Aléa et Cryptographie avec un point de vue dynamique (I & II)

Randomness and cryptography with a dynamical point of view

Pierre LIARDET

en collaboration avec

Alexis BONNECAZE

(Université d'Aix-Marseille)

Partie 1

tirer au hasard = événement imprévisible = aléa

Partie 2

Générateurs de nombres aléatoires (RNG)

1. tirer au hasard = événement imprévisible = aléa

Usages

- au quotidien : jeux de hasard : loteries, roulettes de casino, dés, cartes,...
- échantillonnages
- simulation numérique
- analyse/calcul numérique ; méthode (quasi)-Monte Carlo
- tests de validité de certains algorithmes
- algorithmes stochastiques (amélioration itérative, recuit simulé, algorithmes génétiques, des fourmis,...)
- distribution uniforme, pré-assignée
- générateurs de nombres
- • •

Définitions de l'aléa

a. mots aléatoires (effectif)

Complexité de Kolmogorov $K(w)$ d'un mot binaire w : "plus petit programme qui calcule w sur une machine de Turing" (Kolmogorov 1965, Chaltin 1966, Martin-Löf 1966).

Plus précis : pour une machine universelle de Turing U, $K(w)$ est la plus petite longueur de l'entrée d'un mot binaire e dans U (+ code de la table des transitions) qui donne en sortie w.

$\to$ $K(w)$ dépend de U mais il est défini à une constante additive près. On a toujours $K(x) \leq |w|$ (longueur du mot).

Définition. *Un mot est "aléatoire" au sens de Kolmogorov si $K(w) \geq |w|$ (complexité maximale parmi celles des mots de même longueur). On peut affaiblir en supposant seulement $K(w) \geq |w| - c$ (mot c-aléatoire).*

Conséquence rassurante : pour m assez grand, le mot 0^m n'est pas aléatoire!

$\rightarrow$ Le calcul de $K(w)$ est fonction de K ; il est défini à une constante additive près indépendante de $|w|$.

$\rightarrow$ Le problème " $\cdot$ est aléatoire au sens de Kolmogorov" est indécidable.

$\rightarrow$ Si on restreint les ressources en calcul (exemple "calcul en au plus n^3 étapes successives pour une entrée de longueur n"), la complexité qui en résulte est effectivement calculable.

Exemple de calcul : $K(\text{entier } n \text{ en binaire}) \leq \log_2 n + \mathcal{O}(1)$.

b. Suites binaires aléatoires (effectif)

La définition de Kolmogorov ne se prolonge pas aux suites infinies (Martin-Löf).

Solution 1

– remplacer la machine de Turing universelle par une machine de Turing "préfixe" P : si P calcule $P(e)$ et si e est préfixe de e', alors P calcule $P(e')$ et $P(e) = P(e')$.

– permet de déterminer la complexité préfixe $KP(w)$ de Kolmogorov.

– le mot binaire infini $x = x_0x_1x_2\cdots$ est "aléatoire" si

$$\lim_n \frac{KP(x_0\cdots x_{n-1})}{n} = 1\,.$$

Dans ce cas, x passe tous les tests statistiques, tels que :

- $\lim_n \frac{1}{n}\sum_{i<n} x_i = \frac{1}{2}$;
- x est k-équirépartie pour tout $k \geq 1$ (donc est 2-normal).

Solution 2

Passer par la notion de sous-ensemble A de $[0,1]$ Martin-Löf mesurable de mesure (de Lebesgue) 1. L'intersection $\mathcal{M}$ de tous ces sous-ensembles est encore Martin-Löf mesurable de mesure 1. Les éléments de $\mathcal{M}$ sont dits 'aléatoires" au sens de Martin-Löf.

c. Définition statistique (théorique)

Source uniforme sur q symboles

Définition. Générateur de suite $\sigma := \sigma_1\sigma_2\sigma_3\ldots$ de symboles σ_n pris dans un ensemble S de q symboles, telle que les mots $a_1\cdots a_m \in S^m$ apparaissent dans σ avec la fréquence q^{-m} (distribution uniforme). Une telle suite est dite *q-normale*.

