Cours de Cryptanalyse

Ecrit par Marion Candau

Enseignant : M.Guilhem Castagnos

Master 2 Cryptologie et Sécurité Informatique
Université Bordeaux 1

11 décembre 2010

Table des matières

Chapitre 1

Introduction

1.1 Cryptographie symétrique

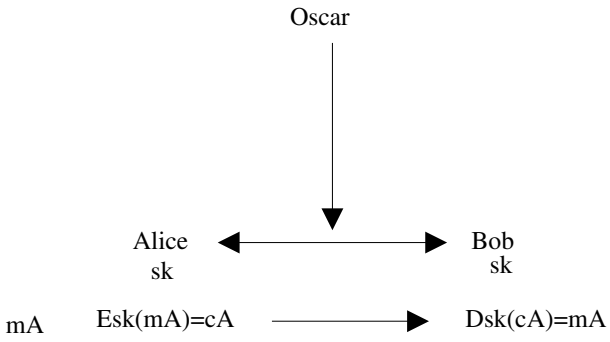


FIG. 1.1 – Schéma de cryptographie symétrique

La communication peut se faire dans les deux sens.

Définition : Algorithme de chiffrement symétrique
On a 3 algorithmes :

- Keygen : algorithme probabiliste avec en entrée un paramètre de sécurité et en sortie la clé secrète.
- Encrypt noté $E_{s_k}(m) = c$: algorithme déterministe avec en entrée le message en clair et la clé secrète et en sortie le message chiffré.
- Decrypt noté $D_{s_k}(c) = m$: algorithme déterministe avec en entrée le message chiffré et la clé secrète et en sortie le message en clair.

Le système est correct si : $D_{s_k}(E_{s_k}(m)) = m$.

Sécurité

Moyens de l'attaquant :

- attaque à chiffré seul
- attaque à clair connu : l'attaquant connait un ou un grand nombre de couples clairs/chiffrés.
- attaque à clair choisi : l'attaquant peut choisir les messages en clair dont il connait les chiffrés correspondants. On parle d'attaques adaptatives s'il choisit les clairs en fonction des résultats de cryptanalyse obtenus.
  Exemple : attaque en boite noire : l'algorithme de chiffrement est exécuté sur une carte à puce.
- attaque à chiffré choisi.

But de l'attaquant :

Principe de Kerkhoffs : l'algorithme de chiffrement/déchiffrement est connu, seule la clé est secrète.

- Retrouver la clé secrète
- Déchiffrer un chiffré : casser la notion de sens unique (OW)
- Retrouver une information sur le clair à partir de son chiffré : casser la sécurité sémantique

En pratique :

Si l'espace des clés est fini, l'attaquant peut retrouver la clé par recherche exhaustive.
Exemple : DES : attaque la plus efficace et facilement parallélisable. L'attaquant possède (m, c), il teste toutes les s_k possibles c'est-à-dire si $D_{s_k}(c) = m$.
Le temps moyen de l'attaque est : $\frac{\text{taille de l'espace des clés}}{2}$.
Deep Crack est une machine construite en 1998 pour 250000$ pour faire de la recherche exhaustive sur le DES. Elle a mis 3 jours à cracker DES. Aujourd'hui la taille des clés symétriques est au minimum 80 ou 128 bits.
Le but du cryptanalyste en cryptographie symétrique est de trouver une attaque plus rapide que la recherche exhaustive.

1.1.1 Chiffrement par flot

L'état E de l'algorithme de chiffrement évolue au cours du temps.
Ce chiffrement est plus rapide que le chiffrement par bloc surtout en matériel, car la complexité en matériel est plus faible. Il est donc utilisé dans les communications GSM (A5/1, 1987), bluetooth (EO), WiFi (WEP,RC4). Il y a

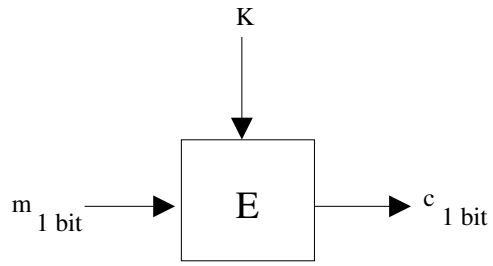


FIG. 1.2 – Schéma de chiffrement par flot

eu en 2004 un concours appelé ESTREAM, avec 35 candidats pour le chiffrement par flot, 7 ont été sélectionnés en 2008.

1.1.2 Chiffrement par bloc

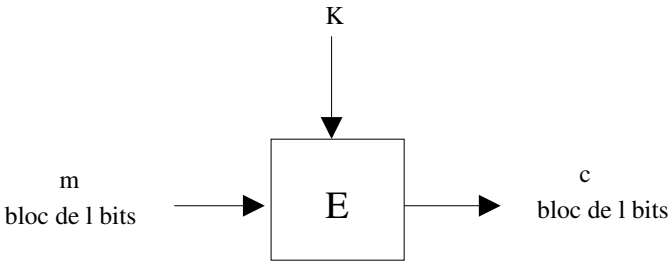


FIG. 1.3 – Schéma de chiffrement par bloc

Un même bloc est toujours chiffré de la même manière avec la même clé. Il faut que la taille des clés, l, soit assez grande, pour éviter les attaques par dictionnaire (on obtient les couples clairs-chiffrés de tous les blocs possibles).

Construction :
Elle est itérative, une même fonction est itérée successivement sur le bloc à chiffrer (tours, rondes) avec des clés de tours dérivées de la clé secrète.
Exemples : DES, en 1974, avec une clé de 56 bits, blocs de 64 bits.
AES, en 2000, clé de 128, 192 ou 256 bits, blocs de 128 bits.

Plusieurs modes d'opération :

- ECB : Electronic Code Book
- OFB : Output FeedBack : chiffrement par flot synchrone à partir d'un chiffrement par bloc.

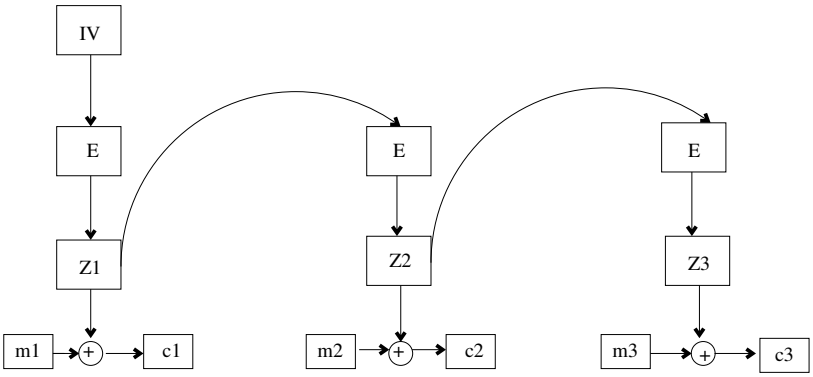


FIG. 1.4 – Schma de l'OFB

1.2 Cryptographie asymétrique

La définition est similaire pour un algorithme de chiffrement asymétrique :
- Keygen : algorithme probabiliste qui crée les clés publiques et privées.
- Encrypt : algorithme également probabiliste qui, à partir du message et de la clé publique du destinataire, crée le message chiffré. Un même message a plusieurs chiffrés possibles.
- Decrypt : algorithme déterministe qui, à partir du chiffré et de la clé privée, trouve le message en clair.

C'est correct si $D_{s_k}(E_{p_k}(m)) = m$.

Différences entre les chiffrements asymétriques et symétriques

La cryptographie asymétrique permet de faire des signatures. Elle est beaucoup plus longue (environ 1000 fois plus long). Elle permet l'échange de clés symétriques. Sa sécurité est basée sur la difficulté d'un problème algorithmique. Elle ne repose plus sur la recherche exhaustive, ça n'a aucun sens de comparer les tailles de clés symétriques et asymétriques pour la sécurité.

Sécurité : Définition d'un attaquant
Moyens :
- Attaques à clairs choisis (CPA)
- Attaques à chiffrés choisis adaptatives ou non (CCA-2 ou CCA-1)

Buts :

- Retrouver s_k (TB)
- A partir d'un chiffré retrouver un message en clair (OW)
- attaquer la sécurité sémantique : le chiffré donne des informations sur le message en clair $\Longleftrightarrow$ indistinguabilité (IND). L'adversaire choisit deux messages m_0, m_1 et il reçoit soit un chiffré de m_0 soit un chiffré de m_1. L'adversaire doit deviner si m_0 ou m_1 a été chiffré. Pour que cela soit le moins possible, il faut que l'algorithme de chiffrement soit probabiliste.

Différents niveaux de sécurité

- le plus faible : TB-CPA
- intermédiaire : IND-CPA
- le plus fort : IND-CCA-2

Pour chaque niveau de sécurité, on fait une réduction algorithmique entre le problème de casser la notion et un problème algorithmique supposé difficile.

Par exemple :

- Casser la notion TB-CPA pour RSA se ramène à factoriser la clé publique $n = pq$.
- Casser la notion OW-CPA pour RSA est équivalent à résoudre l'équation $c = x^e \mod n$.
- Pour El Gamal, casser la notion TB-CPA revient à résoudre un problème de logarithme discret.
- Casser OW-CPA (pour El Gamal) revient à résoudre le problème CDH : tant donnés g, g^a, g^b avec $< g >$ d'ordre q et a et b aléatoire $\mod q$ trouver g^{ab}
- Casser IND-CPA revient à résoudre DDH : décider si un triplet (x, y, z) est de la forme (g^a, g^b, g^{ab})

En général, le cryptanalyste s'attaque aux problèmes sous-jacents : la factorisation pour RSA ou le logarithme discret pour El Gamal. Dans ce cours, on verra les attaques sur RSA qui utilisent des réseaux euclidiens et leur réduction (LLL) : attaque sur le OW-CPA et RSA.

Plan du cours :

- Chiffrement par flot : construction et cryptanalyse

– Chiffrement par blocs : construction et cryptanalyse
– Attaques à base de réduction de réseaux euclidiens en cryptographie asymétrique

Chapitre 2

Rappels sur les corps finis

Définition : Caractéristique d'un anneau $(A, +, \times)$ d'unité 1

Soit $\begin{array}{l} \psi : \mathbb{N} \longrightarrow A \\ n \longmapsto 1 + 1 + \ldots + 1 \text{ (n fois)} \end{array}$

On note $car(A)$ le plus petit entier n_0 tel que $\psi(n_0) = 0$ s'il existe. Sinon on pose $car(A) = 0$.

Soit K un corps fini alors K a un nombre fini d'éléments et K est un corps c'est-à-dire que $(K, +, \times)$ est un anneau et que tout élément différent de 0 ($\in K^*$) est inversible.
$\Rightarrow car(K)$ est un nombre premier.

Proposition :
Soit K un corps fini de caractéristique p, p nombre premier alors K a nécessairement $q = p^n$ avec $n \geqslant 1$ éléments.

Notation : On note $\mathbb{F}_q$ ou $GF(q)$.

Proposition :
Soit K un corps fini avec $car(K) = p$ et K avec $q = p^n$ éléments alors K contient le corps à p éléments $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. De plus, K est un espace vectoriel de dimension n sur $\mathbb{F}_p$: il existe une base $(e_1, \ldots, e_n)$ d'éléments de K telle que :

$$\forall x \in K,\ \exists (a_1, \ldots, a_n) \in (\mathbb{F}_p)^n,\ x = \sum_{i=1}^{n} a_i e_i$$

Réciproquement, pour tout nombre premier p et $n \geqslant 1$, il existe un corps fini à p^n éléments.

Construction :

Soit P un polynôme à coefficients dans $\mathbb{F}_p$, de degré n, irréductible (seuls facteurs possibles : constantes de $\mathbb{F}_p$ et $\frac{p}{\lambda}$ où $\lambda \in \mathbb{F}_p^*$). Alors $\frac{\mathbb{F}_p[X]}{(P(X))}$ est un corps fini à p^n éléments.
Pour tout p premier et $n \geqslant 2$, il existe un polynôme irréductible sur $\mathbb{F}_p$ de degré n : on peut toujours faire cette construction. En pratique, on choisit P avec peu de coefficients pour faciliter les calculs.
On note $\alpha = X \mod P(X)$, par identification de $\mathbb{F}_q$ et $\frac{\mathbb{F}_p[X]}{(P(X))}$, on a $\forall \alpha \in \mathbb{F}_q, P(\alpha) = 0$. $(1, \alpha, \alpha^2, \ldots, \alpha^{n-1})$ est une base de $\mathbb{F}_q$ comme $\mathbb{F}_p$-espace vectoriel de dimension n. En magma, on a : $F < a >:= GF(p^n);$.

Ordre :

Si $\beta \in \mathbb{F}_q$, $\beta \neq 0$, on a $\beta^{q-1} = 1$ car $\mathbb{F}_q \backslash \{0\} = (\mathbb{F}_q)^*$ est le groupe multiplicatif de $\mathbb{F}_q$ et il a $q-1$ éléments. L'ordre de β est le plus petit entier e tel que $\beta^e = 1$. On a $e|(q-1)$.
Réciproquement, si $e|(q-1)$ alors il existe $\varphi(e)$ (φ indicatrice d'Euler) éléments d'ordre e. En particulier, il existe $\varphi(q-1)$ éléments d'ordre $q-1$, ce sont des générateurs de $\mathbb{F}_q^*$ qui est cyclique. Un tel générateur est appelé élément primitif.
Si $\alpha \in \mathbb{F}_q$ est un élément primitif, $\mathbb{F}_q$ est constitué de $0, \alpha, \alpha^2, \ldots, \alpha^{q-1} = 1$.

Remarque :

Si β est d'ordre e, β^i avec $1 \leqslant i \leqslant e$ est d'ordre $\frac{ppcm(e,i)}{i}$. En particulier, β^i est d'ordre e si $pgcd(e,i) = 1$. On en a bien $\varphi(e)$.

Polynôme minimal :

Soit $\beta \in \mathbb{F}_q$ avec $q = p^n$, le polynôme minimal de β est le polynôme unitaire, P sur $\mathbb{F}_p$, de plus petit degré tel que $P(\beta) = 0$.

Propriété : Le polynôme minimal est irréductible sur $\mathbb{F}_p$.

Proposition : Si P est irréductible de degré n sur $\mathbb{F}_p$ admettant $\alpha \in \mathbb{F}_{p^n}$ comme racine, ses autres racines sont $\alpha^p, \alpha^{p^2}, \alpha^{p^3}, \ldots, \alpha^{p^{n-1}}$. Toutes ces racines ont le même ordre.

Remarque : Si $a, b \in \mathbb{F}_{p^n}$, $(a+b)^p = a^p + b^p$ et si $a \in \mathbb{F}_p$, $a^p = a$.

Définition : Le polynôme minimal d'un élément primitif est dit polynôme primitif (toutes ses racines sont primitives).

Proposition : Les sous corps de $\mathbb{F}_{p^n}$ sont les $\mathbb{F}_{p^s}$ avec $s|n$. Les éléments de $\mathbb{F}_{p^n}$ qui sont dans $\mathbb{F}_{p^s}$ sont les racines de $X^{p^n} - X$.

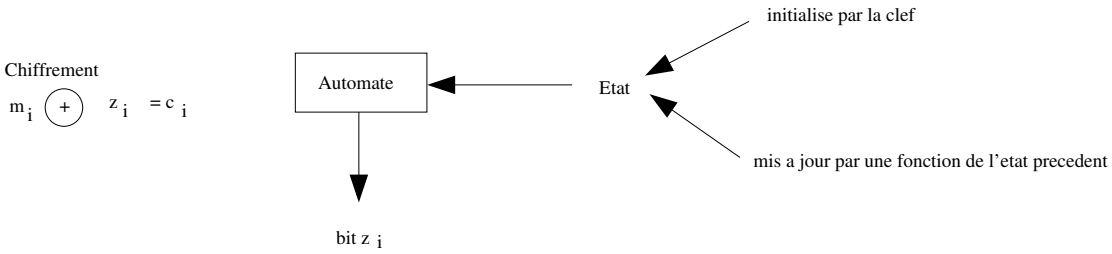
Chapitre 3

Les chiffrements par flot

3.1 Synchrones/auto-synchronisants

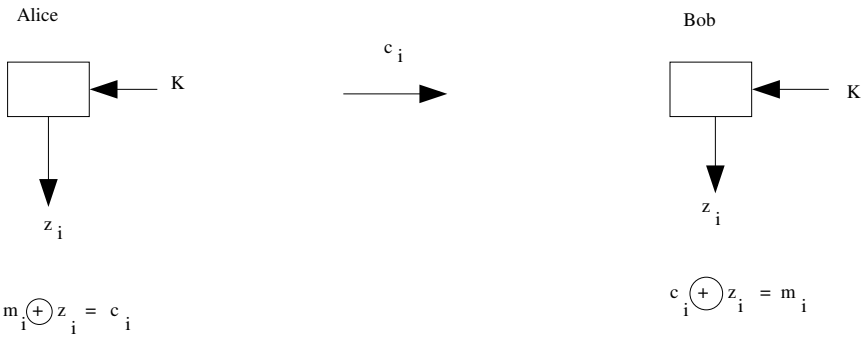
3.1.1 Synchrones

Définition :
La suite chiffrante est générée indépendamment du texte en clair et des textes chiffrés déjà traités. La suite chiffrante ne dépend que de la clé.



Exemple : mode OFB du chiffrement par bloc.

Propriété d'un chiffrement synchrone :
Alice et Bob doivent être synchronisés.



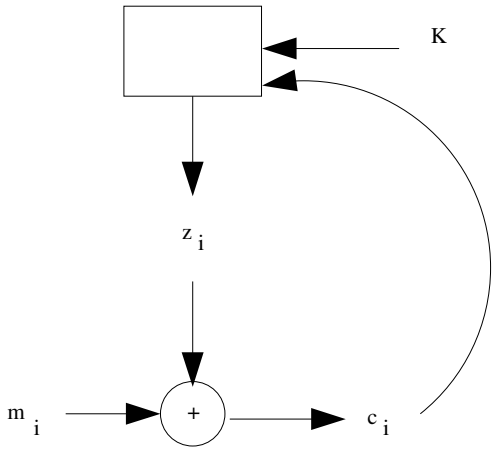
Avantages et inconvénients :

Alice et Bob doivent être synchronisés.
Si des caractères sont perdus ou rajoutés pendant la transmission, la synchronisation est perdue. La solution est l'insertion de marqueurs.
Si un caractère est modifié il perturbe seulement un seul bit du déchiffrement. Pour le détecter on vérifie l'intégrité.

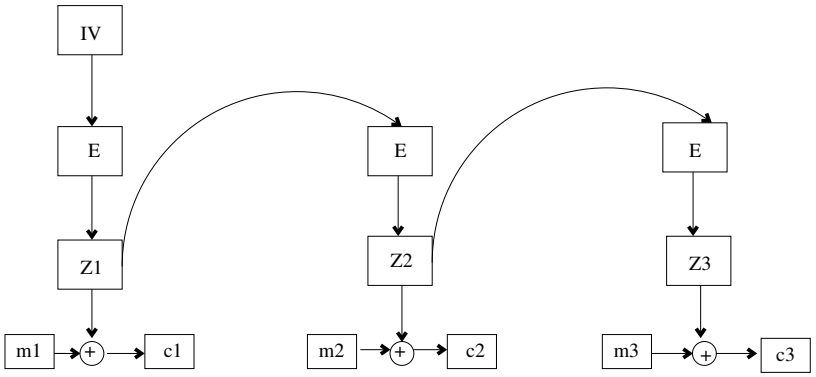
3.1.2 Systèmes auto-synchronisants

Définition :

La suite chiffrante ne dépend que de la clé et du texte chiffré précédemment généré.



