

# **Projet : résistance d'un LFSR filtré aux attaques algébriques**

**Anne CANTEAUT**

INRIA - projet CODES  
B.P. 105

78153 Le Chesnay Cedex

`Anne.Canteaut@inria.fr`

`http://www-rocq.inria.fr/codes/Anne.Canteaut/COURS\_C`

## Règle du jeu

Le projet est à me rendre avant le **11 janvier 2005 (midi)**. Vous devez me fournir les documents suivants :

- l'ensemble des fichiers nécessaires au programme (fichiers source, Makefile, éventuellement des fichiers d'initialisation). Ces fichiers doivent m'être envoyés par mail (si possible au format tar.gz) ;
- un rapport expliquant les différentes options choisies pour la programmation (algorithmes, représentation des données). Ce rapport doit également présenter et commenter les résultats de simulation obtenus à l'aide de ce programme. Les rapports doivent m'être envoyés par mail en format PostScript ou PDF. Les rapports rédigés en LaTeX seront considérés avec bienveillance (une introduction à LaTeX et un exemple sont disponibles sur la page Web du projet).

## Chapitre 1

# Problématique cryptographique

L'objet de ce projet est d'écrire un programme permettant de déterminer si un système de chiffrement de type LFSR filtré résiste aux attaques algébriques.

### 1.1 Chiffrement à flot et LFSRs

Un système de chiffrement à clef secrète à flot (par opposition aux chiffrements par blocs) consiste à additionner bit à bit au texte clair une suite aléatoire de même longueur, appelée *suite chiffrante*. Ce système assure une sécurité parfaite sous la condition que la suite chiffrante soit une suite complètement aléatoire de la même taille que le message à chiffrer. Cependant, comme il n'est en général pas envisageable de partager une clef secrète qui soit aussi longue que le message à chiffrer, on utilise dans la pratique une *suite pseudo-aléatoire* générée de façon déterministe à partir d'un secret commun court qui, lui, peut être échangé plus facilement.

Une méthode classique pour générer une suite binaire pseudo-aléatoire est d'utiliser un registre à décalage à rétroaction linéaire (LFSR pour Linear Feedback Shift Register). Un LFSR de longueur  $L$  est composé d'un registre à décalage contenant une suite de  $L$  bits  $(u_t, \dots, u_{t+L-1})$ , et d'une fonction de rétroaction linéaire.

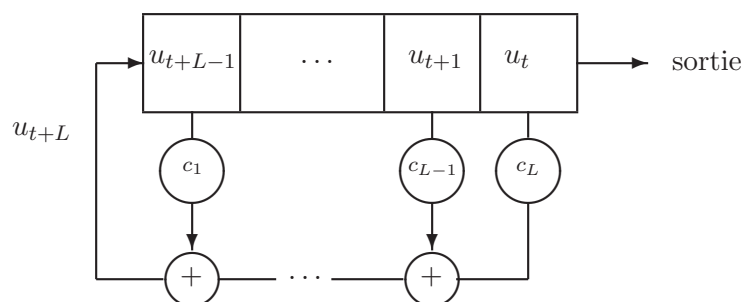


FIG. 1.1 – Fonctionnement d'un registre à décalage à rétroaction linéaire

A chaque top d'horloge, le bit de poids faible  $u_t$  constitue la sortie du registre, et les autres bits sont décalés vers la droite. Le nouveau bit  $u_{t+L}$  placé dans la cellule de poids fort du

registre est donné par une fonction linéaire des bits  $(u_t, \dots, u_{t+L-1})$

$$u_{t+L} = c_1 u_{t+L-1} + c_2 u_{t+L-2} + \dots + c_L u_t \quad (1.1)$$

où les coefficients de rétroaction  $(c_i)_{1 \leq i \leq L}$  sont des éléments de  $\mathbf{F}_2$ .

Les bits  $(u_0, \dots, u_{L-1})$ , qui déterminent entièrement la suite, constituent *l'état initial du registre*.

