CRYPTO À CLÉ SECRÈTE

LES FONCTIONS DE HACHAGE

Cristina Onete
cristina.onete@gmail.com

C'EST QUOI UNE FUNCTION DE HACHAGE

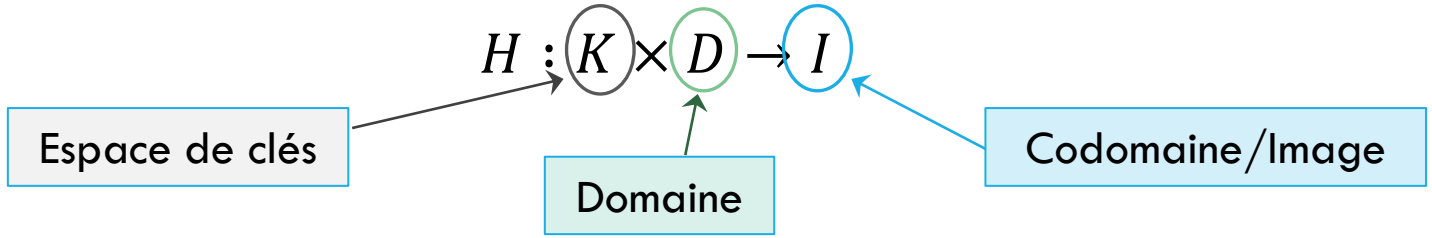
- Une moulinette :
  - ... qui prend en entrée un texte x de taille arbitraire
  - ... et sort un message de taille fixe (une hachée) : H(x)

- Techniquement une fonction de hachage est "publique" :
  - Tout personne peut calculer H(x) à partir de x
  - Une fonction de hachage utilise une clé fixe et bien-connue
  - ... Donc en particulier elle n'est pas "à clé secrète"

Alors pourquoi voir les fonctions de hachage dans ce module ?

LES FONCTIONS DE HACHAGE EN PRATIQUE

- En réalité nous parlons de "familles" de fonctions de hachage



- ... où chaque function est paramétrée par une clé $k \in K$:

$$H_k : D \to I$$

- ... et I est (un sous-ensemble de) $\{0,1\}^\ell$, les chaines de bits d'une taille fixe ℓ

EXEMPLE : LA FAMILLE MD ET SHA1

- La fonction de hachage SHA1 a été proposée en 1995
  - Elle suit le design de la fonction MD4, proposée par Ronald Rivest (le Rivest de RSA)
  - … et celui de MD5 (proposée également par Ronald Rivest)

- MD4 et MD5 prennent en entrée des valeurs $x < 2^{2^{64}} \cong 2^{10^{3*6.4}} \cong 10^{9*6.4}$ bits
  - Et sortent des hachées de taille 128 bits
- SHA1 prend en entrée des valeurs $x < 2^{2^{64}} \cong 2^{10^{3*6.4}} \cong 10^{9*6.4}$
  - Et sortent des hachées de taille 160 bits

Nous verrons des constructions plus tard
D'abord voyons à quoi peuvent servir les fonctions de hachage

LES PROPRIÉTÉS DES FONCTIONS DE HACHAGE

TROIS PROPRIÉTÉS PRINCIPALES

1. La résistance aux préimages (fonction à sens unique) :

   Soient : $k \xleftarrow{\$} K$, x $\xleftarrow{\$} D$ et $y \leftarrow H_k(x)$. Un attaquant reçoit k, y et sort x'.
   La probabilité que $H(x') = y$ est négligeable.

2. La résistance aux deuxième préimages :

   Soient $k \xleftarrow{\$} K$, x $\xleftarrow{\$} D$. Un attaquant reçoit k, x et sort $x' \neq x$.
   La probabilité que $H(x') = H(x)$ est négligeable.