Exemple : mode CFB des chiffrements par blocs.



Caractéristique d'un système auto-synchronisant :
La mise à jour de l'état dépend des t chiffrés précédents.
Dès que t chiffrés consécutifs valides sont transmis, on peut calculer le déchiffrement suivant, c'est donc auto-synchronisant.

Attaque active : modification d'un caractère
Cela modifie le déchiffrement des t bits suivants, c'est détecté par Bob.

Diffusion : un caractère m_i influe sur toute la suite du texte chiffré.
Cela rend plus difficile les attaques statistiques (basé sur la redondance du texte en clair). En pratique, il n'existe aucun chiffrement par flot auto-synchronisant (non basé sur CFB) sûr.

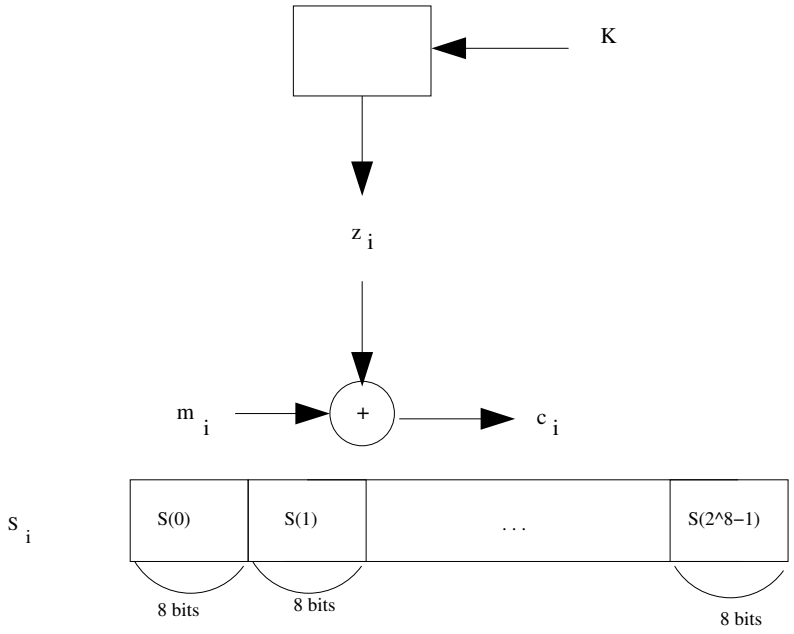
Dans la suite, on verra uniquement les systèmes synchrones additifs.

3.2 Chiffrement à flot additif synchrone

Ils sont inspirés du chiffrement de Vernam (sûr au sens de la théorie de l'information).
On aura un générateur pseudo-alatoire initialisé par une clé K. C'est un procédé déterministe, Bob doit pouvoir déchiffrer.

Exemple : RC4 (Rivest, 1987)
L'état interne est noté S de 8×2^8 bits, il représente les images d'une permutation de $\{0, \ldots, 2^8 - 1\}$ c'est-à-dire une permutation des mots binaires de 8 bits.

La clé secrète K est de longueur l avec $40 \leqslant l \leqslant 128$ bits. On note $K[i]$ le i-ème bit.



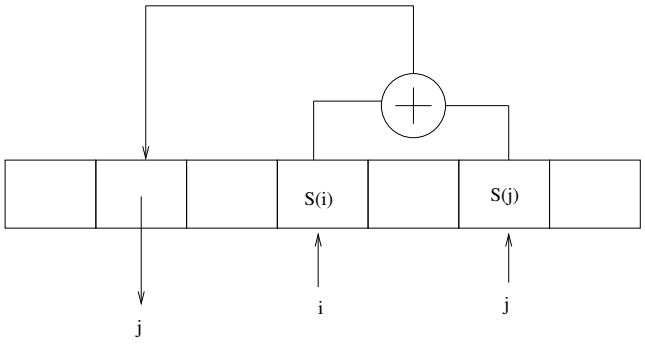
Initialisation :
La permutation S est l'identité : $0 \mapsto 0$, $1 \mapsto 1$, ..., $2^8 - 1 \mapsto 2^8 - 1$.

On pose $j = 0$
Pour $i = 0$ à $2^8 - 1$ **Faire**
$j \leftarrow (j + S[i] + K[i \mod l]) \mod 2^8$
Echanger $S[i]$ et $S[j]$
Fin pour

Mise à jour de l'état :
$i \leftarrow (i + 1) \mod 2^8$
$j \leftarrow (j + S[i]) \mod 2^8$
changer $S[i]$ et $S[j]$

Sortie :
$j \leftarrow S[S[i] + S[j] \mod 2^8]$
Cela est très rapide en logiciel, les principales attaques sont dues à de mauvaises utilisations (WEP).

Propiétés des chiffrements à flot additifs :
Problème : la même clé K produit la même suite chiffrante. Si M_1 et M_2



sont deux chaines de bits de même longueur, alors :

$$E_k(M_1) = M_1 \oplus Z$$

$$E_k(M_2) = M_2 \oplus Z$$

Donc $E_k(M_1) \oplus E_k(M_2) = M_1 \oplus M_2$. De même si on connait $E_k(M_1)$ et M_1, on a $Z = E_k(M_1) \oplus M_1$.

Solution :
L'IV est un vecteur d'initialisation utilisé en plus de K.

Initialisation du vecteur :
Etat initial : $E_0 = h(K, IV)$. Ensuite :
Pour $i = 0$ à $l - 1$ **Faire**
$z_i = g(E_i)$ (bit de suite chiffrante)
$E_{i+1} = f(E_i)$ (mise à jour de l'état)
Fin pour
Cela produit $z_0, z_1, \ldots, z_{l-1}$.
Avec une même clé K, on peut produire plusieurs suites chiffrantes en changeant l'IV.

Taille de clé/IV/état interne à utiliser :
- Taille de K : pour éviter la recherche exhaustive, la taille de K doit etre au moins de 80 ou 128 bits.
- Taille de l'IV : si on a deux fois le même IV avec la même clé K, on a la même suite chiffrante. D'où par exemple, WEP basé sur RC4 et un IV de 24 bits. Par le paradoxe des anniversaires, avec 2^{12} utilisations on retombe sur le même IV.
- Taille de l'état interne : la sortie du GPA au temps $t \geqslant t_0$ ne dépend que de l'état interne au temps t_0. Donc la recherche exhaustive est

impossible sur l'état interne si celui-ci est d'au moins 80 ou 128 bits. Une amélioration est possible : le compromis temps/mémoire :
- Précalculs : P opérations
- Mémoire : M bits de données
- Attaque active : T opérations

Recherche exhaustive :

On connait $z_{t_0}, z_{t_0+1}, z_{t_0+2}, \ldots$. On parcourt les 2^l états possibles, on produit quelques bits de suite chiffrante $\rightarrow s_i = z_{t_0}, z_{t_0+1}, \ldots$ Ils sont candidats pour l'état interne.
On a :
- $P = 0$
- M faible : constante $\times l$ bits.
- $T = 2^l$ opérations

Attaque par dictionnaire :

On note $\varphi : \{0,1\}^l \longrightarrow \{0,1\}^l$ qui à $E_{t_0} \longmapsto (z_{t_0}, z_{t_0+1}, \ldots, z_{t_0+l-1})$ avec E_{t_0} qui est l'état interne $t = t_0$ et $(z_{t_0}, z_{t_0+1}, \ldots, z_{t_0+l-1})$ les l bits suivants de suite chiffrante. En général, φ est "quasiment" une bijection.
On pré-calcule les 2^l images possibles pour φ : $\varphi(x_i) = y_i$. On stocke (x_i, y_i) dans une table triée selon les y_i. Lors de la phase active, on récupère $(z_{t_0}, z_{t_0+1}, \ldots, z_{t_0+l})$. On cherche y dans la table et x_i est ainsi l'état interne E_{t_0}.
On a :
- $P = 2^l$ opérations
- $M = 2^l \times 2^l$ bits
- T : immédiat

Compromis temps-mémoire

On pré-calcule les images de $2^{l/2}$ états internes pris au hasard pour φ, $\varphi(x_i) = y_i$. On stocke (x_i, y_i) dans une table triée selon les y_i. Dans la phase active, on récupère $(z_{t_0}, z_{t_0+1}, \ldots, z_{t_0+2^{l/2}+l-1})$ c'est-à-dire $2^{l/2} + l$ bits consécutifs de suite chiffrante. On a donc $2^{l/2}$ chaines de l bits de suite chiffrante. Lors de l'attaque active, pour y parcourant ces $2^{l/2}$ chaines de l bits, on regarde si y apparait parmi les y_i. Si oui, on retrouve l'état interne x_i, et on a une bonne probabilité pour que ça arrive.
On a :
- $P = 2^{l/2}$ opérations
- $M = 2^{l/2} \times 2^l + 2^{l/2}$
- $T = 2^{l/2}$ opérations

Conséquence : si on veut une sécurité de 2^k opérations, on prend un état interne de taille $2 \times k$ (en pratique 2×80 ou 2×128 bits).

Pourquoi le compromis temps/mémoire fonctionne ?

On a un espace E de cardinal m : on crée une liste L_1 de K_1 éléments de E triés au hasard avec une probabilité uniforme sans remise. On crée une deuxième liste L_2 de cardinal K_2 tirés au hasard avec remise. Quelle est la probabilité d'avoir une collision entre L_1 et L_2 ($L_1 \cap L_2 \neq \emptyset$) ?

- Probabilité que le premier élément de L_2 soit différent de tous les éléments de L_1 :
$$\frac{m - K_1}{m} = 1 - \frac{K_1}{m}$$
- Probabilité que le premier élément de L_1 soit différent de tous les éléments de L_2 :
$$\frac{m - K_1}{m} = 1 - \frac{K_1}{m}$$
- Probabilité que tous les éléments de L_2 soient différents de ceux de L_1 :
$$q(K_1, K_2, m) = \prod_{i=1}^{K_2} \left(1 - \frac{K_1}{m}\right) = \left(1 - \frac{K_1}{m}\right)^{K_2}$$
Or on a $\forall x \in \mathbb{R}$, $1 - x \leqslant e^{-x}$. D'où :
$$q(K_1, K_2, m) \leqslant \left(e^{\frac{-K_1}{m}}\right)^{K_2} = e^{\frac{-K_1 K_2}{m}}$$
- Probabilité d'avoir une collision :
$$P(K_1, K_2, m) = 1 - q(K_1, K_2, m) \geqslant 1 - e^{\frac{-K_1 K_2}{m}}$$

Pour le compromis temps mémoire :

$K_1 = K_2 = 2^{l/2}$ et $m = 2^l$, d'où $e^{\frac{-K_1 K_2}{m}} = e^{-1}$. D'où :

$$\text{Proba de collision } \geqslant 1 - e^{-1} = 0,63$$

Sécurité du GPA :

La connaissance de la suite chiffrante ne doit pas permettre de retrouver la clé K. Encore plus fort : la connaissance de N bits consécutifs ne permettent pas facilement de prévoir (avec probabilité $>> \frac{1}{2}$) la valeur du bit suivant. En particulier, la sortie du GPA doit être le plus proche possible d'une suite aléatoire.

Attaque par distingueur : le but est de déterminer si on a affaire à une

suite produite par un GPA ou à une suite aléatoire. On peut exploiter les biais statistiques dans la suite chiffrante. Si proba$(z_i = 0) = \frac{1}{2} + \epsilon$, on a, si $c_i = m_i \oplus z_i$, Proba$(c_i = m_i) = \frac{1}{2} + \epsilon$. Le biais fait fuir de l'information sur le message.

Critères statistiques pour la sortie d'un GPA

Le critère le plus classique est le critère de Golomb (1982). On note $s = (s_1 \dots s_N)$ une période de la suite chiffrante.

1. Dans chaque période, le nombre de zéro est à un près le même que le nombre de 1.
2. Une série (rum) est une succession de bits identiques encadrés par deux bits différents : 100001 (longueur 4), 0110 (longueur 2). Dans chaque période si S est l'ensemble des séries, si $2^k \leqslant |S| < 2^{k+1}$, alors on a :

$$\frac{|S|}{2} \text{ séries de longueur 1}$$

$$\frac{|S|}{4} \text{ séries de longueur 2}$$

$$\vdots$$

$$\frac{|S|}{2^k} \text{ séries de longueur } k$$

   et pour chaque longueur, autant de séries de 1 que de séries de 0.
3. La fonction d'auto-corrélation prend 2 valeurs suivant que $t = 0$ ou $t \neq 0$, $C(t) = \sum_{i=0}^{N-1} (-1)^{S_i + S_{i+t}}$. D'où S est indépendant de ses translatés.

$$C(0) = N$$

$$C(1) = C(2) = \ldots = C(N)$$

Exemple :

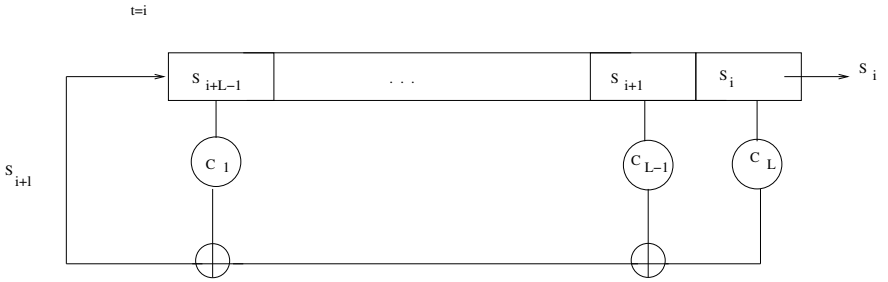
La suite de période 0,1,1,0,0,1,0,0,0,1,1,1,1,0,1 vérifie les critères de Golomb. On peut construire des GPA satisfaisant les 3 critères de Golomb : les LFSR.

Chapitre 4

LFSR

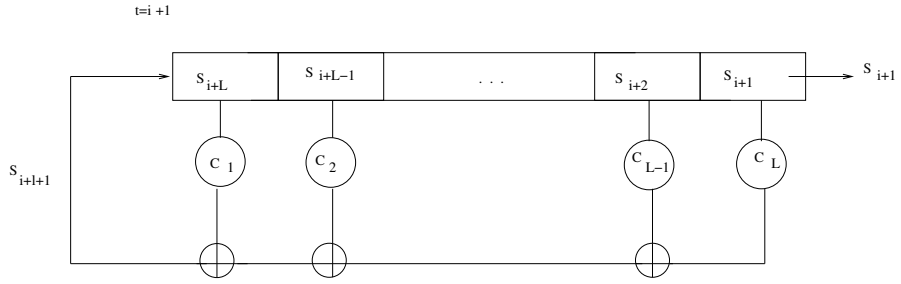
Linear Feedback Shift Register = Registres à décalage à rétroaction linéaire

Définition :
Un LFSR de longueur l est constitué d'un registre de longueur l noté $(s_i, s_{i+1}, \ldots, s_{i+l-1})$ au temps i, mis à jour par une fonction linéaire.

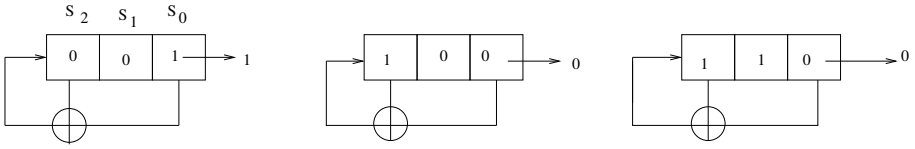


A chaque temps, on sort le bit i, on décale le contenu d'origine vers la droite et à gauche on met :

$$s_{i+l} = c_l s_i + c_{l-1} s_{i+1} + \ldots + c_1 s_{i+l-1}$$



Exemple : $L = 3$; $c_1 = c_3 = 1$, $c_2 = 0$



$$s_{i+3} = s_i + s_{i+2}$$

La suite produite est 1,0,0,1,1,1,0... A $t = 7$ on a le même registre qu'à $t = 0$, on a donc une période de 7 dans la suite chiffrante.

Proposition :

L'état initial $(s_0, \ldots, s_{L-1})$ détermine entièrement la suite produite $s = (s_i)_{i \geqslant 0}$. La suite s est une suite à récurrence linéaire d'ordre L.

Période :

Propriété :

La suite s produite par un LFSR est ultimement périodique s'il existe une pré-période de longueur i_0 telle que la suite $(s_i)_{i \geqslant i_0}$ est périodique. Si T est la période alors $s_{i+T} = s_i \ \forall i \geqslant i_0$. La période $T \leqslant 2^L - 1$ et si $c_L = 1$ la suite est périodique d'où $s_i = s_{i+T} \ \forall i \geqslant 0$

Démonstration :

Notons $R_i = \begin{pmatrix} s_i \\ s_{i+1} \\ \vdots \\ s_{i+L-1} \end{pmatrix}$ registre du LFSR au temps i. R_i peut prendre 2^L valeurs. Si on a $R_i = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ tous les registres suivants sont nuls et la suite produite est nulle à partir de $i : \forall j \geqslant i$, $s_j = 0$.

On suppose que le registre du LFSR n'est jamais nul. Parmi $[R_0, R_1 \ldots, R_{2^L-1}]$ au moins 2 registres sont égaux. Supposons que $R_{i_0} = R_{i_0+T}$. La suite de registre $[R_{i_0}, R_{i_0+1}, \ldots, R_{i_0+T-1}]$ se répète indéfiniment. En particulier, on a

$s_i = s_{i+T}$ pour tout $i \geqslant i_0$ avec $T \leqslant 2^L - 1$. On pose A de taille $L \times L$:

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & 0 \\ & & & \ddots & 1 \\ c_L & c_{L-1} & \dots & c_2 & c_1 \end{pmatrix}$$

$\forall i \geqslant 0$, on a :

$$R_{i+1} = \begin{pmatrix} s_{i+1} \\ s_{i+2} \\ \vdots \\ s_{i+L} \end{pmatrix} = A \times \begin{pmatrix} s_i \\ s_{i+1} \\ \vdots \\ s_{i+L-1} \end{pmatrix} = A \times R_i$$

$\forall n \geqslant 0$, on a $R_n = A^n R_0$
$\det(A) = c_L$ donc si $c_L \neq 0$, A est inversible et la condition $R_{i_0+T} = R_{i_0}$ devient :

$$A^{i_0} R_0 = A^{i_0+T} R_0 \Rightarrow R_0 = A^T R_0 = R_T$$

Donc $\forall i \geqslant 0,\ s_i = s_{i+T}$, s est périodique.
Dans la suite on supposera toujours que $c_L = 1$, donc que (s) est périodique.

Définition :
Si la suite produite par un LFSR de longueur L est de période $2^L - 1$, on dit que la suite est maximale. On parle aussi de m-suite ou m-séquence.

Remarque :
Si on a une m-suite, le registre du LFSR passe par des $2^L - 1$ états non nuls pendant une période si l'initialisation est non nulle. En particulier, deux suites produites par le même LFSR avec des initialisations différentes sont décalées l'une de l'autre.

Retour à l'exemple : $L = 3, c_1 = c_3 = 1$
$R_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ $R_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ $R_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$ $R_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ $R_4 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ $R_5 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ $R_6 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ $R_7 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = R_0$
La période est de $7 = 2^3 - 1$. C'est donc une m-suite.

Définition :

On associe à une suite de $\mathbb{F}_2$, $s = (s_n)_{n \geqslant 0}$ la série génératrice :

$$s(X) = \sum_{n \geqslant 0} s_n X^n$$

Si s est produite par un LFSR de longueur L, son polynôme de rétroaction est :

$$f(X) = 1 + c_1 X + c_2 X^2 + \ldots + c_L X^L \in \mathbb{F}_2$$

(Dans l'exemple, on a : $1 + X + X^3$).
On pose $c_0 = 1$.

