Propriétés de sécurité des fonctions de hachage

Soit $h : \{0,1\}^* \to \{0,1\}^n$ une fonction de hachage que l'on suppose résistante aux collisions. Soit h' la fonction suivante :

$$h' : \begin{cases} \{0,1\}^* & \to & \{0,1\}^{n+1} \\ x & \mapsto & \begin{cases} 0\|x \text{ si } |x| = n \\ 1\|h(x) \text{ sinon} \end{cases} \end{cases}$$

Question 1. *Montrer que h' est résistante aux collisions.*

Solution. *Rappelons la définition d'une attaque par collision : « peut-on écrire un algorithme plus rapide que la borne d'anniversaire qui produise une collision ? »*

Ici, la seule hypothèse dont nous disposons est que h est résistante aux collisions. Nous allons donc réduire *la résistance aux collisions de h' à celle de h. Nous faisons la preuve par contraposition : nous montrons que si h' n'est pas résistante aux collisions, alors h ne l'est pas non plus.*

Supposons que h' ne soit pas résistante aux collisions. Il existe alors une attaque par collision, c'est-à-dire un algorithme $\mathcal{A}$ qui produit une collision plus vite que la borne d'anniversaire générique ($2^{n/2}$). Supposons que $\mathcal{A}$ produise une paire (x, y).

Nous remarquons que $h'(x)$ ne peut pas commencer par 0, car cela impliquerait $h'(x) = 0|x = h'(y) = 0|y$, ce qui mènerait à une contradiction (nous supposons $x \neq y$). Donc $h'(x)$ commence par 1 et $h'(x) = 1|h(x) = h'(y) = 1|h(y)$, ce qui implique $h(x) = h(y)$. Nous pouvons utiliser l'algorithme $\mathcal{A}$ comme une attaque par collision contre h. Cela conclut la preuve.

Question 2. *Montrer que h' n'est pas résistante aux préimages.*

Solution. *Remarquons que h n'est pas supposée être résistante aux préimages, mais elle n'est pas non plus supposée être cassée. Nous ne pouvons simplement rien affirmer concernant sa résistance aux préimages.*

Rappelons la définition d'une attaque par préimage : « un algorithme plus rapide que la recherche exhaustive, et qui réussit avec une probabilité constante ». Cette dernière partie est importante, car il est acceptable que l'attaque ne réussisse pas pour toutes *les cibles possibles. Vous pouvez voir cela ainsi : lorsque vous attaquez la fonction de hachage, la cible sera en général choisie aléatoirement, et si l'attaque échoue, vous la relancez simplement.*

Concevons donc un tel algorithme. En entrée, on prend une cible t choisie uniformément au hasard. Si t commence par 0 alors $t = 0|t'$ et t' est une préimage valide de t, suivant la définition de h'. Si t commence par 1, nous échouons simplement. Nous réussissons donc avec probabilité 1/2 sur des entrées aléatoires, ce qui est suffisant pour revendiquer une attaque.

Variations de Merkle-Damgård

On cherche à concevoir une fonction de hachage sûre basée sur la construction Merkle-Damgård. Dans la suite de cet exercice, on supposera que les blocs de message sont tous complets. De plus, on utilisera une construction de Merkle-Damgård à *deux* fonctions de compression (h, h'), pour laquelle *aucun padding n'est nécessaire*. Un exemple est représenté sur la figure 1. Les deux fonctions $h, h' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ prennent en entrée un bloc de n bits et une valeur de chaînage de n bits, et sont considérées comme des fonctions aléatoires indépendantes.

La complexité en temps des algorithmes sera comptée en évaluations des fonctions h et h'. Lorsque des algorithmes sont demandés, vous pouvez les écrire sous forme de pseudocode peu détaillé.

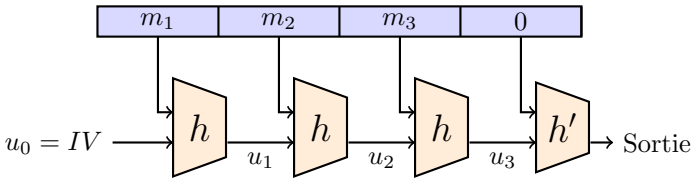

FIGURE 1 – Fonction MD typiquement considérée pour cet exercice.