3. La résistance aux collisions (3 définition, paramétrisées par $s \in \{0,1,2\}$) :

   Un attaquant sort $2 - s$ valeurs du domaine D. Ensuite on choisit $k \xleftarrow{\$} K$.
   L'attaquant est donné k et sort s valeurs dans D.
   La probabilité que les $2 - s + s = 2$ valeurs donnent une collision est négligeable.

RÉSISTANCE AUX PRÉIMAGES

1. La résistance aux préimages (fonction à sens unique) :

   Soient : $k \xleftarrow{\$} K$, x $\xleftarrow{\$} D$ et $y \leftarrow H_k(x)$. Un attaquant reçoit k, y et sort x'.
   La probabilité que $H(x') = y$ est négligeable.

- La probabilité ici est prise sur :
  - Les choix de k et x
  - Les choix aléatoires de l'attaquant

- Intuition : La probabilité qu'on trouve une préimage (une inverse) de la valeur y est faible

  Attention : ceci se joue au choix aléatoire de k, x !

RÉSISTANCE AUX COLLISIONS

3. La résistance aux collisions (3 définition, paramétrisées par $s \in \{0,1,2\}$) :

   Un attaquant sort $2 - s$ valeurs du domaine D. Ensuite on choisit $k \xleftarrow{\$} K$.
   L'attaquant est donné k et sort s valeurs dans D.
   La probabilité que les $2 - s + s = 2$ valeurs sont distinctes et donnent une collision est négligeable.

- Disons que $s = 0$. L'attaquant doit choisir x_1, x_2 avant d'avoir la clé k.
  - Si l'espace de clés est large alors cela rajoute une difficulté pour l'adversaire

- Disons que $s = 2$. L'attaquant reçoit la clé k avant de devoir choisir x_1, x_2
  - Cet attaquant est plus puissant que celui du cas $s = 0$

LA RÉSISTANCE AUX DEUXIÈMES PRÉIMAGES

2. La résistance aux deuxième préimages :

   Soient $k \xleftarrow{\$} K$, x $\xleftarrow{\$} D$. Un attaquant reçoit k, x et sort $x' \neq x$.
   La probabilité que $H(x') = H(x)$ est négligeable.

- L'attaquant reçoit la clé et doit trouver une collision avec x

Quelle est la différence avec la résistance aux collisions avec s = 2 ?

RELATIONS ENTRE LES PROPRIÉTÉS

- Propriété 2 : résistance aux deuxième préimages
- Propriété 3 : résistance aux collisions

- **Proposition** : Disons qu'une famille de fonctions H a la propriété 3 $\forall k \in K$. Cela implique la propriété 2 pour la fonction H (pour toute k).
- **Preuve** : Supposons qu'il y ait un attaquant $\mathcal{A}$ qui peut trouver des deuxièmes préimages contre H, avec une probabilité p non-négligeable. C'est à dire, étant donné $k \xleftarrow{\$} K$, x $\xleftarrow{\$} D$, l'attaquant sort $x' \neq x$ tel que $H(x) = H(x')$, avec une probabilité non-négligeable.

Nous allons construire l'attaquant $\mathcal{B}$ qui peut trouver des collisions. Cet attaquant reçoit $k \xleftarrow{\$} K$, choisit x $\xleftarrow{\$} D$ et donne ces valeurs à $\mathcal{A}$, qui lui retourne $x' \neq x$. L'attaquant $\mathcal{B}$ a donc trouvé sa collision avec probabilité p non-négligeable. Ce qui contredit notre hypothèse.

RELATIONS ENTRE LES PROPRIÉTÉS (CONT'D)

- Propriété 2 : résistance aux deuxième préimages
- Propriété 3 : résistance aux collisions

- Est-ce que l'inverse est vrai ?
- C'est à dire : étant donné une famille H qui a la propriété 2, est-ce qu'elle a la propriété 3 ?

RELATIONS ENTRE LES PROPRIÉTÉS (CONT'D)

- Propriété 1 : résistance aux préimages (sens unique)
- Propriété 3 : résistance aux collisions

Quelle est la relation entre ces deux propriétés ?

