

Cryptanalyse — 4TCY902U

Responsable : G. Castagnos

Projet sur la cryptanalyse algébrique de A5/2

À faire en binôme. Vos programmes clairs et bien commentés ainsi qu'un rapport contenant les réponses aux questions (démonstrations, explications du code Sage) sont à rendre avant le vendredi 2 décembre 23:59 par mail à guilhem.castagnos@math.u-bordeaux.fr

Le projet consiste en une cryptanalyse de l'algorithme de chiffrement par flot A5/2 en s'inspirant de la méthode proposée par Barkan, Biham et Keller en 2003¹.

A5/2 est constitué de 4 LFSR, notés $\text{LFSR}_1, \text{LFSR}_2, \text{LFSR}_3$ et LFSR_4 , de longueurs respectives 19, 22, 23 et 17 et de polynômes de rétroactions respectifs $P_1 = x^{19} + x^{18} + x^{17} + x^{14} + 1$, $P_2 = x^{22} + x^{21} + 1$, $P_3 = x^{23} + x^{22} + x^{21} + x^8 + 1$ et $P_4 = x^{17} + x^{12} + 1$. Il utilise une clef $K = (K_0, \dots, K_{63})$ de 64 bits et un IV $= (IV_0, \dots, IV_{21})$ de 22 bits et il produit une suite chiffrante z de longueur fixe $N = 228$ bits. On notera à chaque instant $R_1 = (R_{1,0}, \dots, R_{1,18}), R_2 = (R_{2,0}, \dots, R_{2,21}), R_3 = (R_{3,0}, \dots, R_{3,22})$ et $R_4 = (R_{4,0}, \dots, R_{4,16})$, les registres des 4 LFSR.

La phase d'initialisation, A5/2 – init, est décrite dans l'encadré suivant :

Mettre à zéro les registres R_1, R_2, R_3, R_4

Pour i allant de 0 à 63,

Mettre à jour les $\text{LFSR}_1, \text{LFSR}_2, \text{LFSR}_3, \text{LFSR}_4$ en ignorant leurs sorties

$R_{1,18} = R_{1,18} + K_i, R_{2,21} = R_{2,21} + K_i, R_{3,22} = R_{3,22} + K_i, R_{4,16} = R_{4,16} + K_i$

Pour i allant de 0 à 21,

Mettre à jour les $\text{LFSR}_1, \text{LFSR}_2, \text{LFSR}_3, \text{LFSR}_4$ en ignorant leurs sorties

$R_{1,18} = R_{1,18} + IV_i, R_{2,21} = R_{2,21} + IV_i, R_{3,22} = R_{3,22} + IV_i, R_{4,16} = R_{4,16} + IV_i$

On fixe à 1 certains bits des registres : $R_{1,3} = 1, R_{2,5} = 1, R_{3,4} = 1, R_{4,6} = 1$.

On note $\text{Maj}(a, b, c)$ la fonction majorité. On rappelle que $\text{Maj}(a, b, c) = 0$ si et seulement si la majorité des bits a, b et c vaut 0, par exemple $\text{Maj}(0, 1, 0) = 0$ et $\text{Maj}(1, 1, 0) = 1$.

A5/2 utilise la fonction de mise à jour A5/2 – step décrite dans l'encadré suivant :

¹. *Instant Ciphertext-Only Cryptanalysis of GSM encrypted communication*, Elad Barkan, Eli Biham, Nathan Keller, CRYPTO 2003

Calculer $m = \text{Maj}(R_{4,6}, R_{4,13}, R_{4,9})$
 Si $R_{4,6} = m$, le LFSR₁ est mis à jour en ignorant sa sortie
 Si $R_{4,13} = m$, le LFSR₂ est mis à jour en ignorant sa sortie
 Si $R_{4,9} = m$, le LFSR₃ est mis à jour en ignorant sa sortie
 Le LFSR₄ est mis à jour en ignorant sa sortie
 Calculer $y_1 = R_{1,0} + \text{Maj}(R_{1,3}, R_{1,4} + 1, R_{1,6})$
 Calculer $y_2 = R_{2,0} + \text{Maj}(R_{2,8}, R_{2,5} + 1, R_{2,12})$
 Calculer $y_3 = R_{3,0} + \text{Maj}(R_{3,4}, R_{3,9} + 1, R_{3,6})$
 Le bit de sortie de A5/2 – step est $y_1 + y_2 + y_3$

Après l'exécution phase d'initialisation, A5/2 – init, la production de $N = 228$ bits de suite chiffrante, se fait de la manière suivante :

- Exécuter 99 fois la fonction A5/2 – step en ignorant son bit de sortie
- Exécuter $N = 228$ fois la fonction A5/2 – step, en utilisant ses bits de sortie pour produire la suite chiffrante, z .

I Programmer (avec Sage) le chiffrement A5/2. Il sera utile pour la suite de diviser le code en plusieurs fonctions (initialisation, mise à jour de l'état, production de suite chiffrante). Pour vérifier votre code, on pourra trouver dans

https://www.math.u-bordeaux.fr/~gcastagn/Cryptanalyse/A5_2-test-vector.sage

un exemple de suite chiffrante créée par A5/2.

