TD1 : Généralités, chiffrements par flots

1. **LFSR.**

   On considère le LFSR associé à la fonction de rétroaction

   $$f(x_0, x_1, x_2, x_3, x_4, x_5) = x_2 \oplus x_5$$

   avec la graine 001000.

   (a) Calculez les 10 premiers bits de sortie. Commentaire ?

   (b) On suppose que lors de six itérations successives numétotées N à $N + 5$, les bits produits par le LFSR sont dans l'ordre : $0, 1, 0, 1, 0, 1$. Quel est l'état interne avant l'étape N ? Quels sont les bits suivants ?

2. **Malléabilité des chiffrements par flots**

   Dans cet exercice, nous considérons un chiffrement par flot, noté E, paramétré par une clé secréte K.

   (a) Rappelez le principe général de fonctionnement d'un chiffrement par flot. Étant donnés un message en clair M, une clé K, comment le chiffré C est-il obtenu ?

   Supposons qu'Alice ait envie de faire un virement bancaire de 100 euros à Mallory. Pour cela, elle utilise un système de chiffrement par flot E dont seules elle et sa banque connaissent la clé privée K. Alice chiffre donc l'ordre de virement M qu'elle envoie alors à sa banque.

   Mallory est capable d'intercepter et de modifier ce message chiffré C avant que la banque d'Alice ne le reçoive. Elle ne connait pas M, mais elle sait que les ordres de virement sont des chaînes de caractères de la forme suivante :

   `M = ⟨date⟩:⟨nonce⟩:⟨emetteur⟩:⟨destinataire⟩:⟨montant⟩:⟨commentaire⟩`

   où nonce est une chaîne aléatoire de 8 chiffres décimaux, que la banque aura transmise à Alice juste avant que celle-ci ne prépare son ordre de virement.

   (b) À quoi sert ce nonce ?

   Dans le cas d'Alice et Mallory, le message est donc de la forme suivante :

   `M = 2019-01-28:⟨nonce⟩:Alice:Mallory:100:⟨commentaire⟩`

   (c) Comment Mallory peut-elle faire pour obtenir 999 euros de la part d'Alice ?

3. **Réseaux de Feistel**

   (a) Décrire la fonction inverse d'un tour de réseau de Feistel.

   (b) Nommons F la fonction interne d'un réseau de Feistel pour $2n = 32 + 32$. Pour une valeur de clé de tour K fixée, la fonction $x \to F(K, x)$ est donc une fonction de 32 bits vers 32 bits. Combien existe-t-il de fonctions distinctes de 32 bits vers 32 bits ?

   (c) Si on considère l'ensemble des valeurs possibles de la clé de tour (qu'on suppose sur 48 bits), que pouvez-vous dire de la proportion que repr´esentent les fonctions de la forme $x \to F(K, x)$ qui peuvent ^etre ainsi construites, parmi toutes les fonctions de 32 bits vers 32 bits ?

(d) Considérons un réseau de Feistel à 16 tours, avec une clé maîtresse de 56 bits. Montrer que pour une valeur donnée de la clé maîtresse, la fonction de chiffrement est une permutation d'un ensemble à 2^{64} éléments. Combien de telles permutations existent ? Que pouvez-vous dire de l'ensemble des permutations qui peuvent être construites grâce à cette structure de réseau de Feistel, en faisant varier les clés ? Quelle est leur proportion ?

4. **Meet in the middle**

Les algorithmes de chiffrement symétrique et de déchiffrement symétrique sont publics, et la sécurité repose sur une clé secrète K, qui sert à la fois à chiffrer et à déchiffrer.

Pour chiffrer le message M à l'aide de la clé secréte K, Alice calcule le chiffré $C = ENC(K, M)$. Pour déchiffrer le message chiffré C à l'aide de la clé K, Bob calcule $M = DEC(K, C)$. Un attaquant a découvert M et C, mais le chiffrement ne permet pas d'en déduire K. Or l'attaquant voudrait connaître la clé K pour pouvoir lire les prochains messages chiffrés d'Alice. Il sait seulement que K est un nombre composé de n bits en binaire. Pour découvrir K, l'attaquant décide d'essayer de chiffrer M avec toutes les valeurs possibles de la clé pour tenter de retrouver C : c'est une attaque par recherche exhaustive. Puisqu'il sait que la longueur de la clé qu'il cherche est de n bits, il devra tester, dans le pire des cas, tous les nombres composés de n bits, et il y en a 2^n.

Pour avoir une meilleure sécurité, Alice a l'idée de chiffrer deux fois son message avec deux clés différentes K_1 et K_2 :

$$C = ENC(K_2, ENC(K_1, M))$$

De cette façon, elle se dit que, en supposant que K_1 et K_2 comportent toutes les deux n bits, l'attaquant devra effectuer dans le pire des cas $2^n \times 2^n = 2^{2n}$ tests pour découvrir les deux clés.

Donner un moyen de découvrir les deux clés beaucoup plus efficace.