MajecSTIC 2007

Caen

Attaques par collision basées sur la perte d'entropie causée par des fonctions aléatoires

Andrea Röck

INRIA Rocquencourt, Projet CODES

andrea.roeck@inria.fr

Résumé:

Nous considérons le problème de la perte d'entropie dans un chiffrement à flot qui emploie une fonction aléatoire pour mettre à jour son état interne. Pour un nombre limité d'itérations, nous présentons une nouvelle approximation de l'entropie d'état. Ceci améliore la borne supérieure connue donnée par la taille de l'image. Ensuite, nous présentons deux attaques par collision qui sont basées sur la perte d'entropie après plusieurs itérations. Pour les deux attaques, nous fournissons une analyse détaillée de la complexité.

Mot-clés : entropie, fonctions aléatoires, chiffrement à flot.

1 Introduction

Le chiffrement à flot est une méthode classique pour chiffrer de grandes quantités de données. Il fait partie de la famille des chiffrements symétriques, c'est-à-dire qu'il emploie la même clé pour le chiffrement et le déchiffrement. L'idée est que le chiffrement à flot considère le texte clair comme une suite de longueur arbitraire, contrairement au chiffrement par bloc qui traite des blocs de taille constante. Dans la pratique, la taille d'un élément est un bit ou un octet.

Un exemple classique de chiffrement à flot est le masque jetable qui utilise une clé secrète de la même taille que la suite de texte clair. La clé est combinée avec le texte clair élément par élément à l'aide d'une addition modulaire. Si les éléments ont une taille d'un bit, l'addition correspond à un XOR. Shannon a montré dans [Shannon 49] que le masque jetable obtient une sécurité théorique parfaite si la clé correspondante est utilisée seulement une fois et consiste en des éléments choisis de façon totalement aléatoire. Par sécurité théorique parfaite nous voulons dire que le texte chiffré ne donne aucune information sur le texte clair si la clé est inconnue.

Le grand inconvénient du masque jetable est la taille nécessaire de clé secrète. Le plus souvent, pour éviter ce problème, le chiffrement à flot ne combine pas le texte clair directement avec la clé mais avec une suite pseudo–aléatoire appelée suite chiffrante. Une

telle suite est générée à partir d'un clé courte mais il est très difficile en pratique de la distinguer d'une véritable suite aléatoire.

Un modèle pour un tel chiffrement à flot est représenté sur la Fig. 1. Pour générer une suite chiffrante nous commençons à partir d'un état initial et appliquons itérativement la fonction de mise à jour Φ. Soit S_k la valeur de l'état après k itérations de la fonction Φ

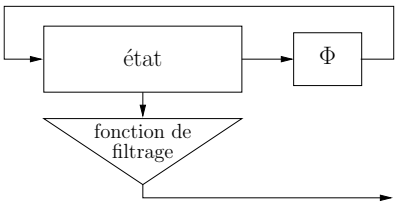

FIG. 1 – *Principe du chiffrement à flot.*

pour $k \geq 0$, alors nous mettons à jour l'état interne par $S_{k+1} = \Phi(S_k)$. En appliquant la fonction de filtrage sur l'état, nous extrayons la suite chiffrante. Pour des fonctions de mise à jour et de filtrage fixées, la suite chiffrante dépend seulement de l'état initial. Si celui-ci est déterminé seulement par la clé secrète nous pouvons produire une seule suite chiffrante. Un vecteur d'initialisation (IV) connu est alors utilisé pour créer plusieurs suites chiffrantes à partir de la même clé. Dans le cas le plus simple l'état initial est déterminé par les deux valeurs.

Dans ce qui suit nous travaillerons avec des modèles statistiques de chiffrement à flot. Un attaquant connaît seulement la loi de distribution de l'espace des clés, le plus souvent supposée uniforme. On modélise alors ce qui dépend de la clé par des variables aléatoires. De tels modèles nous permettent d'étudier des propriétés statistiques du chiffrement à flot comme par exemple l'entropie de l'état. L'entropie en théorie de l'information a été introduite par Shannon dans [Shannon 48] et est une mesure de la quantité d'information contenue dans une variable aléatoire. Elle est maximale si la variable prend toutes les valeurs possibles avec la même probabilité. Le modèle statistique que nous utilisons est décrit dans la Section 1.1.

