Université Pierre et Marie Curie
Master de Sciences & Technologies
Mention Informatique niveau 1
Année universitaire 2009-2010

Science & Technologie du Logiciel
PSTL
Encadrants : Guénaël Renault [1]
et Ludovic Perret[1]

Rapport de **PSTL**

Cassage de code avec des polynômes

Par
Mohamed Amine Najahi [2] et Anissa Feukam[2]

Janvier-Juin 2010

1. {ludovic.perret,guenael.renault}@lip6.fr
2. {mohamed.najahi,anissa.feukam}@etu.upmc.fr

Table des matières

Introduction

Bien que très ancienne, la cryptologie ou la science du secret, a dû attendre la deuxième moitié du XX^{ème} siècle pour acquérir le statut de science à part entière. En effet, longtemps considérée comme un art ou un ensemble de techniques du « secret », elle était restée entre les mains des stratèges et diplomates. Avec l'avènement de la théorie de l'information, cette discipline s'est démocratisée et un lien solide avec les mathématiques est désormais établi.

La cryptologie moderne s'est ainsi scindée en deux catégories : la cryptographie et la cryptanalyse. La cryptographie est une discipline visant à garantir la confidentialité, l'intégrité et l'authenticité lors d'échanges d'informations entre deux parties. Elle a des applications dans plusieurs secteurs civils et militaires. D'ailleurs, la sécurité du commerce électronique ainsi que la protection de la vie privée reposent de plus en plus sur cette science.
La cryptographie à clé secrète (ou symétrique) est la forme la plus ancienne de chiffrement. Elle suppose qu'une clef est a priori partagée par les deux parties.
Plus récente, la cryptographie à clef publique (ou asymétrique) s'est quant à elle affranchie de cette contrainte.
Néanmoins, sa souplesse et sa facilité d'adaptation aux besoins concrets de l'industrie ont permis à la cryptographie symétrique de rester très active.
Les chiffrements à clef secrète ne sont pas pour autant infaillibles. En effet, un cryptosytème sans faille, du point de vue mathématique, est concevable. Cependant, sa grande complexité du point de vue implémentation et son coût d'utilisation le rendent prohibitif. Il ne serait donc pas d'une grande utilité pour l'industrie.
Chercher les failles de ces systèmes "imparfaits" est donc le but de la cryptanalyse. Cette discipline regroupe tous les moyens de casser les cryptosystèmes, et de retrouver des messages sans posséder leur clef.

Organisation du document

Nous commencerons ce rapport par une introduction aux cryptosystèmes à clef secrète.
On décrira ensuite le fonctionnement d'une famille de chiffrements par bloc, en l'occurrence Katan et Ktantan [2]. L'utilité de ces chiffrements et leurs limites seront bien sûr abordés.
La dernière partie concernera la cryptanalyse algébrique. Une introduction aux attaques algébriques sera proposée. Pour finir, nous dévoilerons les résultats de notre attaque sur Katan/Ktantan.

1 Présentation générale des chiffrements par bloc

1.1 Aperçu historique

La cryptographie à clef secrète se décompose en 2 grandes catégories :
- le chiffrement par bloc.
- le chiffrement par flot

Les cryptosystèmes par bloc agissent sur des données de taille fixe. Le texte clair est donc découpé avant d'être chiffré.
Les chiffrement par flot commencent par générer une suite chiffrante de la même taille que le message à crypter. Ensuite, une transformation bit à bit entre cette suite et le message produit le chiffré.
Dans cette introduction au chiffrement par bloc, nous allons présenter 3 chiffrements :
- le chiffrement de César.
- le chiffrement de Vigenère.
- le chiffrement de Vernam.

1.1.1 *Le chiffrement de César :*

Il est considéré comme l'un des premiers cryptosystèmes à clef secrète. La clef K correspond à un chiffre entre 0 et 25. Pour chiffrer un message, chaque lettre de ce dernier est décalée de K positions (décalage dans l'ordre alphabétique). Déchiffrer le message revient à décaler dans le sens inverse.

M	$B_{(1)}$	$O_{(14)}$	$N_{(13)}$	$J_{(9)}$	$O_{(8)}$	$U_{(20)}$	$R_{(17)}$
$K=1$	$C_{(2)}$	$P_{(15)}$	$O_{(14)}$	$K_{(10)}$	$P_{(9)}$	$V_{(21)}$	$S_{(18)}$
$K=2$	$D_{(3)}$	$Q_{(16)}$	$P_{(15)}$	$L_{(11)}$	$Q_{(10)}$	$W_{(22)}$	$T_{(19)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$K=12$	$N_{(13)}$	$A_{(0)}$	$Z_{(25)}$	$V_{(21)}$	$U_{(20)}$	$G_{(6)}$	$D_{(3)}$

Tableau 1 – *Plusieurs chiffrés du mot « BONJOUR » avec des clefs différentes.*

En pondérant les lettres de l'alphabet par des coefficients (de 0 à 25), ces deux opérations peuvent être vue comme une addition dans le groupe $(\mathbb{Z}/26\mathbb{Z}, +)$. Le chiffrement revient à additionner le coefficient de la lettre avec K. On déchiffre en ajoutant $26 - K$, toujours dans $\mathbb{Z}/26\mathbb{Z}$.

1.1.2 *Le chiffrement de Vigenère :*

L'idée du chiffrement de vigenère est d'utiliser le chiffrement de César, mais avec un décalage spécifique pour chaque lettre. Pour cela, on utilise un tableau (26×26) composé des 26 lettres (en entrée et en sortie) de l'alphabet, écrites dans l'ordre, mais décalées de ligne en ligne d'un caractère (vers la gauche). Les lettres du mot à coder se repèrent sur la $1^{ère}$ colonne du tableau, et celles de la clef sur la $1^{ère}$ ligne du tableau. Pour coder un message, on choisit un mot de longueur arbitraire qui sera la clef. On écrit ensuite cette clé sous le message à coder, en la répétant aussi souvent que nécessaire pour que sous chaque lettre du message à coder, on trouve une lettre de la clé. Pour coder, on regarde dans le tableau l'intersection de la ligne de la lettre à coder avec la colonne de la lettre de la clé.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	[R]$_9$	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	[V]$_1$	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	[Y]$_{13}$	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	[U]$_7$	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[A]$_{11}$	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	[W]$_{12}$	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	[H]$_6$	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	[G]$_4$	[H]$_{10}$	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	[F]$_2$	G	H	I	J	K	[L]$_8$	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	[L]$_5$	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	[S]$_3$	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Message	C	R	Y	P	T	O	G	R	A	P	H	I	E
Clef	T	O	U	R	S	T	O	U	R	S	T	O	U
Chiffré	V_1	F_2	S_3	G_4	L_5	H_6	U_7	L_8	R_9	H_{10}	A_{11}	W_{12}	Y_{13}

Tableau 2 – *Chiffrement du mot « CRYPTOGRAPHIE » par la clef « TOURS ».*

1.1.3 *Le chiffrement de Vernam :*

Le chiffrement de Vernam n'est pas un cryptosystème par bloc, mais a été, comme on le verra plus loin, le précurseur de la cryptographie symétrique moderne. Le principe de ce chiffrement est simple, pour chiffrer un message M, de taille, disons N, il faut générer un masque aléatoire K de même taille, et ensuite calculer $C = M * K$ qui sera donc le chiffré. Le symbole $*$ peut représenter plusieurs opérateurs binaires (l'addition $\oplus$ dans $\mathbb{F}_2$ convient si $M \in \mathbb{F}_2^N$, l'addition des lettres de l'alphabet pondérées, modulo 26 est valide si $M \in \mathcal{A}^N$,...). K joue ici le rôle de clef, et sera donc partagée de manière confidentielle par les deux protagonistes. Tandis que C peut transiter par tout canal non sécurisé sans risque, puisqu'aucune caractéristique ne permet de le lier au message initial M.

Le chiffrement de Vernam est parfois appelé masque jetable car aucune clef ne doit être réutilisée. En effet, soient $(M_i, C_i)_{i>1}$ une suite de couples message/chiffré par la même clef K. On a $C_1 = M_1 \oplus K$ et $C_i = M_i \oplus K$ d'où $C_1 \oplus C_i = M_1 \oplus M_i$. Les C_i étants publics, tout attaquant qui parviendrait à recouvrir M_1 serait capable de déchiffrer tous les C_i car $M_i = C_1 \oplus C_i \oplus M_1$. Claude Shannon a démontré dans [1], que ce chiffrement était « parfait », dans le sens où seule une attaque par recherche exhaustive pouvait révéler un message à partir de son chiffré. Cependant, la mise en œuvre pratique du chiffrement de Vernam pose deux problèmes : l'un logique et l'autre logistique.

