Attaques algébriques

Résultats récents

Morgan BARBIER et Pierre-Louis CAYREL

Université de Limoges, XLIM-DMI,
123, Av. Albert Thomas 87060 Limoges Cedex France
barbiermorgan@yahoo.fr
pierre-louis.cayrel@xlim.fr

RÉSUMÉ. *De nos jours, les algorithmes de chiffrement à flot basés sur des registres à décalages à rétroaction linéaire (LFSR) sont très utilisés. Dans (Courtois* et al., *2003), Nicolas Courtois décrit une attaque menée contre les registres à décalage filtrés TOYOCRYPT et LILI* 128 *et dans (Courtois, 2003) il décrit une autre attaque algébrique appelée attaque algébrique rapide. On illustre l'efficacité des attaques algébriques en les appliquant à la cryptanalyse d'un système de chiffrement à flot classique le LFSR. De plus on présente brièvement une optimisation des attaques algébriques sur* E_0, *schéma de chiffrement à flot utilisé dans le Bluetooth. Enfin, on présente une nouvelle approche d'attaque algébrique introduit par Håvard Raddum et Igor Semaev.*

ABSTRACT. *Today, stream ciphers algorithms based on linear feed back shift register (LFSR) are widely used. In (Courtois* et al., *2003), Nicolas Courtois describes an attack against the filtered shift registers TOYOCRYPT and LILI* 128, *and in (Courtois, 2003) he describes another kind of attack called fast algebraic attack. We illustrate the effectiveness of algebraic attacks by applying the cryptanalysis of a classical stream cipher : the LFSR. Then, we briefly present the idea of an optimization of the algebraic attacks against* E_0, *encryption scheme used in the Bluetooth standard. Finally, we present a new approach of the algebraic attacks introduced by Håvard Raddum and Igor Semaev.*

MOTS-CLÉS : *Cryptographie, attaques algébriques, AES, TOYOCRYPT,* E_0

KEYWORDS: *Cryptography, algebraic attacks, AES, TOYOCRYPT,* E_0

1. Introduction

Les attaques algébriques sont des attaques à clair connu qui exploitent des relations algébriques entre les bits du clair, ceux du chiffré et ceux de la clef secrète. La connaissance de plusieurs couples clairs-chiffrés fournit un système d'équations dont les inconnues sont les bits de la clef secrète. Ces derniers peuvent alors être retrouvés en résolvant le système, ce qui est possible s'il est de degré faible, de petite taille ou qu'il possède une structure particulière.

Ces attaques sont utilisées contre, par exemple :

– Stream Ciphers (algorithme de chiffrement à flot) E_0 (Bluetooth, 2001, Armknecht *et al.,* 2008), TOYOCRYPT (Courtois *et al.,* 2003), LILI 128 (Courtois *et al.,* 2003)

– Block Ciphers (algorithme de chiffrement par bloc) A.E.S. (Courtois *et al.,* 2002), Serpent (Courtois *et al.,* 2002)

– Public Key Ciphers (algorithme de chiffrement à clef publique) Matsumoto-Imai (Patarin, 1996), HFE (Kipnis *et al.,* 1999)

On utilise l'idée de ces attaques dans le cadre du schéma de chiffrement à flot. Enfin, on détaille de nouveaux résultats sur les schémas de chiffrement par blocs et E_0.

1.1. *Chiffrement à flot*

Les algorithmes de chiffrement à flot sont utilisés dans le chiffrement symétrique. Le protocole se déroule de la manière suivante :

On note l'expéditeur Alice et le destinataire Bob. Alice souhaite envoyer un message sous forme de bits de texte clair à Bob. On note p_t les bits de clair. Alice et Bob possèdent un secret commun sous la forme d'une clef notée **K** et un générateur aléatoire de bits commun (par exemple un registre à décalage). En utilisant ce registre avec la clé **K** ils obtiennent donc un secret commun noté z_t. Alice fait une addition bit à bit entre son message clair p_t et sa clef z_t et obtient donc un message chiffré noté c_t. Elle envoie ce message à Bob qui est en possession de z_t et de c_t ; en faisant à son tour une somme bit à bit il obtient le message clair d'Alice c'est-à-dire p_t.

Le problème d'une telle méthode est qu'Alice et Bob doivent avoir un secret commun, la clé. Le plus grand problème est la taille de ce secret. On peut convenir que 128 bits de secret sont faciles à partager alors que 100 000 bits ne le sont pas. Après s'être mis d'accord sur leur secret commun de 128 bits, Alice et Bob utilisent un registre à décalage pour créer de nouveaux bits de secret commun. C'est-à-dire qu'ils génèrent une suite de bits pseudo-aléatoire à partir de leurs 128 bits de départ.