Toutefois la longueur du registre reste en pratique trop faible pour se mettre à l'abri d'une attaque à clair connu : il suffit de connaître  $L$  bits consécutifs d'un couple clair-chiffré pour retrouver l'initialisation du registre. Même si les coefficients de rétroaction  $(c_i)_{1 \leq i \leq L}$  sont inconnus (et font partie de la clef secrète), ils peuvent être retrouvés grâce à l'algorithme de Berlekamp-Massey à partir de la connaissance de  $2L$  bits consécutifs de suite chiffrante.

## 1.2 Les LFSRs filtrés

L'algorithme de Berlekamp-Massey rend impossible l'utilisation d'un registre à décalage à rétroaction linéaire pour générer une suite chiffrante destinée à servir de suite pseudo-aléatoire dans un chiffrement à flot. Pour résister à cette attaque tout en conservant les bonnes propriétés des LFSRs, on utilise classiquement un procédé appelé *LFSR filtré*, qui consiste à utiliser comme suite chiffrante la sortie d'une fonction appliquée à certains bits de l'état du registre.

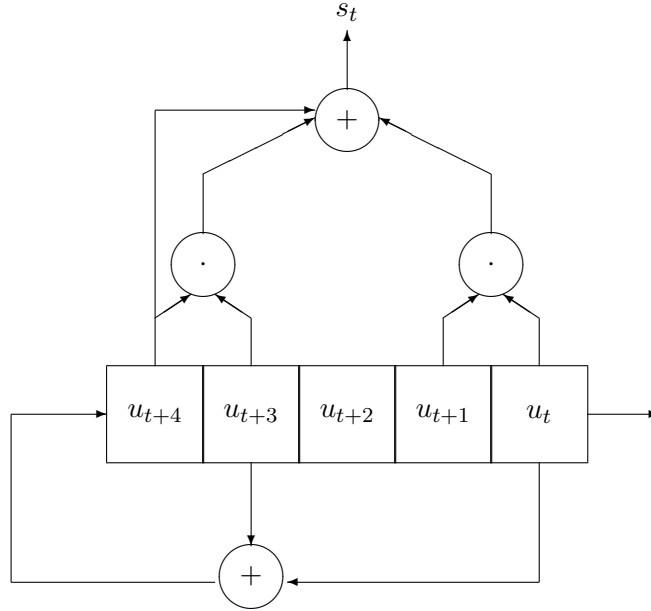
Toute fonction de  $n$  bits vers un bit peut s'écrire comme un polynôme à  $n$  variables, par exemple :

$$f(x_1, x_2, x_3, x_4) = x_1 x_2 + x_3 x_4 + x_1 .$$

Cette représentation polynomiale s'appelle la forme algébrique normale de la fonction (abrégée généralement en anglais par ANF). On choisira toujours pour  $f$  une fonction dite équilibrée, c'est-à-dire qui vaut 1 en exactement la moitié des valeurs de  $(x_1, \dots, x_n)$ .

La figure suivante montre un exemple d'utilisation de cette fonction à 4 variables pour filtrer un registre de longueur 5. Les 4 entrées de la fonction à l'instant  $t$  sont ici choisies en

position  $t + 4$ ,  $t + 3$ ,  $t + 1$  et  $t$  de la suite produite par le LFSR.



Si on note  $s_t$  le bit qui sort de la fonction à l'instant  $t$  (*i.e.* le bit  $t$  de la suite chiffrante) et  $(u_{t+4}, \dots, u_t)$  l'état du LFSR à cet instant, on a :

$$s_t = u_{t+4}u_{t+3} + u_{t+1}u_t + u_{t+4} .$$

### 1.3 Attaque algébrique de base

Une des raisons pour lesquelles il ne faut pas choisir comme fonction de filtrage une fonction de petit degré est qu'un tel choix rendrait le système vulnérable à une attaque dite *algébrique* de base. En effet, si la fonction est de petit degré  $d$ , chaque bit de suite chiffrante,  $s_t$ , s'écrit comme une fonction de degré  $d$  en les  $L$  bits de l'état initial, puisque l'état du registre à l'instant  $t$  est une fonction linéaire de son état initial. En reprenant l'exemple ci-dessus, on peut exprimer  $s_t$  comme une fonction de degré 2 en  $(u_0, \dots, u_4)$  qui sont les 5 bits d'initialisation du registre :