Récemment, plusieurs chiffrements à flot qui utilisent des fonctions de mise à jour non–bijectives ont été proposés. D'ailleurs, dans certains cas cette fonction semble se comporter comme une fonction aléatoire comme pour le chiffrement à flot MICKEY [Babbage 05]. Utiliser une fonction non–bijective signifie que certaines valeurs de l'état peuvent être produites par plus d'une valeur. Par conséquent, après avoir appliqué une telle fonction de mise à jour, le nombre d'états possibles ainsi que l'entropie seront réduits. Au contraire, si un chiffrement à flot utilise une permutation pour mettre à jour son état, chaque valeur est produite par exactement une autre valeur. Ainsi, seul l'ordre des probabilités change et l'entropie ne varie pas. Dans cet article nous prouverons que l'utilisation d'une fonction aléatoire nous permet d'estimer la perte moyenne d'entropie après plusieurs itérations de la fonction de mise à jour. De plus, nous examinerons si un attaquant est capable d'exploiter cette perte d'entropie pour monter une attaque efficace. D'abord nous allons présenter le modèle avec lequel nous allons travailler.

1.1 Notre Modèle de Chiffrement à Flot

Nous reprenons la structure d'un chiffrement à flot comme représenté dans la Fig. 1. La particularité de notre modèle est que Φ est une fonction aléatoire ce qui nous permet de calculer quelques propriétés statistiques de notre chiffrement à flot.

Définition 1 *Soit $\mathcal{F}_n = \{\varphi \mid \varphi : \Omega_n \to \Omega_n\}$ l'ensemble de toutes les fonctions de l'ensemble $\Omega_n = \{\omega_1,\omega_2,\ldots,\omega_n\}$ dans lui-même. Nous disons que Φ est une* fonction aléatoire *si elle prend chaque valeur $\varphi \in \mathcal{F}_n$ avec la même probabilité $Pr[\Phi = \varphi] = 1/n^n$.*

Pour une définition plus complète d'une fonction aléatoire nous nous référons au livre de Kolchin [Kolchin 86].

Soit la valeur de l'état interne après k itérations, S_k, une variable aléatoire qui prend des valeurs dans Ω_n. La loi de probabilité de l'état initial est définie par $\{p_i\}_{i=1}^n$ tels que

$$p_i = Pr[S_0 = \omega_i].$$

Sauf mention contraire, nous supposerons une distribution uniforme, c'est à dire $p_i = 1/n$ pour tout $1 \leq i \leq n$. Pour $k > 0$ nous définissons $S_{k+1} = \Phi(S_k)$ où nous utilisons la même valeur de Φ pour toutes les itérations. Alors

$$p_i^{\Phi}(k) = Pr[\Phi^{(k)}(S_0) = \omega_i]$$

est la probabilité que l'état interne soit ω_i après avoir appliqué k fois Φ sur l'état initial. Notons $p_i^{\varphi}(k)$ la même probabilité mais pour une fonction déterminée $\varphi \in \mathcal{F}_n$. La notation ci-dessus nous permet de définir l'entropie de l'état après k itérations de Φ

$$H_k^{\Phi} = \sum_{i=1}^{n} p_i^{\Phi}(k) \log_2 \left(\frac{1}{p_i^{\Phi}(k)} \right).$$

Dans cet article nous sommes intéressés par les espérances où la moyenne est prise sur l'ensemble de toutes les fonctions $\varphi \in \mathcal{F}_n$. Pour accentuer que nous parlons d'une espérance, nous l'écrirons en gras. Ainsi

$$\mathbf{H_k} = \mathbf{E}(H_k^{\Phi})$$

est l'espérance de l'entropie de l'état interne après k itérations.

La suite de l'article suivant est divisé en deux sections principales. Dans la Section 2, nous discutons des différentes possibilités pour estimer l'entropie d'état. Nous donnons un bref aperçu des résultat précédents de [Flajolet 90] et [Hong 05] dans la Section 2.1. Puis, dans la Section 2.2, nous présentons une nouvelle estimation qui est, pour des nombre limités d'itérations, plus précise que les précédentes. Dans la Section 3, nous étudions deux attaques par collisions contre le chiffrement à flot MICKEY version 1 [Babbage 05] présentés dans [Hong 05]. Pour la première attaque, nous montrons que nous gagnons seulement un facteur sur la complexité en espace si nous augmentons la complexité en données du même facteur. Pour la deuxième attaque nous démontrons que, contrairement à ce qui est espéré des résultats [Hong 05], les complexités sont équivalentes à une recherche de collision directement dans les états initiaux. Pour finir nous présentons une nouvelle variante de ces attaques qui nous permet de réduire la complexité en espace, cependant la complexité en données reste la même.

2 Estimation d'entropie

Dans cette section nous discutons l'estimation de l'entropie de l'état interne et quelques paramètres utiles des fonctions aléatoires.