Caractérisation par automates

Rappels : Automate α sur un alphabet $A = \{0, 1, \ldots, r-1\}$:

$$\alpha := (E, e_0, \Phi, \tau, S) \text{ avec}$$

E : espace des états (en nombre fini) avec sélection d'un état initial e_0 ;

Φ ensemble des instructions $\Phi_a : E \to E$, $a \in A$;

S ensemble des symboles de sortie ;

$\tau : E \to S$ fonction de sortie.

Fonctionnement : associer à tout mot $w := w_1\cdots w_n \in A^n$ le symbole

$$f_\alpha(w) = \tau \circ \phi_{w_n} \circ \cdots \circ \phi_{w_1}(e_0)\,.$$

$\to$ L'application $f_\alpha : A^* \to S$ est dit automatique, d'automate α.

Plusieurs automates α définissent la même fonction automatique $f = f_\alpha$. Il est intéressant de minimiser le nombre d'états de α.

• *Caractérisation* : $f_\alpha : A^* \to S$ est automatique si la relation d'équivalence (de Nerode) $\sim$ sur A^* :

$$w \sim w' \Longleftrightarrow \forall\, v \in A^*, f_\alpha(wv) = f_\alpha(w'v)$$

n'a qu'un nombre fini de classe $\bar{w}$.

$\to$ f_α est alors automatique pour l'automate (minimal) $\bar{\alpha} := (A^*/\sim, \bar{\wedge}, \bar{\Phi}, \bar{\tau})$ où $\bar{\wedge}$ est la classe du mot vide, $\bar{\phi}_a(\bar{w}) = \overline{wa}$ et $\bar{\tau}(\bar{w}) = f_\alpha(w)$.

Théorème de la prédiction (A. Broglio-P. L.). *La suite σ de symboles de S, $\#S = q$, est q-normale si, et seulement si, pour tout automate α d'alphabet S , on a*

$$\lim_n \frac{\#\{k\,;\ 0 \le k < n,\ f_\alpha(\sigma_1 \cdots \sigma_k) = \sigma_{k+1}\}}{n} = \frac{1}{q}\,.$$

Normalité et prédiction

Le théorème précédent dit qu'une suite σ de q symboles est q-normale s'il n'existe pas de fonction automatique $f : S^* \to S$ qui puisse "prédire" les digits suivants, plus précisément :

$$\limsup_n \frac{\#\{k\,;\ 0 \le k < n,\ f_\alpha(\sigma_1 \cdots \sigma_k) = \sigma_{k+1}\}}{n} > \frac{1}{q} \qquad (*)$$

Cette condition peut être affaiblie :

Définition. *Une suite $\sigma : n \mapsto S$ est dite ℓ-prédicable si elle vérifie (*) pour une application automatique f ne dépendant que des ℓ derniers digits $\sigma_{k-\ell+1}, \ldots, \sigma_k$ (pour $k \ge \ell$)*

Théorème. *σ est q normale si, et seulement si, pour tout ℓ elle n'est pas ℓ-prédicable.*

Normalité et aléa

• Peut-on définir une suite aléatoire sur q symboles comme une suite q-normale ?

D'après D. Knuth, ce n'est pas un bon choix. On aimerait que la suite σ vérifie :

Règle (R5). *Toute suite extraite $(\sigma_{s_k})_k$ calculable de σ (i.e. il existe un algorithme effectif $\mathcal{A}$ tel que $\{s_k\,;\ k = 1, 2, 2, \ldots\} = \{n\,;\ \mathcal{A}(n, \sigma_1, \ldots, \sigma_n) = 1\}$) est q-normale.*

Il suffit en fait de supposer plus simplement que chaque symbole apparaît dans $(\sigma_{s_k})_k$ avec la fréquence $1/q$.