Proposition :

Une suite s est produite par un LFSR de polynôme de rétroaction $1+c_1X+c_2X^2+\ldots+c_LX^L$ si et seulement si son développement en série formelle est $s(X) = \dfrac{g(X)}{f(X)}$ où $g \in \mathbb{F}_2[X]$ est un polynôme de degré infrieur ou égal au degré de f. De plus, g est entièrement déterminé par l'état initial du registre :

$$g(X) = \sum_{i=0}^{L-1} X^i \times \left(\sum_{j=0}^{i} c_j s_{i-j} \right)$$

Démonstration :

Supposons s produite par un LFSR de polynôme de rétroaction $f(X) = 1 + c_1 X + c_2 X^2 + \ldots + c_L X^L$. On calcule :

$$s(X)f(X) = (s_0+s_1X+s_2X^2+\ldots)(c_0+c_1X+c_2X^2+\ldots+c_LX^L) = g_0+g_1X+g_2X^2+\ldots$$

On a :
$g_0 = s_0 c_0$
$g_1 = s_1 c_0 + s_0 c_1$
$g_2 = s_0 c_2 + s_1 c_1 + s_2 c_0$
$\forall i, 0 \leqslant i < L$

$$g_i = \sum_{j=0}^{i} c_j s_{i-j}$$

Pour tout $i \geqslant 0$,

$$g_{L+i} = \sum_{j=0}^{L} c_j s_{L+i-j} = c_0 s_{L+i} + c_1 s_{L+i-1} + \ldots + c_L s_i$$

Or $c_1 s_{L+i-1} + \ldots + c_L s_i = s_{L+i}$ (c'est la formule de calcul de la rétroaction). D'où : $g_{L+i} = s_{L+i} + s_{L+i} = 0$. Donc g est un polynôme de degré $< L$ avec cette expression :

$$g(X) = \sum_{i=0}^{L-1} X^i \times \left(\sum_{j=0}^{i} c_j s_{i-j} \right)$$

On peut voir un LFSR comme un automate qui calcule la division selon les puissances croissantes de 2 polynômes de $\mathbb{F}_2$.

Retour à l'exemple : $L = 3, c_1 = c_3 = 1$
$f(X) = 1 + X + X^3 = 1 + c_1 X + c_2 X^2 + c_3 X^3$.
On a $g(X) = s_0 c_0 + (s_0 c_1 + s_1 c_0)X + (c_0 s_2 + c_1 s_1 + c_2 s_0)X^2 = 1 + X$ et $s(X) = \dfrac{g(X)}{f(X)}$.

$$\begin{array}{lllll|l} 1 & + & X & & & 1 + X + X^3 \\ & X^3 & & & & \hline \\ X^4 & + & X^6 & & & 1 + X^3 + X^4 + X^5 + \ldots \\ X^5 & + & X^6 & + & X^7 & \\ & & & & & \end{array}$$

On a : $1 + X^3 + X^4 + X^5 + \ldots = s(X) = s_0 + s_1 X + s_2 X^2 + s_3 X^3 + \ldots$. On a donc : $s_0 = 1$, $s_1 = 0$, $s_2 = 0$, $s_3 = 1$, $s_4 = 1$, $s_5 = 1$, ...
Si $s(X) = \dfrac{g(X)}{f(X)}$ et si $g(X) = g_1(X) \times P(X)$ et si $f(X) = f_1(X) \times P(X)$ alors $s(X) = \dfrac{g_1(X)}{f_1(X)}$.

Définition :

Soit s une suite binaire produite par un LFSR de longueur L, son polynôme de rétroaction minimal est l'unique polynôme unitaire, f, de $\mathbb{F}_2[X]$ tel qu'il existe $g \in \mathbb{F}_2[X]$ avec $\deg(g) < \deg(f)$ et $PGCD(g, f) = 1$ tel que $s(X) = \dfrac{g(X)}{f(X)}$.
La complexité linéaire de s est égale au degré de f, on la note $\Lambda(s)$. C'est la longueur du plus petit LFSR qui permet de produire s.

Retour à l'exemple :

$g(X) = 1 + X$ et $f(X) = 1 + X + X^3$. On a $PGCD(g, f) = 1$ d'où f est le polynôme minimal de rétroaction de la suite s.

Dès que f est irréductible, c'est le polynôme minimal du LFSR. Si f est irréductible de degré L, on a $\Lambda(s) = L$.

Proposition :
Si le polynôme minimal du LFSR est primitif de degré L, alors la suite s produite est une m-séquence : elle est périodique de période $2^L - 1$. De plus, $\Lambda(s) = L$.

Démonstration :

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & 0 \\ & & & \ddots & 1 \\ c_L & c_{L-1} & \dots & c_2 & c_1 \end{pmatrix}$$

On peut montrer que le polynôme caractéristique de A ($\det(A - XI_L)$) est $\tilde{f}(x) = X^L + c_1 X^{L-1} + \ldots + c_{L-1}X + c_L$.
Si $f(X) = 1 + c_1 X + c_2 X^2 + \ldots + c_L X^L$ alors $\tilde{f}(X) = X^L f\left(\frac{1}{X}\right)$ (polynôme réciproque de f).
Les valeurs propres de A sont les racines (dans $\mathbb{F}_{2^L}$) de $\tilde{f}$, ce sont aussi les $\frac{1}{\alpha}$ où α est racine de f.
Comme f est primitif, il a L racines distinctes d'ordre $2^L - 1$. De même pour $\tilde{f}$, on note $\alpha_1, \alpha_2, \ldots, \alpha_L$ les L racines d'ordre $2^L - 1$ de $\tilde{f}$.
Dans $\mathbb{F}_{2^L}$, A est diagonalisable :

$$P^{-1}AP = D = \begin{pmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \alpha_L \end{pmatrix}$$

De plus, on a

$$A^k = I_L \Leftrightarrow D^k = P^{-1}A^kP = \begin{pmatrix} \alpha_1^k & 0 & \dots & 0 \\ 0 & \alpha_2^k & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \alpha_L^k \end{pmatrix} = I_L \Leftrightarrow (2^L - 1)|k$$

Si s est de période T, on a : $R_T = A^T R_0 = R_0 = 1 \times R_0$. Donc R_0 est un vecteur propre associé à la valeur propre 1 de A^T. Cela arrive si $(2^L - 1)|T$. Comme $T \leqslant 2^L - 1$, on a $T = 2^L - 1$ et donc s est une m-séquence.

Remarque :
La réciproque est vraie : si s est une m-séquence, son polynôme de rétroaction minimal est primitif.

Retour à l'exemple :
$f(X) = 1 + X + X^3$. f est primitif donc la suite s est une m-séquence de période $2^3 - 1 = 7$.

Propriétés statistiques des LFSR :
Une m-séquence produite par un LFSR va vérifier les 3 critères de Golomb.

Démonstration :
La premier critère est qu'à un près on a autant de zéros que de un dans une période. Dans une période chaque registre R_i $(s_i, s_{i+1}, \ldots, s_{i+L-1})$ apparait exactement une fois. En particulier, s_i prend 2^{L-1} fois la valeur 1 et $2^{L-1} - 1$ fois la valeur 0 dans une période. De même pour les autres critères.

En conclusion, une m-séquence produite par un LFSR est un bon candidat pour faire du chiffrement à flot rapide, avec une grande période et de bonnes propriétés statistiques.

Problème : Quelle donnée peut être secrète ?
Si les c_i : $c_1, \ldots, c_L$ sont connus, alors à partir de $(s_i, \ldots, s_{i+L-1})$, on peut produire les bits suivants : $s_{i+L}, s_{i+L-1} \ldots$ et on peut aussi revenir en arrière "en rembobinant" le LFSR. Donc pour une utilisation en chiffrement par flot, les c_i doivent être secrets.

Problème :
On peut retrouver les c_i à partir de $2L$ bits de suite chiffrante.

Proposition :
Si s est une suite avec $\Lambda(s) = L$, alors si on connait $2L$ bits conscutifs de suite, on peut calculer les c_i en inversant un système linéaire de taille $L \times L$.

Démonstration :

On a la relation :

$$\begin{matrix} R_i \\ R_{i+1} \\ \vdots \\ R_{i+L-1} \end{matrix} \begin{pmatrix} s_i & s_{i+1} & \dots & s_{i+L-1} \\ s_{i+1} & s_{i+2} & \dots & s_{i+L} \\ \vdots & & & \vdots \\ s_{i+L-1} & \dots & \dots & s_{i+2L-2} \end{pmatrix} \begin{pmatrix} c_L \\ c_{L-1} \\ \vdots \\ c_1 \end{pmatrix} = \begin{pmatrix} s_{i+L} \\ s_{i+L+1} \\ \vdots \\ S_{i+2L-1} \end{pmatrix}$$

Notons B la matrice dans cette relation.

On peut construire ce système si on connait $s_i, s_{i+1} \dots, s_{i+2L-1}$. Si B est inversible, on peut retrouver $(c_1, c_2, \dots, c_L)$. On peut montrer que si s est de complexité linéaire L alors B est inversible. On peut faire mieux :

Théorème :

Si $(s_n)_{n\geqslant 0}$ est une suite produite par un LFSR avec $\Lambda(s) = L$ alors l'algorithme de Berlekamp-Massey détermine l'unique LFSR de longueur $\Lambda(s)$ qui engendre $(s_n)_{n\geqslant 0}$ à partir de $2\Lambda(s)$ bits conscutifs de (s_n), de complexité quadratique en $\Lambda(s)$.

Définition :

Etant donnés une suite finie $s = (s_0, s_1, \dots, s_{n-1})$ et $f(X) = 1 + c_1X + \dots + c_LX^L$, on dit que (L, f) engendre s si :

$$s_j = \sum_{i=1}^{L} c_i s_{j-i}$$

pour $j = L, \dots, n-1$.
Le plus petit L convenable est appelé complexité linéaire et noté $\Lambda(s)$.
Si $s = (s_0, \dots, s_{n-1})$, on note $s^{(k)} = (s_0, \dots, s_{k-1})$ la suite tronquée pour $k \leqslant n$ de telle sorte que $s^{(n)} = s$. L'algorithme de Berlekamp-Massey permet de calculer une suite (L_k, f_k) pour $k = 1, \dots, n$ avec L_k minimal, $L_k = \Lambda(s^{(k)})$. La suite des $(\Lambda(s^{(k)}), f_k)$ est appelée le profil de complexité de s. Pour une suite aléatoire, on doit avoir $\Lambda(s^{(k)}) = \dfrac{k}{2}$. L'algorithme de Berlekamp-Massey utilise le théorème suivant :

Théorème :

Supposons que pour $k \geqslant 1$, on a calculé un couple (L, f) associé à $s^{(k)}$ avec $L = \Lambda(s^{(k)})$.

Soit $d_k = s_k + \sum_{i=1}^{L} c_i s_{k-i}$ avec c_i coefficients de f.

– Si $d_k = 0$ alors (L, f) engendre $s^{(k+1)}$ et $\Lambda(s^{(k+1)}) = L$.
– Si $d_k = 1$, on note m le plus grand entier tel que $m < k$ et $\Lambda(s^{(m)}) < L$ et soit g tel que $(\Lambda(s^{(m)}), g)$ engendre $s^{(m)}$. Alors $\Lambda(s^{(k+1)}) = \max(L, k+1-L)$ et $s^{(k+1)}$ engendrée par $f + x^{k-m}g$.

Algorithme 1 Algorithme de Berlekamp-Massey

Entrées: $s^{(n)} = (s_0, \ldots, s_{n-1})$
Sorties: $(\Lambda(s^{(k)}), f_k)$ qui engendre $s^{(k)}$ pour $k = 1, \ldots, n$.

$f \leftarrow 1$; $L \leftarrow 0$; $m \leftarrow -1$; $g \leftarrow 1$; $k \leftarrow 0$;
Tant que $k < n$ **Faire**
  $d_k \leftarrow s_k + \sum_{i=1}^{L} c_i s_{k-i}$;
  Si $d_k = 1$ **Alors**
    $h \leftarrow f$; $f \leftarrow f + gx^{k-m}$;
    Si $L \leqslant \frac{k}{2}$ **Alors**
      $L \leftarrow k + 1 - L$;
      $m \leftarrow k$;
      $g \leftarrow h$;
    Fin si
  Fin si
  $k \leftarrow k + 1$;
Fin tant que

Exemple : $s = (0, 1, 1, 0, 0, 1, 0, 1, 0, 1)$

k	s_k	d_k	L	f	m	g
0	0	0	0	1	-1	1
1	1	1	2	$1 + X^2$	1	1
2	1	1	2	$X^2 + X + 1$	1	1
3	0	0	2	$X^2 + X + 1$	1	1
4	0	1	3	$X^3 + X^2 + X + 1$	4	$X^2 + X + 1$
5	1	0	3	$X^3 + X^2 + X + 1$	4	$X^2 + X + 1$
6	0	1	4	$X^4 + X + 1$	6	$X^3 + X^2 + X + 1$
7	1	1	4	$X^3 + X^2 + 1$	6	$X^3 + X^2 + X + 1$
8	0	1	5	$X^5 + X^4 + 1$	8	$X^3 + X^2 + 1$
9	1	0	5	$X^5 + X^4 + 1$	8	$X^3 + X^2 + 1$

Donc $(5, X^5 + X^4 + 1)$ engendre s et $\Lambda(s) = 5$.
En pratique, si s est une suite infinie de complexité linéaire L, alors avec $2L$

termes consécutifs de s, l'algorithme de Berlekamp-Massey retourne L et le polynôme de rétroaction de degré L qui engendre s en $O(L^2)$ opérations.

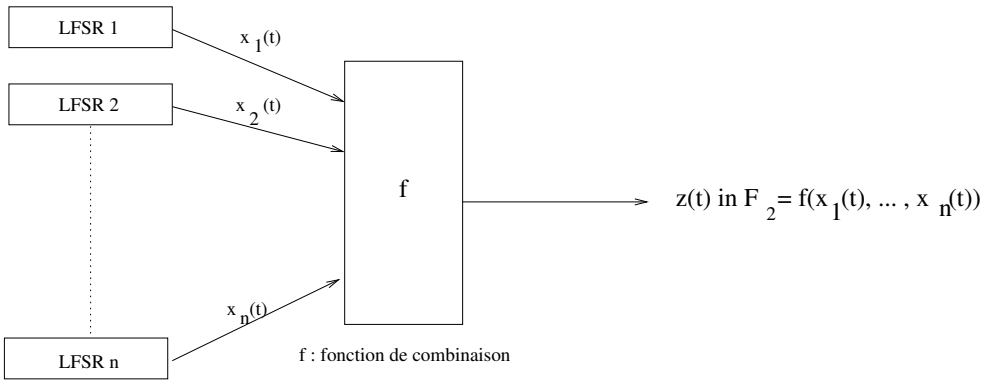
Conséquence :

Si on utilise un LFSR pour produire une suite chiffrante d'un chiffement à flot additif, il faut une complexité linéaire L très élevée pour éviter l'algorithme de Berlekamp-Massey. Si on veut que BM $\approx 2^{80}$ opérations, il faut $L \approx 2^{40}$ et un LFSR de longueur 2^{40} est complètement inutilisable. Une solution pour augmenter la complexité linéaire de manière pratique est les LFSR combinés.

Chapitre 5

Combinaisons de LFSR

Construction : n LFSR avec des rétroactions et des longueurs différentes.



Public : Les n LFSR, c'est-à-dire les n polynômes de rétroaction et la fonction f.

Privé : Les n initialisations des LFSR.

La fonction $f : (\mathbb{F}_2)^n \to \mathbb{F}_2$ est appelée une fonction booléenne à n variables.

5.1 Fonctions booléennes

Définition :
On appelle fonction booléenne vectorielle à n variables et m composantes

une application de $(\mathbb{F}_2)^n$ vers $(\mathbb{F}_2)^m$. C'est la juxtaposition de m fonctions booléennes : $(f_1(x_1,\ldots,x_n), f_2(x_1,\ldots,x_n),\ldots,f_m(x_1,\ldots,x_n))$. Elle est aussi utilisée pour le chiffrement par blocs : S-box.

Définition :
On appelle support de la fonction booléenne f l'ensemble des valeurs $u \in (\mathbb{F}_2)^n$ tel que $f(u) \neq 0$ noté $supp(f)$. Le poids de f est le cardinal de $supp(f)$ et est noté $wt(f)$. Si f, g sont deux fonctions booléennes à n variables, on a :

$$d(f,g) = wt(f+g)$$

$d(f,g)$ est le nombre de $u \in (\mathbb{F}_2)^n$ tel que $f(u) \neq g(u)$. On peut représenter f par la liste de ses images :

$$V_f = [f(0,\ldots,0), f(0,\ldots,0,1),\ldots,f(1,\ldots,1)]$$

Le poids de f est le poids de Hamming de V_f.
Il y a 2^{2^n} fonctions booléennes à n variables. Si $n=3$, 2^8 fonctions booléennes à 3 variables. Si $n=6$, 2^{64} fonctions booléennes à 6 variables. Il est donc impossible de trouver par recherche exhaustive une fonction booléenne ayant de "bonnes" propriétés.

Représentation booléenne Table de vérité de la fonction.
Par exemple :

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

On a $wt(f) = 4$ et $supp(f) = \{(0,0,1),(1,0,1),(1,1,0),(1,1,1)\}$.

f est équilibrée si $\#\{u \in (\mathbb{F}_2)^n, f(u)=0\} = \#\{\mathbb{F}_2)^n, f(u)=1\} = 2^{n-1}$. Cela est utile pour les combinaisons de LFSR, ça assure que la suite générée $(z_n)_{n\geqslant 0}$ a autant de 0 que de 1.

Forme algébrique normale d'une fonction booléenne

Retour à l'exemple :

$f(u) = 1 \Leftrightarrow u = (0,0,1)$ ou $u = (1,0,1)$ ou $u = (1,1,0)$ ou $u = (1,1,1)$.
$(0,0,1)$ correspond à $(1+x_1)(1+x_2)x_3 = 1$.
$(1,0,1)$ correspond à $x_1(1+x_2)x_3 = 1$.
$(1,1,0)$ correspond à $x_1x_2(1+x_3) = 1$.
$(1,1,1)$ correspond à $x_1x_2x_3 = 1$.

On a donc :

$$\begin{aligned} f(x_1,x_2,x_3) &= (1+x_1)(1+x_2)x_3 + x_1(1+x_2)x_3 + x_1x_2(1+x_3) + x_1x_2x_3 \\ &= (1+x_2)x_3[1+x_1+x_1] + x_1x_2(1+x_3+x_3) \\ &= x_1x_2 + x_2x_3 + x_3 \end{aligned}$$

Définition :

La forme algébrique normale d'une fonction booléenne f à n variables est l'unique polynome Q_f de $\dfrac{\mathbb{F}_2[x_1 \dots x_n]}{(x_1^2 - x_1, x_2^2 - x_2, \dots, x_n^2 - x_n)}$ tel que $f(u_1, \dots, u_n) = Q_f(u_1, \dots, u_n)$ $\forall u \in (\mathbb{F}_2)^n$. Dans $\mathbb{F}_2$, si $u \in \mathbb{F}_2$, $u^2 = u, u^3 = u^2, \dots, u^k = u$ $\forall k \geqslant 1$. Evaluer x^k, $k \geqslant 1$ sur $\mathbb{F}_2$ est équivalent à évaluer x. Tous les éléments de $\dfrac{\mathbb{F}_2[x_1 \dots x_n]}{(x_1^2 - x_1, x_2^2 - x_2, \dots, x_n^2 - x_n)}$ peuvent être représentés par des sommes de monomes en $x_1, \dots, x_n$ avec des degrés pour chaque x_i inférieur ou égal à 1.