2.1 Travail précédent

Flajolet et Odlyzko proposent, dans [Flajolet 90], plusieurs paramètres de fonctions aléatoires pour analyser leur graphe fonctionnel. Un graphe fonctionnel d'une fonction φ

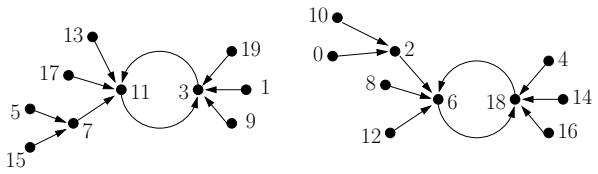

FIG. 2 – *Exemple d'un graphe fonctionnel pour* $\varphi(x) = x^2 + 2 \pmod{20}$.

est un graphe orienté qui a une flèche dirigée du sommet x au sommet y si et seulement si $\varphi(x) = y$. Un exemple pour $\varphi(x) = x^2 + 2 \pmod{20}$ est présenté sur la Fig. 2. Pour des fonctions sur un ensemble fini, un tel graphe se compose d'une ou plusieures composantes séparées, où chacune des composantes est constituée d'un cycle d'arbres, c'est-à-dire que les nœuds dans le cycle sont la racine d'un arbre. Dans le cas d'une permutation, le graphe fonctionnel est consisté seulement d'un ou plusieurs cycles séparés sans arbres.

Pour trouver l'espérance d'un paramètre d'une fonction aléatoire, Flajolet et Odlyzko construisent la fonction génératrice du graphe fonctionnel associée à ce paramètre. Puis, ils obtiennent une valeur asymptotique de l'espérance à l'aide d'une analyse des singularités de la fonction génératrice. Toutes les valeurs asymptotiques sont prises pour n tendant vers $+\infty$. Nous donnons ici la définition de certains des paramètres examinés. La longueur maximale de queue est le nombre maximal d'étapes avant arriver à un cycle. Un r–nœud est un nœud dans le graphe avec exactement r nœuds entrants ce qui est équivalent à une pré–image de taille r. La taille d'image est le nombre de points dans le graphe qui sont accessible. Les valeurs asymptotiques de ces paramètres sont:

- l'espérance du nombre de points de cycle $\mathbf{pc}(n) \sim \sqrt{\pi n/2}$,
- l'espérance de la longueur maximale de queue $\mathbf{mq}(n) \sim \sqrt{\pi n/8}$,
- l'espérance du nombre de r–nœuds $\mathbf{rn}(n,r) \sim \frac{n}{r!e}$ et
- l'espérance de la taille d'image après k itérations $\mathbf{ti}(n,k) \sim n(1 - \tau_k)$ où $\tau_0 = 0$ et $\tau_{k+1} = e^{-1+\tau_k}$.

Pour toutes ces valeurs, l'espérance est prise sur toutes les fonctions dans $\mathcal{F}_n$.

Dans [Hong 05], Hong et Kim emploient l'espérance de la taille d'image pour obtenir une borne supérieure de l'entropie de l'état après k itérations. Ils utilisent le fait que l'entropie est toujours plus petite ou égale que le logarithme du nombre de points qui ont une probabilité plus grande que zéro. Après un nombre fini d'étapes, chaque point dans le graphe fonctionnel arrive à un cycle, ainsi la taille de l'image ne peut pas être inférieure au nombre de points des cycles. Par conséquent, la borne supérieure de l'espérance de l'entropie de l'état interne

$$\mathbf{H_k} \leq \log_2(n) + \log_2(1 - \tau_k) \tag{1}$$

est valide tant que $\mathbf{ti}(n,k) > \mathbf{pc}(n)$. Nous pouvons voir que, pour cette borne, la perte d'entropie ne dépend que de k et pas de n.

Dans la Fig. 3, nous comparons pour $n = 2^{16}$ les valeurs de cette borne avec des valeurs empiriques. Pour calculer ces valeurs nous avons choisi 10^4 fonctions à l'aide du générateur de nombres aléatoires HAVEGE [Sendrier 02], puis nous avons calculé l'entropie moyenne en supposant que l'état initial est distribué uniformément. Même si n n'est pas très grand, notre exemple suffit pour comprendre les relations entre les facteurs différents. La figure montre que si k reste plus petit que $\mathbf{mq}(n)$ cette borne est valide et ne chute pas sous $\log_2(\mathbf{pc}(n))$.

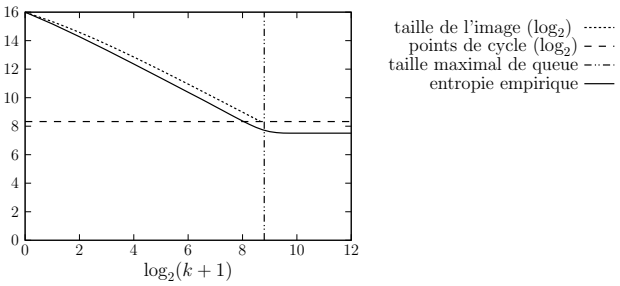

FIG. 3 – *Borne supérieure et valeur empirique de l'entropie moyenne pour* $n = 2^{16}$.