Message	Corps de M	Message aléatoire	Opération	Chiffré
$FEUKAM$	$\mathcal{A}$	$WRCBQY$	Addition mod 26	$\begin{array}{rcccccc} & F_{(5)} & E_{(4)} & U_{(20)} & K_{(10)} & A_{(0)} & M_{(12)} \\ + & W_{(22)} & R_{(17)} & C_{(2)} & B_{(1)} & Q_{(16)} & Y_{(24)} \\ \hline \text{mod } 26 & (27) & (21) & (22) & (11) & (16) & (36) \\ & B_{(1)} & V_{(21)} & W_{(22)} & L_{(11)} & Q_{(16)} & K_{(10)} \end{array}$
111000	$\mathbb{F}_2$	011011	Xor $\oplus$	$\begin{array}{r} 111000 \\ \oplus \quad 011011 \\ \hline 100011 \end{array}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Tableau 3 – *Exemples de chiffrements de Vernam.*

Problème 1 Le message M a la même taille que la clef K. Si l'on disposait d'une méthode sécurisée d'échange de clef (en l'occurrence K ici), pourquoi ne pas l'utiliser directement pour s'échanger M.[3] .

Problème 2 La génération de données aléatoires se révèle être un problème très difficile et dans tous les cas très coûteux.

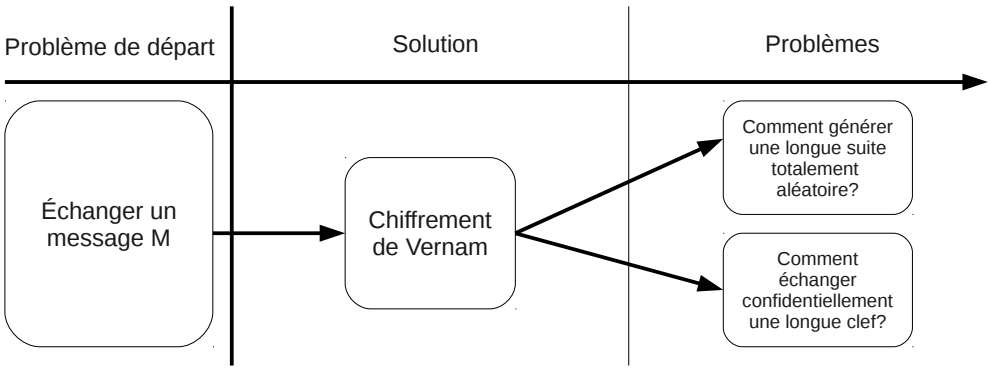


FIGURE 1 – *Paradoxe du chiffrement de Vernam.*

C'est en renonçant à un chiffrement parfait que furent introduits les chiffrements par bloc et par flot.

Le chiffrement par flot, héritier direct du chiffrement de Vernam, génère à partir d'une clef k de taille petite et fixée, et de manière déterministe, le message K de taille N, à l'apparence aléatoire. La suite de l'algorithme est la même que dans le cas du chiffrement de Vernam où $C = M * K$.

Par ailleurs, nous allons voir que les cryptosystèmes par bloc se sont détachés de ce paradigme.

3. Un scénario dans lequel les deux protagonistes échangent plusieurs clefs à l'avance lors d'une éventuelle rencontre ou par le biais d'une valise diplomatique est bien sûr envisageable. Ceci reste coûteux et restreint le champ d'application du chiffrement. Néanmoins, cette méthode a été et reste encore utilisée par les États.

1.2 Confusion et diffusion

Un des problèmes inhérents à la cryptologie est celui de quantifier la sécurité des algorithmes de chiffrement. Shannon fut le premier à s'y attaquer. Ses recherches aboutirent à un résultat fondamental. Ce dernier stipule que l'accumulation de plusieurs couches de confusion et de diffusion devrait offrir un bon degré de sécurité.

1.2.1 Confusion

Dans les chiffrements modernes, la confusion peut être assimilée à la substitution (par exemple la SBox du DES, le SubBytes de l'AES...), son but est essentiellement d'empêcher toute corrélation entre le message et son chiffré à l'issue d'une étape de chiffrement.

1.2.2 Diffusion

Son but est d'assurer un effet avalanche dans le corps du chiffrement. Les permutations sont généralement considérées comme un bon exemple de diffusion. Bien que négligée pendant longtemps, la découverte des méthodes cryptanalytiques différentielles [3] et linéaires[4] l'ont remise à l'ordre du jour. On sait maintenant qu'une mauvaise permutation peut réduire le nombre de Sbox actives [4] et rendre ainsi l'algorithme plus vulnérable aux attaques statistiques.

1.2.3 Combinaison de confusion et diffusion

Bien qu'énoncée à la volée par Shannon, cette idée s'est révélée très fructueuse et l'on constate que désormais, tous les chiffrements sont construits de manière itérative. On décrit une couche de confusion et diffusion, on associe un nombre de tours, disons T, à notre algorithme, et on obtient un chiffrement où l'entrée du tour k n'est autre que la sortie du tour $k-1$. A l'issue du tour $k = T$, le résultat de cette couche de confusion et de diffusion est le chiffré C.
Plusieurs schéma de description de tours existent aujourd'hui. Ci-dessous, nous illustrons deux des schémas les plus connus.

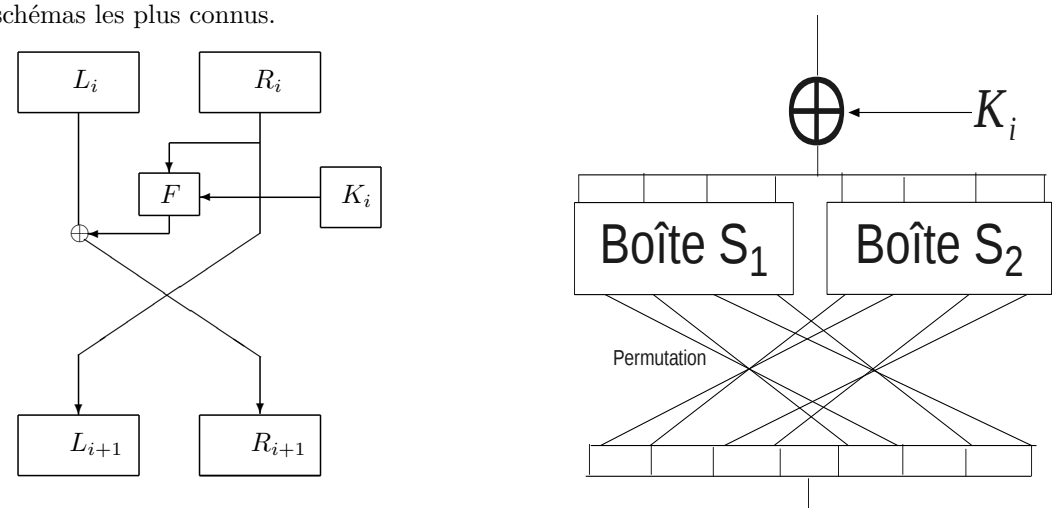


FIGURE 2 – *Schéma de Feistel (à gauche) et schéma SPN (à droite).*

4. La notion de Sbox active est trop complexe pour être décrite ici. Le document [3] en donne une explication pertinente.

Dans un schéma de Feistel, l'entrée du tour i est découpée en deux parties, L_i et R_i. L_i passe ensuite par une fonction de confusion F qui dépend de R_i. R_i reste intacte pour garantir la réversibilité.

$$\begin{array}{c} \textit{Chiffrement} \\ \left\{ \begin{array}{l} L_{i+1} = R_i \\ R_{i+1} = L_i \oplus F_{K_i}(R_i) \end{array} \right. \end{array} \qquad \begin{array}{c} \textit{Déchiffrement} \\ \left\{ \begin{array}{l} L_i = R_{i+1} \oplus F_{K_i}(L_{i+1}) \\ R_i = L_{i+1} \end{array} \right. \end{array}$$

Le schéma SPN (**S**ubstitution **P**ermutation **N**etwork) est moins compliqué. Le message entier passe par une étape de substitution (représentée ici par les Boîtes S). Une permutation achève ensuite le tour. Pour déchiffrer, il suffit d'inverser les étapes : commencer par appliquer la permutation inverse, puis la substitution réciproque.