RELATIONS ENTRE LES PROPRIÉTÉS (CONT'D)

- Propriété 1 : résistance aux préimages (sens unique)
- Propriété 3 : résistance aux collisions

- Prenons une fonction de hachage avec $D = I = \{0,1\}^{512}$ tel que :

$$H(x) = x$$

- Cette fonction de hachage est **résistante aux collisions** -- car il n'y a pas de collision
- Mais elle n'est **pas résistante aux préimages**

La Propriété 3 n'implique pas la Propriété 1

RELATIONS ENTRE LES PROPRIÉTÉS (CONT'D)

- Propriété 1 : résistance aux préimages (sens unique)
- Propriété 3 : résistance aux collisions

- Supposons une famille *H* tel que : $H : K \times D \to I$ a les deux propriétés
- On construit $D' \coloneqq \{x||b,\ x \in D; b \in \{0,1\}\}$ (on "allonge" les entrées par un bit)
- Prenons une famille H' tel que : $H'\colon K \times D' \to I$ et $H'(x||b) = H(x)\ \forall\, x \in D, b \in \{0,1\}$
  - La famille H' est résistante aux préimages, car H l'est
  - Mais elle ne résiste pas aux collisions (pouvez-vous trouver une collision ?)

La Propriété 1 n'implique pas la Propriété 3

UNE NOTE SUR LA RÉSISTANCE AUX COLLISIONS

TROUVER DES COLLISIONS

- Plusieurs méthodes existent actuellement :
  - Certaine méthodes exploitent les spécificités de certaines fonctions de hachage
  - Mais on a également des attaques génériques

- Exemple d'attaque générique :
  - prendre une entrée aléatoirement $x \xleftarrow{\$} D$, calculer $h = H(x)$
  - itérer sur q autres éléments de D jusqu'à trouver une collision sur h
  - si aucune collision trouvée, alors on a échoué

Quelle probabilité a l'adversaire de gagner ?
Sous quelles conditions ?

UNE STRATÉGIE PLUS MALINE

- L'adversaire a q essais pour trouver une collision

- Il les dépensera de la façon suivante :
  - Pour $i = 1,2, \dots, q$:
    - on prend $x_i \xleftarrow{\$} D$ et on calcule et stocke $h_i = H(x_i)$
    - Si $\exists\ 1 \leq j < i$ et $h_i = h_j$ tandis que $x_i \neq x_j$ alors on a trouvé une collision et on arrête
  - Si aucune collision trouvée on a échoué

Quelle est la probabilité de gagner?

FONCTIONS DE HACHAGE CASSÉES

- MD4 : premières collisions trouvées en 1995 et des collisions améliorées avec le temps
  attaque théorique pour trouver des préimages
- MD5 : collisions trouvables dans quelques secondes sur un PC ordinaire
  premières attaques en 2004, et possible à trouver des collisions sur des certificats
  utilisé dans le malware Flame en 2012
- SHA-1 : premières collisions en 2017, passage vers SHA-2 et SHA-3

PARENTHÈSE : LA SÉCURITÉ PROUVABLE

C'EST QUOI LA SÉCURITÉ PROUVABLE ?

- Une méthodologie puissante, qui permet la "sécurité par design"
- 3 éléments principaux : syntaxe, modèle de l'adversaire, jeu de sécurité

- Syntaxe : les algorithmes qui caractérisent une primitive cryptographique
  - Fonctions de hachage : l'algorithme H de hachage prend en entrée une clé *k* et une valeur *x* et sort un hash *h*

C'EST QUOI LA SÉCURITÉ PROUVABLE ?