$$s_0 = u_3u_4 + u_0u_1 + u_4 .$$

En utilisant le fait que la suite  $(u_t)_{t \geq 0}$  produite par le LFSR vérifie la récurrence  $u_t = u_{t-2} + u_{t-5}$ , on déduit qu'à l'instant  $t = 1$ , on a

$$\begin{aligned} s_1 &= u_4u_5 + u_1u_2 + u_5 \\ &= u_0u_4 + u_3u_4 + u_1u_2 + u_3 + u_0 , \end{aligned}$$

puisque  $u_5 = u_3 + u_0$ . Au temps  $t = 2$ , on a

$$s_2 = u_0u_1 + u_1u_2 + u_3u_4 + u_1u_3 + u_2u_3 + u_1 + u_4 .$$

La connaissance de  $N$  bits de suite chiffrante permet donc d'écrire un système de  $N$  équations de degré 2 à 5 variables. Un tel système peut se résoudre grâce à des algorithmes de résolution

de systèmes algébriques tels les algorithmes de base de Gröbner. Une méthode moins efficace mais plus simple consiste à assimiler tous les monômes de degré inférieur ou égal au degré des équations à des nouvelles variables. Dans l'exemple, on pose donc  $x_0 = u_0, \dots, x_4 = u_4$ ,  $x_5 = u_0u_1, x_6 = u_0u_2, \dots, x_{14} = u_3u_4$ . Chaque équation de degré 2 s'écrit donc comme une équation linéaire en  $x_0, \dots, x_{14}$ , par exemple

$$s_2 = x_6 + x_9 + x_{14} + x_{10} + x_{12} + x_1 + x_4 .$$

La donnée de 15 équations de cette forme fournit donc un système linéaire de 15 équations à 15 inconnues que l'on peut résoudre par une simple élimination de Gauss. La complexité de l'algorithme est donc de l'ordre de

$$\left[ \sum_{i=1}^d \binom{L}{i} \right]^3$$

où  $d$  est le degré de la fonction de filtrage  $f$  et  $L$  la longueur du LFSR. Ce nombre d'opérations n'est donc plus accessible dès que le degré de la fonction est élevé, quand on considère des registres de longueur cryptographique, i.e., quand  $L$  dépasse 100.

## 1.4 Attaque algébrique évoluée

Toutefois, en 2003, Courtois et Meier ont proposé une amélioration de cette attaque, qui peut parfois aboutir même lorsque le degré de la fonction de filtrage est élevé. L'attaque fonctionne dès lors qu'il existe des relations de bas degré entre la sortie de la fonction et ses entrées. Plus précisément, l'attaquant recherche des fonctions  $g$  et  $h$  de petit degré qui vérifient

- pour tout  $(x_1, \dots, x_n)$ ,  $g(x_1, \dots, x_n)f(x_1, \dots, x_n) = 0$ ,
- ou pour tout  $(x_1, \dots, x_n)$ ,  $h(x_1, \dots, x_n)[1 + f(x_1, \dots, x_n)] = 0$ .

Si de telles fonctions  $g$  ou  $h$  de degré  $d$  existent, on peut engendrer un système d'équations de degré  $d$  de la manière suivante :

- si  $s_t = 1$ , on a  $g(u_t, \dots, u_{t+L-1}) = 0$  où  $(u_t, \dots, u_{t+L-1})$  est l'état du registre à l'instant  $t$  ;
- si  $s_t = 0$ , on a  $h(u_t, \dots, u_{t+L-1}) = 0$ .

En exprimant l'état du registre à l'instant  $t$  comme une fonction linéaire de l'état initial, on obtient comme précédemment un système d'équations de degré  $d$  en  $L$  variables (les bits de l'état initial), que l'on peut résoudre par les techniques évoquées plus haut.