1.3 Aspect mathématique

Un chiffrement par bloc, comme son nom l'indique, agit sur un bloc de donnée de taille fixe. On peut donc, représenter la phase de chiffrement d'un tel algorithme par une fonction E, qui prend comme paramètres : un bloc de donnée P de taille n et une clef K de taille k. Elle renvoie une donnée C de taille n. On définit de même la fonction que l'on note D et qui a le même ensemble de définition que E.
Ainsi :

$$\begin{array}{rccc} E: & \mathbb{F}_q^n \times \mathbb{F}_q^k & \to & \mathbb{F}_q^n \\ & (P,K) & \mapsto & C \end{array} \qquad \begin{array}{rccc} D: & \mathbb{F}_q^n \times \mathbb{F}_q^k & \to & \mathbb{F}_q^n \\ & (C,K) & \mapsto & P \end{array}$$

avec

$$E(D(C,K),K) = D(E(C,K),K) = C$$

Supposons maintenant que la clef K est fixée. C'est le cas par défaut dans beaucoup de chiffrements orientés Hardware comme Ktantan qu'on verra plus loin. On peut alors simplifier cette écriture en définissant deux nouvelles fonctions E_K et D_K telles que :

$$\begin{array}{rccc} E_K: & \mathbb{F}_q^n & \to & \mathbb{F}_q^n \\ & P & \mapsto & C = E(P,K) \end{array} \qquad \begin{array}{rccc} D_K: & \mathbb{F}_q^n & \to & \mathbb{F}_q^n \\ & C & \mapsto & P = D(C,K) \end{array}$$

et :

$$E_K(D_K(C)) = D_K(E_K(C)) = C$$

On voit ainsi que pour toute clef K, E_K est une bijection de $\mathbb{F}_q^n$ sur lui même, et que D_K est sa réciproque. On note désormais :

$$E_K^{-1} = D_K \quad et \quad Id = E_K \circ D_K$$

Les fonctions E et D, conformément au principe de Kerckhoffs[5] sont des fonctions dont la description est rendue publique, contrairement aux fonctions E_K et D_K auxquelles seule la connaissance de K donne accès. Le but de toute attaque par force brute est donc de retrouver D_K sans connaître K. Le principe est évident : essayer toutes les clefs possibles.

5. Axiome de Kerckhoffs : « *La description du cryptosystème doit être publique, la clef doit être la seule information confidentielle.* »
Shannon le reformula ainsi : « *L'adversaire connaît le système.* »

Algorithme 1 Attaque par force brute[6]

Entrées: E, M, C *et* q^k(Le nombre de clefs possibles)
Sortie: K
  $j \leftarrow 1$
  $K \leftarrow K_1$
  Tantque $[(E(M,K) \neq C)$ et $(j < q^k)]$ **Faire**
    $j \leftarrow j+1$
    $K \leftarrow K_j$ $(tq \quad \forall i < j, \quad K_j \neq K_i)$
  Fin Tantque
  Si $(j \leq q^k)$ **Alors**
    Retourner K
  Sinon
    Print *C n'est pas un chiffré valide de* M.
  FinSi

La complexité d'une telle attaque est de l'ordre de q^k en temps (q étant le cardinal du corps auquel appartient K, et k sa longueur). Pour donner un ordre d'idée, on considère aujourd'hui que 2^{80} est une barrière de complexité sûre d'un point de vue cryptographique. A cause de la facilité de mise en œuvre de cette attaque et de son caractère universel, on ne dira qu'un cryptosystème est potentiellement sûr que si le coût d'une recherche exhaustive est supérieur à 2^{80}. De même, on dira qu'une attaque casse un cryptosystème si la complexité de sa mise en œuvre est en dessous de la complexité d'une attaque par force brute.

2 Katan/Ktantan, un exemple de chiffrement par bloc

2.1 Contexte de Katan/Ktantan

Katan et Ktantan (qui signifient rapide et très rapide en hébreu) forment une famille de chiffrements proposée en 2009 par Christophe De Cannière, Orr Dunkelman et Miroslav Knežević. Ce sont des chiffrements destinés aux environnements restreints tels que les cartes à puce et les capteurs. Plusieurs contraintes sont imposées par de tels environnements. On peut citer par exemple le manque de mémoire, l'autonomie énergétique réduite et l'impossibilité d'effectuer des opérations complexes. On s'attend ainsi à ce que Katan/Ktantan soient rapides, peu « gourmands » en mémoire, et enfin, à ce que leurs opérations intermédiaires ne soient pas très complexes.
En effet, les six chiffrements que l'on va présenter sont divisés en deux familles. Katan est composée de trois chiffrements par bloc qu'on notera Katan_n avec $n \in \{32, 48, 64\}$, où n est la taille du bloc. Ktantan est composée de trois cryptosystèmes différents mais de même taille que Katan. Nous les noterons, de même, Ktantan_n avec $n \in \{32, 48, 64\}$. La seule différence entre Katan_n et Ktantan_n est la méthode utilisée pour dériver les sous-clefs à partir de la clef maître.

En effet, grand nombre de puce RFID ont une durée de vie limitée. La clef de chiffrement peut donc être codée en dur (elle ne peut être modifiée, une fois fixée). Ktantan, dont l'algorithme

6. Bien que considérée ici dans un scénario à clair connu, l'attaque par force brute est aussi efficace dans un contexte où seul le chiffré est connu. L'exemple le plus fréquent est celui où le clair est constitué de codes ASCII : on déchiffre avec toutes les clefs possibles jusqu'à tomber sur un message qui correspond à un code ASCII correct.

Chiffrement	Taille du bloc	Longueur de clef	Structure	Nombre de tours
Katan_32 Ktantan_32	32	80	SPN	254
Katan_48 Ktantan_48	48	80	SPN	254
Katan_64 Ktantan_64	64	80	SPN	254

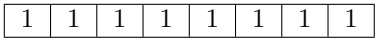
Tableau 4 – *Tableau récapitulatif de la famille Katan/Ktantan*

de cadencement de clef est plus compact (plus sécurisé que Katan) est destinée à ce type de hardware.

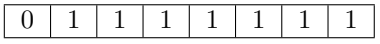
2.2 Compteur de tours

Puisque 254 tours sont nécessaires à tous les chiffrements que l'on va présenter, la question se pose quant au choix d'un compteur de 8-bit. Pour plusieurs raisons, les auteurs ont préféré utiliser un registre à décalage à rétroaction linéaire, on l'appellera LFSR/Counter. Les LFSR sont couramment utilisés en cryptographie pour générer des suites de nombres pseudo-aléatoires. En effet, on sait que la période d'un registre à décalage binaire, quand le polynôme P de rétroaction est primitif, est de 2^d-1 où d est le degré de P. L'idée sélectionnée par les auteurs est la suivante :

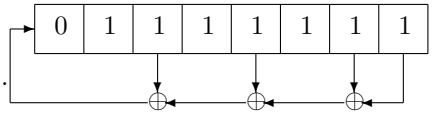
- Initialiser les 8 bits du LFSR/Counter à 1.



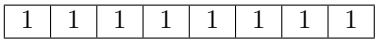
- Son polynôme de rétroaction $P = x^8 + x^7 + x^5 + x^3 + 1$ est primitif. Sa période vaut $2^8 - 1 = 255$.
- Effectuer un seul décalage.



- Commencer le chiffrement.
  Décaler le LFSR/Counter à chaque tour de chiffrement.

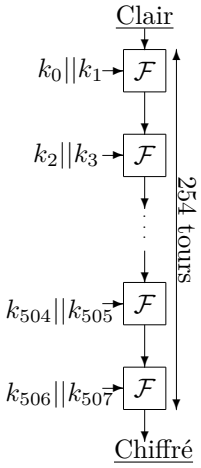


- Arrêter le chiffrement quand les 8 bits du LFSR/Counter retrouvent l'état initial.



2.3 Algorithme de cadencement de clef, point de divergence entre Katan et Ktantan

Katan_n et Ktantan_n enchaînent 254 tours avant de produire un chiffré.
Chaque tour nécessitant deux clefs d'un bit chacune, il faut à priori, générer $2 \times 254 = 508$-bits à partir de la clef maître. La clef maître dans Katan_n et Ktantan_n, rappelons-le est de 80-bits.



2.3.1 Génération de clefs dans Katan

L'alrorithme de cadencement de clef, pour Katan_32 , Katan_48, et Katan_64 est assez simple. En effet c'est un registre à décalage à rétroaction linéaire (On l'appellera LFSR/Clef). Le polynôme de rétroaction est le suivant :

$$x^{80} + x^{61} + x^{50} + x^{13} + 1$$

Il est primitif, de degré 80 et a un poids de Hamming de 5. La clef de 80 bits est chargée dans ce registre (le bit de poids faible de la clef, k_0 est chargé à la position 0).

