Master CSI 2 2019-2020

Cryptanalyse — 4TCY902U
Responsable : G. Castagnos

TP 8 — Collisions sur une fonction de hachage

Le but de ce TP est de trouver des collisions sur les N premiers bits des hachés de la fonction de hachage SHA256, grâce à la méthode ρ de Pollard. Pour SHA256, on utilisera la fonction suivante utilisant la bibliothèque `hashlib` de Python. À un entier de N bits, on retourne l'entier correspondant aux N premiers bits de son haché par SHA256.

```
import hashlib
sha2 = hashlib.sha256
N = 24

def sha2Trunc(m):
    hashHexa = sha2(str(m)).hexdigest()
    return ZZ(hashHexa[:N//4],16)
```

Soit la fonction :

$$h : \{0, \dots, 2^N - 1\} \to \{0, \dots, 2^N - 1\},$$

correspondant à la fonction `sha2Trunc` ci dessus. Soit $x_0 \in \{0, \dots, 2^N - 1\}$. On considère la suite des itérés : $x_n = h(x_{n-1})$ pour $n \geqslant 1$. Cette suite est périodique. On désigne par $x_0, \dots, x_{c-1}$ la pré-période et par ℓ la longueur de la période. C'est à dire que $x_0, \dots, x_{c+\ell-1}$ sont tous distincts et $x_i = x_{i+\ell}$ pour $i \geqslant c$.

1 Programmer l'algorithme de détection de cycle de Floyd : à partir d'un entier $x_0 \in \{0, \dots, 2^N - 1\}$, on calcule simultanément x_i et x_{2i} pour $i \geqslant 1$ jusqu'à trouver une égalité. On obtient ainsi un élément x_i avec $c \leqslant i < c + \ell$.

2 Écrire un algorithme prenant en entrée un élément x_i trouvé précédemment et retournant la valeur de la période ℓ (faire « un tour de cycle » à partir de x_i).

3 Programmer le calcul de la collision : si $c > 0$ et $\ell > 1$, on a

$$h(x_{c-1}) = x_c = x_{c+\ell} = h(x_{c+\ell-1})$$

et $x_{c-1} \neq x_{c+\ell-1}$. Pour cela, partir de x_0 et x_ℓ et calculer x_i et $x_{\ell+i}$ jusqu'à trouver une égalité. En déduire une collision sur les N premiers bits de SHA256 avec $N = 16, 24, 32, 40, \ldots$

Remarque : pour trouver la collision, on aurait pu directement utiliser l'indice i trouvé à la question 1 qui donne un multiple de la période.

4 On cherche maintenant à trouver une collision sur les hachés de deux messages textes intelligibles. Pour cela, on se donne deux messages textes m et m' avec $m \neq m'$, écrit en majuscules. Par exemple, « ALICE CERTIFIE QU'ELLE DOIT NEUF EUROS ET CINQUANTE CENTIMES A OSCAR. » et « ALICE CERTIFIE QU'ELLE DOIT MILLE DEUX CENTS EUROS ET CINQUANTE CENTIMES A OSCAR.».

On considère l'ensemble $\mathscr{M}$ des messages textes obtenus à partir de m en basculant une ou plusieurs lettres en minuscules. De même on construit un ensemble $\mathscr{M}'$ à partir de m'. On cherche ensuite un élément de $\mathscr{M}$ et un élément de $\mathscr{M}'$ ayant le même haché.

Pour cela, on considère une fonction injective g de $\{0, \ldots, 2^N - 1\}$ à valeurs dans $\mathscr{M} \cup \mathscr{M}'$ associant à un entier de bits $b_0, \ldots, b_{N-1}$, un message de $\mathscr{M}$ (resp. de $\mathscr{M}'$) si $b_0 = 0$ (resp. si $b_0 = 1$), construit en basculant en minuscule la i-ème lettre de m (resp. de m') si $b_i = 1$.

Programmer cette fonction g et trouver une collision en appliquant vos programmes à la fonction $\tilde{h} \circ g$, où $\tilde{h} : \mathscr{M} \cup \mathscr{M}' \to \{0, \ldots, 2^N - 1\}$, correspond à la fonction `sha2Trunc` appliquée sur les éléments de $\mathscr{M}$ et $\mathscr{M}'$.