2.2 Nouvel estimateur d'entropie

La taille d'image fournit une borne supérieure assez large pour l'espérance de l'entropie de l'état interne. Nous allons donner une estimation plus précise en utilisant les méthodes indiquées dans [Flajolet 90].

Pour une fonction donnée $\varphi \in \mathcal{F}_n$, soit ω_i un r–nœud. Ceci est équivalent au fait que ω_i est produit par exactement r valeurs initiales différentes après une itération. Ainsi, si la distribution initiale d'état est uniforme, ce nœud a une probabilité de $p_i^{\varphi}(1) = r/n$.

Soit $\mathbf{rn_k}(n,r)$ l'espérance de nombre de points qui sont atteints par exactement r nœuds après k itérations. Alors, dans le cas d'une distribution initiale uniforme l'espérance de l'entropie est

$$\mathbf{H_k} = \sum_{r=1}^{n} \mathbf{rn_k}(n,r) \frac{r}{n} \log_2\left(\frac{n}{r}\right) = \log_2(n) - \sum_{r=1}^{n} \mathbf{rn_k}(n,r)\ \frac{r}{n}\ \log_2(r). \quad (2)$$

L'égalité à droite provient du fait que $\sum_{r=1}^{n} r\ \mathbf{rn_k}(n,r) = n$ comme chacun des n nœuds a exactement un successeur et sera compté dans un $\mathbf{rn_k}(n,r)$.

Pour estimer $\mathbf{rn_k}(n,r)$ pour une itération, nous employons directement le résultat pour l'espérance de r–nœuds, donné dans [Flajolet 90], comme $\mathbf{rn_1}(n,r) = \mathbf{rn}(n,k) \sim \frac{n}{r!e}$. Pour plus d'une itération nous allons définir un nouveau paramètre.

Nous appelons nœud de cycle les nœuds du graphe fonctionnel qui sont comprit dans un cycle et qui sont donc racine d'un arbre, et nœud d'arbres les autres. Nous devons faire cette distinction parce qu'un nœud de cycle n'a pas seulement les nœuds entrants de son arbre mais également les nœuds entrants de son cycle.

Pour définir notre nouveau paramètre nous utilisons le fait qu'un nœud d'arbre qui est exactement atteint par r nœuds après k itérations a

- j nœuds entrants, où $1 \leq j \leq l$. Chacun de ces nœuds sont racine d'un arbre de profondeur supérieure ou égale à $k-1$ et sont atteint respectivement par $i_1, \ldots, i_j$ nœuds après $k-1$ itérations tels que $i_1 + \cdots + i_j = r$.
- un nombre arbitraire de nœuds entrants qui sont racines d'arbres de profondeur strictement plus petite que $k-1$.

Comme dans [Flajolet 90] nous construisons la fonction génératrice associée à notre paramètre. À l'aide d'une analyse des singularités nous obtenons le résultat asymptotique

$$\mathbf{rn_k}(n,r) \sim n\ c_k(r), \quad (3)$$

avec $c_k(r)$ définie par

$$c_k(r) = \begin{cases} 1/e\ r! & \text{pour } k = 1 \\ f_1(k)/e \sum\limits_{[i_1,\ldots,i_j]\in Par(r)} c_{k-1}(i_1)\ \cdots\ c_{k-1}(i_j)\ f_2([i_1,\ldots,i_j]) & \text{autrement.} \end{cases} \tag{4}$$

où

- $f_1(k)$ viens de l'ensemble des arbres avec une profondeur strictement plus petite que $k-1$. Nous avons

$$f_1(k) = \begin{cases} e^{1/e} & \text{pour } k = 1 \\ e^{f_1(k-1)/e} & \text{autrement.} \end{cases}$$

- $Par(r)$ est l'ensemble de toutes les partitions de l'entier r.
  Par exemple pour $r = 4$ nous avons $Par(4) = \{[1,1,1,1],[1,1,2],[2,2],[1,3],[4]\}$.
- $f_2([i_1,\ldots,i_j])$ est un terme de correction.
  Si certaines valeurs $i_{x_1},\ldots,i_{x_\ell}$ avec $1 \le x_1 < \cdots < x_\ell \le j$ sont identiques – $i_{x_1} = \cdots = i_{x_\ell}$ – nous devons multiplier par le facteur $1/\ell!$ pour compenser cette multiplicité, c'est à dire $f_2([1,1,1,1,2,2,3]) = \frac{1}{4!2!1!}$.