- Une méthodologie puissante, qui permet la "sécurité par design"
- 3 éléments principaux : syntaxe, modèle de l'adversaire, jeu de sécurité

- Syntaxe : les algorithmes qui caractérisent une primitive cryptographique
- Modèle de l'adversaire : l'adversaire a accès à toute entrée et algorithme publique
  - De plus on peut lui donner accès "par oracle" à des entrées privées
  - Exemple (chiffrement à clé publique) : l'adversaire a de toute façon le pouvoir de chiffrer (publique)
    de plus on peut lui donner un accès par oracle au déchiffrement

C'EST QUOI LA SÉCURITÉ PROUVABLE ?

- Une méthodologie puissante, qui permet la "sécurité par design"
- 3 éléments principaux : syntaxe, modèle de l'adversaire, jeu de sécurité

- Syntaxe : les algorithmes qui caractérisent une primitive cryptographique
- Modèle de l'adversaire : l'adversaire a accès à toute entrée et algorithme publique
- Jeu de sécurité : interaction entre l'adversaire et les parties honnêtes (challenger)
  - L'adversaire gagne s'il peut casser la modélisation de la propriété de sécurité qu'on souhaite avoir

LA RÉSISTANCE AUX PRÉIMAGES

- Syntaxe : algorithme H prenant une entrée $k \leftarrow K$ et une entrée $x \leftarrow D$ et sortant $h \in I$
- Adversaire : a accès à H
- Jeu de sécurité :

$k \xleftarrow{\$} K$
$x \xleftarrow{\$} D$
$h \leftarrow H(k, x)$
$x^* \leftarrow A(k, h)$

générés par le challenger

L'adversaire prend en entrée k et h
Il doit deviner une preimage valide de h

A gagne ssi. $H(x^*) = h$
et $x^* \in D$

3 JEUX POUR LA RÉSISTANCE AUX COLLISIONS

- Pour la resistance aux collisions, on peut formaliser 3 jeux (3 degrees de sécurité)
- Différences principale : connaissance (ou non) de la clé

$k \xleftarrow{\$} K$
$(x_1, x_2) \leftarrow A(k)$

A gagne ssi. :
$H(x_1) = H(x_2)$
et $x_1 \neq x_2$
et $x_1, x_2 \in D$

$x_1 \leftarrow A()$
$k \xleftarrow{\$} K$
$x_2 \leftarrow A(k)$

A gagne ssi. :
$H(x_1) = H(x_2)$
et $x_1 \neq x_2$
et $x_1, x_2 \in D$

$x_1, x_2 \leftarrow A()$
$k \xleftarrow{\$} K$

A gagne ssi. :
$H(x_1) = H(x_2)$
et $x_1 \neq x_2$
et $x_1, x_2 \in D$

Quel est l'adversaire le plus faible ?
Quelle est la notion la plus forte ?

LA DEUXIÈME PRÉIMAGE

- Pourriez-vous définir le jeu de sécurité de la sécurité contre les deuxième préimages ?

A faire en TD !

LE MODÈLE DE L'ORACLE ALÉATOIRE

A QUOI SERVENT LES FONCTIONS DE HACHAGE ?

- Pour garantir l'intégrité des données (sans garantir en même temps leur confidentialité) :
  - Les logiciels qui gèrent le versioning (comme Git) utilisent des fonctions de hachage pour avoir un identifiant (court) de chaque version
  - Ceci permet aux utilisateurs de vérifier la suite des versions, de revenir en arrière à une certaine version, etc.
  - Quelle propriété doivent avoir ces fonctions de hachage alors ?

- Une fonction de hachage peut être utilisé pour "cacher" les mots de passe des utilisateurs
  - Dans la base de données ce qui sera stocké c'est $(nom, H(mdp))$
  - Quand l'utilisateur cherchera à s'authentifier, alors il enverra le nom et un mot de passe
  - De quelle(s) propriétés a-t-on besoin ?

