Introduction à la cryptographie
TD4 – Fonctions de Hachage

Cécile Pierrot

24 Janvier 2019

1 Fonction de hachage à collisions fortes difficiles

Soit $f : \{0,1\}^{2m} \longrightarrow \{0,1\}^m$ une fonction de hachage. Soit maintenant une deuxième fonction de hachage définie par

$$h : \begin{array}{ccc} \{0,1\}^{4m} & \longrightarrow & \{0,1\}^m \\ x_1||x_2 & \mapsto & f(f(x_1)||f(x_2)) \end{array}$$

où $||$ désigne l'opération de concaténation.

Question 1. Montrer que si f est à collisions fortes difficiles [1], alors h est aussi à collisions fortes difficiles.

2 Algorithme Rho de Pollard pour la recherche de collisions

Nous considérons ici une fonction de hachage cryptographique H, qui prend en entrée un message $x \in \{0,1\}^*$ de taille quelconque et produit une empreinte $H(x) \in \{0,1\}^\ell$ de ℓ bits.

Question 1. Rappelez quelques valeurs typiques de ℓ.

Nous cherchons à trouver des collisions pour la fonction de hachage H. Une première méthode pour cela est de prendre une suite de messages m_i distincts, pour $i \geq 0$, puis de calculer $H(m_i)$ pour tout i, en insérant chaque empreinte dans une table de hachage, jusqu'à détecter une collision.

Question 2. Rappelez quel est le nombre d'essais moyen à effectuer avant de détecter une collision. Si l'on considère que l'insertion dans la table de hachage se fait en temps constant, quelle est la complexité en temps et en mémoire de cette recherche de collision ?

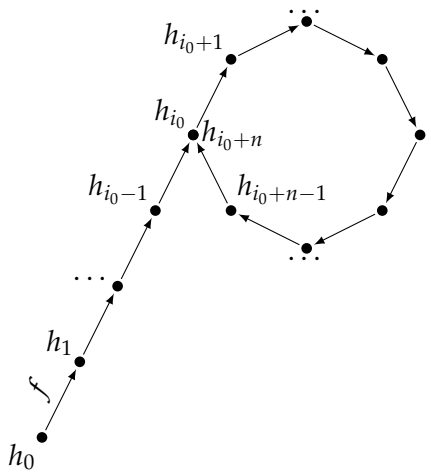
Nous nous dotons d'une fonction R déterministe allant de l'ensemble des empreintes possibles $\mathcal{H} = \{0,1\}^\ell$ à l'ensemble des messages $\mathcal{M} = \{0,1\}^*$.
Il est à noter que R peut très bien être l'identité, puisque chaque empreinte est un mot de ℓ bits et peut donc aussi bien être considérée comme un message. La seule contrainte que nous imposons sur R est qu'elle soit *injective*, c'est-à-dire qu'elle n'ait pas de collision : pour tous h et $h' \in \mathcal{H}$, si $h \neq h'$ alors $R(h) \neq R(h')$.
Nous choisissons alors h_0, une empreinte prise au hasard dans $\mathcal{H}$, et nous considérons la suite d'empreintes $(h_i)_{i\geq 0}$ définie par $h_{i+1} = H(R(h_i))$ pour tout $i \geq 0$. On pourra aussi noter $h_{i+1} = f(h_i) = f^{i+1}(h_0)$, avec la fonction $f = H \circ R$, définie de $\mathcal{H}$ dans $\mathcal{H}$.

1. C'est-à-dire qu'il est calculatoirement difficile d'obtenir deux messages différents x et x' tels que $f(x) = f(x')$.

Nous supposons de surcroît que cette suite se comporte comme une *suite aléatoire* dans l'ensemble $\mathcal{H}$. On parle alors de *marche pseudo-aléatoire* (« pseudo » car la fonction $f = H \circ R$ est déterministe).

Question 3. Montrez que la suite $(h_i)_{i\geq 0}$ est périodique à partir d'un certain rang. En d'autres termes, montrez qu'il existe deux entiers $i_0 \geq 0$ et $n > 0$ tels que $h_{i+n} = h_i$ pour tout $i \geq i_0$.

Dans la suite, nous prendrons pour $i_0 \geq 0$ et $n > 0$ les plus petites valeurs possibles telles que $h_{i+n} = h_i$ pour tout $i \geq i_0$. Nous supposerons aussi que $i_0 > 0$, c'est-à-dire que h_0 ne fait pas partie du cycle. On se retrouve ainsi avec une situation de la forme suivante, qui donne son nom à la méthode :



Question 4. Comment utiliser cette propriété pour trouver une collision de H ? Combien d'éléments de la suite faudra-t-il parcourir en moyenne pour cela ? Quelle est la complexité en temps et en mémoire d'une telle recherche ?

Afin de réduire la complexité en mémoire de cette méthode, on utilise l'algorithme de détection de cycles de Floyd (aussi appelé *méthode du lièvre et de la tortue*). Pour cela, on définit d'abord la sous-suite $(h'_i)_{i\geq 0} = (h_{2i})_{i\geq 0}$ formée des éléments d'indice pair de la suite $(h_i)_{i\geq 0}$. De manière équivalente, on peut aussi voir $(h'_i)_{i\geq 0}$ comme la suite

$$\begin{cases} h'_0 = h_0 \text{ et} \\ h'_{i+1} = f(f(h'_i)), \text{ pour tout } i \geq 0. \end{cases}$$

Question 5. Montrez qu'il existe un indice $j > 0$ tel que les deux suites $(h_i)_{i\geq 0}$ et $(h'_i)_{i\geq 0}$ ont la même valeur à l'indice j, c'est-à-dire que $h_j = h'_j$. Montrez que j est forcément un multiple de n, et que la plus petite valeur de j vérifiant cette condition, notée j_0, est située dans l'intervalle $i_0 \leq j_0 < i_0 + n$.
Astuce : montrez d'abord qu'un tel j est forcément supérieur ou égal à i_0, et que, s'il existe deux indices i et $i' \geq i_0$ tels que $h_i = h_{i'}$, alors la différence $i' - i$ est un multiple de n.

Question 6. Étant donnés H, R et h_0, proposez un algorithme permettant de calculer cet indice j_0, en ne stockant qu'un nombre constant d'empreintes en mémoire. Quelle est sa complexité en temps ?

Question 7. Une fois calculé cet indice j_0, comment retrouver la valeur de i_0 ?
Astuce : considérez les collisions entre les suites $(h_i)_{i\geq 0}$ et $(h''_i)_{i\geq 0} = (h_{j_0+i})_{i\geq 0}$.

Question 8. Déduisez des réponses précédentes un algorithme pour trouver une collision de H. Quelle est sa complexité en temps et en mémoire ?

3 Fonction de hachage basée sur AES

Soit $m = m_1 m_2 \dots m_n$ une chaîne de bits dans laquelle pour chaque $i = 1 \dots n$, m_i est un bloc de 128 bits. On définit une fonction de hachage H qui opère sur les mots binaires de cette forme en posant

- h_0 est un bloc de 128 bits tous nuls ;
- pour chaque $i = 1 \dots n$, $h_i = \mathrm{AES}_{m_i}(h_{i-1})$, où $\mathrm{AES}_K(m)$ est le résultat du chiffrement du bloc m avec la clé K ;
- $H(m) = h_n$.

Question 1. Montrer comment on peut trouver des collisions pour H en appliquant approximativement $c.2^{64}$ fois l'AES, où c est une constante.

Question 2. Etant donnée une chaîne m, montrer comment trouver une chaîne différente m' telle que $H(m) = H(m')$, en appliquant approximativement 2^{64} fois l'AES..