Dans (3) nous ignorons que pour un nœud de cycle nous devons compter non seulement des nœuds entrants d'arbre mais aussi des nœuds entrants de cycle. Cependant, si n est assez grand, l'espérance du nombre de nœuds de cycle est $\sqrt{\pi n/2}$ qui est seulement une petit partie de tous les nœuds. Ainsi l'erreur introduite est négligeable si k, et par conséquent la nombre de nœuds qui arrive au cycle, n'est pas trop grand.

Le calcul de $c_k(r)$ comme présenté dans (4) n'est pas très pratique. En introduisant une fonction $D(k,r,m)$ nous pouvons obtenir une expression plus simple

$$\begin{aligned} c_k(r) &= \begin{cases} 1/r!e & \text{si } k = 1 \\ D(k,r,1)f_1(k)/e & \text{autrement} \end{cases} \\ D(k,r,m) &= \begin{cases} 1 & \text{si } r = 0 \\ 0 & \text{si } 0 < r < m \\ \sum_{u=0}^{\lfloor r/m \rfloor} \frac{c_{k-1}(m)^u}{u!} D(k,r-m\ u,m+1) & \text{autrement,} \end{cases} \end{aligned}$$

qui nous permit de calculer tous les de $c_k(r)$ pour $k \le K$ et $r \le R$ avec une complexité en temps de $O\left(K\ R^2 \log(R)\right)$.

Revenons à l'estimation d'entropie dans (2). En utilisant l'expression (3) nous pouvons écrire

$$\mathbf{H_k} \sim \log_2(n) \underbrace{- \sum_{r=1}^{R} c_k(r)\ r\ \log_2(r)}_{(a)} \underbrace{- \sum_{r=R+1}^{n} c_k(r)\ r\ \log_2(r)}_{(b)},$$

où (a) représente l'estimation de la perte d'entropie qui ne dépend pas de n et (b) un terme correctif. Dans la Fig. 4 nous pouvons voir que la valeur $c_k(r)\ r\ \log_2(r)$ baisse vite lorsque r augmente. Cependant, pour k grand la décroissance devient plus lente. Ainsi, si nous voulons que le terme (b) soit négligeable pour k grand nous avons besoin d'une grande valeur de R. Dans la Table 1 nous comparons notre estimateur d'espérance d'entropie

$$\mathbf{H_k} \sim H_k(R) = \log_2(n) - \sum_{r=1}^{R} c_k(r)\ r\ \log_2(r) \tag{5}$$

avec la borne supérieure (1) et la valeur empirique présenté dans la Fig. 3. Nous pouvons voir que pour un petit k nous obtenons une estimation qui est beaucoup plus précise que la borne supérieure (1). Cependant, pour k grand nous avons besoin d'un grand R et notre méthode deviens plus coûteuse.

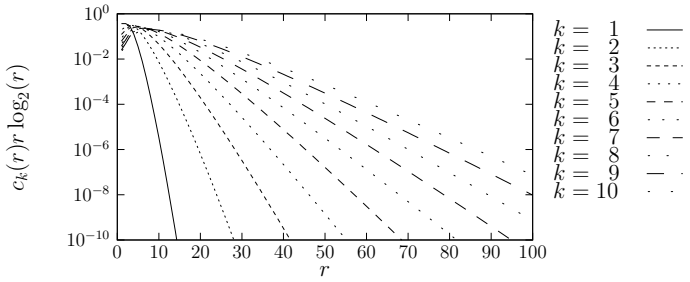

FIG. 4 – *Évaluation de $c_k(r)\ r\ \log_2(r)$ en fonction de k et r.*

k		0	1	2	3	$\cdots$	10	$\cdots$	50	$\cdots$	100
valeur empirique, $n = 2^{16}$		0.0000	0.8273	1.3458	1.7254	$\cdots$	3.1130	$\cdots$	5.2937	$\cdots$	6.2529
taille d'image (1)		0.0000	0.6617	1.0938	1.4186	$\cdots$	2.6599	$\cdots$	4.7312	$\cdots$	5.6913
$H_k(R)$ **(5)**	$R = 50$	0.0000	0.8272	1.3457	1.7254	$\cdots$	3.1084	$\cdots$	2.6894	$\cdots$	1.2524
	$R = 200$	0.0000	0.8272	1.3457	1.7254	$\cdots$	3.1129	$\cdots$	5.2661	$\cdots$	5.5172
	$R = 1000$	0.0000	0.8272	1.3457	1.7254	$\cdots$	3.1129	$\cdots$	5.2918	$\cdots$	6.2729

