université de BORDEAUX

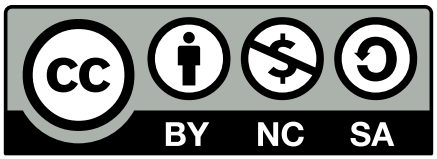
Implémentation de fonctions en éponge

Amélie Guémon
Ida Tucker

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

Master CSI, Université de Bordeaux, France

24 avril 2016



Plan

Définition : fonction de hashage

Une fonction de hashage est une application qui associe à un ensemble de départ infini $\{0,1\}^*$ un ensemble d'arrivée fini $\{0,1\}^n$ constitué de chaînes de bits de taille n.

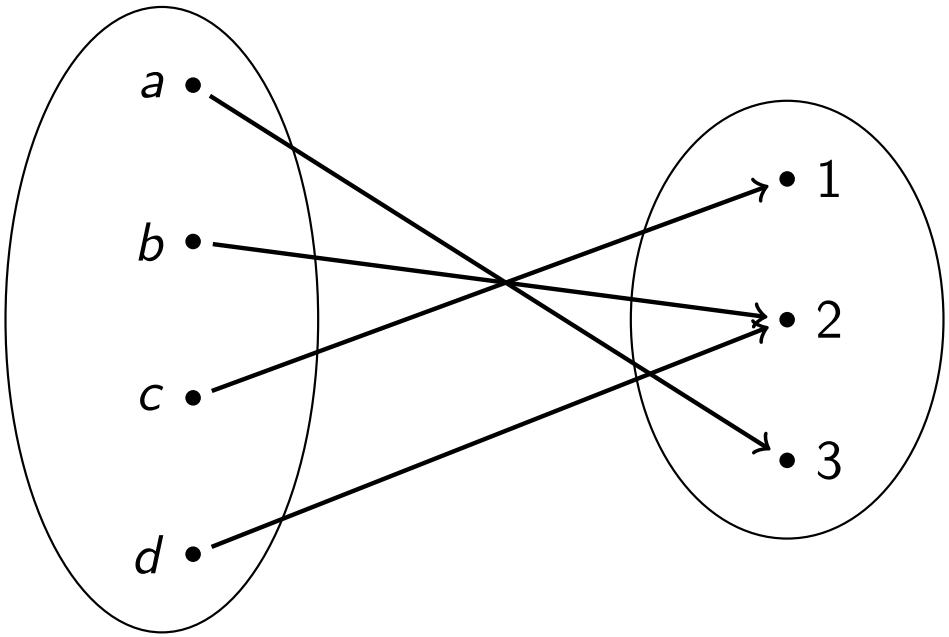


Figure: Collision dans une fonction de Hachage

Propriétés requises

- **Résistance à la Pré-image** : Pour un hash 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 $H(x) = H(x')$.