1.2. *Chiffrement par bloc*

La principale différence entre le chiffrement par bloc (en anglais block cipher) et le chiffrement par flot vient du découpage des données en blocs de taille généralement fixe (souvent une puissance de deux comprise entre 32 et 512 bits). Les blocs sont ensuite chiffrés les uns après les autres. Il est possible de transformer un chiffrement par bloc en un chiffrement par flot en utilisant un mode d'opération comme ECB (chaque bloc chiffré indépendamment des autres) ou CFB (on chaîne le chiffrement en effectuant un XOR entre les résultats successifs).

Une liste non-exhaustive de chiffrements par bloc :

– DES, l'ancêtre conçu dans les années 70, a été passablement étudié

– AES, le remplaçant du DES

– Blowfish et Twofish, des alternatives à AES

2. Attaques algébriques

2.1. *Chiffrement à flot*

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 algébrique de base. En effet, si la fonction est de petit degré d, chaque bit de la 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.

2.1.1. *L'exemple du LFSR*

On peut exprimer s_t comme une fonction de degré 2 en $(u_0, \ldots, u_4)$ 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 $(u_t)_{t\geq 5}$ produite par le L.F.S.R. vérifie la récurrence $u_t = u_{t-2} + u_{t-5}$ on en déduit qu'à l'instant $t = 1$, on a :

$$s_1 = u_4u_5 + u_1u_2 + u_5. \; s_1 = u_0u_4 + u_3u_4 + u_1u_2 + u_3 + u_0.$$

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 que les algorithmes de bases 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,

il s'agit de la linéarisation. Dans l'exemple, on pose donc $x_0 = u_0, \ldots, x_4 = u_4, x_5 = u_0u_1, x_6 = u_0u_2, \ldots, x_{14} = u_3u_4$. Chaque équation de degré 2 s'écrit donc comme une équation linéaire en $x_0, \ldots, x_{14}$, 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 : $[\sum_{i=1}^{d} \binom{l}{i}]^3$ où d est le degré de la fonction de filtrage f et l la longueur du L.F.S.R. 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, c'est-à-dire quand l dépasse 100.

2.1.2. *Améliorations des algorithmes de recherche d'annulateurs dans le cas de E_0*

Récemment dans (Armknecht *et al.,* 2008), un algorithme amélioré pour déterminer des annulateurs est décrit. L'idée est de déterminer un ensemble appelé support pour lequel on va déduire un système d'équations dépendantes de la clé (secrète). Les auteurs proposent un moyen de décrire cet ensemble (dans le cas des registres à décalages avec mémoire en général et l'applique au cas particulier de E_0), ainsi qu'un moyen de déterminer les annulateurs de cet ensemble. Ils prouvent ainsi la non-existence d'annulateurs de degré 3 en considérant entre 7 et 9 bits consécutifs produits par le registre, le lecteur intéressé peut se référer à (Armknecht *et al.,* 2008).

2.2. *Chiffrement par blocs : attaque récente*

Dans (Raddum *et al.,* 2007), H. Raddum et I. Semaev proposent une nouvelle attaque algébrique. L'idée générale de cette attaque est de mettre en équation toutes les solutions possibles et de retirer au fur et à mesure les solutions qui ne peuvent correspondre. Au final, on se retrouve alors avec toutes les clés qui nous permettent de passer du clair au chiffré. Si on ne trouve pas l'unicité de la solution on peut par exemple ajouter au système un autre couple de clair/chiffré. La nouveauté de cette attaque est introduite dès la mise en équation.

2.2.1. *Mise en équation*

Pour les cryptosystèmes symétriques ce sont les fonctions de chiffrement non linéaires qui font monter le degré des équations, et qui rendent donc difficile une attaque algébrique. Par exemple, pour les cryptosystèmes Data Encryption Standard (D.E.S.), Advanced Encryption Standard (A.E.S.) et PRESENT, ce sont les Sbox qui jouent le rôle de fonction non linéaire.
Les auteurs proposent de poser pour chaque partie non linéaire un système linéaire qui est constitué d'une matrice système et d'une matrice solution (Raddum *et al.,* 2006).