## 1.5 Immunité algébrique de la fonction de filtrage

Pour se mettre à l'abri de ces attaques, il est donc essentiel que toutes les fonctions  $g$  et  $h$  qui ont la propriété décrite ci-dessus soient de degré élevé. Dans la suite, on notera  $\mathcal{AN}(f)$  (ensemble annulateur de  $f$ ) l'ensemble des fonctions  $g$  à  $n$  variables qui vérifient  $g(x_1, \dots, x_n)f(x_1, \dots, x_n) = 0$ . Le paramètre essentiel pour la cryptanalyse est donc le degré minimal des fonctions de  $\mathcal{AN}(f)$  et  $\mathcal{AN}(1 + f)$ . Ce paramètre est appelé *immunité algébrique de  $f$* , noté  $AI(f)$  :

$$AI(f) = \min \deg \{g \in \mathcal{AN}(f) \cup \mathcal{AN}(1 + f)\} .$$

Le but du projet est de calculer cette valeur pour une fonction  $f$  donnée. On peut démontrer qu'il existe toujours une fonction  $g$  dans  $\mathcal{AN}(f)$  ou une fonction  $h$  dans  $\mathcal{AN}(1 + f)$  de degré

inférieur ou égal à  $\lfloor \frac{n+1}{2} \rfloor$ . L'objectif est donc déterminer s'il existe des fonctions de degré strictement inférieur dans ces deux ensembles.

Pour rechercher toutes les fonctions  $g$  de degré inférieur ou égal à un degré  $d$  donné (on prendra généralement  $d = \lfloor \frac{n+1}{2} \rfloor - 1$ ) dans  $\mathcal{AN}(f)$ , on utilise le fait qu'une telle fonction est une combinaison linéaire des monômes de degré inférieur ou égal à  $d$ , et qu'elle doit s'annuler en tous les points  $(x_1, \dots, x_n)$  tels que  $f(x_1, \dots, x_n) = 1$ . En effet, si  $f$  vaut 0 au point  $(x_1, \dots, x_n)$ , on a bien  $g(x_1, \dots, x_n)f(x_1, \dots, x_n) = 0$ . On va donc construire une matrice dont chaque ligne correspond aux valeurs prises par un monôme de degré inférieur ou égal à  $d$  en tous les points où  $f$  vaut 1. Trouver s'il existe une ou des combinaisons linéaires non nulles des monômes qui valent 0 sur tous ces points revient à trouver des combinaisons linéaires des lignes de la matrice qui valent 0. Pour cela, il suffit de faire une élimination de Gauss sur cette matrice.

## 1.6 Exemple

Considérons comme exemple (de taille non cryptographique) la fonction de degré 3 à 4 variables suivante :

$$f(x_1, x_2, x_3, x_4) = x_1 + x_2x_3 + x_1x_3x_4 ,$$

et on veut déterminer s'il existe des fonctions de degré 2 dans  $\mathcal{AN}(f)$ . La fonction  $f$  vaut 1 pour la moitié des valeurs d'entrées, c'est-à-dire sur les 8 quadruplets

$$\mathcal{S}(f) = \{(1,0,0,0), (1,1,0,0), (1,0,1,0), (1,1,1,0), (1,0,0,1), (1,1,0,1), (0,1,1,1), (1,1,1,1)\} .$$

On construit maintenant la matrice dont chaque ligne correspond aux 8 valeurs prises en  $\mathcal{S}(f)$  par un monôme de degré inférieur ou égal à 2. Cette matrice est décrite à la figure 1.2.