Définition de D. Knuth. *Une suite σ de q symboles est dite aléatoire si, pour tout algorithme effectif qui détermine une suite infinie d'entiers t_n positifs distincts comme fonction de n et des valeurs précédentes $\sigma_{t_1}, \ldots, \sigma_{t_{n-1}}$, la suite $(\sigma_{t_n})_n$ vérifie la règle (R5).*

Ainsi, soit x une suite binaire 2-normale et soit x' définie par

$$x'_n = \begin{cases} x_n & \text{si } n \text{ n'est pas une puissance de 2,} \\ 0 & \text{sinon.} \end{cases}$$

La suite x' est encore 2-normale, mais elle ne vérifie pas la règle (R5) ; elle n'est donc pas aléatoire au sens de Knuth.

Test statistiques

Définition pragmatique d'une suite binaire finie aléatoire

$\rightarrow$ Suite qui passe les tests du NIST (National Institut of Standards and Technology).

Soit T un test sur l'ensemble des suites binaires finies de longueur L, muni de la probabilité uniforme U_L. Alors T est une variable aléatoire

$$T : \{0,1\}^L \rightarrow \mathbf{R}$$

Elle détermine une loi de probabilité P_T sur $\mathbf{R}$.

$\rightarrow$ Une suite finie à tester $x \in \{0,1\}^L$ est vue comme un échantillonnage issue d'une suite de variables aléatoires indépendantes X_i, $i = 1, \ldots L$, à valeurs dans $\{0,1\}$ et de loi uniforme.

Il s'agit de fixer les conditions pour que la valeur $T(x_1, \ldots, x_n)$ appartienne à un intervalle validant le test (hypothèse nulle) : la suite testée est alors déclarée *aléatoire* pour le test.

Liste des tests

1. Monobit test

Détermine si la fréquence des 0 est celle statistiquement attendue pour une suite aléatoire vraie. Recommandation : longueur de la suite ≥ 100.

Description :

– poser $X_k = 2x_k - 1$;

– Calculer $S_n = X_n + \cdots + X_n$;

– Calculer la statistique $T = \frac{|S_n|}{\sqrt{n}}$

– Calculer la probabilité $FECN(T) = \frac{2}{\sqrt{\pi}} \int_T^\infty e^{-t^2} dt$ (fonction erreur du complément pour la loi normale)

– Règle de décision : si $FECN(T) < 0,01$ le test n'est pas réussi, la suite est non-aléatoire.

2. Frequency Test within a Block

Recommandation : longueur de la suite ≥ 100.

Description :

– Subdiviser la suite $x_1, \dots x_n$ en $N = \lfloor N/M \rfloor$ blocs disjoints B_i de longueur M,

– Calculer la proportion p_i des 1 dans M_i

– Calculer la statistique $T(x) = 4M \sum_{i=1}^{N} (p_i - 1/2)^2$

– Calculer $FECG(N/2, T/2)$ où

$$FECG(a, \theta) = \frac{1}{\Gamma(a)} \int_{\theta}^{\infty} e^{-t} t^{a-1} dt$$

(fonction erreur du complément pour la loi Γ de paramètre a)

– Règle de décision : si $FECG(N/2, T/2) < 0,01$ le test n'est pas réussi, la suite est non-aléatoire.

3. Runs test

Un *run* de longueur k dans x est une suite ininterrompue de k bits identiques et de longueur maximale. Le test détermine si le nombre de runs de 0 et de 1 est celui (statistiquement) attendu pour une suite aléatoire. Recommandation : longueur $n \geq 100$.

Description :

– Calculer la fréquence des 1 : $p = \frac{1}{n}\sum_{i=1}^{n} x_i$,

– Si $|p - 1/2| \geq 2/\sqrt{n}$ le test a échoué. Sinon :

– calculer la statistique $V_n(x) = 1 + \sum_{k=1}^{n-1} r(k)$ ($r(k) = 0$ si $x_k = x_{k+1}$ et $r(k) = 1$ sinon)

– Calculer $PR(x) = FECN\left(\frac{|V_n(x) - 2np(1-p)|}{2\sqrt{2np(1-p)}}\right)$

– Règle de décision : si $PR(x) < 0,01$ le test n'est pas réussi, la suite est non-aléatoire.