Padding

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

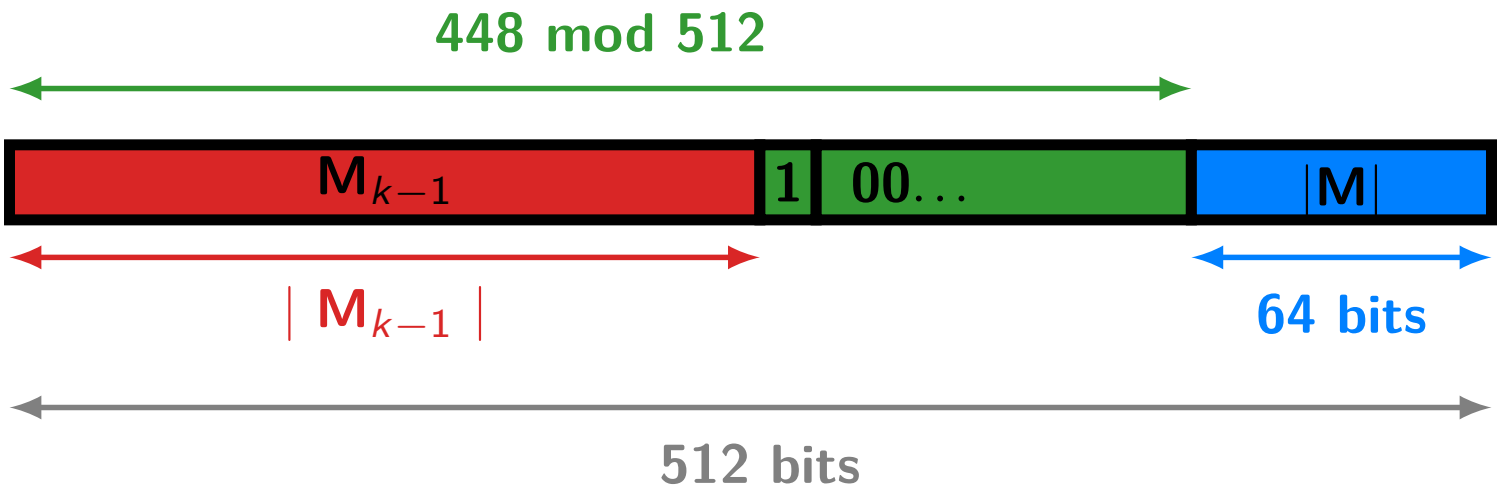


Figure: Merkle-Damgård padding.

Plan

Construction de Merkle-Damgård

La construction de Merkle-Damgård permet de définir des fonctions de hachage en itérant des fonctions de compression.

- Une fonction de compression part d'un ensemble fini vers un ensemble fini.
- Une fonction de hachage part d'un ensemble infini vers un ensemble fini.

Construction de Merkle-Damgård

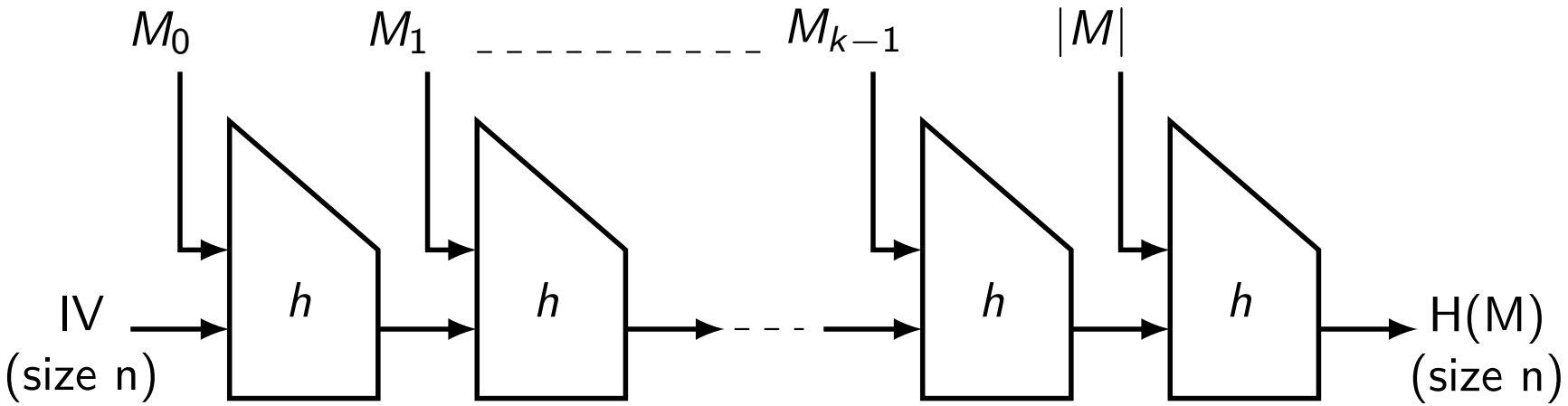


Figure: Merkle-Damgård construction.

- Théorème : Si la fonction de compression h utilisée par la fonction de hachage H l'est aussi.