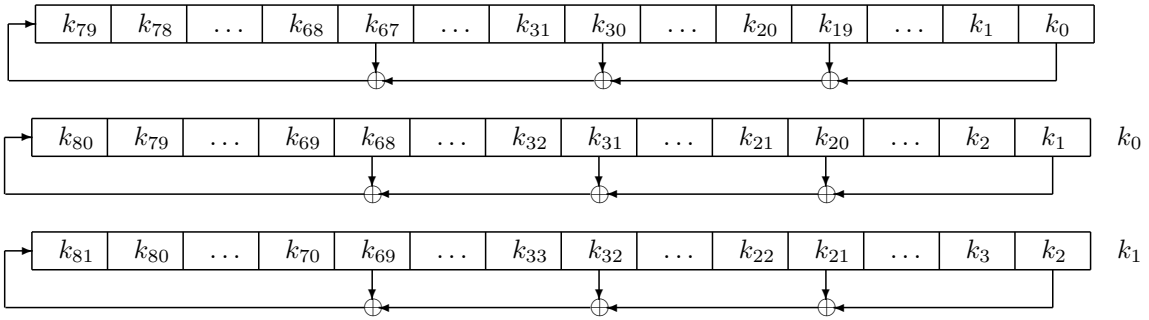


FIGURE 3 – *Illustration de l'état du LFSR/Clef au bout d'un tour de Katan_n*

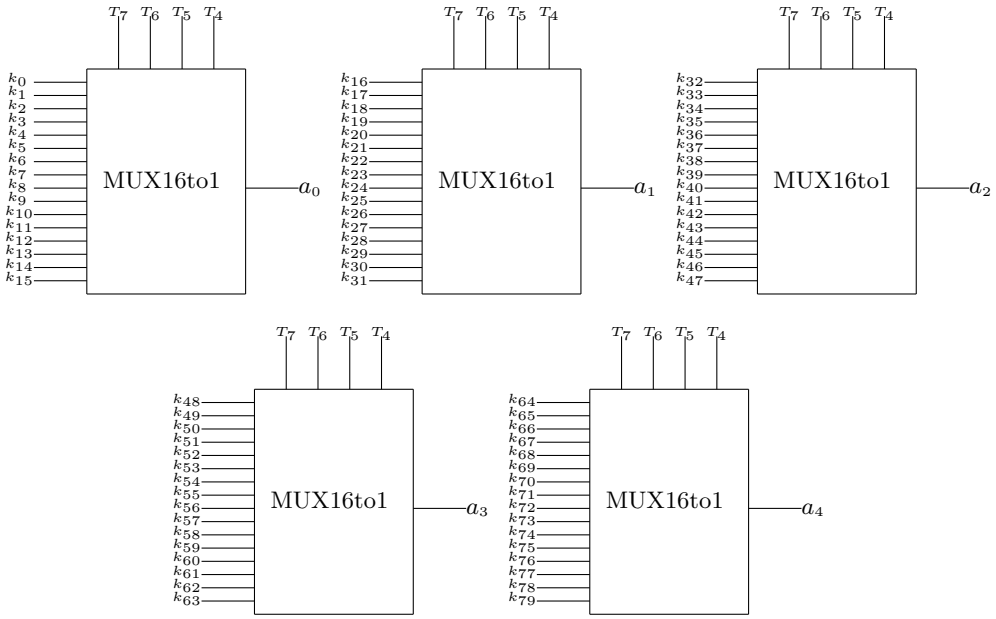
En effet, si l'on note K la clef maître :

$$k_i = \begin{cases} K_i & \text{si } i = 0...79 \\ k_{i-80} \oplus k_{i-61} \oplus k_{i-50} \oplus k_{i-13} & \text{sinon} \end{cases}$$

Chaque tour nécessite deux clefs $k_a||k_b$, et ces clefs sont donc $k_{2i}||k_{2i+1}$. Au premier tour qui est le tour 0, on a $k_a||k_b = k_0||k_1$, au deuxième $k_a||k_b = k_2||k_3$ et ainsi de suite.

2.3.2 Génération de clefs dans Ktantan

L'utilisation d'un registre à décalage n'offre pas une sécurité maximale à la famille Ktantan, car rappelons-le, la clef maître est codée en dur. L'idée retenue est donc de combiner les bits produits par le LFSR/Counter avec ceux de la clef maître K. En effet, cette méthode fait appel à deux types de multiplexeur (nommés MUX16to1 et MUX4to1). La clef principale est tous d'abord découpée en 5 mots de 16 bits chacun : $K = w_4||w_3||w_2||w_1||w_0$ avec $|w_i| = 16$ bits (le bit de poids le plus fort de w_4 étant le bit de poids le plus fort de K).
A chaque tour, on note T la séquence du LFSR/Counter (T_7 est le bit de poids le plus fort, T_0 le bit de poids le plus faible) et on calcule la liste des a_i où : $a_i = \text{MUX16to1}(w_i, T_7T_6T_5T_4)$.



Finalement, les deux clefs de ce tour sont :

$$k_a = \overline{T_3} \cdot \overline{T_2} \cdot (a_0) \oplus (T_3 \vee T2) \cdot \text{MUX4to1}(a_4a_3a_2a_1, T_1T_0),$$
$$k_b = \overline{T_3} \cdot T_2 \cdot (a_4) \oplus (T_3 \vee \overline{T2}) \cdot \text{MUX4to1}(a_3a_2a_1a_0, \overline{T_1T_0}).$$

FIGURE 4 – MUX4to1$(a_4a_3a_2a_1, T_1T_0)$ et MUX4to1$(a_3a_2a_1a_0, \overline{T_1T_0})$

2.4 Algorithme de chiffrement, point commun entre Katan et Ktantan

On se propose de commencer par décrire le principe de chiffrement pour Katan_32 avant d'expliciter les différences avec les cas Katan_48 et Katan_64. Il est inutile de faire de même pour la catégorie Ktantan puisque le principe de chiffrement de Ktantan_n est similaire à celui de Katan_n.
Le message clair (noté P) dans Katan_32 est de 32 bits ; on commencera par le charger dans deux registres L_1 et L_2, de taille respective 13 et 19. Le bit de poids le plus faible P_0 est chargé à la position 0 de L_2 , tandis que le bit de poids le plus fort P_{31} est chargé à l'index 12 de L_1.

Tableau 5 – L_1

P_{31}	P_{30}	P_{29}	P_{28}	P_{27}	P_{26}	P_{25}	P_{24}	P_{23}	P_{22}	P_{21}	P_{20}	P_{19}

Tableau 6 – L_2

P_{18}	P_{17}	P_{16}	P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

Au début de chaque tour, tous les bits sont décalés à gauche (bit i à la position $i+1$), ensuite les deux MSB (bits de poids le plus fort) de $L1$ et $L2$ qui sortent du registre, servent d'argument à deux fonctions, f_a et f_b. Pour achever le tour, le résultat de f_a est placé dans le LSB de L_2 (bit de poids le plus faible) et celui de f_b dans le LSB de L_1.
Décrivons, maintenant les deux fonctions non-linéaires f_a et f_b :

$$f_a : L_1 \longrightarrow \mathbb{F}_2 \qquad \text{et} \qquad f_b : L_2 \longrightarrow \mathbb{F}_2$$

$$f_a(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] \cdot L_1[x_4]) \oplus (L_1[x_5] \cdot IR) \oplus k_a$$
$$f_b(L_2) = L_2[y_1] \oplus L_2[y_2] \oplus (L_2[y_3] \cdot L_2[y_4]) \oplus (L_2[y_5] \cdot L_2[y_6]) \oplus k_b$$

où IR (*Irregular update rule*) est le MSB du LFSR/Counter et $k_a||k_b = k_{2i}||k_{2i+1}$ sont les deux clefs de tour. Quant aux indices x_i avec $i \in \{1,2,3,4,5\}$ et y_j avec $j \in \{1,2,3,4,5,6\}$, ils sont donnés dans le Tableau 7.