Exemple : $x_1x_2 + x_2x_3 + x_3 = x_1^2x_2^5 + x_2x_3^4 + x_3^6$ dans $\dfrac{\mathbb{F}_2[x_1 \dots x_n]}{(x_1^2 - x_1, \dots, x_n^2 - x_n)}$.

Définition :

On appelle degré de f noté $\deg(f)$ le degré du polynome Q_f c'est-à-dire le nombre maximal de termes dans les monomes de Q_f.

Exemple : $x_1x_2 + x_2x_3 + x_3$ est de degré 2.

Une fonction de degré 1 est dite affine : $1 + x_1 + x_3 + x_5$. Si de plus, $f(0, \dots, 0) = 0$, on dit que f est linéaire $(x_1 + x_2 + \dots)$ c'est-à-dire sans termes constants. C'est une forme linéaire de $(\mathbb{F}_2)^n \to \mathbb{F}_2$.

Pour des LFSR combinés, on cherche à casser la linéarité des LFSR, on prend donc des fonctions de degré le plus grand possible pour complexifier

les relations entre les bits de suite chiffrante et les contenus des registres des LFSR. Si f est linéaire, comme la mise à jour du LFSR est linéaire, les bits de sortie dépendent de manière linéaire de l'initialisation des registres c'est-à-dire la clé secrète. Par conséquent, on peut écrire un système linéaire dont les inconnus sont les bits de clé secrète. Si f est de degré supérieur, le système devient complexe.

Notation condensée pour exprimer une fonction booléenne f

Si $u = (u_1, \ldots, u_n)$, on note :

$$x^u \text{ le monome } x_1^{u_1} x_2^{u_2} \ldots x_n^{u_n}$$

On note alors $\forall n \in (\mathbb{F}_2)^n, x = (x_1, \ldots, x_n)$ avec $a_u \in \mathbb{F}_2$:

$$f(x) = \sum_{u \in (\mathbb{F}_2)^n} a_u x^u$$

Exemple :

$$f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3$$

x_1x_2 donne x^u avec $u = (1, 1, 0)$.
x_2x_3 donne x^u avec $u = (0, 1, 1)$.
x_3 donne x^u avec $u = (0, 0, 1)$.
On a donc $f(x) = \sum a_u x^u$ avec $a_{(1,1,0)} = a_{(0,1,1)} = a_{(0,0,1)} = 1$ et les autres sont nuls.

Définition : Transformée de Mobius de $f \to f^\circ$
Si $u, v \in (\mathbb{F}_2)^n$, on dit que $v \leqslant u \Leftrightarrow v_i = 1 \Rightarrow u_i = 1 \forall i \in \{1, \ldots, n\}$.

Exemple :
$(0, 0, 1) \leqslant (1, 0, 1)$ mais $(0, 0, 1) \not\leqslant (1, 1, 0)$ On a :

$$f^\circ(u) = \sum_{\alpha \in (\mathbb{F}_2)^n, \alpha \leqslant u} f(\alpha) \text{ dans } \mathbb{F}_2$$

Propriété :

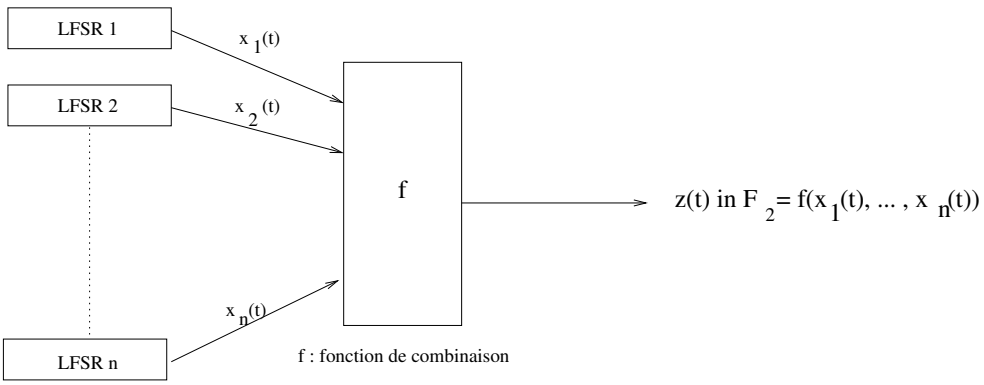
Si f est une fonction booléenne à n variables, $f(x) = \sum a_u x^u$ avec $a_u = f^\circ(u)$, $\forall u \in (\mathbb{F}_2)^n$.

Exemple :

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$f^\circ(0,0,0) = f(0,0,0) = 0$
$f^\circ(0,0,1) = f(0,0,0) + f(0,0,1) = 0 + 1 = 1$
$f^\circ(0,1,0) = f(0,0,0) + f(0,1,0) = 0 + 0 = 0$
$f^\circ(1,0,0) = 0$
$f^\circ(0,1,1) = f(0,0,0) + f(0,1,0) + f(0,0,1) + f(0,1,1) = 0 + 0 + 1 + 0$
$f^\circ(1,0,1) = 0$
$f^\circ(1,1,0) = 1$
$f^\circ(1,1,1) = 0$
Donc $f(x) = x^{(0,0,1)} + x^{(0,1,1)} + x^{(1,1,0)} = x_3 + x_2x_3 + x_1x_2$

5.2 Combinaisons de LFSR



On prend f équilibrée, cela permet d'avoir la même probabilité de sortie pour un 0 que pour un 1. On peut montrer que $(z(t))_{t\in\mathbb{N}}$ est une suite à récurrence linéaire. Il faut que sa complexité linéaire soit élevée. Il faut calculer la complexité linéaire de produit terme à terme de 2 suites à récurrence

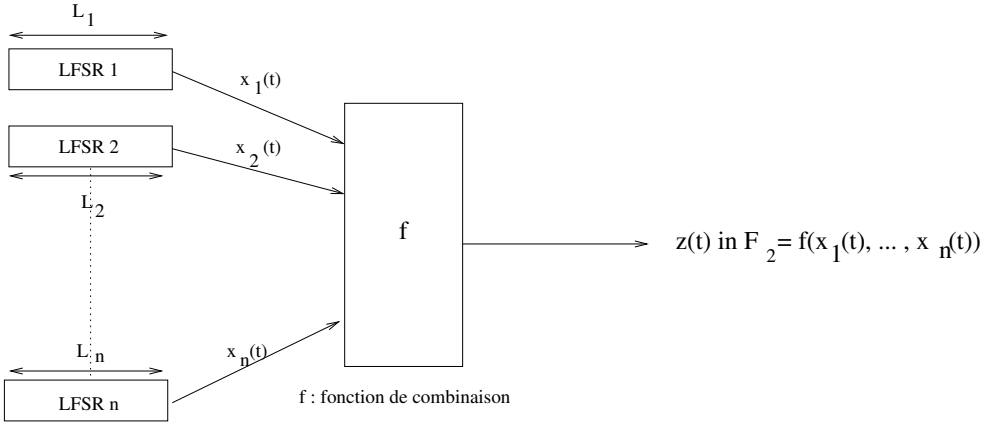
linéaire et de sommes de 2 suites à récurrence linéaire.

On a par exemple :
Si (u_n) et (v_n) sont deux suites à récurrence linéaire de polynomes de rétroaction minimaux f_u et f_v, la somme $(u_n + v_n)$ est une suite à récurrence linéaire de coplexité linéaire :

$$\Lambda(u+v) \leqslant \Lambda(u) + \Lambda(v)$$

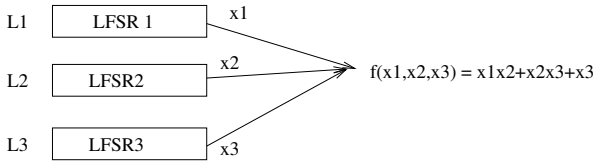
avec égalité si et seulement si $pgcd(f_u, f_v) = 1$.

Proposition :



Si $L_1, \ldots, L_n$ sont deux à deux distinctes et si $L_1 \geqslant 2$ et si les polynomes de rétroactions sont tous primitifs, alors la complexité linéaire de $(z(t))_{t\in\mathbb{N}}$ est $f(L_1, L_2, \ldots, L_n)$ évalué dans $\mathbb{Z}$.

Par exemple : le générateur de Geffe (1973)
On a $L_1 \neq L_2$ et $L_2 \neq L_3$ et $L_1 \neq L_3$ et les polynomes de rétroactions



primitifs. On a donc que la complexité linéaire est de $L_1L_2 + L_2L_3 + L_3$.
On a f équilibrée. La clé secrète est l'initialisation des LFSR.
Pour retrouver la clé secrète à partir de $(z(t))_{t\in\mathbb{N}}$, il faut résoudre un système à $L_1 + L_2 + L_3$ inconnues avec des équations non linéaires.

5.3 Attaque par corrélation sur une combinaison de LFSR

Attaque proposée par Siegentheler en 1985. C'est une attaque de type diviser pour régner, on retrouve l'initialisation de chacun des LFSR indépendamment.

Exemple sur le générateur de Geffe :
$z(t) = x_1(t)x_2(t) + x_2(t)x_3(t) + x_3(t) \Leftrightarrow z(t) = x_1(t)x_2(t) + x_3(t)(1 + x_2(t))$.

Corrélation entre $z(t)$ et $x_1(t)$

$$P[z(t) = x_1(t)] = P[x_2(t) = 1] + P[x_2(t) = 0]P[x_3(t) = x_1(t)] = \frac{1}{2} + \frac{1}{2} \times \frac{1}{2} = \frac{3}{4}$$

Corrélation entre $z(t)$ et $x_3(t)$

$$P[z(t) = x_3(t)] = \frac{3}{4}$$

Corrélation entre $z(t)$ et $x_2(t)$

$$P[z(t) = x_2(t)] = \frac{1}{2} \text{ (pas de corrélation)}$$

On suppose que l'on a récupéré une portion de suite chiffrante $(z(t))_t$. On va retrouver l'initialisation du premier LFSR. On teste les 2^{L_1} initialisations du LFSR 1.

- Pour chaque initialisation, on calcule $(x_1(t))_t$.
- On compare bit à bit $(x_1(t))_t$ et $(z(t))_t$.
- Si on a égalité avec une probabilité de $\frac{3}{4}$ c'est une bonne initialisation.
- Si on a égalité avec une probabilité de $\frac{1}{2}$ c'est une mauvaise initialisation.

On fait de même pour le LFSR 3 : complexité 2^{L_3}. Pour le LFSR 2, on fait une recherche exhaustive : complexité 2^{L_2}. L'attaque prend donc $2^{L_1} + 2^{L_2} + 2^{L_3}$ opérations. Et la recherche exhaustive prendrait $2^{L_1+L_2+L_3}$ opérations.

Pour résumer, si on a une combinaison de n LFSR, on calcule $p_i = P[z(t) = x_i(t)]$. Si $p_i \neq 0,5$, alors on fait une attaque par corrélation sur le registre. Cout de l'attaque :

$$\sum_{j, p_j \neq 0,5} 2^{L_j} + \prod_{j, p_j = 0,5} 2^{L_j}$$

Comment se protéger de ces attaques ?

Définition :
On dit que f est non-corrélée à l'ordre K si pour $X_1, X_2, \ldots, X_n$ des variables aléatoires binaires, la variable aléatoire $f(X_1, X_2, \ldots, X_n)$ est indépendante des $\sum_{i \in I} X_i$ où I est un sous-ensemble de $\{1, \ldots, n\}$ de cardinal au plus K (indépendante $= P[f(X_1, \ldots, X_n) = 0 | \sum X_i = a] = P[f(X_1, \ldots, X_n) = 0]$ avec $a = 0$ ou 1).

On dit que f est K résiliente si f est non-corrélée à l'ordre K et équilibrée. On peut montrer que f est K-résiliente $\Rightarrow \deg(f) \leqslant n-K-1$ avec $K < n-1$. Pour choisir f, on fait le compromis d'augmenter K pour éviter les attaques par corrélation et avoir un $\deg(f)$ suffisant pour avoir une complexité linéaire élevée et éviter les attaques avec Berlekamp-Massey.

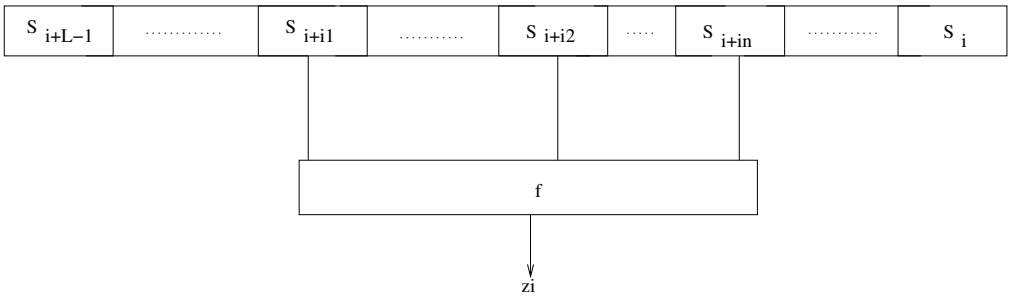
5.4 LFSR filtré

C'est une autre manière de conserver les bonnes propriétés statistiques des LFSR en évitant les attaques du type Berlekamp-Massey. On a un LFSR de longueur L, de rétroaction publique initialisé par $K_{L-1}, K_{L-2}, \ldots, K_1, K_0$ clé secrète qui servira de premier registre du LFSR. On a aussi $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ avec $1 \leqslant n \leqslant L$, une fonction booléenne de filtrage de l'état. On choisit des indices :

$$L - 1 \geqslant i_1 \geqslant i_2 \geqslant \ldots \geqslant i_n \geqslant 0$$

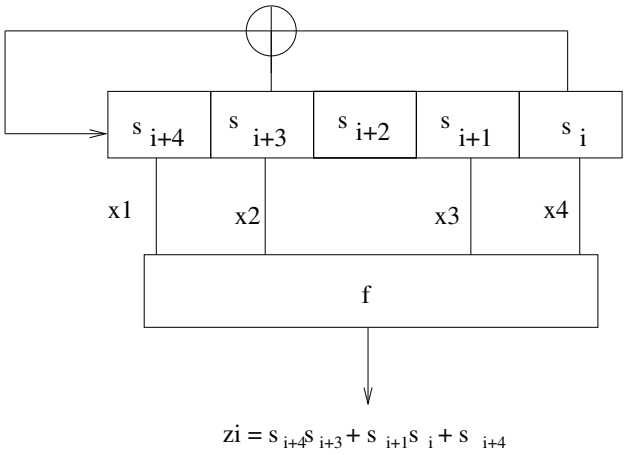
Au temps i, le bit de suite chiffrante z_i est égal à :

$$z_i = f(s_{i+i_1}, s_{i+i_2}, \ldots, s_{i+i_n})$$

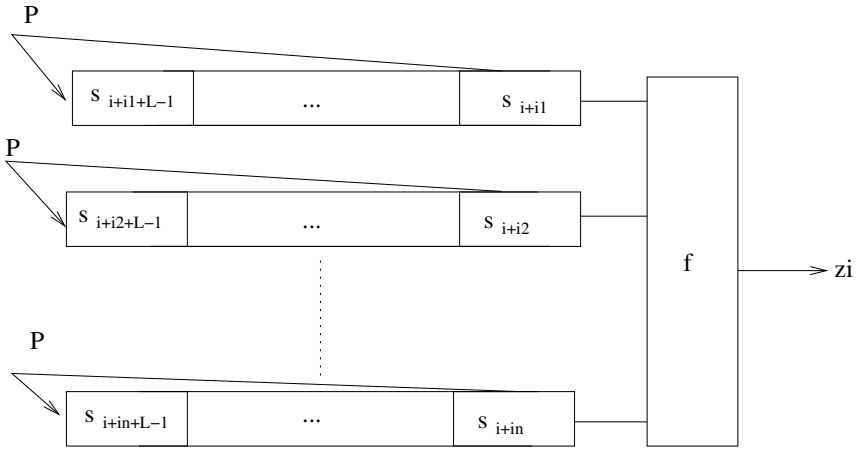


Exemple :

$f(x_1, x_2, x_3, x_4) = x_1x_2 + x_3x_4 + x_1$



Equivalence avec le schéma à combinaison suivant :



Les attaques sur un LFSR filtré ne fonctionnent pas de la même manière que sur un LFSR combiné car on a le même LFSR avec la même longueur et la même rétroaction.

Complexité linéaire d'un LFSR filtré :

Si (z_i) est une suite à récurrence linéaire alors :

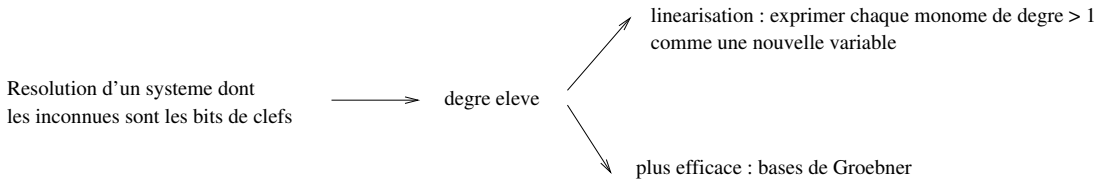
$$\Lambda(z) \leqslant \sum_{K=1}^{d} \begin{pmatrix} L \\ K \end{pmatrix}$$

où L est la longueur du LFSR et d le degré de f.
La période divise $2^L - 1$ en supposant que P est primitif. Si L est premier, alors $\Lambda(z) \approx \left(\begin{array}{c} L \\ d \end{array} \right) = \frac{L!}{d!(L-d)!}$, pour la plupart des f de degré d.

Attaques possibles

- Attaques par corrélations adaptées
- Inversion Attack : elle exploite les zones entre $i_1, i_2, \ldots$. En particulier, $i_1 - i_n$ doit être le plus grand possible proche de $L-1$ et $PGCD_{k,j,k\neq j}(i_k - i_j + 1)$ doit être les plus petits possibles pour éviter les attaques par inversion.
- Attaques algébriques dans le cas d'un LFSR filtré. Le principe général est qu'on fait une attaque à clairs/chiffrés connus des bits de z_i. On a $c_i = m_i + z_i$. Les z_i s'expriment comme équation en les bits de clés.

Principe énoncé par Shannon en 1949 :
Ecrire les équations de chiffrement : exprimer les bits de chiffré en fonction du message en clair et de la clef.



Exemple sur le LFSR filtré

Soit un LFSR de longueur L et de polynome de rétroaction $1 + c_1 X + \ldots + c_L X_L$.

$$A = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & 0 \\ & & & 0 & 1 \\ c_L & c_{L-1} & \ldots & c_2 & c_1 \end{pmatrix}$$

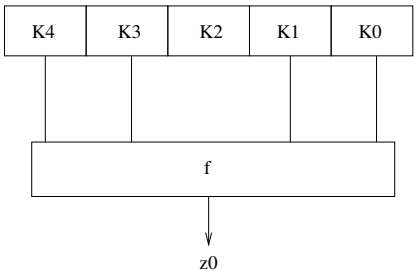
Si R_0 est le registre initial, on a $R_0 = (s_0, s_1, \ldots, s_{L-1}) = (K_0, K_1, \ldots, K_{L-1})$.
Or on a $R_i = A^i R_0$. On a donc un système linéaire qui donne $R_i = (s_i, s_{i+1}, \ldots, s_{i+L-1})$ en fonction de $R_0 = (K_0, K_1, \ldots, K_{L-1})$. z_i est obtenu en appliquant f de degré d à n éléments de R_i. Or les n éléments de R_i sont des équations linéaires en $K_0, K_1, \ldots, K_{L-1}$. D'où z_i s'exprime sur des équations de degré au plus d en $K_0, K_1, \ldots, K_{L-1}$.