Applications

- MD5
- SHA1

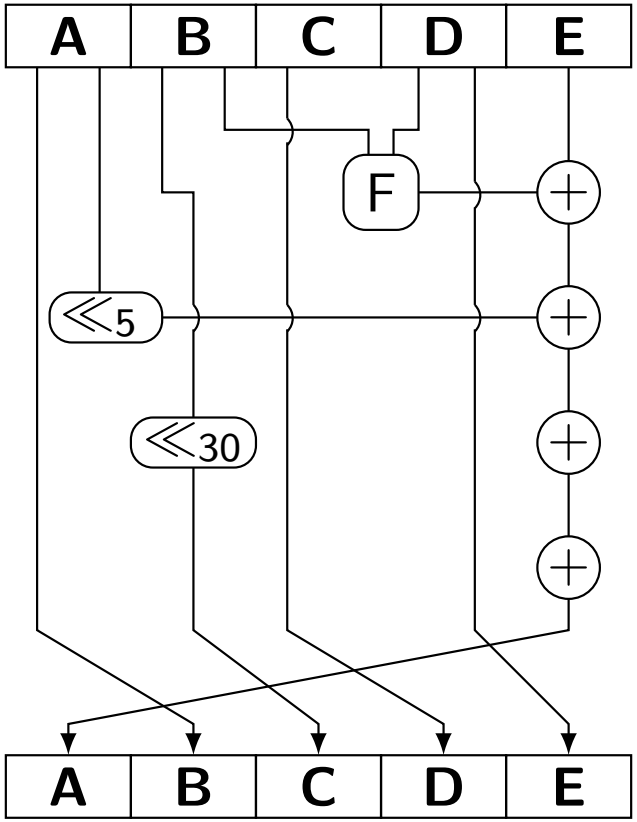


Figure: The i^{th} round in SHA-1 ($0 \leq i \leq 79$).

Plan

Attaque par force brute

Objectif : Trouver des collisions pour une fonction de hachage

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

Algorithme :

- Choisir de façon aléatoire une 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 $H(M) = H(M')$.

Probabilité de succès

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 $> 50\%$:

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

Application numérique

Application à MD5 et SHA1 :

- taille du condensé de MD5 : 128 bits.

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

- taille du condensé de SHA1 : 160 bits.

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

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

Fonctions de hachage cassée

université de BORDEAUX

Définition :
Une fonction de hashage 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 permit de casser de nombreuses fonctions de hachage itérées, basées sur Merkle-Damgård (dont SHA0, SHA1, MD5).

Plan

Etat d'une fonction en éponge

- S : l'état (state) de la fonction $S = R||C$.
- 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.

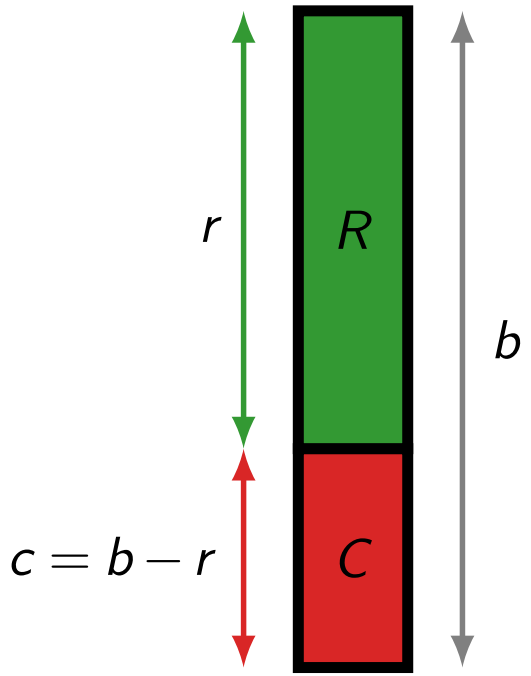


Figure: État d'une fonction en éponge.