Chiffrement	$\lvert L_1\rvert$	$\lvert L_2\rvert$	x_1	x_2	x_3	x_4	x_5
Katan_32/Ktantan_32	13	19	12	7	8	5	3
Katan_48/Ktantan_48	19	29	18	12	15	7	6
Katan_64/Ktantan_64	25	39	24	15	20	11	9

Chiffrement	y_1	y_2	y_3	y_4	y_5	y_6
Katan_32/Ktantan_32	18	7	12	10	8	3
Katan_48/Ktantan_48	28	19	21	13	15	6
Katan_64/Ktantan_64	38	25	33	21	14	9

Tableau 7 – *Indice des paramètres de f_a et f_b*

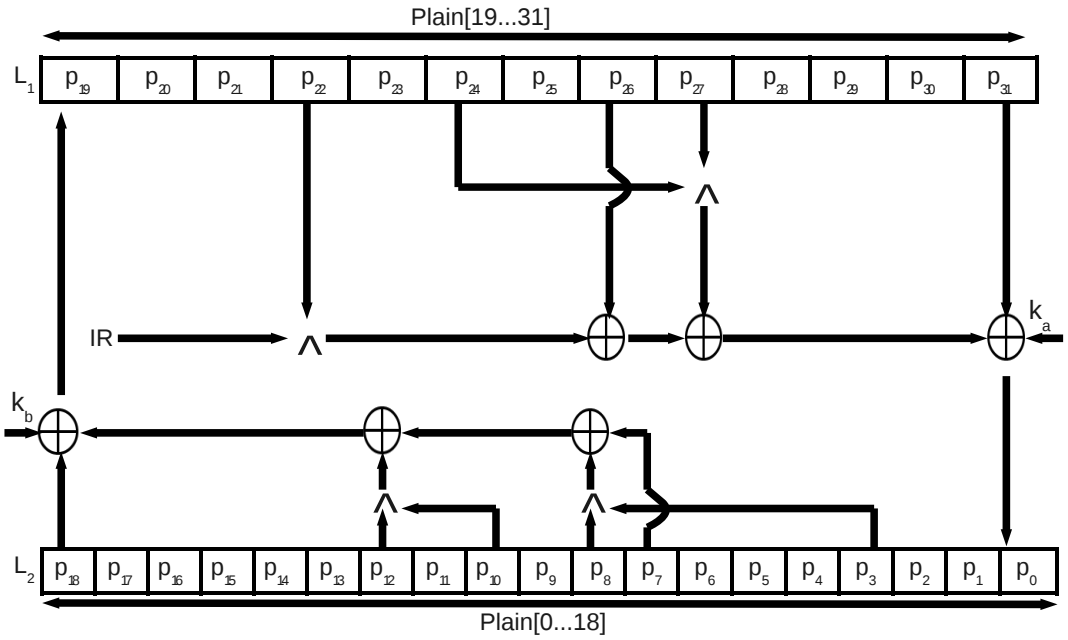


FIGURE 5 – *Schéma d'un tour de Katan_n.*

On peut ainsi remarquer que selon le tour f_a peut prendre deux formes :

$$f'_a(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] \cdot L_1[x_4]) \oplus k_a$$
$$\text{ou} \quad f''_a(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] \cdot L_1[x_4]) \oplus L_1[x_5] \oplus k_a$$

selon que IR est utilisé ou non.

Décrivons maintenant les 4 différences entre Katan_32 et Katan_n où $n \in \{48, 64\}$:
- la taille du bloc.
- la taille des registres L_1 et L_2.
- les indices des bits qui servent de paramètres à f_a et f_b.
- le nombre de fois où f_a et f_b sont appliquées à chaque tour (on le notera nf).

Les 3 premières différences sont explicitées dans la Tableau 7.

Dans le Tableau 8, figure le nombre de fois où les fonctions non-linéaires sont appliquées à chaque tour. En effet, pour Katan_48, f_a et f_b sont appliquées une fois avec les clefs de tour $k_a||k_b$, puis les registres sont décalés, ensuite f_a et f_b sont appliquées une seconde fois avec ces clefs, enfin, les registres sont décalées une dernière fois pour conclure le tour. De même, pour Katan_64 où ce processus est exécuté 3 fois *i.e.* f_a et f_b sont appliquées à 3 reprises, les registres étant décalés à chaque fois.

Chiffrement	nf
Katan_32/Ktantan_32	1
Katan_48/Ktantan_48	2
Katan_64/Ktantan_64	3

Tableau 8 – *Nombre de fois où f_a et f_b sont appliquées à chaque tour.*

2.5 Origine et intérêt de Katan/Ktantan

2.5.1 Origine de Katan/Ktantan

Un des concepteur de Katan/Ktantan, Christophe De Cannière, est l'auteur de Trivium[5], un chiffrement par flot qui a été intensément étudié ces dernières années [7]. En effet, De Cannière et *al.* ont réussi à démontrer à travers Trivium qu'un chiffrement par flot pouvait être simplifié considérablement. Publié depuis 2003, ce cryptosystème s'est révélé résistant à un nombre impressionnant d'attaques (attaque cube, attaque algébrique...). On peut donc légitimement penser que Katan/Ktantan est plus ou moins une adaptation au chiffrement par bloc de Bivium (une variante de Trivium réduite à 2 registres). Néanmoins, une différence existe au niveau des tailles des registres. En effet, Trivium maintient un état interne assez large (288 bits) alors que Katan_n a un état interne de taille n ($n \in \{32, 48, 64\}$). La nature des chiffrements par flot justifie cette différence : Trivium révèle un bit de suite chiffrante à chaque tour, ce qui constitue une information considérable si l'état interne est de taille réduite. En contrepartie, un chiffrement par bloc ne révèle aucune information avant de produire un chiffré final. Préserver un état interne large est donc une précaution qui n'est pas nécessaire dans ce cas.

2.5.2 Intérêt de Katan/Ktantan

La minimisation de la quantité de ressources utilisée est l'objectif principal de Katan/Ktantan. Les auteurs prétendent d'ailleurs que cet objectif est atteint et que Ktantan_48 est aujourd'hui un des chiffrements les plus sobre dans le domaine du hardware. Bien que l'on n'ait pas implémenté les chiffrements avec un langage de description de matériel, on peut néanmoins s'étonner de la simplicité des circuits nécessaires (2 registres à décalage, 2 multiplexeurs...).
Aujourd'hui, on retrouve de plus en plus de RFID dans divers secteurs, avec un besoin croissant de sécurité. Afin de répondre à ces demandes, plusieurs chiffrements ont été mis au point, dont plusieurs chiffrements par bloc. Comme nous pouvons le voir dans le Tableau 9, Katan et Ktantan font partie des meilleurs candidats sur le marché des environnements restreints.

7. Ce chiffrement a fait l'objet d'un PSTL en 2009.
8. Cette implémentation est une version légère de PRESENT-80. Elle utilise moins de mémoire mais elle est beaucoup plus lente.

Chiffrement	Taille de clef	Taille du bloc	Mémoire utilisée (GE)	Vitesse (Kb/s)
AES-128	128	128	3100	0.08
DES	56	64	2309	44.4
DESL	56	64	1848	44.4
PRESENT-80	80	64	1570	200
PRESENT-80 [8]	80	64	1000	11.4
Katan_32	80	32	802	12.5
Katan_48	80	48	927	18.8
Katan_64	80	64	1054	15.1
Ktantan_32	80	32	462	12.5
Ktantan_48	80	48	588	18.8
Ktantan_64	80	64	688	15.1

Tableau 9 – *Comparaison de Katan/Ktantan avec d'autres chiffrements par bloc.*

« *On s'endort cryptographe et on se réveille cryptanalyste.* »
(Marc Girault)

3 La cryptanalyse algébrique :

La cryptanalyse est une discipline qui évalue la sécurité des cryptosystèmes. Elle permet parfois de déchiffrer des messages sans passer par la clef. En effet, si certains cherchent à partager des informations de manière confidentielle, d'autres cherchent à intercepter ces informations. Ne disposant pas de la clef secrète, ils mettent en place des attaques pour retrouver les informations cryptées , et si possible, reconstituer la clef au fur et à mesure.

Il existe plusieurs classes d'attaques dont :

- Les attaques à chiffré connu.
- Les attaques à chiffré choisi.
- Les attaques à clair connu.
- Les attaques à clair choisi.
- Les attaques à chiffré adaptative.

C'est souvent le contexte et l'objectif de l'attaque qui déterminent le scénario à adapter.

3.1 Introduction aux attaques algébriques

> « *Breaking a good cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type.* »(Shannon [1])

La résolution de la plupart des problèmes issus de la modélisation, on le sait aujourd'hui, peut se ramener dans un environnement discret à la résolution d'un système algébrique. Ainsi, une attaque réussie consisterait à remonter d'un chiffrement à son système polynomial et à le résoudre. Cette approche relève de la cryptanalyse algébrique.