TAB. 1 – *Comparaison des différents méthodes pour estimer la perte d'entropie.*

Nous pouvons généraliser le cas d'une distribution initiale arbitraire $P = \{p_1,p_2,\ldots,p_n\}$. L'espérance d'entropie est prit sur tous $\varphi \in \mathcal{F}_n$, ainsi chaque n–uplet d'indices $1 \leq i_1 < i_2 < \cdots < i_r \leq n$ de taille r apparaît le même nombre de fois. En total il y a $n^n\ \mathbf{rn_k}(n,r)$ points qui sont atteint par r nœud après k itérations et il y a $\binom{n}{r}$ n–uplet d'indices possible de taille r. Ainsi chaque élément $(p_{i_1} + \cdots + p_{i_r}) \log_2 \frac{1}{p_{i_1}+\cdots+p_{i_r}}$ apparaît $\frac{n^n\ \mathbf{rn_k}(n,r)}{\binom{n}{r}}$ fois dans la somme. En divisant par n^n nous avons pour l'espérance d'entropie

$$\mathbf{H_k^P} = \sum_{r=1}^{n} \mathbf{rn_k}(n,r) \frac{1}{\binom{n}{r}} \sum_{1 \leq i_1 < \cdots < i_r \leq n} (p_{i_1} + \cdots + p_{i_r}) \log_2 \frac{1}{p_{i_1} + \cdots + p_{i_r}}. \quad (6)$$

Dans l'équation (6) nous pouvons utiliser de nouveau la valeur (3) pour approcher $\mathbf{rn_k}(n,r)$.

Dans ce chapitre nous avons présenté un nouvel estimateur d'entropie. Nous pourrions montrer que si le nombre d'itérations n'est pas trop grand, il est beaucoup plus précis que la limite supérieure donnée par la taille d'image. En plus, la même méthode peut être employé pour n'importe quelle distribution initiale arbitraire.

3 Attaques par collision

Dans [Hong 05], Hong et Kim déclarent que la fonction de mise à jour de la première version du chiffrement à flot MICKEY [Babbage 05] se comporte presque comme une fonction aléatoire concernant la perte d'entropie et la taille de l'image. Ils présentent deux attaques par collision, qui essayent d'exploiter la perte d'entropie de l'état interne, cependant sans analyse détaillée de complexité. Ces attaques sont directement applicables sur notre modèle.

Nous prenons deux états initiaux différents S_0 et S'_0 et appliquons la même fonction impérativement sur les deux. Nous parlons d'une collision si il existe k et k' tel que $S_k = S'_{k'}$. L'idée de Hong et Kim était qu'une entropie réduite induit une probabilité plus grande de collision. Une fois que nous avons trouvé une collision, nous savons que les états suivants sont identiques. En raison du paradoxe des anniversaires, nous supposons qu'avec une entropie de m-bits nous trouvons une collision avec une grande probabilité si nous prenons $2^{\frac{m}{2}}$ valeurs différentes.

Dans cette section nous étudions les deux attaques présentées dans [Hong 05]. Pour chacune d'elles nous donnons une analyse détaillée de la complexité, où nous considérons la complexité en espace et en données ainsi que le nombre d'états initiaux nécessaires. Tous les résultats sont en moyenne pour une attaque ayant une grande probabilité de succès. La complexité en données est le nombre d'états qui sont nécessaires à l'attaque. La complexité en espace est le nombre d'états que nous devons enregistrer pour pouvoir chercher une collision entre eux. Chaque fois nous comparons les résultats avec les complexités d'une recherche d'une collision directement dans des états initiaux choisis d'une façons aléatoire. Les complexités en données et en espace pour une telle recherche sont de l'ordre de $\sqrt{n}$.

Dans toutes les attaques étudiées ici, nous ne considérons jamais le nombre de bits de sortie qui sont nécessaire pour enregistrer un état et pour identifier une collision, car cette valeur dépend de la fonctions de filtrage utilisé. Dans l'exemple de MICKEY, Hong et Kim déclare qu'ils ont besoin d'environ 2^8 bits.

3.1 L'état après k itérations

La première attaque de Hong et Kim prend m états initiaux différents et applique k fois la même instance de Φ sur chacun d'eux. Puis elle cherche une collision dans les m états résultants. Utilisant (1) nous savons que l'espérance de l'entropie après k itérations d'une fonction aléatoire est plus petite que $\log_2(n) + \log_2(1 - \tau_k)$. Hong et Kim conjecturent, en se basant sur des résultats expérimentaux, que celle-ci est approximativement $\log_2(n) - \log_2(k) + 1$. Ainsi, avec une forte probabilité nous trouvons une collision si $m > 2^{(\log_2(n)-\log_2(k)+1)/2} = \sqrt{2n/k}$.

Cette attaque enregistre seulement les dernières valeurs d'itérations et elle cherche une collision dans cet ensemble. Cela donne une complexité en espace de $m \sim \sqrt{2n/k}$ pour k assez grand. Cependant, nous avons appliqué k fois Φ sur tous les états choisis, ce qui donne une complexité en données de $mk \sim \sqrt{2kn}$. C'est à dire que nous augmentons la complexité en données autant que nous réduisons la complexité en espace et le nombre d'états initiaux.

