Chiffrement et déchiffrement à la *Bluetooth*

1 Masque jetable

Le *masque jetable*, également appelé *chiffre de Vernam* est un algorithme de cryptographie inventé par Gilbert Vernam en 1917. Bien que simple, ce chiffrement est le seul qui soit théoriquement impossible à casser, même s'il présente d'importantes difficultés de mise en œuvre pratique.

1.1 Principe

Le chiffrement par la méthode du masque jetable consiste à combiner le message en clair avec une clé présentant les caractéristiques très particulières suivantes :

- La clé doit être une suite binaire aussi longue que le message à chiffrer.
- Les bits composant la clé doivent être choisis de façon totalement aléatoire.
- Chaque clé, ou « masque », ne doit être utilisée qu'une seule fois (d'où le nom de masque jetable).

La méthode de combinaison entre le clair et la clé est simple et sera décrite ci-dessous.
L'intérêt considérable de cette méthode de chiffrement, c'est que si les trois règles ci-dessus sont respectées strictement, le système offre une sécurité théorique absolue, comme l'a prouvé Claude Shannon en 1949:

> L'argument théorique est le suivant, dans son principe : si on ne connaît que le texte chiffré, et que toutes les clés sont équiprobables alors tous les textes clairs sont possibles et avec la même probabilité puisqu'il y a bijection, une fois le chiffré fixé, entre clés et textes clairs. Même une attaque par force brute ne donnera aucune information.

Ce type d'impossibilité, appelé sécurité sémantique, ne repose pas sur la difficulté du calcul, comme c'est le cas avec les autres systèmes de chiffrement en usage. Autrement dit, le système du masque jetable est inconditionnellement sûr.

1.2 Méthode de chiffrement et déchiffrement

Lorsque les données sont sous forme binaire, la méthode se réduit à un calcul particulièrement simple et très efficace en pratique. Le message en clair, à chiffrer, se présente comme une suite de bits. La clé est une autre suite de bits, de même longueur. On traite un à un les bits du clair, en combinant chacun avec le bit de même rang dans la clé, en leur appliquant l'operation *xor* : $\oplus$. Celle-ci est définie par le tableau suivant.

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Le déchiffrement s'effectue en combinant le chiffré avec les bits de la clé. L'application de l'opération *xor* étant simple en informatique, ces traitements peuvent s'effectuer à très grande vitesse.

1.3 Difficultés de mise en œuvre

La première difficulté que présente ce système est la longueur et le nombre des clés nécessaires, avec le problème de leur transmission au correspondant. Ensuite générer des clés réellement aléatoires nécessite des moyens complexes. Enfin garantir l'utilisation unique de chaque clé, même à des années d'intervalle, pose des problèmes d'organisation importants : à défaut, la sécurité du système peut être compromise.

2 Générateur de nombres pseudo-aléatoires

Un générateur de nombres pseudo-aléatoires (*pseudorandom number generator* (PRNG) en anglais) est un algorithme qui génère une séquence de nombres présentant certaines propriétés du hasard. Par exemple, les nombres sont supposés être approximativement indépendants les uns des autres, et il est potentiellement difficile de repérer des groupes de nombres qui suivent une certaine règle (comportements de groupe).
Cependant, les sorties d'un tel générateur ne sont pas entièrement aléatoires ; elles s'approchent seulement des propriétés idéales des sources complètement aléatoires. John von Neumann insista sur ce fait avec la remarque suivante : « Quiconque considère des méthodes arithmétiques pour produire des nombres aléatoires est, bien sûr, en train de commettre un péché ». De vrais nombres aléatoires peuvent être produits avec du matériel qui tire parti de certaines propriétés physiques stochastiques (bruit d'une résistance par exemple).
La raison pour laquelle on se contente d'un rendu pseudo-aléatoire est : d'une part qu'il est difficile d'obtenir de « vrais » nombres aléatoires et que, dans certaines situations, il est possible d'utiliser des nombres pseudo-aléatoires, en lieu et place de vrais nombres aléatoires ; d'autre part, que ce sont des générateurs particulièrement adaptés à une implémentation informatique, donc plus facilement et plus efficacement utilisables.