Les autres tests

4. Test for the Longest Run of Ones in a Block

5. Binary Matrix Rank Test (matrices 32×32 construites par les suites de blocs de longueur 32 de la suite x).

6. Discrete Fourier Transform (Spectral) Test

7. Non-overlapping Template Matching Test

8. The Overlapping Template Matching Test

9. Maurer's "Universal Statistical" Test

10. The Linear Complexity Test

11. The Serial Test

12. The Approximate Entropy Test

13. The Cumulative Sums (Cusums) Test

14. The Random Excursions Test

15. The Random Excursions Variant Test

16. Lempel Ziv compression.

2. Générateurs de nombres aléatoires

Essentiellement deux types de générateurs :

– les “pseudos” ou algorithmiques

– les “vrais” ou physiques,

Générateur de nombres pseudo-aléatoires (PRNG)

Définition. *Cas binaire : algorithme déterministe qui prend en entrée un mot binaire de longueur ℓ (graine) et donne en sortie un mot binaire de longueur L (grand devant ℓ) qui apparaît comme étant aléatoire selon les applications recherchées (test de validation).*

Applications concrètes :

1) cryptographie (clés pour chiffrement à flots) ;

2) évaluation numérique (méthode Quasi-Monte Carlo) ;

3) optimisation (explorations aléatoires *guidées*) ;

4) simulation (évolution de systèmes mathématiques ou physiques).

. . .

Principe général

Pour un ensemble donné E, construire une application $f : E \to E$ telle que E soit l'unique orbite de f. Pour tout $a \in E$, $E = \{f^k(a)\,;\ 0 \leq k < \#E\}$.

Modèle

Soit A un alphabet fini.

Définition générale. *Soit $h : \mathbf{N} \to \mathbf{N}$ une suite croissante telle que $h(n) > n$ pour tout n.*

Un générateur aléatoire de type h est une application $G : A^* \to A^*$ telle que la restriction $G_{|A^n}$ de G à A^n, notée G_n, soit à valeurs dans $A^{h(n)}$ pour tout $n \in \mathbf{N}$.

Il est alors habituel de mettre sur chaque ensemble A^m la loi de probabilité uniforme U_m. L'objectif est que G induise sur les $A^{h(n)}$ des distributions D_n qui asymptotiquement s'approchent de la distribution uniforme $U_{h(n)}$ suivant un cahier de charges spécifiques :

– passer une batterie de tests ;

– D_n ne peut être distinguée de la distribution uniforme $U_{h(n)}$ par un algorithme de complexité polynomiale.

Exemple 1

Générateur congruentiel linéaire : $f : \mathbf{Z}/m\mathbf{Z} \to \mathbf{Z}/m\mathbf{Z}\,, \quad f(x) = ax + c.$

$$x_{n+1} \equiv ax_n + c \,(\mathrm{mod}\, m),$$

avec $(c, m) = 1$.

– le plus ancien et le mieux connu ;

– facile à programmer, temps d'exécution court ;

Problèmes :

– sensible au choix des paramètres a, c, m : inutilisable en cryptographie.

– " live mainly in planes" (W. Givens) ;

– Mais intéressant d'un point de vue dynamique.

Théorème (classique) *La période de f est m (maximum) si et seulement si les trois conditions suivantes sont vérifiées :*

(i) $pgcd(c, m)=1$.

(ii) Si p premier divise m, alors $p \,|\, a-1$.

(iii) Si m est un multiple de 4, alors $4 \,|\, a-1$.