Linéarisation du système obtenu

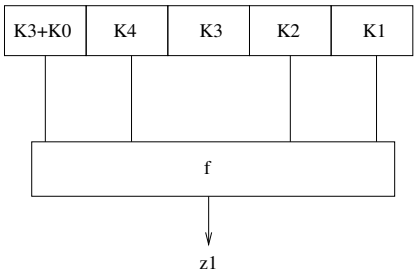
Chaque monome $K_{j_1} \times K_{j_2} \times \ldots \times K_{j_r}$ avec $r \leqslant d$ est exprimé comme une nouvelle variable. On en a $\begin{pmatrix} L \\ r \end{pmatrix}$. On obtient après linéarisation un nombre de variables $\leqslant \sum_{r=1}^{d} \begin{pmatrix} L \\ r \end{pmatrix}$. Cette somme est la complexité linéaire de (z_i).

Exemple :

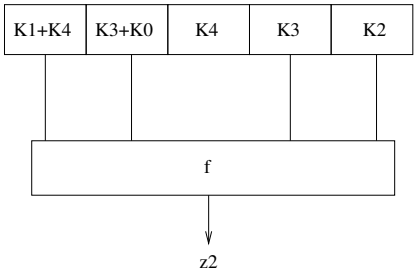
$f(x_1, x_2, x_3, x_4) = x_1x_2 + x_3x_4 + x_1$



$z_0 = K_4K_3 + K_1K_0 + K_4$. Cette équation est de degré 2 en les $(K_0, \ldots, K_4)$.



$z_1 = (K_3 + K_0)K_4 + K_2K_1 + K_3 + K_0 = K_3K_4 + K_0K_4 + K_2K_1 + K_3 + K_0$. Cette équation est de degré 2.



$z_2 = (K_1 + K_4)(K_3 + K_0) + K_3K_2 + K_1 + K_4 = K_1K_3 + K_1K_0 + K_4K_3 + K_4K_0 + K_3K_2 + K_1 + K_4$. Cette équation est de degré 2.
Si on récupère Nz_i, on aura N équations de degré 2 en $K_0, \ldots, K_4$.
Linéarisation : variables suivantes :

$$Y_0 = K_0;\ Y_1 = K_1;\ Y_2 = K_2;\ Y_3 = K_3;\ Y_4 = K_4$$

$$Y_5 = K_0K_1;\ Y_6 = K_0K_2;\ Y_7 = K_0K_3;\ Y_8 = K_0K_4;\ Y_9 = K_1K_2$$

$$Y_{10} = K_1K_3;\ Y_{11} = K_1K_4;\ Y_{12} = K_2K_3;\ Y_{13} = K_2K_4;\ Y_{14} = K_3K_4$$

On a $\left(\begin{array}{c} 5 \\ 1 \end{array}\right) + \left(\begin{array}{c} 5 \\ 2 \end{array}\right) = 5 + 10 = 15$ variables. Si on obtient 15 équations avec 15 inconnues et si elles sont linéairement indépendantes, on peut résoudre le système. La complexité est $\leqslant \left(\sum_{i=1}^{2} \left(\begin{array}{c} 5 \\ i \end{array}\right)\right)^3$ opérations . Cette complexité peut vite devenir grosse :
Si $d = 7$ et $L = 256$ on a :

$$\left(\sum_{i=1}^{7} \left(\begin{array}{c} 256 \\ i \end{array}\right)\right)^3 \approx \left(\begin{array}{c} 256 \\ 7 \end{array}\right)^3 \approx 2^{120}$$

Pour faire baisser le degré des équations : si f est de degré 7 et qu'il existe g de degré 3 (par exemple) tel que $f(x)g(x) = 0$ alors si $z_i = 1$ et $f() = 1$, on a nécessairement pour avoir $f()g() = 0$, $g = 0$ (équation de degré 3).

Attaque algébrique rapide de Courtois/Meyer (2003)

S'il existe $h \neq 0$, $g \neq 0$ de degré $< d$ avec $fg = 0$ ou $(1 + f)h = 0$ et si $z_i = f(s_{i+i_1}, \ldots, s_{i+i_n})$ (équation de degré $\deg(f)$) alors :

- Si $z_i = 1$, on a $g(s_{i+i_1}, \ldots, s_{i+i_n}) = 0$ (équation de degré $\deg(g)$).
- Si $z_i = 0$, on a $h(s_{i+i_1}, \ldots, s_{i+i_n}) = 0$ (équation de degré $\deg(h)$).

Définition :

Si f est une fonction booléenne, on appelle annulateur de f :

$$AN(f) = \{g | fg = 0\}$$

Et on appelle immunité algébrique de f :

$$AI(f) = \min_{g \neq 0} \{\deg(g),\ g \in AN(f) \cup AN(1 + f)\}$$

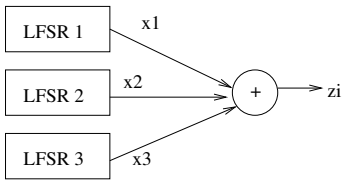
On a $AI(f) \leqslant \deg(f)$ car $f(f + 1) = f^2 + f = f + f = 0 \Rightarrow f \in AN(f + 1)$.
On peut montrer que $AI(f) \leqslant \left\lfloor \frac{n+1}{2} \right\rfloor$ avec n nombre de variables de f.

On a donc un critère de choix pour une fonction booléenne f utilisée dans un LFSR filtré qui est d'avoir $AI(f)$ le plus grand possible.

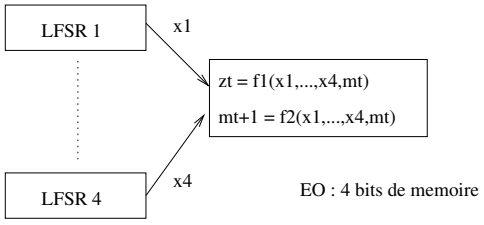
Autres attaques classiques sur les constructions à base de LFSR :
- Attaques par resynchronisation sur l'initialisation des registres. L'attaquant choisit les IV utilisés et exploite la suite chiffrante produite avec l'initialisation.
- Attaques Guess and Determine : On devine une partie de l'état interne et on essaye de trouver la clef à partir de ce qu'on a deviné.

Constructions classiques :

- A5/1 (GSM) : générateur à controle d'horloge



  A chaque tour, on calcule $Maj(x_1, x_2, x_3) = b$ avec Maj qui est la majorité (exemple : $Maj(0, 0, 1) = 0$) et on "clocke" les LFSR i tels que $x_i = b$. Le controle d'horloge permet de casser la linéarité des LFSR.
  Attaque sur A5/1 : compromis temps/mémoire qui a besoin de 2 minutes de messages clairs et qui est une attaque rapide mais nécessite beaucoup de précalculs (300GB de données). Il y a aussi l'attaque sur l'initialisation (2003 par Ekdahl et Johansson) qui nécessite 5 minutes de messages clairs mais qui est une attaque en quelques minutes sans précalculs.
- LFSR combiné avec mémoire : EO (Bluetooth)



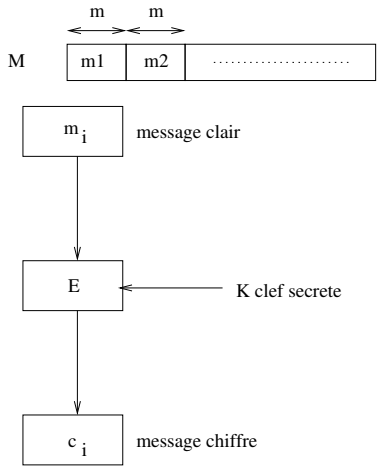
  Attaque par corrélation spécialisée sur EO qui nécessite 2^{38} opérations, 2^{26} bits de suite chiffrante.

La fiche intitulée "Snow 2.0 et Grain" donne d'autres constructions.

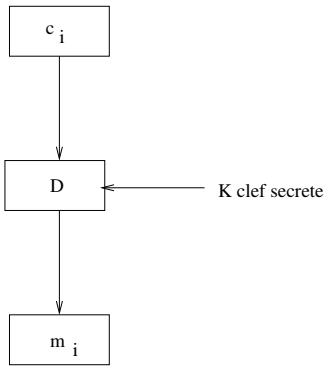
Chapitre 6

Chiffrement par blocs

Définition :
L'algorithme de chiffrement par bloc prend en entrée m bits de message dans $(\mathbb{F}_2)^m$ et donne en sortie un bloc de bits (en général m).

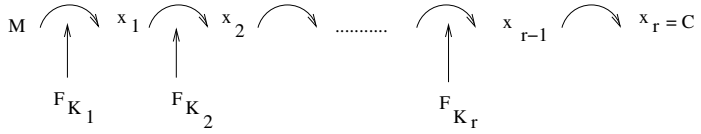


E est une permutation de $(\mathbb{F}_2)^n$ sélectionnée par la clef K parmi les $(2^n)\,!$ permutations possibles. Le déchiffrement est similaire avec $D_K = (E_K)^{-1}$.



Construction

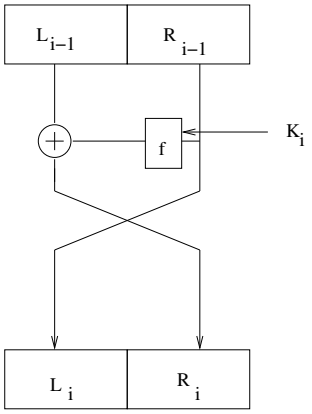
C'est un schéma itératif, avec F la fonction de tour et K_i la clef de tour déduite de K par un algorithme de cadencement de clef.



On a deux types de construction :
- le schéma de Feistel
- le schéma Substitution/Permutation (ou SPN en anglais avec N=network) utilisé pour AES

6.1 Schéma de Feistel

Il a été introduit par Feistel dans les années 70. Le message clair est coupé en deux blocs de meme longueur $\frac{m}{2}$ notés L_0 et R_0.
Au tour i :

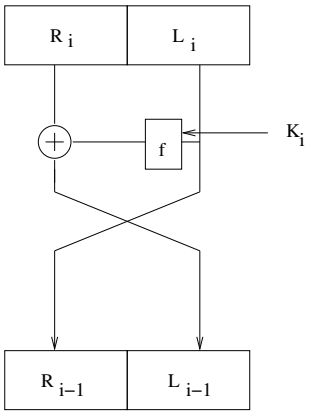


On a :

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f_{K_i}(R_{i-1})$$

Au bout de r tours, on a le chiffré $C = R_r|L_r$. Le déchiffrement de $C = R_r|L_r$ se fait avec le même procédé en utilisant les clefs de tour dans l'autre sens : $K_r, K_{r-1}, \ldots, K_1$.
A la ronde i :



On a

$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus f_{K_i}(L_i)$$

et à la dernière étape on obtient $L_0|R_0$.
On obtient un schéma inversible si on connait les clefs de tour sans hypothèse sur la fonction f. f n'est pas supposée être bijective.

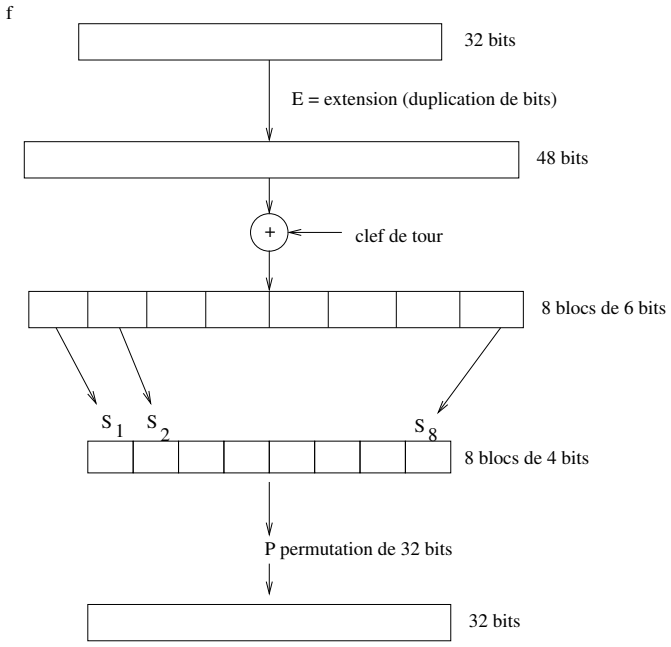
Sécurité théorique des schémas de Feistel

Si f_{K_i} est une fonction pseudo-aléatoire (c'est-à-dire dont les sorties sont indistinguables d'une fonction aléatoire) alors 3 tours de schémas de Feistel donne une permutation pseudo-aléatoire.
Le schéma de Feistel est très utilisé :

- comme composant pour construire des schémas par blocs
- en cryptologie asymétrique : OAEP : Optimal Asymetric Encryption Padding, utilisé dans RSA-OAEP qui utilise un schéma de Feistel à 2 tours. Cela consiste à rajouter de l'aléa et un padding au message et à appliquer des fonctions de hachage.

Exemple de schéma de Feistel

DES : Data Encryption Standard, standard de 1977 à 2000 qui prend en entrée un bloc de 64 bits, utilise des clefs de 56 bits et des clefs de tour de 48 bits. C'est un schéma de Feistel à 16 tours avec une permutation initiale et une permutation finale avec $f : (\mathbb{F}_2)^{32} \to (\mathbb{F}_2)^{32}$.



La fonction E est une fonction linéaire de $(\mathbb{F}_2)^{32} \to (\mathbb{F}_2)^{48}$. La fonction P est aussi une fonction linéaire de $(\mathbb{F}_2)^{32} \to (\mathbb{F}_2)^{32}$. Ces deux fonctions, E et P apportent de la diffusion c'est-à-dire que si l'on change un bit en entrée, il y aura plusieurs bits changés en sortie. La fonction S est une boite S ou S-box et apporte de la confusion c'est-à-dire qu'elle rend les relations en ckef et chiffré complexes. Les boites S ont des critères de design inconnus (dus à la NSA). Elles sont faites pour résister à la cryptanalyse différentielle, redécouverte par Biham et Shamir en 1988 (et qui nécessite 2^{47} messages clairs choisis). Ce sont des fonctions booléennes vectorielles : $(\mathbb{F}_2)^6 \to (\mathbb{F}_2)^4$. Elles sont décrites par la table des 2^6 sorties possibles et choisies à grande distance de Hamming des focntions affines (de degré 1). Ce qui les rend très résistantes à la cryptanalyse linéaire (qui nécessite 2^{43} clairs connus, découverte par Matsui en 1993). Mais ces attaques sont moins réalistes que la recherche exhaustive.

Concept de diffusion-confusion (Shannon 1949) :

- La diffusion a pour but d'empecher les attaques statistiques. Par exemple, si un bit du message en clair est souvent égal à 1 cela ne doit pas se voir sur la distribution des messages chiffrés.
- La confusion permet de rendre la relation entre la clé et le chiffré complexe. Il est toujours difficile de retrouver la clef meme en connaissant un grand nombre de couple clair-chiffré. Chaque bit de chiffré dépend

de plusieurs bits de clé et de manière complexe. On veut qu'un bit de clé changé provoque la modification de tout le chiffré.

Aujourd'hui DES est obsolète car 56 bits de clé ne nous met plus à l'abri de la recherche exhaustive. Il y a une alternative : le 3-DES créé en 1998 et toujours utilisé surtout dans les banques. On a :

$$C = E_{K_3}\left[D_{K_2}\left(E_{K_1}(M)\right)\right]$$

On a trois options pour l'utiliser :

1. $K_1 \neq K_2 \neq K_3$, on a donc une clef de $3 \times 56 = 168$ bits.
2. $K_3 = K_1$ et $K_2 \neq$, on a donc une clef de $2 \times 56 = 112$ bits.
3. $K_1 = K_2 = K_3$, on a donc une seule clef de 56 bits, et c'est équivalent à un DES classique.

Attaque Meet in the Middle

Supposons $C = E_{K_2}(E_{K_1}(M))$ avec $K_1 \neq K_2$ et que (M, C) est connu. On stocke les couples $(M, E_{K_1}(M))$ pour toutes les clefs K_1 possibles. Pour toutes les clefs K_2 possibles, on calcule $D_{K_2}(C)$ et pour le bon (K_1, K_2) on aura $D_{K_2}(C) = E_{K_1}(M)$. Donc si on a égalité, on a un candidat pour (K_1, K_2). La complexité est $2^n + 2^n = 2^{n+1}$ opérations et $C \times 2^n$ bits de mémoire avec n la taille des clefs. En conclusion, on a une sécurité de "n bits" comme la recherche exhaustive sur l'algorithme de chiffrement initial. On n'a donc pas de gain à faire un double chiffrement.

Conséquences sur le Triple DES

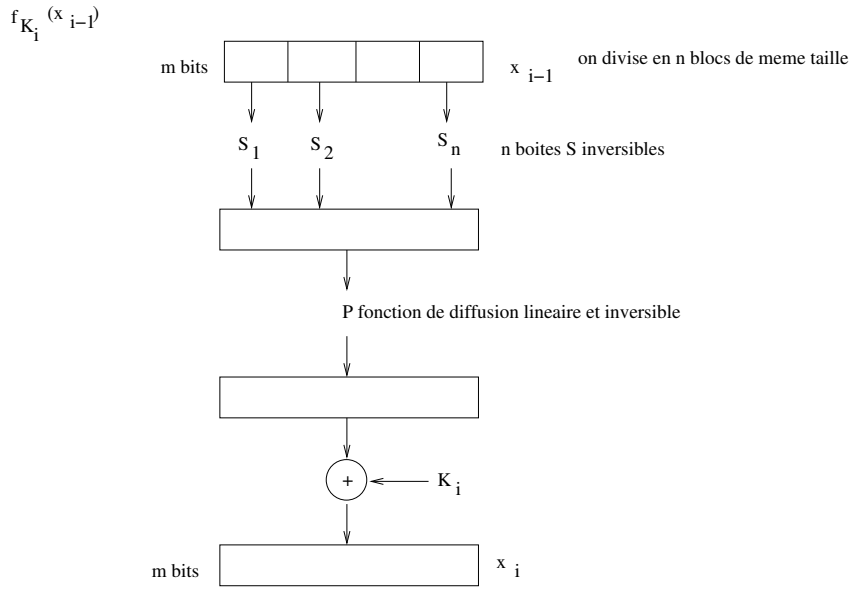
1. L'attaque Meet in the Middle est possible en précaculant les $D_{K_2}(E_{K_1}(M))$ et en calculant les $D_{K_3}(C)$. On a une attaque en 2^{2^n} avec $n = 56$ ce qui nous donne 2^{112}. On a une sécurité de 112 bits.
2. Il n'y a pas d'attaques Meet in the Middle possible mais d'autres attaques sont possibles et on estime la sécurité à environ 80 bits.
3. C'est un DES classique, il a donc 56 bits de sécurité.

6.2 Schéma Substitution/Permutation (SPN)

C'est une autre construction de schémas itératifs avec des fonctions de tour inversibles. Il alterne les opérations de diffusion et de confusion.

Construction

A l'étape initiale, on fait $x_0 = M + K_0$ avec M le message en clair et K_0 la clef de tour.
Puis on calcule $x_i = f_{K_i}(x_{i-1})$ et $C = x_r$ au bout de r tours.



Le déchiffrement se fait en faisant toutes les étapes dans le sens inverse en appliquant P^{-1} et $(S_i)^{-1}$.