On remarque d'abord que ce type d'attaque s'organise autour de deux étapes :

1. Modélisation du problème : consiste à retrouver le système polynomial.
2. Résolution du problème : trouver les solutions du système polynomial.

La première étape est généralement plus accessible : en effet un procédé consiste à exprimer les sorties du cryptosystème en fonction de ses entrées et de la clef. Bien sûr, rien n'empêche de se ramener à travailler sur un tour et d'introduire des variables intermédiaires.
On remarquera néanmoins que certaines primitives cryptographiques (Katan/Ktantan en fait partie) se prêtent plus facilement que d'autres à cet exercice. Parmi les chiffrements qui s'y prêtent mal, figurent ceux qui font appel à une ou plusieurs Sbox. L'interpolation des Sbox une par une est nécessaire dans ce cas de figure.
Rappelons d'autre part que la résolution d'un système polynomial multivarié reste un problème NP-complet largement étudié. D'ailleurs, plusieurs algorithmes et heuristiques y sont associés. On peut citer l'algorithme de Buchberger [6], F4 [7] , F5[8], ElimLin [9], la famille XL[10], sans oublier les SAT-solvers[11] qui procèdent par force brute améliorée sur $\mathbb{F}_2$.
Résoudre le système est équivalent à trouver une information secrète. En fonction de la modélisation choisie et du scénario de l'attaque, cette information représente soit la clef, soit le message initial.
Pour ce projet, nous avons choisi de monter notre attaque sur Katan_32 et Ktantan_32. Nous commencerons par modéliser Katan_32, puis on utilisera les bases de Gröbner pour essayer de résoudre le système obtenu. L'algorithme utilisé est F4 de Faugère, implémenté dans Magma.
Vu la ressemblance des deux chiffrements, on ne s'attardera pas sur la modélisation de Ktantan_32. En conclusion, nous donnerons les résultats de nos expériences sur ces deux cryptosystèmes.

3.2 Attaque algébrique sur Katan_32

Les Sbox de Katan/Ktantan sont les fonctions non-linéaires f_a et f_b, et les permutations sont les décalages des registres vers la gauche. Pour mieux discerner ces deux composantes de confusion et de diffusion, nous avons fait le choix de représenter, par la Figure 6, Katan_32 sous la forme de schéma SPN (Réseau de substitutions et permutations). Cette forme de représentation a deux avantages :

1. Elle facilite la comparaison de Katan_32 avec les algorithmes de chiffrements les plus répandus. Le schéma SPN est en effet une des deux méthodes de construction les plus utilisées.
2. Elle montre de façon explicite le caractère itératif de l'algorithme (Nombre de tours, Particularités de chaque tour, ...).

3.2.1 Modélisation de Katan_32

3.2.1.1 Notation

Désormais nous désignerons les variables du texte clair par $x_{31}x_{30}\cdots x_1x_0$. Le but de ce changement est de rester cohérent avec notre implémentation Magma. De même, on désignera par $k_{79}k_{78}\cdots k_1k_0$ les variables de la clef maître K. Enfin, on introduira des variables intermédiaires $y_n\cdots y_1y_0$ (on cherchera à établir la valeur de n ultérieurement).

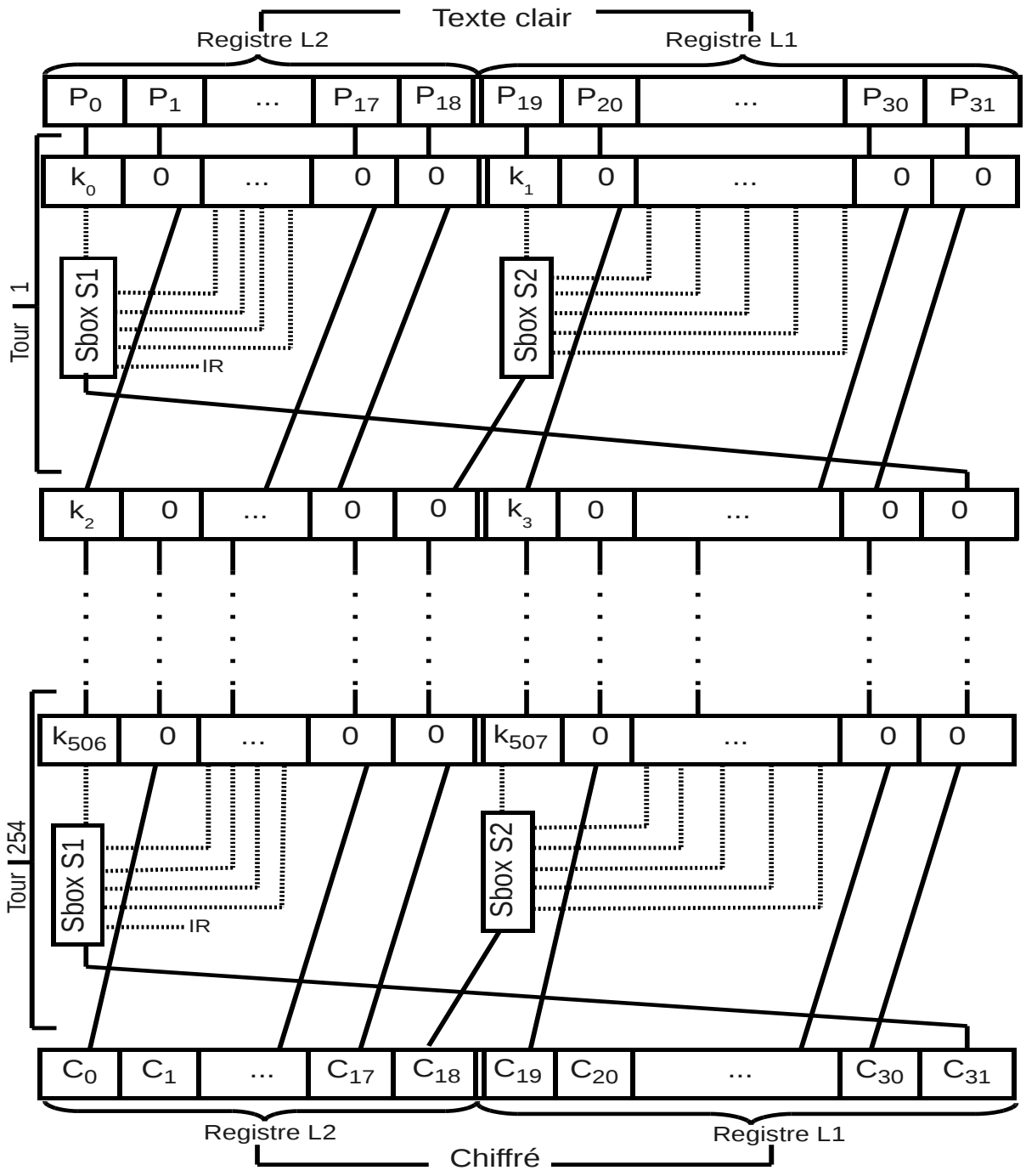


FIGURE 6 – *Katan_32 (ou Ktantan_32) sous forme de schéma SPN.*

3.2.1.2 Nombre de variables

Pour faire des expériences, nous serons ramenés à réduire le nombre de tours, qui vaut 254 initialement. Il est indispensable alors de changer différents paramètres en fonction de ce nouveau nombre de tours.
Par exemple, si l'on se ramène à $N = 100$ tours, il est inutile de prévoir $254 \times 2 = 508$ sous-clefs. Chaque tour nécessitant 2 clefs, $2 \times N$ sous-clefs suffisent.
De même, les variables intermédiaires y_l seront introduites dans les registres, aux indices où figure une équation de degré plus grand que 1. A priori, à l'issue de chaque tour, seulement 2 équations de degré plus grand que 1 seront rajoutées (elles correspondent au résultats de f_a et f_b). Nous en déduisons que $2 \times N$ variables intermédiaires sont nécessaires pour un tel système (N représente toujours le nombre de tours).
On étend alors notre notation pour les clefs qu'on note désormais $k_{(2\times N)-1}k_{(2\times N)-2}\cdots k_1k_0$. Si le nombre de tour N est inférieur à 40, alors $k_{(2\times N)-1}k_{(2\times N)-2}\cdots k_1k_0$ correspondent aux $(2 \times N)^{\text{ème}}$ premières variables de la clef maître K. Si $N > 40$ alors $k_{(2\times N)-1}k_{(2\times N)-2}\cdots k_{81}k_{80}$ correspondent à des sous-clefs générées par l'algorithme de cadencement de clef, tandis que les variables $k_{79}k_{78}\cdots k_1k_0$ correspondent aux variables de la clef maître.
Ces adaptations du système en fonction du nombre de tours correspondent à notre procédure Magma *Fixer_tours* (Algorithme 2).

