Fonctions de hachage

Gardons à l'esprit qu'un algorithme de recherche de collisions (resp. préimages, secondes préimages) est un algorithme probabiliste, qui n'a besoin que de réussir en moyenne.

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

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

Où $\|$ est une concaténation.

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

Solution. *Recall the definition of a collision attack : "can we write an algorithm faster than the birthday bound that outputs a collision ?"*

Here, the only assumption we have is that h is collision resistant. So we will reduce *the collision resistance to that of h. We do the proof by contraposition : we show that if h' is not collision resistant, then h is not collision resistant either.*

Suppose that h' is not collision resistant. There exists a collision attack, i.e., we have an algorithm $\mathcal{A}$ that outputs a collision faster than the generic birthday bound. Suppose that $\mathcal{A}$ outputs a pair (x, y).

We notice that $h'(x)$ cannot start with 0, because this would mean $h'(x) = 0\|x = h'(y) = 0\|y$ leading to a contradiction (we're supposed to have $x \neq y$). Thus $h'(x)$ starts with 1 and $h'(x) = 1\|h(x) = h'(y) = 1\|h(y)$ implying $h(x) = h(y)$. We can use the algorithm $\mathcal{A}$ as a collision attack on h. This concludes the proof.

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

Solution. *Notice that h is not assumed to be preimage resistant, but it is also not assumed to be broken. We simply cannot say anything on its preimage resistance.*

Recall the definition of a preimage attack : "can we write an algorithm faster than the exhaustive search, and which succeeds with constant probability ?". The last part is important, because it's OK if the attack does not succeed on all *possible targets. You can think of this as follows : when you attack the hash function, the target preimage will typically be chosen at random, and if the attack fails, you simply re-run it.*

So, let's design such an algorithm. Input a target t taken uniformly at random. If t starts with 0 then $t = 0\|t'$ and t' is a valid preimage of t, following the definition of h'. If t starts with 1, we just fail. Then we succeed with probability 1/2 on random inputs, which is enough to claim an attack.

Autour des définitions

Soit $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ un schéma de signature sûr sur l'espace de messages $\{0,1\}^n$. On définit un nouveau schéma de signature $(\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}')$ utilisant deux paires de clés de signature / vérification $(\mathsf{pk}_0, \mathsf{sk}_0)$ et $(\mathsf{pk}_1, \mathsf{sk}_1)$.

Question 3. *Dans cette question on sépare le message m en deux et on signe ses moitiés :*

$$\begin{cases} m := m_0 \| m_1 \\ \mathsf{Sign}'((\mathsf{sk}_0, \mathsf{sk}_1), m)) := \mathsf{Sign}(\mathsf{sk}_0, m_0), \mathsf{Sign}(\mathsf{sk}_1, m_1) \\ \mathsf{Verify}'((\mathsf{pk}_0, \mathsf{pk}_1), m, (\sigma_0, \sigma_1))) := \mathsf{Verify}(\mathsf{pk}_0, m_0, \sigma_0) \wedge \mathsf{Verify}(\mathsf{pk}_1, m_1, \sigma_1) \end{cases}$$

Est-ce que ce schéma est sûr ?

Solution. *Ce n'est pas sûr. À partir d'une signature de $(m_0 \| m_1)$ et de $(m'_0 \| m'_1)$ on peut créer une signature de $(m_0 \| m'_1)$ par exemple.*

On définit maintenant un schéma qui accepte si une des deux signatures est valide :

$$\begin{cases} \mathsf{Sign}'((\mathsf{sk}_0, \mathsf{sk}_1), m)) := \mathsf{Sign}(\mathsf{sk}_0, m), \mathsf{Sign}(\mathsf{sk}_1, m) \\ \mathsf{Verify}'((\mathsf{pk}_0, \mathsf{pk}_1), m, (\sigma_0, \sigma_1))) := \mathsf{Verify}(\mathsf{pk}_0, m, \sigma_0) \vee \mathsf{Verify}(\mathsf{pk}_1, m, \sigma_1) \end{cases}$$

On va démontrer que ce schéma est sûr.

Question 4. *Soit $\mathcal{B}$ un adversaire dans le jeu EUF-CMA pour la signature $(\mathsf{Sign}', \mathsf{Verify}')$. On définit un adversaire $\mathcal{A}$ dans le jeu EUF-CMA pour la signature $(\mathsf{Sign}, \mathsf{Verify})$, qui joue le rôle de challenger pour $\mathcal{B}$.*