Pour cela, on considère les variables dans $\mathbb{F}_2$ qui représentent tous les bits issus des fonctions non linéaires et de la clé. Dans la figure 1, les "X" représentent tous les

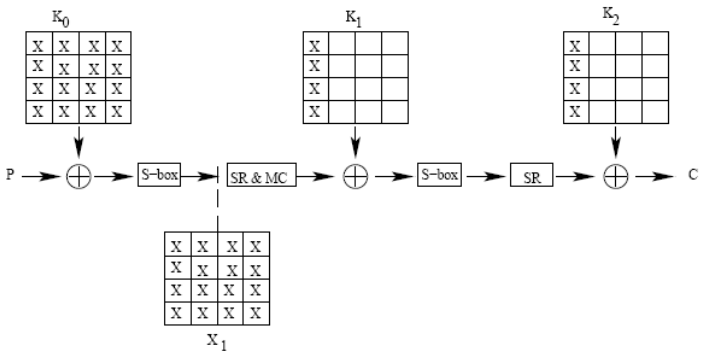

Figure 1. *A.E.S. réduit á deux tours.*

$$\begin{pmatrix} \ell_0 \\ \vdots \\ \ell_7 \\ \ell_8 \\ \vdots \\ \ell_{15} \end{pmatrix} X \stackrel{\circ}{=} \underbrace{\begin{bmatrix} 0 & 1 & \cdots & E & F \\ 0 & 0 & \cdots & F & F \\ 3 & C & \cdots & B & 6 \\ 6 & 7 & \cdots & B & 1 \end{bmatrix}}_{Sbox}$$

Figure 2. *Mise en équation d'une Sbox de l'A.E.S.*

octets qui sont mis en équations.

De la manière que les variables sont définies, chaque variable peut être exprimée en combinaison linéaire. Ces combinaisons linéaires dépendent des variables, des bits du clair et des bits du chiffré. Pour construire la matrice système on exprime pour les premières lignes les relations linéaires qui lient les bits des variables d'entrée de la fonction non linéaire aux autres variables. Pour les dernières lignes de la matrice système, on fait de même pour les bits de sortie.
Ensuite, on construit la matrice solution du système linéaire en mettant en face des équations d'entrées toutes les entrées possibles et en dessous la sortie correspondante. La figure 2 représente un exemple de mise en équation d'une Sbox de l'A.E.S. où les ℓ_i représentent les relations linéaires.

Les auteurs proposent de définir un nouvel objet mathématiques : le symbole représenté par la figure 2 qui est un couple de matrices, dont le premier membre représente le système et le second l'ensemble des solutions possibles.

2.2.2. *Résolution*

Pour résoudre ce système les auteurs définissent des opérations sur les symboles : la mise en accord, et le collage.

2.2.2.1. Mise en accord

La mise en accord de deux symboles $S_i = (A_i, L_i)$ et $S_j = (A_j, L_j)$ est un procédé qui a pour but de rendre consistant le système concaténé.

Définition 2.1 *On dit que le symbole S_i est en accord avec S_j si $\forall a_i \in L_i$, $\exists a_j \in L_j$ tel que*

$$\begin{pmatrix} A_i \\ A_j \end{pmatrix} X = \begin{pmatrix} a_i \\ a_j \end{pmatrix}$$

soit consistant.

Définition 2.2 *On dit que les symboles S_i et S_j sont en accord si S_i est en accord avec S_j et si S_j est en accord avec S_i.*

Pour mettre en accord deux symboles, on cherche à supprimer les solutions qui rendent le système inconsistant

$$\begin{matrix} k_i\{ \\ k_j\{ \end{matrix} \begin{pmatrix} A_i \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ A_j \end{pmatrix} = \begin{pmatrix} a_i \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ a_j \end{pmatrix}. \qquad [1]$$

L'idée principale est de trouver une matrice U telle que

$$U \begin{pmatrix} A_i \\ A_j \end{pmatrix} = \begin{pmatrix} B \\ 0 \end{pmatrix} \} r \ , \qquad [2]$$

où B est une matrice de rang plein. Comme on travaille dans un corps de caractéristique deux, on remarque facilement avec l'égalité [1], pour que les symboles soient en accord il faut que $U \begin{pmatrix} a_i \\ 0 \end{pmatrix}$ et $U \begin{pmatrix} 0 \\ a_j \end{pmatrix}$ soient égaux pour les r dernières coordonnées.

Remarque 2.1 *On remarque que si U n'existe pas, alors $\begin{pmatrix} A_i \\ A_j \end{pmatrix}$ est de rang plein. Donc le système est consistant et les symboles sont en accord.*

Exemple 2.1 On se propose de regarder comme exemple un "toy cipher", représenté par la figure 3. Les opérations appliquées au clair sont les fonctions utilisées dans l'A.E.S.