2 On se place juste après la phase d'initialisation. On suppose connu le registre R_4 du LFSR₄. On pose $R_1 = (x_0, \dots, x_{18})$, $R_2 = (x_{19}, \dots, x_{40})$ et $R_3 = (x_{41}, \dots, x_{63})$ où les x_i sont des inconnues (sauf 3 x_i que l'on sait être égaux à 1). Montrer que durant toutes les étapes de la production de suite chiffrante de A5/2, on peut exprimer les contenus des registres R_1, R_2, R_3 au moyen d'équations linéaires en les x_i .

3 Écrire un code Sage permettant d'exprimer ces équations linéaires. Pour cela, déclarer les inconnues par `BPR = BooleanPolynomialRing(64, 'x')` ; `v = BPR.gens()` et utiliser les matrices de rétroaction des LFSR.

4 Donner la forme algébrique normale de la fonction booléenne Maj. En déduire que les bits de suite chiffrante z peuvent s'exprimer par des équations quadratiques en les x_i .

5 Écrire un code Sage permettant d'exprimer ces N équations quadratiques.

[6] Montrer qu'au plus 655 monômes de degré au plus deux peuvent apparaître dans ces équations (sachant que l'on connaît la valeur de trois x_i). Créer avec Sage la liste M de ces monômes. On note $L = 655$ la longueur de M.

[7] Linéariser ces équations avec Sage. Pour cela, créer une matrice $N \times L$ à coefficients dans \mathbf{F}_2 dont la i -ième ligne contient un 1 à la colonne j si et seulement si le monôme M_j apparaît dans l'équation i . Créer également un vecteur de longueur N qui contient un 1 à la coordonnée i si un 1 apparaît dans l'équation i . On pourra utiliser la méthode `.monomials()` appliquée à une équation pour obtenir la liste des monômes qui la constitue.

[8] On suppose dans les trois questions suivantes que $N = 700$. Récupérer dans le fichier

https://www.math.u-bordeaux.fr/~gcastagn/Cryptanalyse/A5_2-700.sage

une suite chiffrante de 700 bits créée par A5/2 et la valeur du registre R_4 à la fin de la phase d'initialisation. Quel était le contenu des registres R_1, R_2, R_3 ? Pour résoudre un système linéaire, on pourra utiliser la méthode `.solve_right()` appliquée à une matrice.

[9] On cherche maintenant à retrouver la clef secrète. Montrer que lors de l'étape d'initialisation, après la première for, l'état du LFSR₁ est égal à

$$X = \sum_{i=0}^{63} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{63-i} \end{pmatrix}$$

où A_1 est la matrice de rétroaction du LFSR₁. Montrer qu'après le deuxième boucle for, l'état du LFSR₁ est égal à

$$A_1^{22}X + \sum_{i=0}^{21} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21-i} \end{pmatrix}$$

[10] LIV utilisé pour créer la suite chiffrante de 700 bits était nul. Dédurre de la question précédente, avec Sage, les équations exprimant les contenus des registres R_1, R_2, R_3 et R_4 en fin de phase d'initialisation en fonctions des bits (K_0, \dots, K_{63}) . Quel était la clef secrète utilisée pour construire cette suite de 700 bits ?

On revient pour toute la suite au cas $N = 228$ bits. Récupérer dans le fichier

https://www.math.u-bordeaux.fr/~gcastagn/Cryptanalyse/A5_2-3frames.sage

3 suites chiffrantes z_0, z_1 et z_2 chacune de 228 bits et produites avec A5/2 initialisé avec la même clef K et respectivement avec les IV $(0, \dots, 0)$, $(0, \dots, 0, 1)$ et $(0, \dots, 0, 1, 0)$ (Ce cas correspond à l'utilisation d'A5/2 dans le protocole GSM).

I1 On se place après la phase d'initialisation de l'exécution de A5/2 ayant produit la suite z_0 . Comme à la question 2, on suppose connu le registre R_4 du LFSR₄ et on pose $R_1 = (x_0, \dots, x_{18})$, $R_2 = (x_{19}, \dots, x_{40})$ et $R_3 = (x_{41}, \dots, x_{63})$ où les x_i sont des inconnues. Montrer comment déduire de R_1, R_2, R_3 et R_4 , l'état des registres après la phase d'initialisation des exécutions de A5/2 ayant produit les suites z_1 et z_2 .

I2 Déduire de la question précédente et de ce qui a été fait précédemment pour le cas $N = 700$, une attaque permettant de retrouver la clef K utilisée.

I3 Programmer cette attaque (la valeur du registre R_4 après la phase d'initialisation de l'exécution de A5/2 donnant z_0 est donnée dans le fichier contenant z_0, z_1 et z_2).

I4 En déduire une attaque (théorique) permettant de retrouver la clef K sans connaître R_4 . Vu le temps pris par l'attaque de la question I3, combien de temps prendrait l'exécution de cette attaque complète ?

I5 Proposer des changements dans la conception d'A5/2 le mettant à l'abri de cette attaque.