| monôme   | $x_1x_2x_3x_4$ |      |      |      |      |      |      |      |
|----------|----------------|------|------|------|------|------|------|------|
|          | 1000           | 1100 | 1010 | 0110 | 1001 | 1101 | 0111 | 1111 |
| 1        | 1              | 1    | 1    | 1    | 1    | 1    | 1    | 1    |
| $x_1$    | 1              | 1    | 1    | 0    | 1    | 1    | 0    | 1    |
| $x_2$    | 0              | 1    | 0    | 1    | 0    | 1    | 1    | 1    |
| $x_3$    | 0              | 0    | 1    | 1    | 0    | 0    | 1    | 1    |
| $x_4$    | 0              | 0    | 0    | 0    | 1    | 1    | 1    | 1    |
| $x_1x_2$ | 0              | 1    | 0    | 0    | 0    | 1    | 0    | 1    |
| $x_1x_3$ | 0              | 0    | 1    | 0    | 0    | 0    | 0    | 1    |
| $x_1x_4$ | 0              | 0    | 0    | 0    | 1    | 1    | 0    | 1    |
| $x_2x_3$ | 0              | 0    | 0    | 1    | 0    | 0    | 1    | 1    |
| $x_2x_4$ | 0              | 0    | 0    | 0    | 0    | 1    | 1    | 1    |
| $x_3x_4$ | 0              | 0    | 0    | 0    | 0    | 0    | 1    | 1    |

FIG. 1.2 – Matrice représentant les valeurs des monômes de degré inférieur ou égal à 2 sur l'ensemble  $\mathcal{S}(f)$

Un pivot de Gauss appliqué à cette matrice nous montre qu'il y a 3 lignes nulles corres-

pondant aux fonctions

$$\begin{aligned} &1 + x_1 + x_2 + x_1x_2 \\ &1 + x_1 + x_3 + x_1x_3 \\ &1 + x_1 + x_4 + x_1x_4 + x_2x_3 + x_3x_4 . \end{aligned}$$

Il y a donc  $2^3 - 1 = 7$  fonctions  $g$  non nulles de degré inférieur ou égal à 2 dans  $\mathcal{AN}(f)$  : il s'agit de toutes les combinaisons linéaires non nulles des 3 fonctions précédentes.

De même, l'ensemble annulateur de  $1 + f$  est obtenu par le même algorithme où les colonnes de la matrice correspondent cette fois-ci aux points où  $1 + f(x_1, \dots, x_4) = 1$ , i.e., où  $f(x_1, \dots, x_4) = 0$ . On trouve alors 7 fonctions de degré 2 dans  $\mathcal{AN}(1 + f)$ , qui sont les combinaisons linéaires non nulles des fonctions

$$\begin{aligned} &x_1 + x_1x_3 \\ &x_1 + x_1x_2 + x_1x_4 \\ &x_1x_2 + x_2x_3 . \end{aligned}$$

## Chapitre 2

# Mise en œuvre

Le programme à écrire doit donc prendre comme arguments le nom d'un fichier décrivant la fonction (sous forme polynômiale) et le degré  $d$  correspondant au degré maximal des fonctions  $g$  que l'on recherche dans  $\mathcal{AN}(f)$  et  $\mathcal{AN}(1+f)$ . Typiquement, on prendra pour  $d$  la valeur  $\lfloor \frac{n+1}{2} \rfloor - 1$  où  $n$  est le nombre de variables de la fonction.

La fonction sera représentée dans le fichier sous la forme suivante :

- la première ligne donne le nombre de variables ;
- les lignes suivantes donnent la forme algébrique normale de la fonction.

Par exemple,

```
4
x1 + x2x3 + x1x3x4
```

On souhaite que le programme affiche les dimensions des espaces  $\mathcal{AN}(f)$  et  $\mathcal{AN}(1+f)$ , ainsi qu'une base des fonctions qui annulent  $f$  (puis  $1+f$ ) et le degré de ces fonctions. L'affichage du degré est important car, dans l'exemple précédent, il permet de conclure que les fonctions dans  $\mathcal{AN}(f)$  sont toutes de degré 2 (et non de degré 1). Par exemple, on affichera

```
1 + x1 + x2 + x1x2          degre = 2
1 + x1 + x3 + x1x3          degre = 2
1 + x1 + x4 + x1x4 + x2x3 + x3x4      degre = 2
```

Dimension de  $\mathcal{AN}(f)$  = 3

```
*****
+ x1 + x1x3          degre = 2
+ x1 + x1x2 + x1x4    degre = 2
+ x1x2 + x2x3         degre = 2
```

Dimension de  $\mathcal{AN}(1+f)$  = 3

Le nombre de variables raisonnable pour faire tourner le programme sera de l'ordre de  $n \sim 10 - 15$ .