Exemple

L'algorithme de Rijndael issu d'un concours de 1997 à 2000, concu par Daemen et Rijmen et choisit pour AES (Advanced Encryption Standard). Il prend en entrée un bloc de 128 bits et utilise des clefs de 128, 192 ou 256 bits et fait selon la longueur de la clé 10, 12 ou 14 tours. Voici le détail de la fonction de tour :
La fonction P est la composition de 2 fonctions appelées ShiftRows et MixColumns. Le bloc de 128 bits est disposé en matrice 4×4 d'octets. Chaque case de la mémoire est un octet identifié avec un élément du corps $\mathbb{F}_{2^8}$ le corps à 256 éléments au moyen d'un polynome irréductible fixé. L'état est une matrice de $\mathcal{M}_4(\mathbb{F}_{2^8})$.
La fonction ShiftRows effectue un décalage sur les lignes de la matrice. La fonction MixColumns multiplie l'état par une matrice fixe de $\mathcal{M}_4(\mathbb{F}_{2^8})$ inversible. Pour la fonction S, on utilise une seule boite S appliquée sur chaque octet de l'état : $S : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$. C'est la composition de 2 opérations :

1.

$$\mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$$
$$x \longmapsto x^{-1} \text{ si } x \neq 0$$
$$0 \longmapsto 0$$

Ce qui revient à faire :

$$x \longmapsto x^{254}$$

2. une fonction affine : on identifie le résultat de la première opération avec un élément de $\mathbb{F}_2^8$ et on calcule $Ay + b$ où $A \in \mathcal{M}_8(\mathbb{F}_2)$ et $b \in (\mathbb{F}_2)^8$ fixés. Elle sert à éviter les attaques algébriques.

Chapitre 7

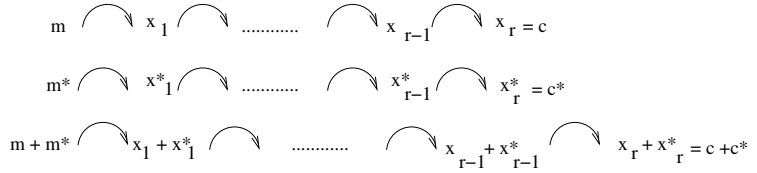
Cryptanalyse des chiffrements par blocs

Il y a bien évidemment la recherche exhaustive sur la clé.

7.1 Cryptanalyse différentielle

Elle a été redécouverte par Biham et Shamir en 1990, mais elle était connue lors de la conception du DES dans les années 70. C'est une attaque à clairs choisis sur les schémas par blocs itératifs.

Principe général :
On considère 2 messages clairs m et m^* et on appelle différence $m + m^*$. On s'intéresse à l'évolution des différences dans le schéma itératif.



On s'intéresse à $x_{r-1} + x^*_{r-1}$.
On note $P_{A,B} = P(x_{r-1} + x^*_{r-1} = B | m + m^* = A)$. On suppose avoir trouvé A et B tel que $P_{A,B}$ soit élevé.
L'attaque consiste à récupérer un grand nombre de clairs-chiffrés (m, c) et (m^*, c^*) tels que $m + m^* = A$. Pour chaque $[(m, c), (m^*, c^*)]$, pour chaque valeur possible K de K_r, on calcule $F_K^{-1}(c) = z$ et $F_K^{-1}(c^*) = z^*$. Si $z+z^* = B$ alors on incrémente un compteur associé à K. Quand $K = K_r$, on a $z+z^* = B$ avec une probabilité de $P_{A,B}$, donc le compteur pour K_r est incrémenté à $P_{A,B} \times N$ où N est le nombre de couples $(m, c), (m^*, c^*)$. Quand $K \neq K_r$,

on a $z + z^* = B$ avec une probabilité uniforme sur les B possibles donc si la largeur des blocs est n, on a une probabilité de $\frac{1}{2^n}$. Donc le compteur des mauvaises clefs est incrémenté $\frac{1}{2^n} \times N$ fois. Si $P_{A,B} >> \frac{1}{2^n}$, on trouve K_r en regardant le plus grand compteur.

2 problèmes :

1. Comment trouver $P_{A,B}$ élevé ?
2. Comment faire une recherche exhaustive sur K_r ? (on peut se restreindre à rechercher quelques bits de K_r qui permettent de vérifier si $z + z^* = B$ ou non).

7.1.1 Calcul de $P_{A,B}$ sur un schéma SPN

On va regarder comment évolue une différence sur chaque étape d'un SPN.

- Sur les étapes linéaires :

$$x \to f \to y$$

$$x^* \to f \to y^*$$

  avec f linéaire ; on a $f(x + x^*) = f(x) + f(x^*) = y + y^*$. Donc si $x + x^* = \alpha$, on a $y + y^* = f(\alpha)$ avec probabilité 1.
- Ajout de clefs de tours :

$$x \to \oplus K \to y = x \oplus K$$

$$x^* \to \oplus K \to y^* = x^* \oplus K$$

  Si $x + x^* = \alpha$ alors $y + y^* = x + K + x^* + K = x + x^* = \alpha$ avec une probabilité de 1.
  Remarque : la différence ne dépend pas des clefs de tours.
- Sur les boites S :
  On suppose $S : (\mathbb{F}_2)^s \to (\mathbb{F}_2)^s$. Soit D une matrice de $2^s \times 2^s$. On a :

$$D[\alpha, \beta] = Card\{(x, x^*) \in (\mathbb{F}_2)^s \times (\mathbb{F}_2)^s | \; x + x^* = \alpha \text{ et } S(x) + S(x^*) = \beta\}$$

  avec α et β, blocs de s bits identifiés aux entiers de 0 à $2^s - 1$.
  On a :

$$D = \begin{bmatrix} 2^s & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & & \\ 0 & & & \end{bmatrix}$$

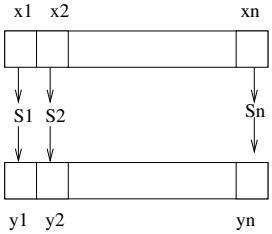
Sur $D[0,.]$, on a $\alpha = 0$ donc $x + x^* = 0 \Rightarrow x = x^*$ donc $S(x) = S(x^*) \Rightarrow S(x) + S(x^*) = 0 \Rightarrow \beta = 0$.
$D[0,0] = \#\{(x, x^*) | x = x^* \text{ et } S(x) = S(x^*)\} = 2^s$
$\forall i \neq 0,\ D[0,i] = 0 = \#\{x = x^* \text{ et } S(x) \neq S(x^*)\}$
$\forall i \neq 0,\ D[i,0] = 0$ car S est une bijection.
Tous les coefficients sont pairs. En effet, si (x, x^*) est tel que $x + x^* = \alpha$ et $S(x) + S(x^*) = \beta$ alors (x^*, x) est tel que $x^* + x = \alpha$ et $S(x^*) + S(x) = \beta$.
On a $P_{\alpha,\beta} = P[S(x) + S(x^*) = \beta | x + x^* = \alpha] = \dfrac{D[\alpha, \beta]}{2^s}$

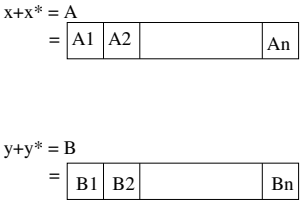
$$x \rightarrow S \rightarrow y$$

$$x^* \rightarrow S \rightarrow y^*$$

$$x + x^* = \alpha \rightarrow y + y^* = \beta \text{ avec probabilité } P_{\alpha,\beta}$$

On en déduit la probabilité pour plusieurs boites.

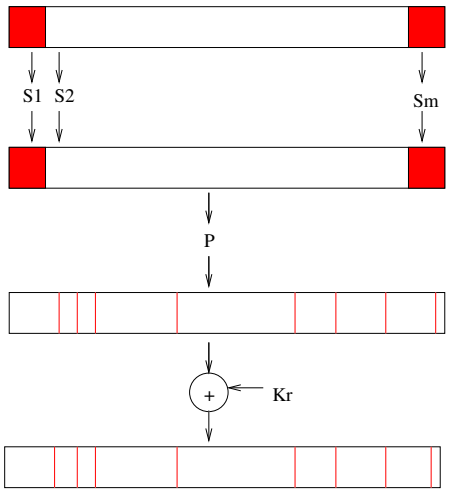




$$P_{\alpha,\beta} = P[y + y^* = \beta | x + x^* = \alpha] = \prod_i P[y_i + y_i^* = \beta_i | x_i + x_i^* = \alpha_i]$$

avec $P[y_i + y_i^* = \beta_i | x_i + x_i^* = \alpha_i]$ obtenue avec la matrice D_i.
On trouve $P[x_{r-1} + x_{r-1}^* = B | m + m^* = A]$.

7.1.2 Comment faire une recherche exhaustive sur K_r



$x_{r-1} + x^*_{r-1} = B$. Par exemple, $B = [1\text{er bloc}|0|\ldots\ldots\ldots|0|\text{dernier bloc}]$

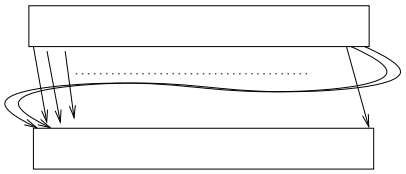
On dit que S_1 et S_n sont les boites actives. On note $I \subset \{1 \ldots n\}$ les images par P des indices de sortie des boites actives (dans l'exemple : $P(\{1, 2, 3, 4, 29, 30, 31, 32\})$, 8 boites S 4 bits $\rightarrow$ 4 bits).

On a $c + c^*$ qui sont différents sur les bits d'indices dans I et $c_j = c^*_j$ si $j \in \{1, \ldots, n\}\backslash I$. Pour vérifier si $x_{r-1} + x^*_{r-1} = B$, il faut que c et c^* soient égaux sur $\{1, \ldots, n\}\backslash I$ et il suffit de chercher les bits de K_r d'indices I.
Au final, on a :

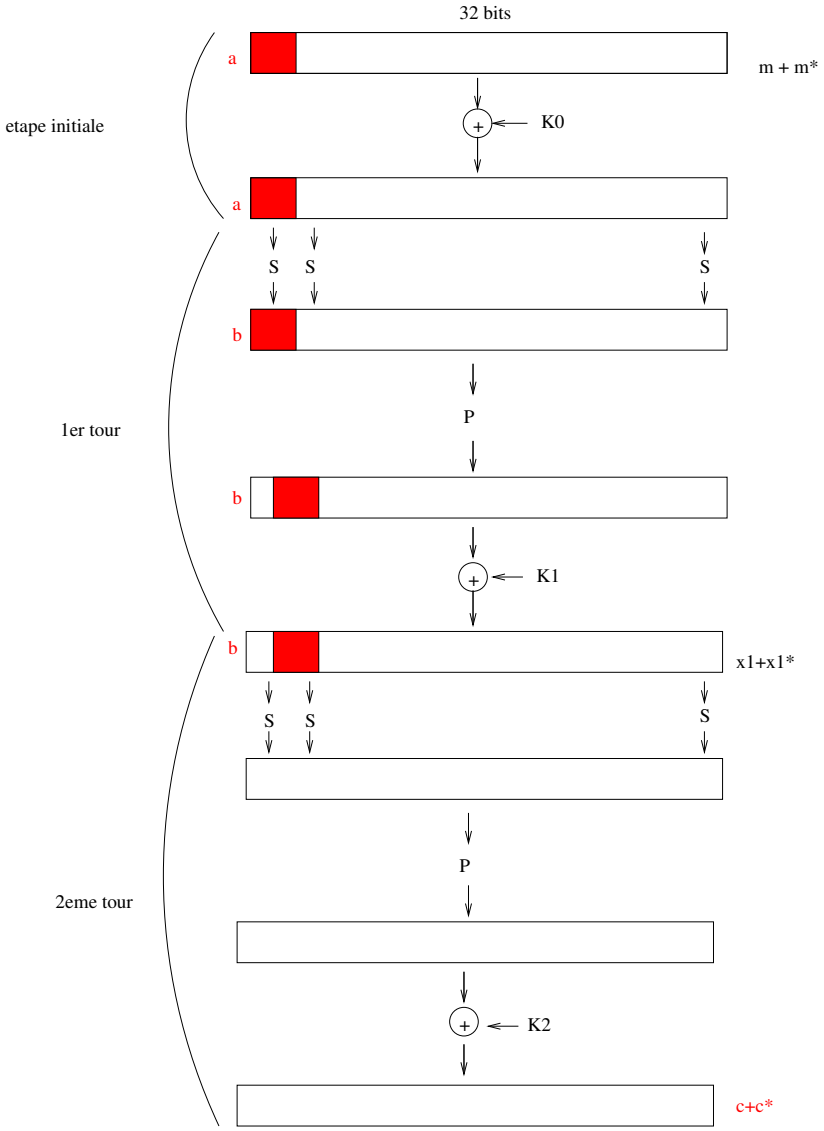
- On itère sur l'ensemble des (c, c^*) avec $m + m^* = A$. Si $c = c^*$ sur les bits d'indices $\{1, \ldots, n\}\backslash I$:
- pour chaque clef K de $\#I$ bits, on calcule $z = \tilde{F}^{-1}(c, k)$ et $z^* = \tilde{F}^{-1}(c^*, K)$ où $\tilde{F}^{-1}$ est la fonction partielle de la fonction de dernier tour sur les bits d'indices I.
- Si $z + z^* = \beta$ sur les indices de boites actives, on incrémente un compteur pour K.
- La clef la plus incrémentée donne les bits d'indices I de K_r. On réitère sur d'autres boites actives pour trouver d'autres bits de K_r. On finit éventuellement par une recherche exhaustive sur la clef secrète du SPN.

Exemple sur B32 B32 est un SPN avec des blocs de 32 bits. Il est fait pour illustrer la cryptanalyse différentielle/linéaire.

La substitution c'est 8 fois la même boite S : 4 bits $\rightarrow$ 4 bits.
La permutation effectue un décalage de 2 bits vers la droite.



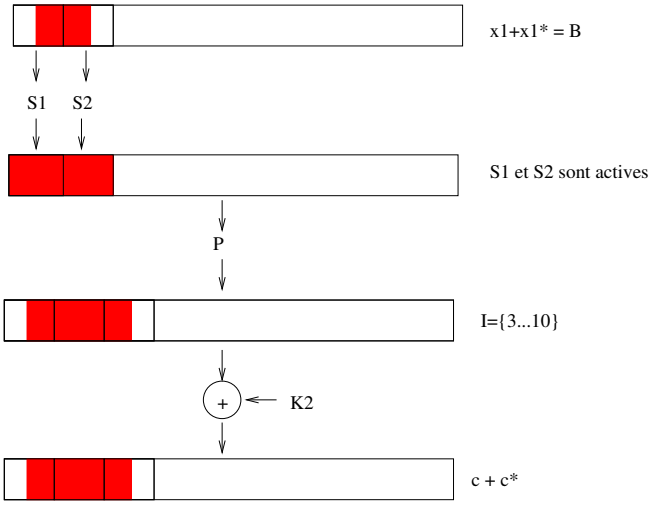
Cela permet une diffusion très faible. Ci-dessous le schéma d'une version à 2 tours avec 3 clefs K_0, K_1, K_2 de 32 bits.



On construit la table des différentielles $D_{\alpha,\beta}$ pour la boite S. On a par exemple : $P_{\alpha,\beta} = P[S(x) + S(x^*) = \beta | x + x^* = \alpha]$ avec $\alpha = [1111]$ et $\beta = [1001]$. $P_{\alpha,\beta} = \dfrac{10}{16}$ très élevée.

On pose $A = [11110\ldots0]$ et on prend (m, m^*) avec $m + m^* = A$. On pose $B = [0010010\ldots0]$, on a alors :

$$P(x_1 + x_1^* = B | m + m^* = A) = \frac{10}{16}$$



Algorithme : Etant donnés des couples (m, c), (m^*, c^*) avec $m + m^* = A$. Pour chaque (c, c^*), si $c = c^*$ sur les indices $\{1, 2, 11, \ldots, 32\}$, alors pour chaque clé K de 8 bits, on pose $\tilde{c} = c[3...10]$ et $\tilde{c^*} = c^*[3...10]$.

$$c \rightsquigarrow \tilde{c} \rightsquigarrow \tilde{c} \oplus K \rightarrow S^{-1}(\text{1er bloc})S^{-1}(\text{2eme bloc}) = z$$

$$c^* \rightsquigarrow \tilde{c^*} \rightsquigarrow \tilde{c^*} \oplus K \rightarrow S^{-1}(\text{1er bloc})S^{-1}(\text{2eme bloc}) = z^*$$

On teste si $z + z^* = [00B00]$ et si oui on incrémente un compteur pour K. Pour la bonne sous-clef, on incrémente $\frac{10}{16} \times Nb$ de couples. Pour une mauvaise clef on incrémente $\frac{1}{16} \times Nb$ de couples. On retrouve 8 bits de K_2, on itère avec d'autres boites actives.

7.2 Cryptanalyse linéaire

Elle a été découverte en 1992 par Matsui. C'est une attaque à clair connus sur les schémas de chiffrement par blocs itératifs.

Principe général

Trouver des équations linéaires qui relient des bits de message clair, de message chiffré et de clefs, vraies avec probabilité différentes de $\frac{1}{2}$.

Remarque :
On a des équations non linéaires (de degré très élevé) reliant clairs/chiffrés et clefs vraies avec probabilité 1. On essaye de trouver des équations linéaires parfois vérifiées.

Si $x = (x_1, \ldots, x_n) \in (\mathbb{F}_2)^n$, une équation linéaire en les bits de x est de la forme $a_1x_1+a_2x_2+\ldots+a_nx_n = b$ avec $b \in \{0,1\}$ et $a_i \in \{0,1\}\ \forall i \in \{1,\ldots,n\}$. On pose $a = (a_1, \ldots, a_n)$ et on note $a.x = a_1x_1 + a_2x_2 + \ldots + a_nx_n$. Les équations linéaires sur les bits de x sont de la forme $a.x = b$. Si x est aléatoire, pour a fixé, l'équation $a.x = b$ est vérifiée avec probabilité $\frac{1}{2}$: on considère la forme linéaire :

$$f : (\mathbb{F}_2)^n \to \mathbb{F}_2$$

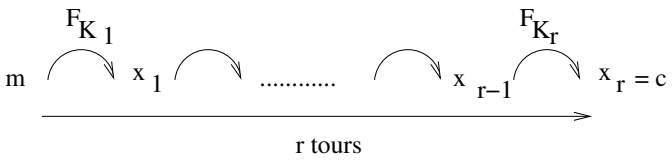
$$x \mapsto a.x$$

On a : $a.(x + y) = a.x + a.y$.
$\dim(\mathbb{F}_2)^n = \dim Ker(f) + \dim Im(f) = n$. On suppose f non nulle, $a \neq 0$, donc $Im(f) = \{0,1\} = \mathbb{F}_2$ et $\dim Im(f) = 1$ et donc $\dim Ker(f) = n - 1$. C'est-à-dire qu'il y a $2^{n-1}x$ tel que $a.x = 0$. L'équation $a.x = 0$ a 2^{n-1} solutions, elle est donc vraie avec probabilité $\frac{2^{n-1}}{2^n} = \frac{1}{2}$. L'équation $a.x = 1$ a 2^{n-1} solutions, elle est donc vraie avec probabilité $\frac{2^{n-1}}{2^n} = \frac{1}{2}$.