Étant donné une fonction de compression $h : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, l'itération de h sur les blocs $m_1, \ldots, m_\ell$ en partant de IV est notée :

$$h^*(IV, m_1, \ldots, m_\ell) \ ,$$

par exemple :

$$h^*(IV, m_1, m_2, m_3) = h(h(h(IV, m_1), m_2), m_3) \ .$$

Ainsi la fonction Merkle-Damgård complète, notée H, a pour expression :

$$H(m_1, \ldots, m_\ell) = h'(h^*(IV, m_1, \ldots, m_\ell), 0) \ .$$

Question 3. *On commence avec une taille de bloc (et de chaînage) de 128 bits. Quel est le niveau de sécurité de la fonction H contre les collisions ? (Il n'est pas nécessaire de détailler l'algorithme d'attaque).*

Solution. *Seulement 64 bits de sécurité.*

Question 4. *On modifie maintenant la fonction h'. Elle se comporte toujours comme une fonction aléatoire, mais sa sortie est étendue à 256 bits :*

$$h' : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{256}$$

Quel est le niveau de sécurité de la nouvelle fonction H contre les collisions ? Justifiez cette réponse par un algorithme d'attaque simple dont vous estimerez la complexité.

Solution. *Trouver une paire m, m' en collision juste après l'IV : $h(IV, m) = h(IV, m') := c$ en utilisant votre algorithme de recherche de collisions préféré. Dans les deux cas la sortie est $h'(0, c)$ donc c'est la même.*

Question 5. *On utilise maintenant deux fonctions de compression $h, h' : \{0,1\}^{256} \times \{0,1\}^{256} \to \{0,1\}^{256}$. On définit une nouvelle construction basée sur Merkle-Damgård, donnée par l'algorithme suivant :*

Entrée : *ℓ blocs de message $(m_1, \ldots, m_\ell)$ de 512 bits chacun*

1. *Séparer les blocs en moitiés comme suit : $m'_1 \| m''_1 = m_1, \ldots, m'_\ell \| m''_\ell = m_\ell$*
2. *Calculer $t_0 = h'(h^*(0, m'_1, \ldots, m'_\ell), 0)$*
3. *Calculer $t_1 = h'(h^*(1, m''_1, \ldots, m''_\ell), 0)$*
4. *Renvoyer $t_0 \oplus t_1$*

Donnez un algorithme d'attaque en préimage contre cette fonction et estimez sa complexité. Est-elle plus ou moins sûre qu'une fonction Merkle-Damgård classique à 256 bits de sortie ?

Solution. *L'algorithme d'attaque : trouver une collision entre t_0 et t_1. Formellement, on définit deux fonctions $f(m'_1, \ldots, m'_\ell) := h'(h^*(0, m'_1, \ldots, m'_\ell), 0)$ et $g(m''_1, \ldots, m''_\ell) := h'(h^*(1, m''_1, \ldots, m''_\ell), 0)$*

et on cherche une paire de ℓ-tuples de blocs qui donne la même sortie pour f et g. C'est le même coût que la collision pour une fonction seule[1]*.*

La fonction a $n/2$ bits de sécurité en préimage. C'est moins sûr qu'un Merkle-Damgard classique. En effet, MD classique a n bits de sécurité en préimage (contrairement à la seconde préimage). On pourrait aussi raisonner sur les secondes préimages : on a vu en cours une attaque en seconde préimage sur MD, mais elle n'atteint la complexité $2^{n/2}$ que pour des messages très longs, ici l'attaque est plus puissante puisqu'on est sur des messages courts.

Dans la suite de cet exercice, on définit une fonction de Merkle-Damgård *avec checksum*, de la manière suivante :

$$HC(m_1, \ldots, m_\ell) = h'\left(h^*(0, m_1, \ldots, m_\ell), \bigoplus_{i=1}^{\ell} m_i\right)$$

Le dernier bloc, précedemment 0, est maintenant le XOR de tous les blocs de message (la checksum). Blocs et valeurs de chaînages sont de taille n, et nous nous intéressons exclusivement à des complexités asymptotiques en n.
Nous allons montrer une attaque en seconde préimage sur cette fonction.

Question 6. *La première étape est de construire une multicollision à $2n$ blocs, c'est-à-dire une série de $2n$ paires de blocs (m_i^0, m_i^1) tels que :*

$$\exists U, \forall b_1, \ldots, b_{2n}, h^*(0, m_1^{b_1}, \ldots, m_{2n}^{b_{2n}})) = U$$

Donner un algorithme pour cette étape, et sa complexité asymptotique.

Solution. *Complexité : $\mathcal{O}(n2^{n/2})$.*

Algorithme : c'est la multicollision vue en cours. Un pseudocode très basique suffit : initialiser $c = 0$. Itérer $2n$ fois : trouver une paire m, m' telle que $h(c, m) = h(c, m')$. Mettre à jour $c \leftarrow h(c, m)$. Renvoyer les $2n$ paires.