Nous verrons plus de cas d'usage en TD

UNE PROPRIÉTÉ PLUS FORTE

- Parfois nous utilisons les fonctions de hachage pour la dérivation de clés
  - Typiquement, un protocole d'échange de clé (comme DH) nous donne une valeur "racine"
  - Mais, cette valeur est souvent dans une structure algébrique (groupe fini, etc.)
  - Pour la dérivation de clé, c'est mieux d'avoir une valeur distribuée uniformément sur $\{0,1\}^{\ell}$
    - Par exemple $H(racine)$

- Si la sortie d'une fonction de hachage H_k doit être "pratiquement aléatoire" =>

  besoin d'une propriété plus forte que les trois vues précédemment

Notamment, H_k doit être un oracle aléatoire

C'EST QUOI UN ORACLE ALÉATOIRE ?

- Une idéalisation des fonctions de hachage
- On interprète $H_k: D \rightarrow \{0,1\}^{\ell}$ comme une function $RO: D \rightarrow \{0,1\}^{\ell}$

- Deux propriétés :
  - **Le caractère aléatoire** : à chaque x correspond un $RO(x)$ tiré aléatoirement de l'ensemble $\{0,1\}^{\ell}$
  - **La cohérence** : si on demande deux fois la valeur x à RO on aura la même valeur de retour

- Intuition : un écart d'un bit en entrée donne deux sorties vraiment différentes

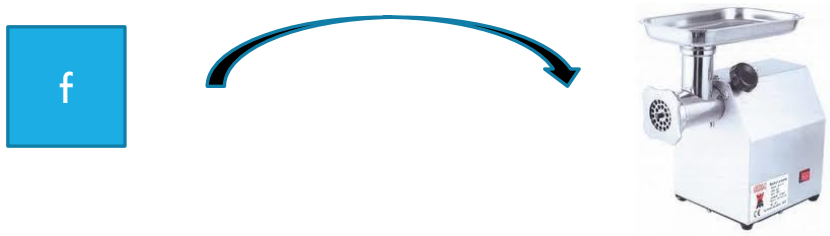
L'UTILITÉ DE L'ORACLE ALÉATOIRE

- Le modèle de l'oracle aléatoire est une idealisation
- Aucune fonction de hachage n'est un vrai oracle aléatoire
  - Mais on croit que certaines fonctions pourraient s'approcher à un RO
  - Techniquement ceci veut dire qu'un adversaire "ne voit pas la différence" entre la fonction et un RO
- Si un oracle aléatoire n'existe pas, à quoi sert-il ?
  - Principalement dans les preuves de sécurité de primitives/protocoles plus complèxes
  - Dans le modèle de l'oracle aléatoire toutes les parties honnêtes et l'adversaire ont accès au RO, ainsi remplaçant leur usage de la vraie fonction de hachage

LE DESIGN DES FONCTIONS DE HACHAGE

LA CONSTRUCTION MERKLE-DAMGAARD

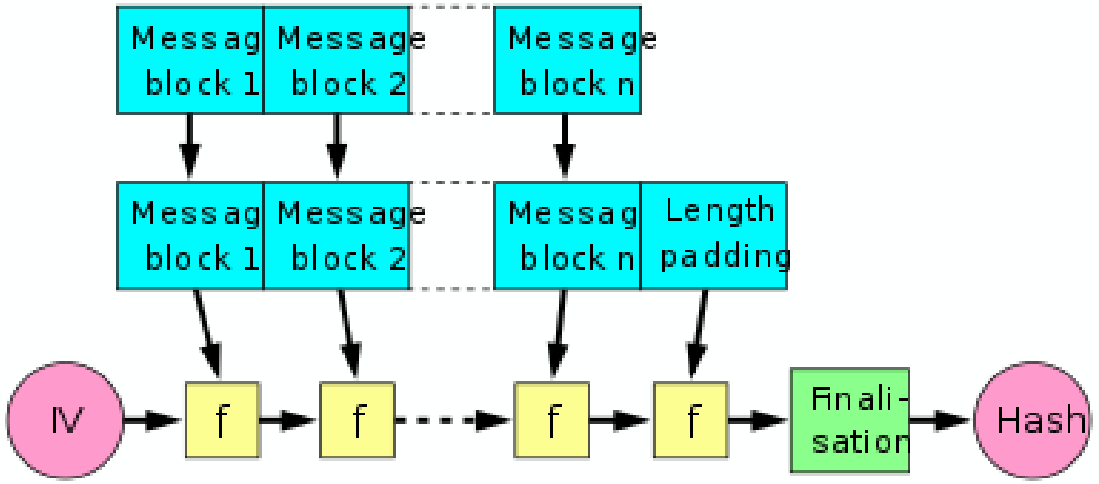
- Présentée dans la these de doctorat de Ralph Merkle en 1979
- Décrit comment construire une fonction de hachage résistante aux collisions
  - A partir d'une fonction de compression resistante aux collisions et à sens unique