Pour la cryptanalyse linéaire, si $E_K(m) = c$, on cherche des équations de la forme $\alpha m + \beta c + \gamma K = 0$ avec probabilité $\frac{1}{2} + \epsilon$ avec $\epsilon > 0$. On peut faire une attaque sur le chiffrement global si on a $\alpha m + \beta c + \gamma K = 0$ avec probabilité $\frac{1}{2} + \epsilon,\ \epsilon > 0$.

C'est une attaque à clair connu, pour chaque (m, c), on calcule $\alpha m+\beta c \in \{0,1\}$. Si on trouve plus de 0 que de 1 alors $\gamma K = 0$ sinon $\gamma K = 1$. On en déduit une équation linéaire sur les bits de K. On réitère pour avoir assez d'équations linéaires pour retrouver K.

Cryptanalyse linéaire sur le dernier tour d'un SPN
On note :



On note $P_{\alpha,\beta}$ la probabilité $P[\alpha.m + \beta.x_{r-1} = 0]$. On suppose avoir trouvé (α, β) tels que $P_{\alpha,\beta} = \frac{1}{2} \pm \epsilon$ avec $\epsilon > 0$. On fait une attaque à clairs connus, pour chaque couple (m, c), pour chaque valeur possible K de K_r, on calcule $z = F_{K_r}^{-1}(c)$. Si $\alpha.m+\beta.z = 0$ alors on incrémente un compteur pour K. Si N est le nombre de couples (m, c) utilisés, pour la bonne clef K_r, on a $z = x_{r-1}$ et l'équation est vraie avec une probabilité de $\frac{1}{2} \pm \epsilon$. Donc le compteur de K_r est incrémenté de $\frac{N}{2} \pm \epsilon N$. Pour $K \neq K_r$, $z \neq x_{r-1}$ et l'équation $\alpha.m+\beta.z = 0$ est vraie avec une probabilité de $\frac{1}{2}$. Donc le compteur est incrémenté d'environ $\frac{N}{2}$. Le compteur le plus éloigné de $\frac{N}{2}$ donnera K_r.

Remarque : Si K et K' sont telles que $F_{K'}^{-1}(c) = F_K^{-1}(c)$, alors les compteurs de K et K' sont incrémentés de la meme manière. On restreint donc la recherche exhaustive sur les bits de la clef de dernier tour qui permettent de calculer les bits de z qui interviennent dans l'équation $\alpha.m + \beta.z = 0$ (les bits z_i de z qui correspondent aux i tels que $\beta_i = 1$.

Comment calculer les $P_{\alpha,\beta}$?

Regardons la propagation d'une équation linéaire dans un SPN :

– Effet d'une permutation :

On suppose que l'on a $\alpha.m + \beta.z = 0$ avec une probabilité de $\frac{1}{2} + \epsilon$, $\epsilon > 0$.

$$m \rightarrow \ldots \rightarrow y \rightarrow P \rightarrow z = P(y)$$

avec P permutation des bits de y, c'est à dire de $\{1, \ldots, n\}$.

$$\forall i \in \{1, \ldots, n\},\ y_i = z_{P(i)}$$

Si on note $j = P(i)$, on a $P^{-1}(j) = i$. Donc $y_i = z_{P(i)},\ \forall i \in \{1, \ldots, n\}$ devient $y_{P^{-1}(j)} = z_j,\ \forall j \in \{1, \ldots, n\}$. On a :

$$\beta.y = \sum_{i=1}^{n} \beta_i y_i = \sum_{i=1}^{n} \beta_i z_{P(i)} = \sum_{j=1}^{n} \beta_{P^{-1}(j)} z_j = P(\beta).z$$

Donc si on a une équation $\alpha.m+\beta.y = 0$ avec une probabilité de $\frac{1}{2}+\epsilon$, on a aussi $\alpha.m + P(\beta).z = 0$ avec une probabilité de $\frac{1}{2} + \epsilon$.

– Effet de l'ajout d'une clé :

$$m \to \ldots \to y \to \oplus K \to z = y + K$$

On suppose $\alpha.m + \beta.y = 0$ avec probabilité $\frac{1}{2} + \epsilon$.

$$\alpha.m + \beta.(z + K) = 0 \Leftrightarrow \alpha.m + \beta.z + \beta.K = 0$$

avec la meme probabilité.
Si α, β, K sont fixés, $\beta.K$ est fixé et ne dépend pas du message en clair m. Donc soit $\beta.K = 0$ soit $\beta.K = 1$. Donc soit $\alpha.m + \beta.z = 0$ avec une probabilité de $\frac{1}{2} + \epsilon$, soit $\alpha.m + \beta.z = 1$ avec une probabilité de $\frac{1}{2} + \epsilon$ et donc $\alpha.m + \beta.z = 0$ est vraie avec une probabilité de $\frac{1}{2} - \epsilon$. D'où dans les deux cas, on a $\alpha.m + \beta.z = 0$ avec une probabilité de $\frac{1}{2} \pm \epsilon$.

– Effet des S-box :

$$x \text{ de } s \text{ bits } \to S \to S(x) \text{ de } s \text{ bits}$$

On crée une table L de taille $2^s \times 2^s$ telle que :

$$L[\alpha, \beta] = Card\{x \in (\mathbb{F}_2)^s,\ \alpha.x + \beta.S(x) = 0\}$$

On a alors :

$$P[\alpha.x + \beta.S(x) = 0] = \frac{L[\alpha, \beta]}{2^s}$$

On prend un couple (α, β) tel que $\frac{L[\alpha, \beta]}{2^s}$ soit éloigné de $\frac{1}{2}$. Grâce à lui, on en déduit une probabilité d'avoir une équation linéaire du type $A.m + B.x_{r-1} = 0$. On utilise le lemme suivant :

Lemme d'empilement : Soit $Z_1, Z_2, \ldots, Z_r$ des variables aléatoires indépendantes booléennes avec $P[Z_i = 0] = \frac{1}{2} \pm \epsilon\ \forall i \in \{1, \ldots, r\}$ alors $P[Z_1 + Z_2 + \ldots + Z_r = 0] = \frac{1}{2} \pm 2^{r-1} \prod_{i=1}^{r} \epsilon_i$.

Par exemple, si $\alpha.m + \beta.x_1 = 0$ avec probabilité $\frac{1}{2} + \epsilon_1$ et si $\beta.x_1 + \gamma.x_2 = 0$ avec probabilité $\frac{1}{2} + \epsilon_2$ et si on note (1) le premier tour et (2) le deuxième tour, on a :

(1)+(2) : $\alpha.m + \beta.x_1 + \beta.x_1 + \gamma.x_2 = \alpha.m + \gamma.x_2$ avec proba $P[(1)+(2) = 0]$

On a $(1) + (2) = 0$ si :

– $(1) = (2) = 0$ avec probabilité $\left(\frac{1}{2} + \epsilon_1\right)\left(\frac{1}{2} + \epsilon_2\right)$

– $(1) = (2) = 1$ avec probabilité $\left(1 - \frac{1}{2} - \epsilon_1\right)\left(1 - \frac{1}{2} - \epsilon_2\right)$

On suppose que (1) et (2) sont indépendantes et on a :

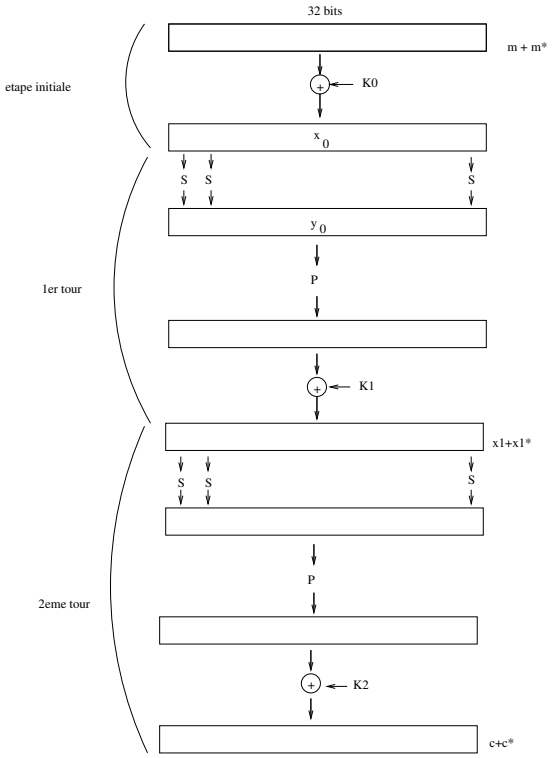
$$P[(1) = 0 \text{ et } (2) = 0] = P[(1) = 0]P[(2) = 0]$$

Donc :

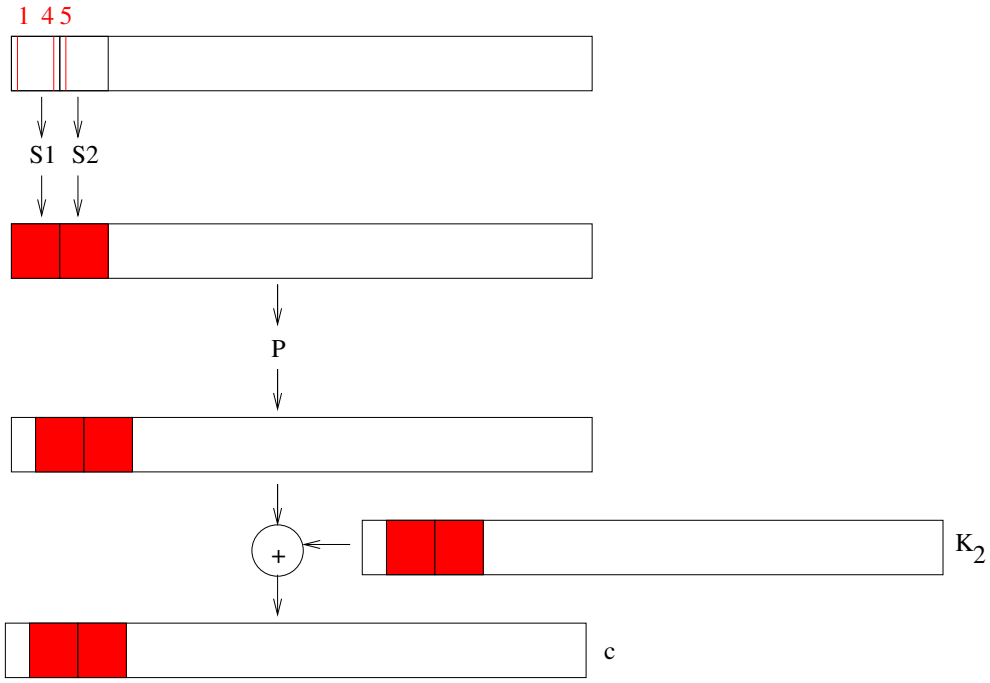
$$\begin{aligned} P[(1) + (2) = 0] &= \left(\frac{1}{2} + \epsilon_1\right)\left(\frac{1}{2} + \epsilon_2\right) + \left(\frac{1}{2} - \epsilon_1\right)\left(\frac{1}{2} - \epsilon_2\right) \\ &= \frac{1}{4} + \frac{\epsilon_1\epsilon_2}{2} + \epsilon_1\epsilon_2 + \frac{1}{4} - \frac{\epsilon_1\epsilon_2}{2} + \epsilon_1\epsilon_2 \\ &= \frac{1}{2} + 2\epsilon_1\epsilon_2 \end{aligned}$$

La démonstration du lemme se fait par récurrence de manière similaire.

Exemple récapitulatif sur B32 à 2 tours :



On veut une équation linéaire $\alpha.m+\beta.x_1 = 0$ avec probabilité $\frac{1}{2} \pm \epsilon$. On part d'une équation sur une boite S. De la table L de S, on déduit une équation de type $a.x + b.S(x) = 0$ avec probabilité $\frac{1}{2} \pm \epsilon$ avec ϵ grand. On a donc une équation qui relie les bits de x_0 et y_0, et on l'étend en une équation qui relie m et x_1 (à travers la permutation et les ajouts de clés K_0 et K_1). On trouve donc une équation $\alpha.m + \beta.x_1 = 0$ vraie avec probabilité $\frac{1}{2} \pm \epsilon$. On attaque le dernier tour et on suppose par exemple que $\beta = (1, 0, 0, 1, 1, 0, \ldots, 0)$.



Si $x_1 = \left(x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(32)}\right)$ alors $\beta.x_1 = x_1^{(1)} + x_1^{(4)} + x_1^{(5)}$. Ici on fait une recherche exhaustive sur 8 bits de K_2 qui permettent de vérifier si $\alpha.m + \beta.x_1 = 0$.

Chapitre 8

Réseaux euclidiens et cryptanalyse

8.1 Réseaux euclidiens

Définition :

Un réseau euclidien est un arrangement régulier de points de $\mathbb{R}^n$. $\mathbb{R}^n$ est un espace vectoriel de dimension n sur $\mathbb{R}$. On considère $\|.\|$, la norme euclidienne sur $\mathbb{R}^n$.
$\forall x, y \in \mathbb{R}^n,\ x = (x_1, x_2, \ldots, x_n),\ y = (y_1, y_2, \ldots, y_n)$, on a le produit scalaire :

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$$

et la norme : $\|x\| = \sqrt{\langle x, x \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2}$
Plus préciséement, un réseau euclidien est un sous-groupe discret, L, de $(\mathbb{R}^n, +)$.

- sous groupe : $0 \in L$, $-x \in L$, $\forall x, y \in L,\ x + y \in L$.
- discret : pas de point d'accumulation c'est-à-dire si $x \in L$, il existe une boule de centre x qui ne contient aucun autre point de L

Exemple : $n = 1$, $\mathbb{Z} \subset \mathbb{R}$, $\mathbb{Z}$ est un réseau de $\mathbb{R}$.
$\forall d \leqslant n,\ \mathbb{Z}^d \subset \mathbb{R}^n$ est un réseau de $\mathbb{R}^n$.

Propriétés :

Si $b_1, \ldots, b_d$ sont des vecteurs linéairement indépendants de $\mathbb{R}^n$, et si on

considère $L = \left\{ v \in \mathbb{R}^n,\ v = \sum_{i=1}^{d} \lambda_i b_i \text{ avec } \forall i,\ \lambda_i \in \mathbb{Z} \right\}$ alors L est un réseau de $\mathbb{R}^n$.
Tous les réseaux peuvent s'écrire sous cette forme ou alors avec $b_1, \ldots, b_d \in \mathbb{Q}^n$ et non nécessairement linéairement indépendants.
Si $(b_1, \ldots, b_d)$ sont linéairement indépendants, on dit que c'est une base de L (non unique mais il en existe au moins une).
$\forall v \in L$, il existe un unique $(\lambda_1, \ldots, \lambda_d) \in \mathbb{Z}^d$ tel que :

$$v = \sum_{i=1}^{d} \lambda_i b_i$$

Si $d = n$, on dit que L est de rang plein. On représente un réseau par sa base dans une matrice $d \times n$ exprimant en ligne les coordonnées des b_i dans la base canonique de $\mathbb{R}^n$.

$$\mathcal{M} = \begin{pmatrix} \leftarrow & b_1 & \rightarrow \\ \leftarrow & b_2 & \rightarrow \\ & \vdots & \\ \leftarrow & b_d & \rightarrow \end{pmatrix}$$

En cryptologie, on prend des b_i à coordonnées entières : $\mathcal{M} \in \mathcal{M}_{d \times n}(\mathbb{Z})$.

Exemple : $\mathbb{Z}^2 \subset \mathbb{R}^2$
Si $b_1 = (1, 0)$ et $b_2 = (0, 1)$ alors (b_1, b_2) est une base de $\mathbb{Z}^2$. $\mathcal{M} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ représente $\mathbb{Z}^2$.

Si B et B' sont 2 bases d'un même réseau L, alors il existe une matrice $P \in GL_d(\mathbb{Z})$ (inversible et de déterminant ± 1) tel que $B' = PB$. Toutes les bases ont le même nombre de vecteurs, c'est la dimension du réseau.

Exemple : $b'_1 = (1, 1)$ et $b'_2 = (0, 1)$ est une base de $\mathbb{Z}^2$

$$B' = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \times \mathcal{M}$$

et $\det(P) = \det \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = 1$ donc P est bien une matrice de permutation.
Mais $b''_1 = (2, 0)$ et $b''_2 = (0, 1)$ n'en est pas une.

$$B'' = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \times \mathcal{M}$$

mais $\det(P) = \det \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} = 2$ et P n'est pas une matrice de permutation entre B'' et $\mathcal{M}$.

Définition : Matrice de Gram d'un réseau L
Si B est une base de L, $B = (b_1, \ldots, b_d)$ alors :

$$G = B \times B^T = \begin{pmatrix} \leftarrow & b_1 & \rightarrow \\ \leftarrow & b_2 & \rightarrow \\ & \vdots & \\ \leftarrow & b_d & \rightarrow \end{pmatrix} \times \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ b_1 & b_2 & \ldots & b_d \\ \downarrow & \downarrow & & \downarrow \end{pmatrix} = (< b_i, b_j >)_{1 \leqslant i,j \leqslant d}$$

Le déterminant de Gram $\det G > 0$. On appelle déterminant du réseau :

$$\det(L) = \sqrt{\det G}$$

Remarque : le déterminant ne dépend pas de la base choisie. Si $B' = PB$ est une autre base alors $G = B'B'^T = PBB^TP^T$.

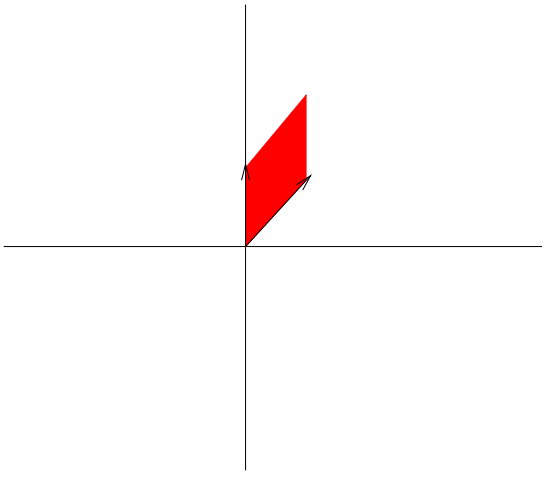
$$\det G = (\det P)^2 \times \det(BB^T) = 1 \times \det G$$

Si $L \subset \mathbb{R}^n$ est de rang plein ($d = n$) alors B est une matrice carrée $n \times n$ donc $\det G = (\det B)^2$ et $\det L = |\det B|$.

$\det L$ est le volume du parallélotope formé par les vecteurs de base,

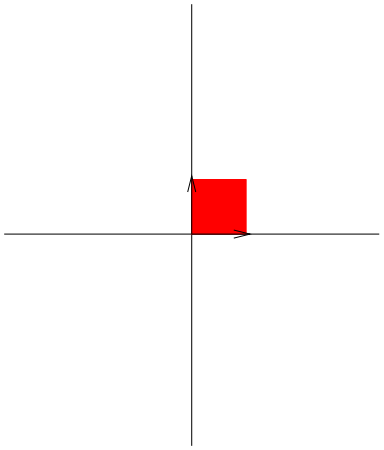
$$\left\{ \sum_{i=1}^{d} x_i b_i, x_i \in [0,1] \right\}$$

Sur l'exemple, $L = \mathbb{Z}^2$ et $B' = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$.



$$\{x_1 b_1 + x_2 b_2, x_1, x_2 \in [0,1]\}$$

On a $\det L = |\det B'| = 1 = |\det B|$ avec $B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$



Définition : minimum d'un réseau L
On note $\lambda_1(L) = \min L = \min_{x \in L \setminus \{0\}} \|x\|$. En général trouver un vecteur atteignant le minimum d'un réseau est un problème difficile. On le note le problème SVP (Shortest Vector Problem), conjecturé NP-dur.