## 2.1 Lecture de la forme algébrique normale

Une fonction permettant de lire et d'interpréter la forme algébrique normale de la fonction, et le fichier en-tête donnant son interface, sont fournis sur

[http://www-rocq.inria.fr/codes/Anne.Canteaut/COURS\\_C/PROJET/lecture\\_anf.c](http://www-rocq.inria.fr/codes/Anne.Canteaut/COURS_C/PROJET/lecture_anf.c)  
[http://www-rocq.inria.fr/codes/Anne.Canteaut/COURS\\_C/PROJET/lecture\\_anf.h](http://www-rocq.inria.fr/codes/Anne.Canteaut/COURS_C/PROJET/lecture_anf.h)

Cette fonction prend comme arguments un `unsigned int *` destiné à recevoir l'adresse du nombre de variables et le nom de fichier contenant la fonction. Elle retourne un tableau de  $2^n$  éléments de type `unsigned short`, tel que l'élément d'indice  $x$  vaut 1 si sa décomposition en base 2 correspond à un monôme présent dans la forme algébrique normale (et 0 si ce monôme n'est pas présent). Dans toute la suite, les monômes sont représentés par des entiers de la façon suivante : le monôme  $x_{i_1}x_{i_2}\dots x_{i_d}$  est représenté par l'entier  $2^{i_1-1} + 2^{i_2-1} + \dots + 2^{i_d-1}$ , car on suppose que les variables sont numérotées de 1 à  $n$ . Ainsi, la fonction à 4 variables de l'exemple sera représentée par un tableau de 16 éléments dont les seuls éléments égaux à 1 sont ceux d'indice  $1 = 2^0$  (pour  $x_1$ ), d'indice  $6 = 2^2 + 2^1$  (pour  $x_2x_3$ ) et d'indice  $13 = 2^3 + 2^2 + 2^0$  (pour  $x_1x_3x_4$ ).

## 2.2 Représentation de la fonction sous forme du vecteur de ses valeurs

Pour représenter la fonction de manière plus maniable, écrire une fonction qui, à partir du tableau renvoyé par la fonction précédente, retourne un tableau d'`unsigned int` correspondant à la suite des valeurs de  $f$  en tous les  $n$ -uplets  $(x_1, \dots, x_n)$ . Ce tableau contiendra  $2^n/(8 * \text{sizeof}(\text{unsigned int}))$  éléments. La valeur en un point  $(v_1, \dots, v_n)$  de la fonction se calcule à partir de la forme algébrique normale par la relation suivante :

$$f(v_1, \dots, v_n) = \sum_{u \preceq v} a[u] \bmod 2$$

où la relation d'ordre partiel  $x \preceq y$  entre deux mots de  $n$  bits,  $x$  et  $y$ , signifie que  $x_i \leq y_i$  pour tout  $i$ ,  $1 \leq i \leq n$ . Les  $a[u]$  sont les coefficients de la forme algébrique normale (égaux à 0 ou 1) retournés par la fonction précédente. Avec notre exemple, si on veut calculer la valeur de  $f$  en  $(v_1, \dots, v_4) = (1, 0, 1, 0)$ , on considère tous les mots de 4 bits,  $u$ , tels que  $u \preceq (1, 0, 1, 0)$ , c'est-à-dire tous les mots tels que  $u_2 = u_4 = 0$  et  $u_1$  et  $u_3$  sont quelconques. On a donc

$$\begin{aligned} f(1, 0, 1, 0) &= a[0, 0, 0, 0] + a[0, 0, 1, 0] + a[1, 0, 0, 0] + a[1, 0, 1, 0] \\ &= 0 + 0 + 1 + 0 = 1. \end{aligned}$$

On fera en sorte que cette fonction calcule également le poids de  $f$ , c'est-à-dire le nombre de points où  $f$  vaut 1. On rappelle qu'en pratique, les fonctions utilisées ont pour poids  $2^{n-1}$ .