Le plus important pour cet algorithme est que chaque nouvelle collision est calculée à partir de la valeur de chaînage actuelle c, qu'on doit mettre à jour.

Nous admettons le résultat suivant.
Étant donnée une cible $t \in \{0,1\}^n$ quelconque, il existe (avec très grande probabilité) un choix de blocs $(b_1, \ldots, b_{2n})$ dans les paires de la multicollision, tel que :

$$\bigoplus_{i=1}^{2n} m_i^{b_i} = t \ .$$

De plus ce choix peut être calculé en temps $\mathcal{O}(n^3)$ à l'aide d'une résolution de système linéaire.

Question 7. *Soit $P = (p_1, \ldots, p_{2^k})$ un message de longueur 2^k. Soient :*

$$u_0 := IV, u_1 := h(u_0, p_1), \ldots, u_{2^k} := h(u_{2^k-1}, p_{2^k}),$$

les valeurs de chaînage dans la fonction. Donner un algorithme simple pour trouver un bloc m^* tel que $h(U, m^*) \in \{u_1, \ldots, u_{2^k}\}$, et donner sa complexité asymptotique.

1. Pourquoi ? Méthode simple : définissez une seule fonction à n bits de sortie F telle que $F(0\|x) = f(x)$ et $F(1\|x) = g(x)$, cherchez une collision de F, recommencez jusqu'à ce que (par hasard) les deux éléments de la paire aient des premiers bits distincts. Je n'aurais pas demandé qu'on me donne ces détails en examen.

Solution. *$\mathcal{O}(2^{n-k})$ par recherche exhaustive, comme l'attaque en seconde préimage sur MD.*

Question 8. *Soit $m_1, \ldots, m_{2n}, m^*$ un message de $2n+1$ blocs tel que pour un certain i :*

- $h^*(IV, m_1, \ldots, m_{2n}, m^*) = h^*(IV, p_1, \ldots, p_i)$
- $m_1 \oplus m_2 \oplus \ldots \oplus m_{2n} \oplus m^* = p_1 \oplus \ldots \oplus p_i$

Montrer que $m_1, \ldots, m_{2n}, m^*, p_{i+1}, \ldots, p_{2^k}$ est une seconde préimage du message P originel.

Solution. *Faire le calcul : il s'agit de montrer que :*

$$HC(p_1, \ldots, p_{2^k}) = HC(m_1, \ldots, m_{2n}, m^*, p_{i+1}, \ldots, p_{2^k})$$

$$h'(h^*(0, m_1, \ldots, m_{2n}, m^*, p_{i+1}, \ldots, p_{2^k}), m_1 \oplus \ldots \oplus m_{2n} \oplus m^* \oplus p_{i+1} \ldots \oplus p_{2^k}) = h'(h^*(0, p_1, \ldots, p_{2^k}), \bigoplus p_i)$$

On va donc montrer deux égalités. Égalité de la dernière valeur de chaînage :

$$h^*(0, m_1, \ldots, m_{2n}, m^*, p_{i+1}, \ldots, p_{2^k}) = h^*(0, p_1, \ldots, p_{2^k})$$

venant du fait que par définition de m^* :

$$h^*(IV, m_1, \ldots, m_{2n}, m^*) = h^*(IV, p_1, \ldots, p_i)$$

et qu'on rajoute ensuite les blocs $p_{i+1}, \ldots, p_{2^k}$. Deuxième égalité, celle de la checksum, venant de :

$$m_1 \oplus m_2 \oplus \ldots \oplus m_{2n} \oplus m^* = p_1 \oplus \ldots \oplus p_i$$
$$\implies m_1 \oplus \ldots \oplus m_{2n} \oplus m^* \oplus p_{i+1} \ldots \oplus p_{2^k} = \bigoplus p_i \ .$$

Question 9. *En déduire une attaque en seconde préimage sur Merkle-Damgård avec checksum, et donner sa complexité.*

Solution. *Il faut mettre ensemble les questions précédentes, et le plus important est d'avoir le bon ordre dans les opérations. Voici les étapes de l'attaque :*

1. *On construit la multicollision, on récupère la valeur finale U ;*
2. *On calcule m^* (qui ne dépend que de U et des p_i) ;*
3. *On trouve un message $m_1, \ldots, m_{2n}$ dans la multicollision qui a la bonne valeur pour la checksum.*

La complexité est $2^{n-k} + n2^{n/2} + n^3$ (le dernier facteur étant négligeable), comme une attaque en seconde préimage sur MD standard.