Conséquence dynamique Notons $\mathbf{Z}_p$ l'anneau des entiers p-adiques et soit $\varphi : \mathbf{Z}_p \to [0,1]$ l'application de Monna en base p : $\varphi(a_0, a_1, a_2, \ldots) := \sum_{k=0}^{\infty} \frac{a_k}{p^{k+1}}$.

Corollaire *Soit p premier et a, c entiers tels que $p \not| \, c$, $p \,|\, a-1$ si $p \neq 2$ et $4 \,|\, a-1$ si $p = 2$, alors la transformation $T : \mathbf{Z}_p \to \mathbf{Z}_p$ définie par*

$$T(x) = ax + c$$

est uniquement ergodique et pour tout $\xi_0 \in \mathbf{Z}_p$ la discrépance de la suite $\xi : n \mapsto \varphi(T^n(\xi_0))$ est d'ordre minimal :

$$N D_N(\xi) \leq C_{a,c} \log N \, .$$

Exemple 2

Registre à décalage

Construire, pour $\ell \geq 1$ donné, des suites binaires périodiques, de période T, telles que les mots de longueur ℓ apparaissent avec une fréquence $1/2^\ell$ (donc $T \geq 2^\ell$).

Exemple de construction

La suite périodique cherchée est vue comme un mot ***circulaire*** (de De Bruijn). Sa longueur est 2^ℓ. On construit en fait un mot circulaire de longueur $2^\ell - 1$ où tous les mots de longueur ℓ, sauf 0^ℓ, apparaissent ; on insère ensuite un 0 en bonne place.

• Méthode

– Choisir un polynôme $P(X)$ irréductible de degré ℓ sur le corps $\mathbf{F}_2$.

– Développement en série formelle :

$$\frac{1}{P(X)} = 1 + a_1 X + a_2 X + \dots \quad (a_k \in \mathbf{F}_2)\,.$$

– La suite des a_k vérifie une relation de récurrence linéaire d'ordre ℓ. Un bloc B de ℓ coefficients successifs détermine tous ceux qui suivent.

– Si T est la période de la suite a, il y a T blocs B consécutifs distincts et $\frac{1-X^T}{P(X)}$ est un polynôme sur $\mathbf{F}_2$.

– Donc les racines ζ de P vérifie $\zeta^T = 1$.

– En prenant $P(X)$ polynôme primitif (les racines sont d'ordre maxi $2^\ell - 1$ dans le groupe cyclique $\mathbf{F}_{2^\ell}$) on obtient une suite $a_0, a_1, \ldots,$ cherchée.

– Il y a $\frac{\varphi(2^\ell-1)}{\ell}$ polynômes primitifs sur $\mathbf{F}_2$ de degré ℓ.

• Réalisation : registre à décalage linéaire

Il s'agit de calculer les a_i d'une suite récurrente linéaire

$$a_i = a_{i-1}h_{\ell-1} + \cdots + a_{i-\ell+1}h_1 + a_{i-\ell}h_0$$

associée au polynôme compagnon $h(X) = X^\ell - h_{\ell-1}X^{\ell-1} - \cdots - h_0$ (rappel : on est dans $\mathbf{F}_2$, donc $1 = -1$), avec $h(X)$ irréductible.

