INFO002 : cryptologie, 2021–2022
TD 4 : hash, MAC

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 17, poste : 94 22
email : Pierre.Hyvernat@univ-smb.fr
www : http://www.lama.univ-smb.fr/~hyvernat/

Exercice 1 : empreintes et MAC

Question 1. Quelle différence y a t'il entre une empreinte et un MAC ?

Question 2. On suppose qu'un attaquant peut calculer 2^{128} empreintes. Pourquoi faut-il utiliser une empreinte de taille 256 si on souhaite empêcher la recherche de collisions ?

Question 3. Alice et Bob veulent jouer à pile ou face par email. Décrivez un protocole pour faire ceci... (On parle de "mise en gage", ou "commitment scheme".)

Question 4. De nombreuses fonctions de hachage cryptographiques sont construites à partir d'une *fonction de compression* à sens unique : $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ (construction de Merkle-Damgard).

Décrivez cette méthode.

Question 5. Voici, en pseudo-Python, une fonction de recherche de collisions pour la fonction `H` :

```
s = "une grande chaine de caractères ..."
all_s = {s}
all_h = {H(s)}
while True:
    s = f(s)  # on fait une modification de s
    h = H(s)
    assert s not in all_s
    if h in all_h:
        break  # on a trouvé une collision
    all_s.add(s)
    all_h.add(h)
for t in all_s:
    if h == H(t):
        print(f"collision: {s} / {t}")
        break
else:
    assert(False)
```

Le paradoxe des anniversaires implique qu'il faut en moyenne calculer $2^{n/2}$ empreintes de taille n avant de trouver une collision.

Quelle est la complexité en espace de ce morceau de code ?

Question 6. Un autre algorithme de recherche (moins naïf) est le suivant : en partant d'une valeur x_0, on cherche une collision dans la suite

$$x_0, x_1 = H(x_0), x_2 = H(x_1), \dots, x_{i+1} = H(x_i), \dots\dots$$

En effet, cette suite est forcément (ultimement) périodique.

Pour ceci, on utilise l'algorithme du lièvre et de la tortue (Floyd) en calculant en "parallèle" les suites x_i et x_{2i} jusqu'à trouver une valeur t.q. $x_i = x_{2i}$.

- Écrivez une boucle qui cherche une telle valeur de i vérifiant $x_i = x_{2i}$. Quelle est la complexité en espace de votre algorithme ?
- Une fois ce i trouvé, on sait que $x_k = x_{k+i}$ dès que $k \geq i$. Écrivez une boucle qui cherche *le premier* k qui fonctionne. (On aura donc $x_{k-1} \neq x_{k-1+i}$.)
- Comment peut-on garantir que cette valeur k est différente de 0 ?
- Déduisez en une collision pour H. Quelle est la complexité en espace de l'algorithme complet ?

Exercice 2 : "Message Authentification Codes"

Question 1. Rappelez la définition de sécurité calculatoire pour les MAC.

Question 2. On suppose que $F(\_,\_)$ est une "bonne" fonction de chiffrement par blocs. Elle donne un MAC sûr pour les messages de 1 bloc.

Pour les messages de plusieurs blocs, on considère les MAC suivants, où k est une clé secrète partagée par Alice et Bob, et $m_1 \,||\, m_2 \,||\, ... \,||\, m_l$ est un message clair de l blocs.

- Alice tire un bloc r aléatoire et calcule

$$c = F(k, r) \oplus F(k, m_1) \oplus \ldots \oplus F(k, m_l).$$

  Le MAC est alors (r, c).

- Alice calcule

$$c_i = \underbrace{F(k, F(k, F(k, ...F(k,}_{i \text{ fois}} m_i)...))$$

  Le MAC est alors $c_1 \,||\, c_2 \,||\, \ldots \,||\, c_l$ et fait la même taille que le message.

- Alice tire un bloc c_0 aléatoire et calcule le chiffrement F-CBC

$$c_{i+1} = F(k, m_{i+1} \oplus c_i)$$

  Le MAC est alors $c_0 \,||\, c_1 \,||\, \ldots \,||\, c_l$ et fait un bloc de plus que le message.

- comme précédemment, mais le MAC contient alors uniquement le dernier bloc c_l (et du vecteur d'initialisation c_0).
- comme précédemment, mais le bloc c_0 est fixé à $0 \ldots 0$. (Difficile)

Montrez qu'aucun de ces systèmes d'authentification n'est sûr au sens de la question précédente.

Question 3. CBC-MAC pour une fonction de chiffrement par blocs F fonctionne presque comme le dernier essai de la question précédente : le MAC est le dernier bloc du chiffrement CBC avec un vecteur d'initialisation à $0 \ldots$. Pour corriger le problème évoqué plus haut, on ajoute un bloc initial contenant la taille du message : $m_1 \,||\, \ldots \,||\, m_l$ est transformé en $l \,||\, m_1 \,||\, \ldots \,||\, m_l$.

Nous allons montrer que si la taille l est ajoutée à la fin du message, le MAC correspondant n'est pas sûr !

- Calculez les MAC c_1 et c_2 des message m_1 et m_2, chacun de 1 bloc, lorsque leur taille est ajoutée en fin.
- Calculez le MAC c_3 du message $m_1 \,||\, 1 \,||\, m_3$.
- Que pouvez-vous dire du MAC c_4 du message $m_2 \,||\, 1 \,||\, (c_1 \oplus c_2 \oplus m_3)$?
- Concluez.

Question 4. Une méthode de construction d'un MAC à partir d'une fonction de hachage est d'utiliser

$$\mathrm{MAC}(k, m) = H(k||m)$$

MD5, SHA-1 et la famille SHA-2 utilisent la construction de Merkle-Damgard (cf. exercice précédent). Expliquez comment créer un nouveau MAC valide à partir de

- un message m (sur un nombre entier de blocs),
- sont MAC calculé comme si dessus.

L'attaquant ne connait bien sûr pas la clé utilisée...

Indice : essayez d'agrandir le message avec un nouveau bloc.