Exemple : $L = \mathbb{Z}^2 \subset \mathbb{R}^2$.
Les vecteurs courts sont $(1,0)$, $(0,1)$, $(-1,0)$, $(0,-1)$ de norme 1.

$$\min L = 1$$

Dans $\mathbb{Z}^n \subset \mathbb{R}^n$, il y a $2n$ vecteurs courts $(0,0,\ldots,0,1,0,\ldots,0)$ de norme 1.

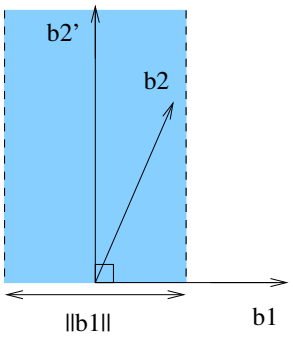
$$\min \mathbb{Z}^n = 1$$

Pour trouver le minimum d'un réseau L, on essaye de construire une bonne base de L, dont les vecteurs vont approcher des vecteurs courts du réseau.

Cas de la dimension 2 : Lagrange-Gauss

On aimerait avoir une base orthogonale proche d'une base orthogonale avec des vecteurs courts.

Définition : On dit que (b_1, b_2) est réduite au sens de la réduction de Lagrange-Gauss si $\|b_1\| \leqslant \|b_2\|$ et si (b_1, b_2') est une base orthogonale directe de $\mathbb{R}^2$ alors : $b_2 \in \{\lambda_1 b_1 + \lambda_2 b_2,\ \lambda_1, \lambda_2 \in \mathbb{R}\}$ avec $|\lambda_1| \leqslant \dfrac{1}{2}$ et $\lambda > 0$.



Propriétés :

On peut montrer que si (b_1, b_2) est réduite alors b_1 est un vecteur court du réseau : $\min L = \|b_1\|$.

Algorithme 2 Algorithme de Lagrange-Gauss

Entrées: (b_1, b_2) base de L telle que $\|b_1\| \leqslant \|b_2\|$.
Sorties: une base réduite de L

Répéter

$b_2 \leftarrow b_2 - \left\lfloor \dfrac{< b_1, b_2 >}{\|b_1\|^2} \right\rceil b_1$

Si $\|b_1\| > \|b_2\|$ **Alors**

échanger b_1 et b_2

Sinon

Retourner (b_1, b_2) ou

Retourner $(b_1, -b_2)$

Fin si

Jusqu'à ce que la base soit réduite

Remarque :

Cet algorithme est proche de l'algorithme d'Euclide. Euclide prend en entrée un réseau $a\mathbb{Z} + b\mathbb{Z}$ de $\mathbb{R}$ et il trouve $g = PGCD(a, b)$: il réduit $a\mathbb{Z} + b\mathbb{Z}$ en $g\mathbb{Z}$.

On peut montrer que l'algorithme de Lagrange-Gauss se termine et a une complexité quadratique et il sort une base réduite au sens de Lagrange-Gauss et le premier vecteur, b_1, est un vecteur court du réseau : $\min L$. En dimension 2, on résout SVP en complexité polynomiale. En dimension supérieure, on n'a plus d'algorithme polynomial pour résoudre SVP.

Cas de la dimension supérieure

On va utiliser l'orthogonalisation de Gram-Schmidt.

Soit $(b_1, \ldots, b_d)$ linéairement indépendants dans $\mathbb{R}^n$.
On pose $b'_1 = b_1$ et $b'_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b'_j$ avec $\mu_{i,j} = \dfrac{< b_i, b'_j >}{\|b'_j\|^2}$ pour $2 \leqslant i \leqslant d$ et $\mu_{i,i} = 1$.
$(b'_1, \ldots, b'_d)$ est l'orthogonalisé de Gram-Schmidt de $(b_1, \ldots, b_d)$.

Attention : si $(b_1, \ldots, b_d)$ est la base d'un réseau L, $(b'_1, \ldots, b'_d)$ n'est plus une base de L ($\mu_{i,j} \notin \mathbb{Z}$).

Propriété :

$$< b'_i, b'_j >= 0, \ \forall i \neq j$$

$vect(b_1, \ldots, b_i) = vect(b'_1, \ldots, b'_i)$ est le $\mathbb{R}$-espace vectoriel engendré par $b_1, \ldots, b_i$ et $\forall i$, $b'_i - b_i \in vect(b_1, \ldots, b_{i-1})$.
Ces trois proriétés caractérisent l'orthogonalisé de Gram-Schmidt.

Interprétation matricielle

$$b_1 = b'_1$$
$$b_2 = b'_2 + \mu_{2,1} b'_1$$
$$\vdots$$
$$b_d = b'_d + \mu_{d,1} b'_1 + \ldots + \mu_{d,d-1} b'_{d-1}$$
$$\begin{pmatrix} 1 & & 0 \\ & \ddots & \\ \mu_{i,j} & & 1 \end{pmatrix} \times \begin{pmatrix} b'_1 \\ \vdots \\ b'_d \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_d \end{pmatrix}$$

On a donc $B = UB'$ et $\det U = 1$ (U n'est pas à coefficients entiers !).
Si $G = BB^T$ alors $\det G = \det B'B'^T = \det \begin{pmatrix} \|b'_1\|^2 & & 0 \\ & \ddots & \\ 0 & & \|b'_d\|^2 \end{pmatrix}$. Donc $\det G = \prod_{i=1}^{d} \|b'_i\|^2$ et $\det L = \prod_{i=1}^{d} \|b'_i\|$.

On va définir une notion de base réduite en comparant la base du réseau à son orthogonalisé de Gram-Schmidt. En dimension 2, on a (b_1, b_2) est réduite si $\|b_1\| \leqslant \|b_2\|$ et $b_2 = \lambda_1 b_1 + \lambda_2 b'_2$ avec $|\lambda_1| > \dfrac{1}{2}$ et $\lambda_2 > 0$. ($b_1 = b'_1, b'_2$) est

une base orthogonale. En fait, (b'_1, b'_2) est l'orthogonalisée de Gram-Schmidt de (b_1, b_2) en prenant $\lambda_2 = 1$ et $\lambda_1 = \mu_{2,1}$.

$$b_2 = b'_2 + \mu_{2,1} b'_1 \text{ avec } |\mu_{2,1}| \leqslant \frac{1}{2}$$

Définition :

On dit que $(b_1, \ldots, b_d)$ est la base d'un réseau à la condition de taille si $|\mu_{i,j}| \leqslant \frac{1}{2}$, $\forall 1 \leqslant j < i \leqslant d$ où $\mu_{i,j}$ sont les coefficients obtenus par l'orthogonalisation de Gram-Schmidt de $(b_1, \ldots, b_d)$ c'est-à-dire :

$$b_i - b'_i = \mu_{i,1} b'_1 + \ldots + \mu_{i,j-1} b'_{j-1} \ \forall 2 \leqslant i \leqslant d \text{ avec } |\mu_{i,j}| \leqslant \frac{1}{2}$$

A partir de $(b_1, \ldots, b_d)$ base de L, on peut construire une autre base L qui a la condition de taille en temps polynomial en faisant :

$$b_i \leftarrow b_i - \lceil \mu_{i,j} \rfloor b_j \text{ pour } j = 1 \ldots i-1$$

La condition $\|b_1\| \leqslant \|b_2\|$ va être généralisée en dimension supérieure.

$$\begin{aligned} \|b_2\|^2 = < b_2, b_2 > &= < b'_2 + \mu_{2,1} b'_1, b'_2 + \mu_{2,1} b'_1 > \\ &= \|b'_2\|^2 + \mu_{2,1}^2 \|b'_1\|^2 \\ &= \|b'_2\|^2 + \mu_{2,1}^2 \|b_1\|^2 \end{aligned}$$

$$\|b_1\| \leqslant \|b_2\| \Leftrightarrow \|b'_1\|^2 \leqslant \|b'_2\|^2 + \mu_{2,1}^2 \|b'_1\|^2$$
$$\|b_1\| \leqslant \|b_2\| \Leftrightarrow \|b'_1\|^2 (1 - \mu_{2,1}^2) \leqslant \|b'_2\|^2$$

Comme $|\mu_{2,1}| \leqslant \frac{1}{2}$, on a $1 - \mu_{2,1}^2 \geqslant \frac{3}{4}$ et :

$$\frac{3}{4} \|b'_1\|^2 \leqslant \|b'_1\|^2 (1 - \mu_{2,1}^2) \leqslant \|b'_2\|^2$$

Définition : Hermite (1850)
Si $(b_1, b_2, \ldots, b_d)$ est une base de L, $(b'_1, \ldots, b'_d)$ est son orthogonalisé de Gram-Schmidt et si on a $\mu_{i,j}$ coefficients de Gram-Schmidt alors $(b_1, \ldots, b_d)$ est réduite au sens d'Hermite si :

1. $|\mu_{i,j}| \leqslant \frac{1}{2}$ (condition de taille)
2. $\forall i \geqslant 2$, $\|b'_i\|^2 \geqslant \frac{3}{4} \|b'_{i-1}\|^2$

Hermite donne un algorithme pour réduire une base selon les conditions de la définition. Mais il n'est pas polynomial.

LLL (1982)

On relâche la condition d'Hermite et on obtient un algorithme polynomial.

Définition :

On utilise les mêmes notations que la définition précédente. On dit que $(b_1, \ldots, b_d)$ est LLL-réduite si :

1. $|\mu_{i,j}| \leqslant \frac{1}{2}$ (condition de taille)
2. $\forall i \geqslant 2,\ \|b_i'\|^2 \geqslant \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \|b_{i-1}'\|^2$

Théorème :

Si $(b_1, \ldots, b_d)$ est LLL-réduite alors :

- $\|b_1\| \leqslant 2^{\frac{d-1}{4}} (\det L)^{\frac{1}{d}}$
- $\det L \leqslant \prod_{i=1}^{d} \|b_i\| \leqslant 2^{\frac{d(d-1)}{4}} \det L$
- $\forall x \in L,\ x \neq 0,\ \|b_1\| \leqslant 2^{\frac{d-1}{2}} \|x\|$ (en pratique $\|b_1\|$ est beaucoup plus petit que ça)

Algorithme 3 Algorithme LLL

Entrées: base d'un réseau L

Sorties: une base LLL réduite de L en $O(d^5(\log \mathcal{M})^3)$ où $\mathcal{M} = \max_{1 \leqslant i \leqslant d} \|b_i\|$

Réduire $(b_1, \ldots, b_d)$ pour avoir la condition de taille

S'il existe i tel que la condition 2) ne soit pas vérifiée, on échange b_i et b_{i-1} et on retourne à l'étape 1

L'algorithme setermine en temps polynomial et retourne une base LLL-réduite.

8.2 Application à la cryptographie

Construction en 1978 du crypto-système de Merkle-Hellmann basé sur le problème du sac à dos.

Problème du sac à dos : soit $a_1, \ldots, a_n, s \in \mathbb{N}$, existe-t-il $I \subset \{1, \ldots, n\}$ tel que $s = \sum_{i \in I} a_i$? C'est un problème décisionnel et NP-complet. Une manière de le résoudre est de chercher les a_i.

Définition :
On dit que le sac à dos est à super-croissance si :

$$a_j > \sum_{i=1}^{j-1} a_i, \ \forall j$$

Algorithme glouton : est-ce que $s = \sum_I a_i$?
- Déterminer le plus grand des a_i inférieurs à s.
- Remplacer s par $s - a_i$, et recommencer tant que $s > \min a_i$

Proposition :
Si s se décompose en $a_{i_1}+\ldots+a_{i_l}$, l'algorithme glouton retrouve $a_{i_l}, a_{i_{l-1}}, \ldots, a_1$ et on tombe sur $s = 0$. Sinon l'algorithme ne tombe pas sur $s = 0$.

Merkle-Hellmann Le crypto-système cache une instance faible du problème du sac à dos : un sac à dos à super-croissance.

Crypto-système de Merkle-Hellmann à clef publique

- Génération de clé
  - n : paramètre de sécurité
  - sac à dos à super croissance $(a_1, \ldots, a_n)$, $a_i \approx 2^{n+(i-1)}$ et un entier $N > \sum_{i=1}^{n} a_i$ et μ un entier premier à N.
    On calcule $b_i = a_i\mu \mod N$
  - on a la clé publique : $(b_1, \ldots, b_n)$ et la clé privée $(N, \mu, (a_1, \ldots, a_n))$
- Chiffrement de $m \in \{0,1\}^n$, $m = (m_1, \ldots, m_n)$, $m_i \in \{0,1\}$

$$c = \sum_{i=1}^{n} m_i b_i$$

- Déchiffrement de c : on calcule $s = c\mu^{-1} \mod N$
  On a donc $s = \sum_{i=1}^{n} m_i b_i \mu \mod N = \sum_{i=1}^{n} m_i a_i \mod N$. Mais $\sum_{i=1}^{n} m_i a_i < N$, donc on a $\sum_{i=1}^{n} m_i a_i$ dans $\mathbb{N}$. On résout le sac à dos à super croissance avec l'algorithme glouton et on retrouve m.

Définition : Densité d'un sac à dos $(a_1, \ldots, a_n)$

$$d = \frac{n}{\log(\max_{1 \leqslant i \leqslant n}(a_i))}$$

Pour le sac à dos public de Merkle-Hellmann $(b_1, \ldots, b_n)$, on a $b_i \approx N \approx 2^{2n}$ donc $d = \dfrac{n}{2n} \approx \dfrac{1}{2}$. Si $d < 0,645$ alors on peut résoudre le sac à dos avec LLL.

On veut résoudre le sac à dos $(b_1, \ldots, b_n), c$. On va considérer le réseau L de dimension $n+1$ inclus dans $\mathbb{R}^{n+2}$ engendré par :

$$A = \begin{pmatrix} 1 & & & & Kb_1 \\ & \ddots & & 0 & Kb_2 \\ & & \ddots & & \vdots \\ & 0 & & \ddots & Kb_n \\ & & & 1 & -Kc \end{pmatrix}$$

Avec $A \in \mathcal{M}_{n+1}$ et $K \in \mathbb{N}$

Soit $(e_1, \ldots, e_{n+1})$ les lignes de A. On a $x \in L$ si $x = \displaystyle\sum_{i=1}^{n+1} x_i e_i = (x_1 \ldots x_{n+1})A$.
Si $x = x_1e_1 + x_2e_2 + \ldots x_{n+1}e_{n+1}$ avec $x_1e_1 = x_1(1, 0, \ldots, 0, Kb_1) = (x_1, 0, \ldots, 0, Kb_1x_1)$ alors on a

$$x = (x_1, x_2, \ldots, x_{n+1}, K\sum_{i=1}^{n} x_i b_i - x_{n+1}Kc)$$

Ainsi :

$$\|x\|^2 = \sum_{i=1}^{n+1} x_i^2 + K^2 \left(\sum_{i=1}^{n} x_i b_i - x_{n+1} c \right)^2$$

Si $m = (m_1, m_2, \ldots, m_n)$ est une solution du sac à dos $c = \displaystyle\sum_{i=1}^{n} m_i b_i$, on lui associe le vecteur de L : $V = (m_1, m_2, \ldots, m_{n+1}, 1) \times A = (m_1, m_2, \ldots, 1, 0)$.
On a :

$$\begin{aligned} \|V\|^2 &= \sum_{i=1}^{n} m_i^2 + 1 \text{ avec } m_i \in \{0, 1\} \\ \|V\|^2 &\leqslant n + 1 \end{aligned}$$

Quitte a changer c en $c - \sum_{i=1}^{n} b_i$, on peut supposer :

$$\|V\|^2 \leqslant \frac{n+1}{2}$$

On pose K tel que $K^2 > \frac{n+1}{2}$. Donc si $x \in L$ et $\|x\|^2 < \frac{n+1}{2}$ et si $\|x\|^2 = \sum_{i=1}^{n+1} x_i^2 + K^2 \left(\sum x_i b_i - x_{n+1} c\right) \leqslant \frac{n+1}{2}$ alors :

$$\sum x_i b_i - x_{n+1} c = 0$$

On a donc $\sum x_i b_i = x_{n+1} c$. On peut en fait montrer que si $\|x\|^2 \leqslant \frac{n+1}{2}$ alors $(x_1, \ldots, x_n)$ est une solution du sac à dos avec probabilité 1.

En pratique, on applique LLL au réseau engendré par A. Le premier vecteur de la base réduite trouvée par LLL va donner une solution du sac à dos. Etant donnés c chiffré de $(m_1, m_2, \ldots, m_n)$ par Merkle-Hellmann, LLL permet de retrouver $(m_1, m_2, \ldots, m_n)$.

Attaque de Coppersmith sur RSA (1997)

RSA : $c = m^e \mod N$ tel que $N = pq$. Retrouver m à partir de c c'est trouver une racine du polynôme $X^e - c$ sur $\mathbb{Z}/n\mathbb{Z}$.
Si $m < N^{\frac{1}{e}}$ alors $m^e \mod N = m^e$ dans les entiers car $m^e < \left(N^{\frac{1}{e}}\right)^e = N$.
A partir de $c = m^e$ dans les entiers $\rightarrow m = \sqrt[e]{c}$

L'attaque de Coppersmith sur les messages stéréotypés est :
On connait $1 - \frac{1}{e}$ bits du poids fort de m. Par exemple $e = 3$, $m =$« le mot de passe est : toto »avec « le mot de passe est : »connu.
On a $m = m_0 + m'$ avec m_0 connu et m' inconnu et $\mid m' \mid < N^{\frac{1}{e}}$. Comme $c = m^e = (m_0 + m')^e \mod N$, on a que m' est racine de $c - (m_0 + X)^e \mod N$ avec $\mid m' \mid < N^{\frac{1}{e}}$.

Algorithme de Coppersmith

Soit N un entier à la factorisation inconnue et $f \in \mathbb{Z}/n\mathbb{Z}[X]$ de degré K, unitaire, alors il existe un algorithme qui trouve les entiers x_0 tel que $f(x_0) = 0 \mod N$ avec $\mid x_0 \mid < N^{\frac{1}{k}}$ en temps $O(K^5 \log^9 N)$.

Idée de l'algorithme Construire un polynôme g tel que $g(x_0) = 0$ dans les entiers.

Lemme Howgrave- Graham

Soit g un polynôme de degré K, $m \geqslant 0$ et $X \in \mathbb{N}$. Si :

1. $g(x_0) = 0 \mod N$ avec $\mid x_0 \mid < X$
2. $\|(g_0, g_1X, \ldots, g_kX^k)\| < \dfrac{N^m}{\sqrt{K+1}}$

Alors $g(x_0) = 0$ dans les entiers.
Pour construire g, on considère une famille de ploynômes construits à partir de f :

$$\tilde{f}_{i,j} = X^j f(X)^i N^{m-i}$$
$$\tilde{f}_{i,j}(x_0) = 0 \mod N^m$$

On considère le réseau engendré par les vecteurs des coefficients des $\tilde{f}_{i,j}(X)$. Un vecteur du réseau correspond à un polynôme g tel que (1) $g(x_0) = 0$ mod N^m. En appliquant LLL au réseau, on trouve un vecteur court qui satisfait (2) puis on trouve x_0 comme racine d'un polynôme sur les entiers.

Quelques applications :

- La factorisation $N = pq$ et $\tilde{p}$ tel que $\mid p - \tilde{p} \mid < N^{\frac{1}{4}}$. Alors on peut factoriser p par l'algorithme de Coppersmith.
- Cryptanalyse de RSA avec d petit. Si $d < N^{0,292}$ on peut retrouver d en temps polynomial