On peut se demander, s'il y a des circonstances dans lesquelles nous pouvons employer cette approche sans augmenter la complexité en données. Le réponse est oui, si le chiffrement à flot utilise un ensemble des fonctions telles que l'espérance de la perte d'entropie est plus grande que $2\log_2(k)$ après k itérations. Une telle caractéristique impliquerait que nous n'employons pas de fonctions aléatoires pour mettre à jour l'état, cependant le principe de l'attaque reste la même.

3.2 Avec des états intermédiaires

La deuxième attaque dans [Hong 05] est équivalente à employer $2k - 1$ fois la même fonction Φ sur m états initiaux différents et de chercher une collision dans tous les états intermédiaires de la $k$ème jusqu'à la $(2k - 1)$ème itération. Comme après $k - 1$ itérations nous avons environ $\log_2(n) - \log_2(k) + 1$ bits d'entropie, Hong and Kim suppose que nous avons besoins d'un m tels que $mk \sim \sqrt{n/k}$. Ils déclarent que ce résultat serait un peu trop optimiste puisque des collisions à l'intérieur d'une ligne n'apparaissent normalement pas avec un chiffrement à flot. Cependant, ils annoncent que cette approche représente quand même une menace réaliste pour le chiffrement à flot MICKEY. Nous prouverons que, contrairement à leur conjecture, cette attaque a des complexités du même ordre de grandeur qu'une recherche directe de collision dans les états initiaux.

Considérons un tableau contenant les $2km$ états intermédiaires pour les $2k - 1$ itérations. S'il n'y pas de collisions dans cet ensemble la, il n'y aura pas non plus de collision

dans les km état considérés par l'attaque. Ainsi, la probabilité de succès de l'attaque est encore plus petite que $1 - Pr[sansColTotal]$, où $Pr[sansColTotal]$ est la probabilité qu'il n'y ait aucune collision dans l'ensemble des $2km$ états. Soit $Pr[sansColInit]$ la probabilité qu'il n'y ait pas de collisions dans les m états initiaux. Nous pouvons écrire directement que

$$\begin{aligned} Pr[sansColTotal] &= Pr[sansColTotal \cap sansColInit] \\ &= Pr[sansColTotal|sansColInit]\ Pr[sansColInit]. \end{aligned}$$

Supposons que nous ayons prit m états initiaux différents. Ceci arrive avec une probabilité

$$Pr[sansColInit] = \frac{\binom{n}{m} m!}{n^m}. \tag{7}$$

Dans ce cas nous avons n^n fonctions différentes, où chacune crée $\binom{n}{m} m!$ tableaux différentes. Chaque table peut être produite par plus d'une fonction. Il y a $n\ (n-1)\dots(n-2km+1)$ tableaux différents qui ne contiennent aucune collision. Chacun d'eux apparaîtra pour $n^{n-(2k-1)m}$ fonctions différentes, puisque un tableau détermine déjà $(2k-1)m$ positions de la fonction. Ainsi, nous obtenons la probabilité

$$Pr[sansColTotal|sansColInit] = \frac{n\ (n-1)\dots(n-2km+1)\ n^{n-(2k-1)m}}{n^n \binom{n}{m} m!} \tag{8}$$

pour $m > 0$ et $2km \leq n$. En combinant (7) et (8) nous arrivons à

$$Pr[sansColTotal] = \frac{n(n-1)\dots(n-2km+1)}{n^{2km}} \tag{9}$$

où la probabilité est prise sur toutes les fonctions $\varphi \in \mathcal{F}_n$ et sur tous les états initiaux possibles. C'est exactement la probabilité de non collision parmi $2km$ points aléatoires, nous avons donc besoin d'un m tel que $2km \sim \sqrt{n}$. Cela nous donne une complexité en données de l'ordre de $\sqrt{n}$ et une complexité en espace de $\sim \sqrt{n}/2$.

3.3 Amélioration par des points distingués

En utilisant la méthode connue des points distingués nous pouvons réduire la complexité en espace de la deuxième attaque, cependant, la complexité en données ne change pas.

Considérons un sous-ensemble de Ω_n qui est distingué par une propriété, par exemple que les ℓ bits de poids fort doivent être 0 pour une entier ℓ spécifié, un *point distingué* (PD) est un élément de cet ensemble. Dans notre variante de la deuxième attaque nous itérons φ dans chaque ligne jusqu'à arriver à un point distingué. Dans ce cas nous nous arrêtons et enregistrons le PD. Si nous n'arrivons pas à un PD après un nombre fixé MAX itérations nous nous arrêtons et nous n'enregistrons rien. S'il y a une collision dans un des états, les états suivants sont identique et nous arrêtons avec le même PD. Ainsi, il est suffisant à chercher une collision dans les PDs finaux.

