modified from) poorlydrawnlines.com



Ida Tucker Amelie Guemon.

Hashing algorithms.

<https://github.com/pouwapouwa/HachingAlgo>, 2016.



NIST Computer Security Division.

SHA-3 Standard : Permutation-Based Hash and Extendable-Output Functions.

Number 202. May 2014.



Jean-Guillaume Dumas, Jean-Louis Roch, Eric Tannier, and Sébastien Varrette.

Théorie des codes : Compression, cryptage, correction.

DUNOD, 2007.



Hashcat performance tests.

<http://hashcat.net/oclhashcat/>.

Accessed : 2016-04-13.



Tadayoshi Kohno John Kelsey.

Herding hash functions and the nostradamus attack.

Technical report, National Institute of Standards and Technology, CSE Department, 2006.



Antoine Joux.

Multicollisions in iterated hash functions. application to cascaded constructions.

Technical report, DCSSI Crypto Lab, 2004.



Marc Stevens.

Attacks on Hash Functions and Applications.

PhD thesis, Amsterdam, 2012.



Douglas Stinson.

CRYPTOGRAPHIE Théorie et pratique.

vuibert, 5 edition, 1996.



Christopher Swenson.

Modern Cryptanalysis : Techniques for advanced code breaking.
WILEY, 2008.



Xuejia Lai Xiaoyun Wang, Dengguo Feng and Hongbo Yu.

Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.
Cryptography ePrint Archive : Report 2004/199, 2004.

Questions ?