Algorithme 2 *Fixer_tours*

Entrées:	N	*le nouveau nombre de tours*
	nb_rounds	*le nombre de tours*
	nb_plain	*le nombre de variables du clair*
	nb_keys	*le nombre de variables de clef*
	nb_intemediaires	*le nombre de variables intermédiaires*
	nb_total	*le nombre de variables totales*

Sortie: aucune (change la valeur des variables globales reçues en entrée en fonction de N)
$nb\_rounds \leftarrow N$
$nb\_plain \leftarrow 32$
$nb\_keys \leftarrow 2 \times N$
$nb\_intermediaires \leftarrow 2 \times N$
$nb\_total \leftarrow nb\_plain + nb\_keys + nb\_intermediaires$

3.2.1.3 Initialisation du système

D'abord, on remarque que toutes les variables intervenants dans notre système vérifient une certaine équation. En effet, si $y_l = x_j * k_m$ (avec $* \in \{\oplus, \cdot\}$) alors $y_l \in \mathbb{F}_2$ (car $\oplus$ et $\cdot$ sont les lois de composition interne de $\mathbb{F}_2$ et $(x_j, k_m) \in \mathbb{F}_2^2$). On en déduit que, $\forall i \leq 31, \forall j \leq (2 \times N) - 1, \forall l \leq (2 \times N) - 1$:

1. $x_i^2 - x_i = x_i(x_i - 1) = 0$ (car x_i vaut 0 ou 1).
2. $k_j^2 - k_j = k_j(k_j - 1) = 0$ (car k_j vaut 0 ou 1).
3. $y_l^2 - y_l = y_l(y_l - 1) = 0$ (car y_l vaut 0 ou 1).

Ces équations s'appellent les équations de corps du système. Elles indiquent à notre logiciel de résolution d'équations que les solutions prennent leurs valeurs dans un certains corps ($\mathbb{F}_2$ ici).
La fonction Magma *Initialise_systeme* (Algorithme 3) que nous proposons dans notre implémentation a pour rôle de créer un système vide puis de lui rajouter les équations de corps.

Algorithme 3 *Initialise_systeme*

Entrées: P : *Polynomial Ring (un anneau de polynômes dans lequel nb_total variables ont été initialisées et nommées)*
Sortie: *sys* (*un système d'équations contenant les équation de corps)*
  $sys \leftarrow []$
  $tmp \leftarrow Rank(P)$ {/*Rank renvoie le nombre de variable de P qui est *nb_total**/}
  Pour $i = 1$ à tmp **Faire**
    {/*Ajout de l'équation de corps pour chaque variable*/}
    $sys \leftarrow (sys \quad concaténé \quad à \quad [P.i^2 - P.i])$
    {/*Le second membre ($= 0$) est omis de l'équation*/}
  Fin pour

3.2.1.4 Relations entre clefs

On a vu que le nombre de sous-clefs est inhérent au nombre de tours N. Si ce dernier est inférieur à 40, alors à priori, nous avons besoin de moins de $40 \times 2 = 80$ sous-clefs. Dans ce cas, on ne fait pas appel à l'algorithme de cadencement de clef, les sous-clefs étant les variables de la clef maître.
Par ailleurs, si $(2 \times N) > 80$, il faut générer les $(2 \times N) - 80$ sous-clefs manquantes. Rappelons que le cadencement de clef de la famille Katan se fait à l'aide d'un LFSR, et qu'on a ainsi la relation :

$$\begin{aligned} & k_i = k_{i-80} \oplus k_{i-61} \oplus k_{i-50} \oplus k_{i-13} \text{ pour } i > 80 \\ \Leftrightarrow \quad & k_i \oplus k_{i-80} \oplus k_{i-61} \oplus k_{i-50} \oplus k_{i-13} = 0 \text{ pour } i > 80 \end{aligned}$$

Notre procédure Magma *Key_Generate* 4 simule ainsi le LFSR et rajoute les équations qui relient les sous-clefs entre elles au système.

Algorithme 4 *Key_Generate*

	sys	*le système d'équations*
Entrées:	*nb_rounds*	*le nombre de tours*
	key	*les variables qui symbolisent les sous-clefs dans P*

Sortie: aucune (*le système d'équations sys est mis à jour)*
  {/*si $(2 \times nb\_rounds) \leq 80$, on ne fait rien */}
  Pour i de 81 à $(2 \times nb\_rounds)$ **Faire**
    $sys \leftarrow (sys \quad concaténé \quad à \quad [key[i] + key[i-80] + key[i-61] + key[i-50] + key[i-13]])$
  Fin pour

3.2.1.5 Génération des équations

La dernière étape consiste à générer les équations qui lient l'entrée, la sortie et la clef. Nous supposerons que l'on dispose de la procédure *Chiffrer_tour*, qui effectue un tour de chiffrement Katan_32. Nous allons définir une nouvelle fonction *Substitution*. Étant donné un registre L,

Algorithme 5 *Substitution*

Entrées:	*sys*	*le système d'équations*
	L	*le registre à traiter*
	ind	*l'indice de la prochaine variable intermédiaire disponible*
	P	*Polynomial Ring (l'anneau de polynômes dans lequel les variables ont été initialisées)*
		le système d'équations sys est mis à jour.
Sortie: aucune		*L ne contient plus de variables de degré >1.*
		l'indice ind de la prochaine variable intermédiaire est incrémenté.

$sys \leftarrow (sys \quad concaténé \quad à \quad [P.ind + L[0]])$
$L[0] \leftarrow P.ind$
$ind \leftarrow ind + 1$

cette fonction remplace l'équation quadratique de $L[0]$ par une variable intermédiaire. Le système d'équations est ensuite mis à jour.

La génération des équations se fait en combinant *Chiffrer_tour* et *Substitution*. Le principe est décrit par l'algorithme 6.

Algorithme 6 *Katan_Generate*

Entrées:	*sys*	*le système d'équations*
	key	*les variables représentants les clefs*
	plain	*les variables représentant le texte clair*
	result	*result contient les variables symbolisant le chiffré*
	P	*Polynomial Ring (l'anneau de polynômes dans lequel les variables ont été initialisées)*
		le système d'équations sys est mis à jour.
Sortie: aucune		*L ne contient plus de variables de degré >1.*
		l'indice ind de la prochaine variable intermédiaire est incrémenté.

{/*l'indice de la première variable intermédiaire*/}
$ind \leftarrow nb\_plain + nb\_keys + 1$
Pour i de 0 à 253 **Faire**
Chiffrer_tour
Substitution(sys, L, ind, P)
Fin pour

3.2.2 Résolution du système d'équations polynomiales

Nous allons utiliser la commande *Variety* de Magma. C'est une instruction qui fait appel à F4, un algorithme de résolution de systèmes algébriques multivariés. Étant donné un idéal, F4 utilise l'algèbre linéaire pour calculer sa base de Gröbner. Comme ces notions dépassent largement le cadre de ce PSTL, on les utilisera en boîte noire.

3.2.2.1 Contexte de l'attaque

L'attaque que nous proposons est à clair et chiffré connus. Seule la clef reste à recouvrir. Voici donc le scénario choisi :

1. Choisir un texte clair P (une séquence de 32 bits).

2. Fixer [9] les variables du texte clair dans le système.
3. Choisir une clef K (une séquence de 80 bits).
4. Calculer C le chiffré de P par K.
5. Fixer les variables qui correspondent au résultat dans le système.
6. Lancer la résolution en utilisant la commande $Variety$.

3.2.2.2 Résultats

Les résultats que nous exposerons ont été obtenus avec un Core 2 duo T5800, qui dispose de 4GB de mémoire et dont la fréquence est de 2.00 GHz.
D'abord, on se rend compte immédiatement que pour 254 tours, F4 consomme toute la mémoire disponible et s'arrête rapidement. Notre idée fut donc de réduire le nombre de tours. Ainsi, l'algorithme termine pour un nombre de tour inférieur ou égal à 23 (voir le Tableau 10).
Pour un nombre de tour égal à 24, F4 s'arrête au bout de 11 minutes. L'algorithme consomme 2558.6MB de mémoire et ne peut plus continuer par manque d'espace. Cette méthode atteint

Nombre de tours	Temps de calcul de la variété (en secondes)
10	0.040
15	0.050
20	0.250
21	0.960
22	5.190
23	55.700

Tableau 10 – *Temps de calcul en fonction du nombre de tours.*

vite ses limites.
Une alternative consiste donc à ajouter une dernière étape à notre scénario :
– Fixer certaines variables de la clef.
Nous supposerons donc que N bits de la clef sont connus. Notre implémentation propose plusieurs méthode de « fixation » de clef. Dans la Figure 7 , nous avons fixé les N premiers bits de la clef. Avec seulement 69 variables fixées, F4 s'arrête au bout de 85 minutes à cause d'un manque de mémoire. Néanmoins, retrouver la solution avec seulement 70 bits de clef fixés est un bel exploit. Ce résultat nous garantit une attaque de complexité 2^{70} sur Katan_32 réduit à 100 tours.