- *Initialisation : $\mathcal{A}$ reçoit la clé pk de $\mathcal{C}$. Iel tire un bit b au hasard, ainsi qu'une clé $(\mathsf{pk}', \mathsf{sk}')$, et définit : $\mathsf{pk}_b := \mathsf{pk}, \mathsf{pk}_{1-b} = \mathsf{pk}', \mathsf{sk}_{1-b} = \mathsf{sk}'$.*
- *Requêtes : lorsque $\mathcal{B}$ effectue une requête de signature sur le message m, $\mathcal{A}$ transfère la requête à $\mathcal{C}$ et reçoit $\sigma = \mathsf{Sign}(\mathsf{sk}, m)$. $\mathcal{A}$ renvoie alors à $\mathcal{B}$:*
  - *$\sigma, \mathsf{Sign}(\mathsf{sk}_1, m)$ dans le cas $b = 0$*
  - *$\mathsf{Sign}(\mathsf{sk}_0, m), \sigma$ dans le cas $b = 1$*
- *Finalisation : $\mathcal{B}$ renvoie $(m, (\sigma_0, \sigma_1))$. $\mathcal{A}$ renvoie (m, σ_b).*

Montrer que :

$$\Pr[\mathsf{Verify}(\mathsf{pk}, m, \sigma_b) = 1] \geq \frac{1}{2} \Pr[\mathsf{Verify}'((\mathsf{pk}_0, \mathsf{pk}_1), m, (\sigma_0, \sigma_1))) = 1] \quad .$$

Conclure.

Solution. *On a :*

$$\begin{aligned} &\Pr[\mathsf{Verify}'((\mathsf{pk}_0, \mathsf{pk}_1), m, (\sigma_0, \sigma_1))) = 1] \\ &\qquad \leq \Pr[\mathsf{Verify}(\mathsf{pk}_0, m, \sigma_0) = 1] + \Pr[\mathsf{Verify}(\mathsf{pk}_1, m, \sigma_1) = 1] \\ &\qquad = \Pr[\mathsf{Verify}(\mathsf{pk}_b, m, \sigma_b) = 1 | b = 0] + \Pr[\mathsf{Verify}(\mathsf{pk}_b, m, \sigma_b) = 1 | b = 1] \\ &\qquad = 2 \Pr[\mathsf{Verify}(\mathsf{pk}_b, m, \sigma_b) = 1] \quad . \end{aligned}$$

Ce type d'argument s'applique à plus que deux copies : $\mathcal{A}$ devine à l'avance sur quelle clé l'attaque va avoir lieu. Cela permet de prouver génériquement la sécurité en "multi-clés" ou "multi-utilisateurs" des schémas que l'on utilise.

DSA

L'algorithme de signature DSA ("*Digital Signature Algorithm*") a été standardisé par le NIST en 1991. Aujourd'hui on le retrouve plus couramment sous sa version ECDSA, utilisant des courbes elliptiques.

On considère un grand nombre premier p tel que $p-1$ est divisible par un nombre premier q de taille « moyenne » . Soit g' un générateur de $\mathbb{Z}_p^*$, et $g = (g')^{(p-1)/q}$. On a donc $g^q = 1 \mod p$.

On considère une fonction de hachage $H : \{0,1\}^* \to \mathbb{Z}_q$.

DSA

KeyGen

- $x \leftarrow U(\mathbb{Z}_q^*)$
- $\mathsf{pk}, \mathsf{sk} := g^x, x$

$\mathsf{Sign}(\mathsf{sk}, m)$

- Calculer $h = H(m)$
- $k \leftarrow U(\mathbb{Z}_q^*)$
- $r \leftarrow (g^k \pmod p) \pmod q$
- $s \leftarrow (h + \mathsf{sk}r)k^{-1} \pmod q$
- Renvoyer (r, s)

$\mathsf{Verify}(\mathsf{pk}, m, (r, s))$

- Calculer $h = H(m)$
- $a \leftarrow hs^{-1} \pmod q$
- $b \leftarrow rs^{-1} \pmod q$
- $v \leftarrow (g^a \cdot h^b \pmod p) \pmod q$
- Renvoyer 1 ssi $v = r$.

Question 5. *Prouver que la signature DSA est correcte.*

Solution. *La signature calcule :*

$$s = k^{-1}(h + xr) \mod q$$

donc :

$$k = hs^{-1} + xrs^{-1} \mod q$$

Comme g est d'ordre q on a :

$$g^k = g^{hs^{-1}+xrs^{-1} \mod q} = g^a(g^x)^b \mod p$$

On a donc :

$$r = (g^k \mod p) \mod q = (g^a(g^x)^b \mod p) \mod q = v \ .$$

Question 6. *Peut-on réutiliser la valeur k pour plusieurs signatures ?*

Solution. *Non. En signant deux hachés h, h' on obtiendrait en effet : $(r, (h+xr)k^{-1})$ et $(r, (h'+xr)k^{-1})$ pour la même valeur de k et r (qui est connu).*

Donc un système de deux équations à deux inconnues (x, k) :

$$\begin{cases} h + xr = ka \mod q \\ h' + xr = kb \mod q \end{cases}$$

qui est trivialement soluble. Cette attaque fut notoirement utilisée contre la PlayStation 3 de Sony car la valeur de k était la même pour toutes les signatures.