Construction en éponge

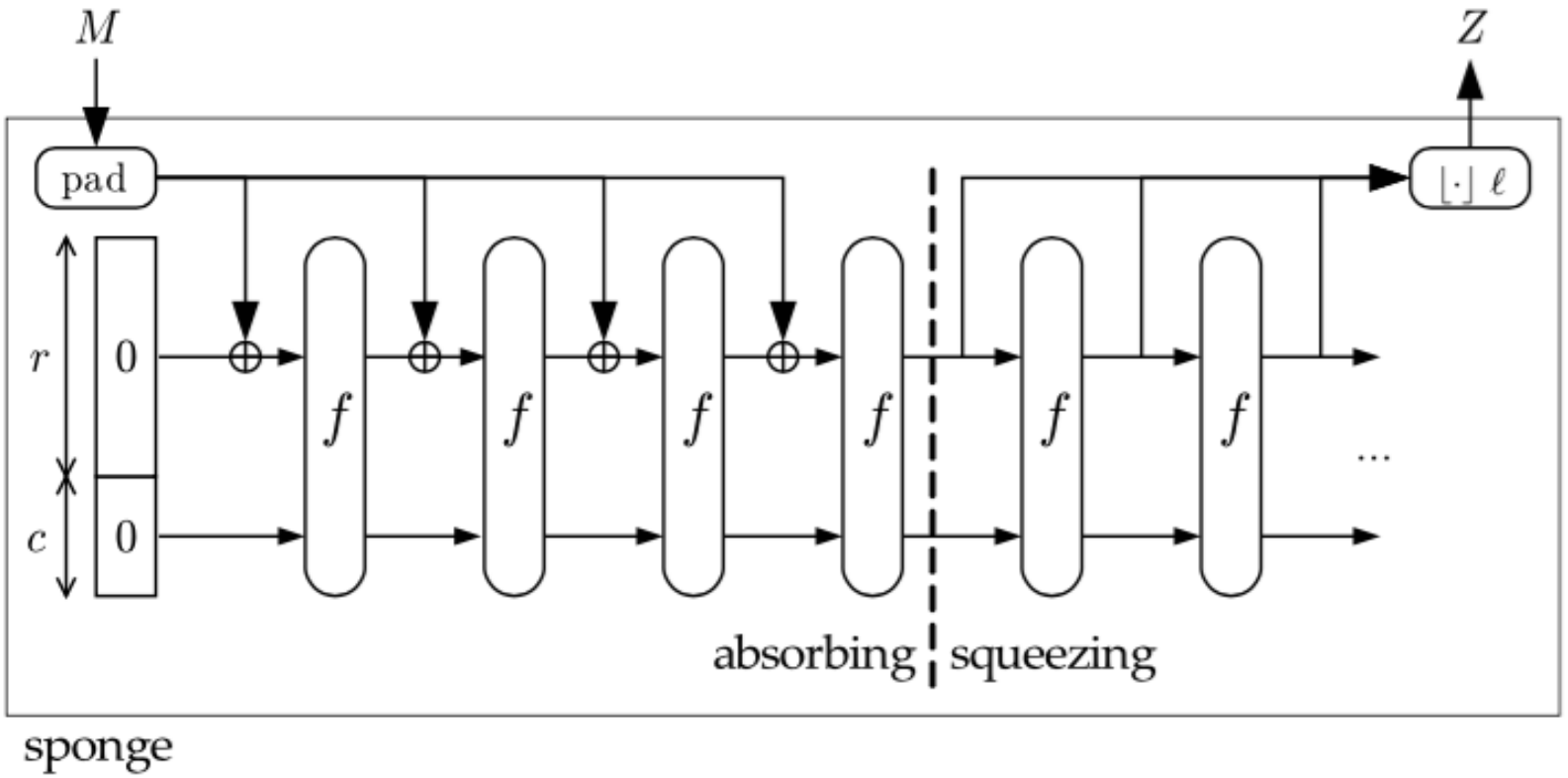


Figure: Construction de fonctions en éponge $Z = \text{SPONGE}[f, pad, r](M, l)$

Keccak formellement

université de BORDEAUX

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

- w : la longueur fixée des chaines 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