Soit d le nombre de points distingués dans Ω_n. Nous supposons que le ratio $c = \frac{d}{n}$ est assez grand pour que, avec une grand probabilité, nous arriver à un PD dans chaque ligne avant la fin de cycle dans le graphe fonctionnel. C'est à dire qu'en moyenne le nombre d'itérations avant d'arriver à un PD est beaucoup plus petit que l'espérance de la longueur de queue plus la longueur de cycle. L'espérance de cette longueur est de l'ordre de $\sqrt{\frac{\pi\ n}{2}}$ [Flajolet 90]. Nous supposons que dans ce cas l'espérance de la longueur d'une ligne est de l'ordre de $1/c$ comme ce serait le cas pour des points aléatoires. Nous supposons aussi que nous avons besoin d'environ $m/c \sim \sqrt{n}$ points donnés pour trouver une collision,

comme dans le cas précédent. Ainsi nous avons obtenons une complexité en données de $\sim \sqrt{n}$ et une complexité en espace de seulement $\sim c\sqrt{n}$. Des résultat empiriques pour $n = 2^{20}$, $0.7 \leq \frac{\log_2(d)}{\log_2(n)} \leq 1$ et $MAX = \sqrt{n}$ confirment nos hypothèses.

Un résumé de toutes les complexités de toutes les attaques est présenté dans la Table 2. Nous pouvons voir que soit la complexité en espace est encore dans la même ordre de grandeur que $\sim \sqrt{n}$ ou elle est réduite par le même facteur que la complexité en données est augmentée.

attaque	**# états initiaux**	**complexité en espace**	**complexité en données**
après k itérations, 3.1	$\sim \sqrt{2n/k}$	$\sim \sqrt{2n/k}$	$\sim \sqrt{2kn}$
avec les états intermédiaires, 3.2	$\sim \sqrt{n}/2k$	$\sim \sqrt{n}/2$	$\sim \sqrt{n}$
avec des PDs, 3.3	$\sim c\sqrt{n}$	$\sim c\sqrt{n}$	$\sim \sqrt{n}$

TAB. 2 – *Complexité des attaques.*

4 Conclusion

Nous avons présenté une nouvelle méthode pour estimer l'entropie de l'état interne d'un chiffrement à flot qui emploie une fonction aléatoire pour la mise à jour. Cet estimateur est basé sur le nombre de points qui produisent la même valeur après k itérations. Son calcul est cher pour un grand nombre d'itérations, cependant pour des petits nombres il est beaucoup plus précis que la borne supérieure donnée par la taille de l'image.

Nous avons également étudié les deux attaques par collision proposées dans [Hong 05]. Nous avons précisé que la première attaque améliore la complexité en espace au coût d'un augmentation de la complexité en données. Nous avons montré que les complexités de la deuxième attaque sont équivalent à la complexité d'une recherche directe de collision dans les valeurs initiales. De plus, nous avons proposé une nouvelle attaque qui réduit la complexité en espace mais laisse la complexité en données constante. L'utilisation des fonctions aléatoires dans un chiffrement à flot introduit le problème de la perte d'entropie, cependant, les attaques proposées basées sur cette faiblesse sont moins efficaces que prévu. Ainsi, le fait d'employer une fonction aléatoire n'est pas suffisant pour qu'un chiffrement à flot soit menacé par une attaque par collision basée sur la perte d'entropie.

Références

[Babbage 05] S. Babbage & M. Dodd. *The stream cipher MICKEY (version 1)*. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/015, 2005. `http://www.ecrypt.eu.org/stream`.

[Flajolet 90] P. Flajolet & A. M. Odlyzko. *Random Mapping Statistics*. Advances in Cryptology, Proc. Eurocrypt'98, vol. 434, pages 329–354, 1990.

[Hong 05] J. Hong & W. H. Kim. *TMD-Tradeoff and State Entropy Loss Considerations of Streamcipher MICKEY*. In INDOCRYPT, pages 169–182, 2005.

[Kolchin 86] V. F. Kolchin. Random mappings. Optimization Software, Inc, 1986.

[Sendrier 02] N. Sendrier & A. Seznec. *HArdware Volatile Entropy Gathering and Expansion: generating unpredictable random numbers at user level*. Rapport technique, INRIA, 2002.

[Shannon 48] C. E. Shannon. *A Mathematical Theory of Communication*. Bell System Technical Journal, vol. 27, pages 379–423 and 623–656, July and October 1948.

[Shannon 49] C. E. Shannon. *Communication Theory of Secrecy Systems*. Bell Systems Technical Journal, vol. 28, no. 4, pages 656—715, 1949.