## 2.3 Construction de la table des monômes

Afin de faciliter le calcul de la matrice représentant la valeur des monômes de degré au plus  $d$  sur les points où  $f$  vaut 1, écrire une fonction qui construit un tableau dont les éléments

correspondent aux monômes de  $n$  variables de degré inférieur ou égal à  $d$ . Pour l'exemple d'une fonction à 4 variables et des monômes de degré au plus 2, le tableau comportera

$$\sum_{i=0}^2 \binom{4}{i} = 1 + 4 + 6 = 11$$

éléments. Ces éléments seront les entiers 0 (pour le monôme 1), 1 (pour  $x_1$ ), 2 (pour  $x_2$ ), 4 (pour  $x_3$ ), 8 (pour  $x_4$ ), 3 (pour  $x_1x_2$ ), 5 (pour  $x_1x_3$ ), 9 (pour  $x_1x_4$ ), 6 (pour  $x_2x_3$ ), 10 (pour  $x_2x_4$ ), 12 (pour  $x_3x_4$ ).

On pourra écrire une fonction intermédiaire qui calcule les coefficients binomiaux  $\binom{n}{i}$  pour  $0 \leq i \leq d$ , et leur somme qui correspond au nombre de monômes considérés.

## 2.4 Construction de la matrice

Écrire une fonction qui retourne, sous forme d'un `unsigned int**`, la matrice dont les lignes correspondent aux valeurs prises par un monôme en l'ensemble des points où  $f$  vaut 1. Cette fonction prendra comme arguments le tableau donnant les valeurs de  $f$ , le poids de  $f$ , le nombre de variables  $n$ , le nombre de monômes et le tableau des monômes construit précédemment. Chaque élément de la matrice correspondra évidemment à  $8 * \text{sizeof}(\text{unsigned int})$  valeurs du monôme.

## 2.5 Élimination de Gauss

Écrire une fonction qui prend comme arguments la matrice précédente, son nombre de lignes (le nombre de monômes), son nombre de colonnes (le poids de la fonction), le nombre de variables et le tableau des monômes, et qui retourne le nombre de lignes nulles de la matrice après élimination de Gauss. Cette fonction affichera, à chaque fois qu'elle trouve une ligne nulle, la forme algébrique normale de la fonction correspondant et son degré. Ces informations seront obtenues en gardant une trace de chaque opération (autrement, à chaque fois qu'une ligne sera remplacée par elle-même plus une autre, on stockera cette information). On remarquera que le degré d'une fonction booléenne correspond au plus grand poids de Hamming des monômes qui la constituent.

On rappelle l'algorithme d'élimination de Gauss sur une matrice  $M$  à  $k$  lignes et  $m$  colonnes. La matrice  $T$  sert à garder la trace des additions de lignes effectuées. Il s'agit d'une matrice carrée dont le nombre de lignes (et de colonnes) est égal au nombre de lignes de  $M$ . Elle est initialisée à l'identité. Dans la suite, la notation  $X_i$  désigne la ligne  $i$  d'une matrice  $X$ ;  $\oplus$  désigne la somme bit à bit de deux lignes.

`lignes_traitees = 0.`

Pour  $i$  de 0 à  $k - 1$ :

- Chercher le premier indice  $j \geq \text{lignes\_traitees}$  tel que  $M_{i,j} \neq 0$ .
- Si  $M_{i,j} = 0$  pour tout  $j$ , la ligne  $i$  est nulle. Incrémenter le nombre de lignes nulles et afficher la fonction correspondant et son degré.
- Sinon :
  - Si  $i \neq j$ , échanger les colonnes  $i$  et  $j$  de  $M$ .

- Pour tout  $\ell$ ,  $i + 1 \leq \ell \leq k$ , tel que  $M_{\ell,i} \neq 0$ 
  - $M_\ell \leftarrow M_\ell \oplus M_i$ .
  - $T_\ell \leftarrow T_\ell \oplus T_i$ .
- Incréments `lignes_traitees`.