$$(a_{i-\ell+1},\dots,a_i) = (a_{i-\ell},\dots,a_{i-1})C_h,$$

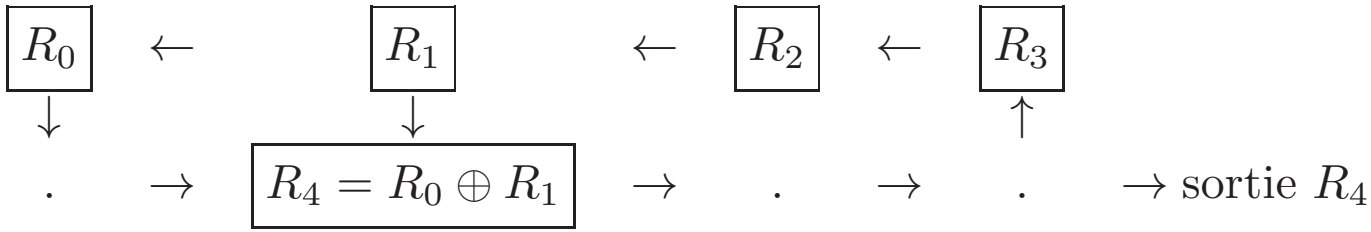
avec

$$C_h = \begin{pmatrix} 0 & 0 & \dots & 0 & h_0 \\ 1 & 0 & \dots & 0 & h_1 \\ 0 & 1 & \dots & 0 & h_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & h_{\ell-1} \end{pmatrix}$$

Exemple : $\ell = 4$, $h(X) = X^4 + X + 1$ ($h_0 = h_1 = 1$, $h_2 = h_3 = 0$) et

$$C_h = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Periode $2^4 - 1$, initialisation $(a_0, a_1, a_2, a_3) \neq (0,0,0,0)$, registre :



Construction générale

$E = \{0,1\}^m$, $f : E \to E$ de la forme

$$f(x_1, \ldots, x_m) = (x_2, x_3, \ldots, g(x_1, \ldots, x_m))$$

avec $g : E \to \{0,1\}$.

On fixe une graine $a_1, \ldots, a_m$ puis par récurrence

$$a_{n+1} = g(a_{n-m+1}, \ldots, a_{n-1}, a_n).$$

En cryptographie g est choisie non linéaire. Au mieux , g est une fonction booléenne courbe.

Exemple 3

Générateurs de type inversif

Intérêt :

– ils ne sont pas linéaires,

– ils utilisent la structure multiplicative des corps finis ou des anneaux de Galois,

– l'inversion dans $\mathbf{F}_{2^8}$ est la partie non linéaire de l'algorithme de chiffrement AES (le standard actuel qui remplace le DES).

• **Générateurs inversifs sur les corps finis** (Eichenhauer et Lehn 1986)

– en caractéristique deux : application $\phi : \mathbf{F}_{2^m} \to \mathbf{F}_{2^m}$ définie par

$$\phi(t) = a_0 t^{-1} + b_0, \text{ si } t \neq 0, \quad \text{et} \quad \phi_0(0) = b_0.$$

Suites engendrées :

$$\text{germe } t_0 \text{ et } t_{n+1} = \phi(t_n).$$

• **Générateurs inversifs sur les anneaux $\mathbf{Z}/2^l\mathbf{Z}$ des entiers modulo 2^l**

(Eichenhauer-Hermann et Grothe 1992).

Difficulté principale : inverser des diviseurs de zéro !

Un bel exemple issue des générateurs congruentiels de type inversif

→ Introduit vers 1987 par Eichenauer et Lehn :

Les acteurs :

– un corps fini K et son projectif $\mathbf{P}_1(K)$,

– une homographie : $f : x \mapsto \frac{ax+b}{cx+d}, \quad x \in \mathbf{P_1}, \quad ad-bc \neq 0, \ f(\infty) = a/c, \ldots$

– L'ICG : $n \mapsto f^n(x_0), \quad x_0 \in \mathbf{P_1}$

(f est supposée d'ordre $\#P_1(K)$: le polynôme caractéristique de la matrice compagnon est primitif).

• ***Nombreuses variations*** :

– cas multidimensionnel (F. Griffin, H. Niederreiter et Shparlinski, 1998-2001,...),

– Modulo M (H. Niederreiter, A. Winterhof, E. El-Mahassni, 2001-2005,...).