9. On fixe x à 0 en ajoutant l'équation x au système.
On fixe x à 1 en ajoutant l'équation $x + 1$ au système.

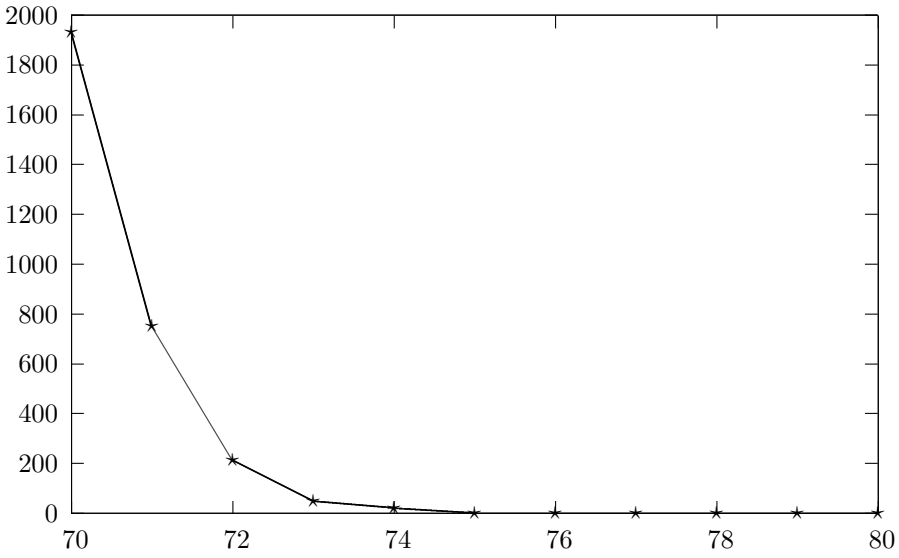


FIGURE 7 – *Katan_32 : Temps de calcul de la variété (en secondes) en fonction de N.*

3.3 Attaque algébrique sur Ktantan_32

3.3.1 Différence avec Katan_32

Seul l'algorithme de cadencement de clef diffère entre ces deux chiffrement. Pour cette raison, nous avons repris toutes les fonctions qui ont servi à modéliser Katan_32. A chaque tour de Ktantan_32, deux bits de la clef maître sont choisis pour servir de clef de tour. Les 80 variables de la clef K suffisent donc à représenter toutes les sous-clefs.

3.3.2 Résultats expérimentaux

Le résultat de l'attaque sur 100 tours (Figure 8) est encore meilleur que celui obtenu pour Katan_32. En fixant seulement les 67 premières variables de la clef, le calcul a duré 3 heures, 55 minutes et 59 secondes avant de renvoyer la variété attendue. Ceci signifie qu'une attaque dont la complexité est de 2^{67} est possible sur Ktantan_32 réduit à 100 tours. La complexité de cette attaque est en dessous de celle d'une recherche exhaustive.
Finalement, on peut affirmer que ces résultats sont assez encourageants. En effet, ses concepteurs ont laissé une très grande marge de sécurité à Katan/Ktantan. La difficulté que rencontrent les attaques vient essentiellement du nombre élevé de tours. D'ailleurs, une idée assez répandue stipule que même un chiffrement faible peut devenir assez redoutable en augmentant son nombre de tour.

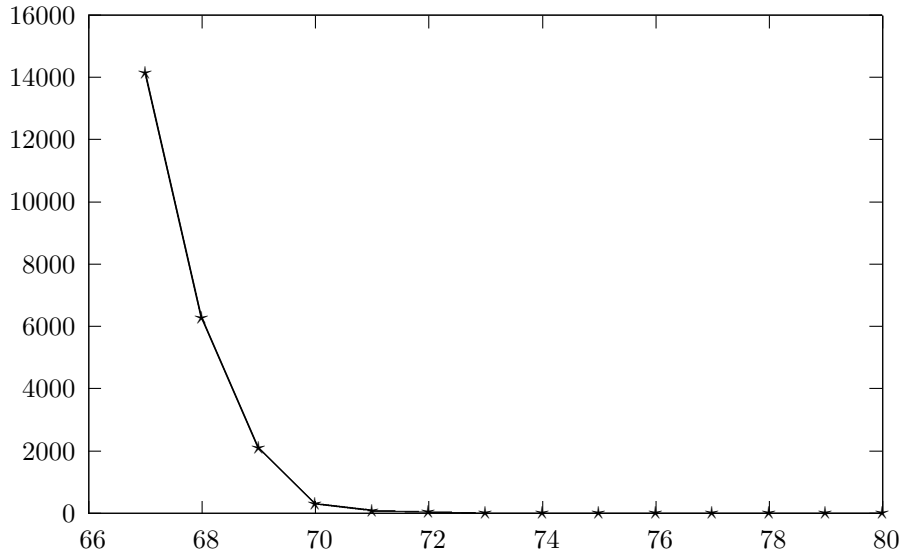


FIGURE 8 – *Ktantan_32 : Temps de calcul de la variété (en secondes) en fonction de N.*

Conclusion

Dans ce projet, on s'est intéressé à l'implémentation d'une famille de chiffrement. Ces chiffrements sont destinés au milieux restreints. Pourtant, leur sécurité semble dépasser celle de certains algorithmes plus classiques. A travers notre attaque algébrique, nous avons pu constater la robustesse de Katan/Ktantan. En effet, ses concepteurs ont prévu une succession de phases défensives contre ce type d'attaque. A ce jour, ces cryptosystèmes restent résistants, et la recherche exhaustive est la meilleur attaque connue. Néanmoins, d'autres pistes restent à explorer. L'utilisation de plusieurs couples clair-chiffré en fait partie. L'association des cryptanalyses différentielles et linéaires à une attaque algébrique est possible également.
Dans tous les cas, nous avons atteint notre objectif de départ, qui était la modélisation d'un algorithme de chiffrement par bloc. Nous l'avons même dépassé en proposant quelques scénarios d'attaques ainsi que leurs résultats.

Références

[1] N. J. A. Sloane and A. D. Wyner, *Claude Elwood Shannon : Collected Papers*. Piscataway, NJ : IEEE Press, 1993.

[2] C. D. Cannière, O. Dunkelman, and M. Knezevic, "Katan and ktantan - a family of small and efficient hardware-oriented block ciphers.," in *CHES* (C. Clavier and K. Gaj, eds.), vol. 5747 of *Lecture Notes in Computer Science*, pp. 272–288, Springer, 2009.

[3] E. Biham and A. Shamir, *Differential cryptanalysis of the data encryption standard*. London, UK : Springer-Verlag, 1993.

[4] M. Matsui, "The first experimental cryptanalysis of the data encryption standard," in *CRYPTO*, pp. 1–11, 1994.

[5] C. D. Cannière and B. Preneel, "Trivium," in *The eSTREAM Finalists*, pp. 244–266, 2008.

[6] B. Buchberger, *An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal*. PhD thesis, 3 2006.

[7] J. C. Faugère, "A new efficient algorithm for computing gröbner bases (F4)," in *Journal of Pure and Applied Algebra*, pp. 75–83, ACM Press, 1999.

[8] J. C. Faugère, "A new efficient algorithm for computing gröbner bases without reduction to zero (F5)," in *ISSAC '02 : Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, (New York, NY, USA), pp. 75–83, ACM, 2002.

[9] N. Courtois and G. V. Bard, "Algebraic cryptanalysis of the data encryption standard," in *IMA Int. Conf.*, pp. 152–169, 2007.

[10] N. Courtois and J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations," in *ASIACRYPT*, pp. 267–287, 2002.

[11] N. Eén and N. Sörensson, "An extensible sat-solver," in *SAT* (E. Giunchiglia and A. Tacchella, eds.), vol. 2919 of *Lecture Notes in Computer Science*, pp. 502–518, Springer, 2003.