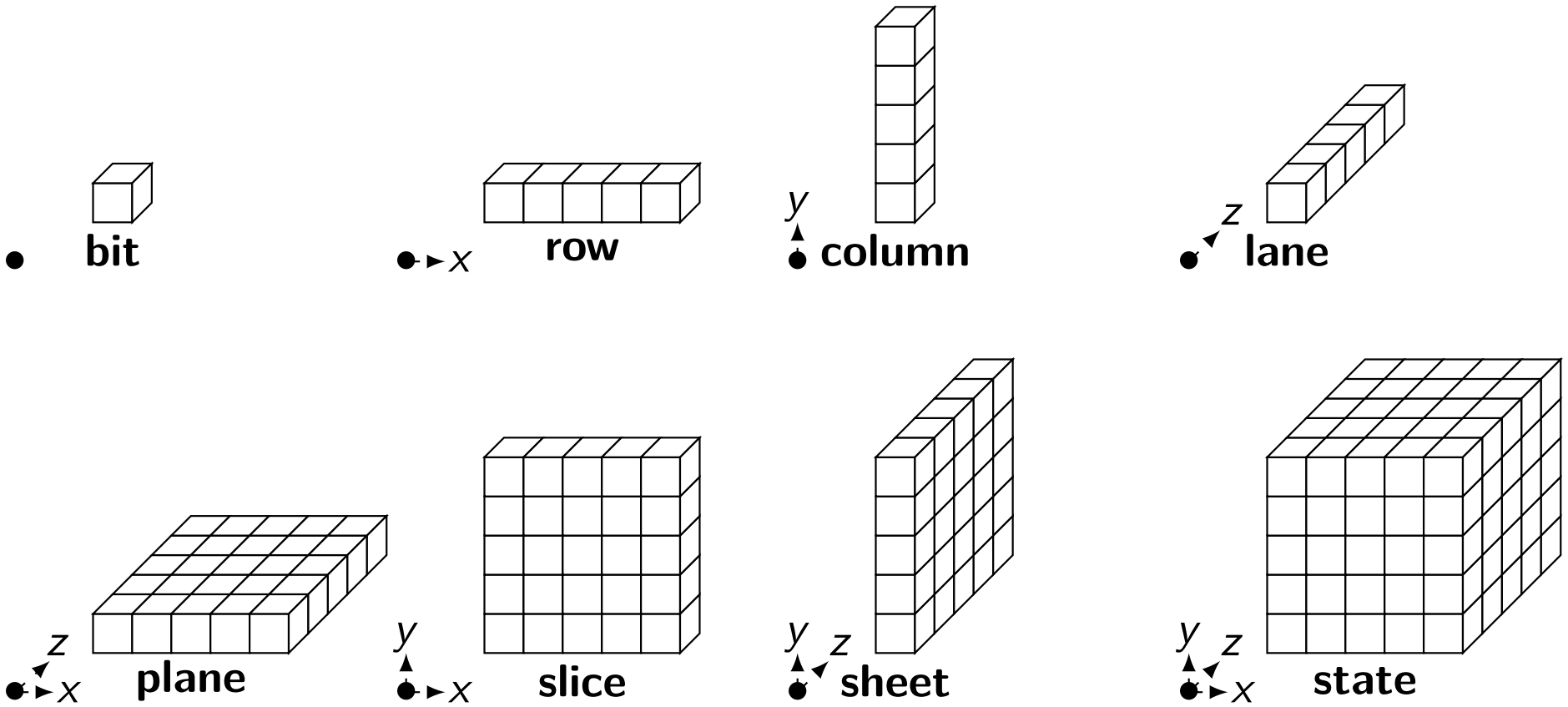


Figure: État manipulé sous forme matricielle A avec $x, y, z \in [\![0,5[\![\times [\![0,5[\![\times [\![0,w[\![$

Une Ronde de Keccak : Rnd

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 : Généralités

université de BORDEAUX

- **2 novembre 2007 :** Publication par le National Institute of Standards and Technology (NIST) annonçant la recherche d'algorithmes candidats pour une nouvelle famille de fonctions de hachage cryptographiques : SHA-3

- **Exigences générales** :
  - Condensés de 224, 256, 384, 512 bits
  - Fournir une alternative à SHA-2 (bien que SHA-2 soit encore utilisable).
  - Processus similaire à la compétition AES

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 : $\mathrm{SHAKE128}(M, l)$ et $\mathrm{SHAKE128}(M, l)$ pour une capacity de 256 bits, et une sortie de longueur l)

Livres et références I


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.

Livres et références II


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.

Livres et références III

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.
Cryptology ePrint Archive : Report 2004/199, 2004.

Questions ?