• Pour notre construction, on part de l' ICG introduit par J. Eichenaueur et H. Grothe (1992) :

– Anneau $R_{2^m} := \mathbf{Z}/2^m\mathbf{Z}$;

– notation $\nu(\cdot)$: valuation 2-adique sur $\mathbf{Z}$ ou R_{2^m} ou $\mathbf{Z}_2 = \varprojlim_m R_{2^m}$ (anneau des entiers 2-adiques), avec $\nu(0) = \infty$ dans tous les cas) ;

– avec a and b unités dans R_{2^m}, définir $T_{a,b} : R_{2^m} \to R_{2^m}$ par

$$\boxed{T_{a,b}(x) = 2^{\nu(x)} a (2^{-\nu(x)} \cdot x)^{-1} + b, \quad T_{a,b}(0) = b \qquad (*)}$$

où $x' = 2^{-\nu(x)} \cdot x$ signifie $x = 2^{\nu(x)} x'$ dans R_{2^m}.

Propriétés

– $T_{a,b}$ est bijectif ;

– $T_{a,b}(1 + 2R_{2^m}) = 2R_{2^m}$ et $T(2R_{2^m}) = 1 + 2R_{2^m}$;

– $T_{a,b}$ est d'ordre 2^m si et seulement si $\nu(a-1) \geq 2$.

Propriétés supplémentaires

Rappel

$$T_{a,b}(x) = 2^{\nu(x)} a (2^{-\nu(x)} \cdot x)^{-1} + b, \quad T_{a,b}(0) = b\,. \qquad (*)$$

• ***Faits élémentaires :***

– La formule $(*)$ définit $T_{a,b}$ sur $\mathbf{Z}_2$,

mais ici, $2^{-\nu(x)} \cdot x$ signifie de décaler le développement binaire de Hensel de x de $\nu(x)$-digits, *i.e.*

$$2^{-\nu(x)} \cdot (0, \ldots, 0, 1, x_{\nu+1}, x_{\nu+2}, \ldots) = (1, x_{\nu+1}, x_{\nu+2}, \ldots)$$

– Notons $[c_0 \ldots c_{k-1}] := \{x_0 x_1 x_2 \ldots \in \mathbf{Z}_2\,;\ c_0 \ldots c_{k-1} = x_0 \ldots x_{k-1}\}$ un cylindre typique de $\mathbf{Z}_2$, de taille k.

$\to$ Si $\nu(a-1) \geq 2$, alors $T_{a,b}$ agit comme une permutation circulaire sur l'ensemble des cylindres de taille k ; elle se superpose à la permutation correspondante sur l'ensemble des cylindres de taille $k-1$.

Conséquences

Theorem (P. L. non publié) *Supposons $\nu(1-a) \geq 2$. La transformation $T_{a,b}$: $\mathbf{Z}_2 \to \mathbf{Z}_2$ est continue, uniquement ergodique de mesure invariante la mesure de Haar ; elle est métriquement conjuguée à l'odomètre $x \mapsto x+1$.*

Preuve. Basée sur l'observation suivante : toute 2^k-ième racine de l'unité ζ est valeur propre de l'isométrie sur $L^2(\mathbf{Z}_2, \mu)$ associée à $T_{a,b}$, de fonction propre $f_k = \sum_{s=0}^{2^k-1} \bar{\zeta}^k \mathbf{1}_{[0^k]} \circ T_{a,b}^s$. ■

L'étude précédente donne :

Corollaire. *Pour tout $x \in \mathbf{Z}_2$, la suite $n \mapsto \varphi(T_{a,b}^n(x))$ est bien équirépartie mod 1 (de discrépance en $\mathcal{O}(\log N/N)$).*

Élargissement du domaine de construction

→ Il existe une classe d'anneaux incluant $\mathbf{Z}_{2^l}$ et $\mathbf{F}_{2^m}$:

• **L'anneau de Galois** $GR(2^l, m)$

Mise en place

→ Unique extension de Galois de degré m de l'anneau $\mathbf{Z}_{2^l}$.

Il est :

– de caractéristique 2^l,

– de taille 2^{lm}.

Exemples :

$$GR(2^l, 1) = \mathbf{Z}_{2^l},$$

$$GR(2, m) = \mathbf{F}_{2^m}.$$

→ Objet algébrique naturel pour étudier les récurrences linéaires sur $\mathbf{Z}_{2^l}$.