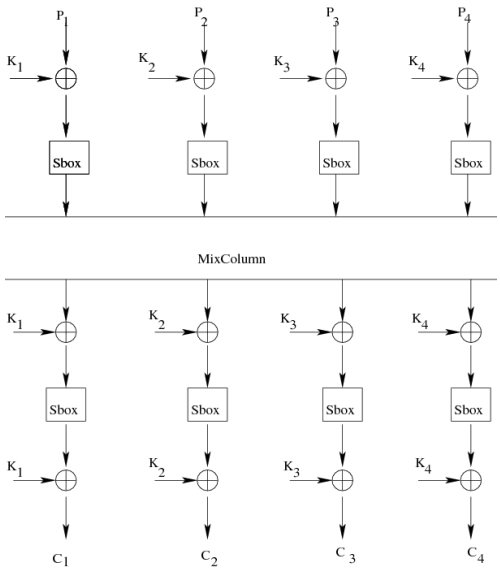

Figure 3. *Toy cipher*

2.2.2.2. Le collage

La mise en équation d'un cryptosystème génère une multitude de symboles. Il se peut que la mise en accord de toute la liste de symboles n'élimine pas toutes les "fausses solutions". C'est dans le but de résoudre ce problème que le collage a été introduit.

Le collage s'effectue sur deux symboles S_i et S_j qui sont en accord. Le but est de créer un nouveau symbole S à partir de S_i et S_j. Les auteurs ont remarqué expérimentalement que le symbole issu du collage S est en général en désaccord avec le reste de la liste. Le collage repose sur le même principe que la mise en accord.

La matrice système de S est la sous matrice de UA de rang plein, en d'autres termes c'est la matrice B dans l'égalité [2]. La matrice solution du symbole est obtenue en additionnant toutes les colonnes de $U \begin{pmatrix} L_i \\ 0 \end{pmatrix}$ et de $U \begin{pmatrix} 0 \\ L_j \end{pmatrix}$ qui ont les mêmes r dernières coordonnées. Il n'est pas difficile de montrer que l'intersection des solutions de S_i et de S_j est égale aux solutions de S.

Le collage permet en général de créer un nouveau symbole qui n'est pas en accord avec les autres symboles. Cependant dans certains cas ce procédé ne permet pas de débloquer la situation. Une solution supplémentaire que proposent H. Raddum et I. Semaev est de deviner un bit.

Notre première implémentation en C de l'algorithme de H. Raddum et I. Semaev trouve toutes les clés en une vingtaine de secondes, en utilisant 270 Mo de mémoire. Tandis que sur la même machine, Magma trouve la solution en un peu moins de 5 heures, en utilisant 7,3 Go de mémoire.

3. Conclusion

On a montré dans cet article le rôle essentiel des attaques algébriques dans la cryptanalyse actuelle. Tant en chiffrement à flot qu'en chiffrement par blocs, les attaques algébriques contribuent fortement à affaiblir la sécurité de ces systèmes. De récents travaux apportent régulièrement des améliorations dans ce sens. On a brièvement présentés ce type d'attaque sur l'exemple du LFSR et E_0 ; et détaillé la mise en place de ces attaques sur les algorithmes de chiffrement par blocs avec la construction de H. Raddum et I. Semaev.

4. Bibliographie

Armknecht F., Cayrel P.-L., Gaborit P., Ruatta O., « Improved algorithm to find equations for algebraic attacks for combiners with memory », *Third International Workshop on Boolean Functions : Cryptography and Applications, Proceedings of BFCA 2007,*, vol. Jean-Francis Michon, Pierre Valarcher, Jean-Baptiste Yunès Eds.,, p. 81-98, 2008.

Bluetooth S., « Specification of the Bluetooth system Version 1.1 1 », 2001.

Courtois N., « Fast Algebraic Attacks on Stream Ciphers with Linear Feedback », *CRYPTO*, vol. 1729, p. 177-194, 2003.

Courtois N., Meier W., « Algebraic Attacks on Stream Ciphers with Linear Feedback », *Eurocrypt*, vol. LNCS 2656, p. 345-359, 2003.

Courtois N., Pieprzyk J., « Cryptanalysis of Block Ciphers with Overdefined Systems of Equations », *Asiacrypt*, vol. LNCS 2501, p. 267-287, 2002.

Kipnis A., Shamir A., « Cryptanalysis of the HFE Public Key Cryptosystem », *Lecture Notes in Computer Science*, vol. LNCS 1666, p. 19-30, 1999.

Patarin J., « Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP) : two families of Asymmetric Algorithms », *Eurocrypt*, vol. Springer Verlag, p. 33-48, 1996.

Raddum H., Semaev I., « New Technique for Solving Sparse Equation Systems », 2006. http ://eprint.iacr.org/.

Raddum H., Semaev I., « Solving MRHS linear equations », *WCC*, Avril, 2007.