- Fonction de compression : $f: \{0,1\}^m \rightarrow \{0,1\}^n$
  - Exemple : on prend en entrée deux valeurs d'une taille l et on sort une valeur de taille l
  - Souvent implémentées en utilisant des chiffres par bloc (block ciphers)

LA CONSTRUCTION MERKLE-DAMGAARD (CONT'D)

- Première étape : padding
- Choisir un IV
- Utiliser à chaque fois un bloc, l'utiliser pour obtenir le prochain
- Finalisation : une fonction qui peut encore compresser la sortie, ou rajouter de la complexité pour l'attaquant



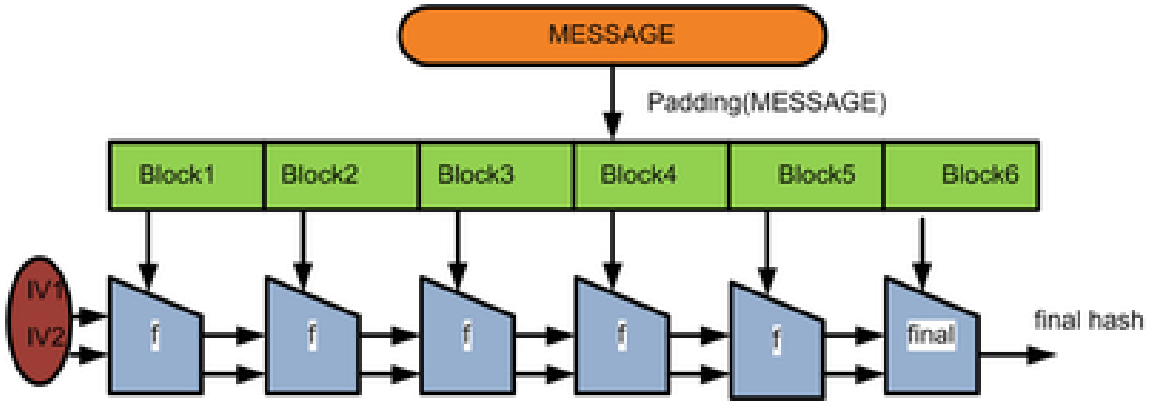
Source : wikipedia.org

PROBLÈMES AVEC CE DESIGN

- Merkle et Damgaard on montré que ce design resistait les collisions...
- ... Mais il présente toujours des faiblesses :
  - Séconde preimage : bien plus facile que par brute force
  - Multicollisions : le coût de trouver plusieurs collisions sur une même valeur à peine plus large que le coût d'une seule collision
  - Herding : une attaque qui trouve des collisions bien plus rapidement que sur un RO
  - Length-extension attacks : étant donné x inconnu et $H_K(x)$, on peut trouver $H_k(pad(x)||y)$ pour un y connu