2.1 Générateurs congruentiels linéaires

Introduits en 1948 par D. H. Lehmer sous une forme réduite, ils vont être généralisés et seront largement utilisés ensuite. Ils reposent sur une simple formule de récurrence :

$$X_{n+1} = (a \cdot X_n + c) \mod m$$

avec X_0 la graine (*seed* en anglais : un nombre employé pour produire une suite pseudo-aléatoire habituellement plus longue) et avec a, c, m des entiers convenablement choisis.
Certains générateurs pseudo-aléatoires peuvent être qualifiés de cryptographiques quand ils font preuve de certaines propriétés nécessaires pour qu'ils puissent être utilisés en cryptologie. Ils doivent être capables de produire une sortie suffisamment peu discernable d'un aléa parfait et doivent résister à des attaques comme par exemple l'injection de données forgées de manière à produire des imperfections dans l'algorithme, ou encore des analyses statistiques qui permettraient de prédire la suite.

2.2 Générateurs par registres à décalage linéaire

Une généralisation des générateurs congruentiels linéaires consiste à ne plus utiliser seulement la précédente valeur pour fabriquer l'élément suivant de la séquence, mais plusieurs des précédentes valeurs, c'est-à-dire que X_n est calculé par des combinaisons linéaires de $X_{n-1}, X_{n-2}, \ldots, X_{n-k}$. Autrement dit :

$$X_n = (a_1 \cdot X_{n-1} + a_2 \cdot X_{n-2} + \ldots + a_k \cdot X_{n-k}) \mod m.$$

Dans ce cas la graine est $(X_{k-1}, X_{k-2}, \ldots, X_0)$. Ces générateurs sont particulièrement intéressants si m est un nombre premier car leur période maximale est alors $m^k - 1$. Ainsi il est possible, même avec un petit module, d'avoir de très grandes périodes. Par exemple, pour générer des suites aléatoires de bits, on choisit $m = 2$. Dans ce cas $a_i \in \{0, 1\}$, les opérations peuvent être réalisées très rapidement sur machine par des *xor* pour l'addition modulo 2 et par des décalages pour générer les bits suivants.
Il existe même des puces spécialisées réalisant les opérations nécessaires. On parle alors de *registres à décalage linéaire*, abrégé par *LFSR* (de l'anglais Linear Feedhack Shift Registers).

Exemple : Le LFSR

$$X_n = X_{n-1} \oplus X_{n-2} \oplus X_{n-4},$$

avec la graine

$$X_0 = 0, X_1 = 1, X_2 = 1, X_3 = 0$$

correspond à $m = 2$, $a_1 = 1$, $a_2 = 1$, $a_3 = 0$, $a_4 = 1$ et au $k = 4$. Les valeurs ainsi obtenues sont : $X_4 = 1$, $X_5 = 0$, $X_6 = 0$, $X_7 = 0$, $X_8 = 1$, $X_9 = 1$, ...

3 Réaliser

Réaliser un programme informatique (dans un langage à votre choix) de chiffrement et de déchiffrement basé sur un masque engendré par un LFSR. La sécurisation de ce protocole utilisera un schéma de type Verman mais avec un générateur pseudo-aléatore à base de LFSR; la graine représentant la clé secrète courte. Le choix du LFSR sera:

1. $$X_n = X_{n-5} \oplus X_{n-6} \oplus X_{n-15} \oplus X_{n-16}$$

   avec la clé secréte courte (= la graine) constituée par les valeurs $X_0, X_1, \ldots, X_{15}$.

2. $$\begin{cases} v_n &= v_{n-5} \oplus v_{n-13} \oplus v_{n-17} \oplus v_{n-25} \\ x_n &= x_{n-7} \oplus x_{n-15} \oplus x_{n-19} \oplus x_{n-31} \\ y_n &= y_{n-5} \oplus y_{n-9} \oplus y_{n-29} \oplus y_{n-33} \\ z_n &= z_{n-3} \oplus z_{n-11} \oplus z_{n-35} \oplus z_{n-39} \end{cases}$$

   Dans ce cas, la clé secrète est constituée de $25 + 31 + 33 + 39$ bits, et le bit secret engendré apres chaque itération est donné par $b = v_n \oplus x_n \oplus y_n \oplus z_n$ [1].

Le programme doit pouvoir chiffrer des données de type texte, image et son.

1. ce LFSR est utilisé dans le protocole de sécurisation *Bluetooth*