MERKLE-DAMGAARD, WIDE-PIPE

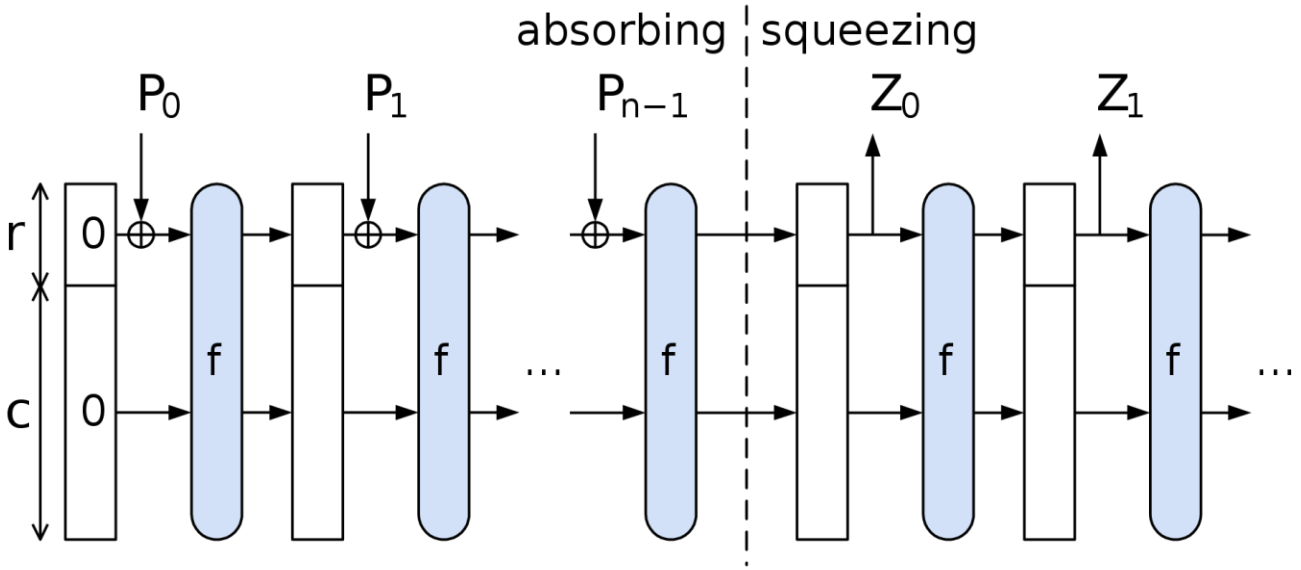
- Introduite par Lucks en 2004
- La fonction de compression prend en entrée un état à $2n$ bits et un bloc du message à m bits et retourne $2n$ bits
- Plus résistante
- Peut être plus rapide grâce à un speed-up connu



Source : wikipedia.org

LES CONSTRUCTIONS EPONGE

- À la base de SHA-3 = Keccak
- Un principe d'absorption, puis de pression de l'éponge
  - Absorption : l'entrée est absorbée dans une partie de la construction par des XORs successifs un état est mis à jour à chaque absorption
  - Pression : on resort des valeurs par des iterations successives



Source : wikipedia.org

LE HASH CHAMÉLÉON

LES HASH CAMÉLÉON

- Une fonction de hachage avec une trappe qui permet de trouver des collisions
  - Les entrées sont du type $H_k(x, r)$ avec un message x et une valeur aléatoire r
  - Sans la trappe, difficile à trouver des collisions
  - Avec la trappe, étant données x, r, x^*, on trouve r^* tel que $H_k(x, r) = H_k(x^*, r^*)$

A quoi cela pourrait servir ?

- Les hash caméléons sont utiles dans les signatures assainissables (sanitizable signatures)
  - Une autorité signe un document (par exemple un dossier médical)
  - Une partie de ce document doit être envoyée à une autre entité (son employeur par exemple) -- mais sans dévoiler beaucoup de détails personnels
  - Pour authentifier le document "assaini", l'autorité a délégué la possibilité à une autre entité (assurance maladie) la trappe d'un hash caméléon
- L'assurance maladie est capable de modifier la signature sans avoir besoin de la clé du signataire