Construction pratique

$$GR(2^l, m) = \mathbf{Z}_{2^l}[X]/(h(X))$$

où $h(X)$ est irreductible, unitaire, de degré m ; relevé d'un polynôme de $\mathbf{F}_2[X]$.

• $R = GR(2^l, m)$ est un anneau local, d'idéal maximal (2), de corps résiduel $R/(2) = \mathbf{F}_{2^m}$.

Les représentants (dit de Teichmüller) des classes de ce quotient sont :

$$\mathcal{T} = \{0, 1, \xi, \xi^2, \ldots, \xi^{2^m-2}\} \text{ avec } \xi \text{ racine de } h.$$

Tout $x \in R$ admet un unique développement 2-adique

$$x = \sum_{i=0}^{l-1} t_i 2^i \quad \text{avec} \quad t_i \in \mathcal{T}\,.$$

Construction arithmétique

Soit K l'extension de $\mathbf{Q}_2$ non ramifiée de degré m et A son anneau des entiers. Alors

$$GR(2^l, m) = A/(2^l).$$

(K est l'extension cyclotomique d'ordre $2^m - 1$ de $\mathbf{Q}_2$, les représentants non nuls de Teichmüller sont les racines $2^m - 1$-ièmes de l'unité).

Avantage de cette construction :

– $GR(2^l, m)$ est une approximation du groupe compact A des entiers de K, utile pour une approche dynamique,

– se généralise à tout corps de nombres local k : extension non ramifiée de corps résiduel isomorphe à une extension donnée du corps résiduel de k.

Construction fondamentale

Fonction inversive sur les anneaux de Galois $R = GR(2^l, m)$

Tout élément de R s'écrit $2^e x$, avec $0 \leq e \leq l$ et $x \in R^*$ (groupe des inversibles) ;

$\rightarrow$ Construction : $\phi : R \rightarrow R$

$$\phi(2^e x) = 2^e a x^{-1} + b, \quad \text{et} \quad \phi(0) = b,$$

avec $a \in \mathcal{T}^*$ et $b \in \mathcal{T}^*$.

Suites engendrées : les orbites suivant ϕ.

Point de vue dynamique

$\rightarrow$ On fait tendre l vers l'infini (capture de l'anneau local A) ; l'application ϕ se prolonge en une transformation

$$T_\phi : A \rightarrow A\,.$$

Théorème (P. Liardet, non publié) *La transformation T est uniquement ergodique, isomorphe à l'odomètre $x \mapsto x + 1$.*

Pour en savoir plus sur les RNG

→ Les exemples donnés ci-dessus ont été choisis parmi une large famille de générateurs en usage dont :

LCG (Linear generator)

SRG (shift register generator)

ICG (Inversive Congruential Generator)

EICG (Explicit Inversive Congruential Generator)

→ Pour une description détaillée de nombreux types de générateurs de nombres et leurs propriétés, leur codes (en C), les tests statistiques usuels et des liens utiles sur ce sujet, consulter le site

http ://random.mat.sbg.ac.at/

→ certains de ces générateurs se prêtent bien à la construction de transformations uniquement ergodiques engendrant des suites dans $[0, 1[$ bien équiréparties mod 1 de faible discrépance.

Du côté cryptographie, que choisir ?

Plusieurs solutions ont été proposées (cas binaire). La plus classique :

Générateur pseudo-aléatoire Blum-Blum-Shub (BBS)

– "prouvé sûr" (cela sera expliqué dans le prochain exposé).

Définition

– choisir deux grands nombres premiers p et q congrus à 3 modulo 4.

– $n = pq$. Motivation : si $x^2 = a$ a une solution mod n, alors il y en a 4 et une seule est un carré. L'inverse de $z \mapsto z^2$ sur les carrés est alors

$$\xi \mapsto \xi^{\frac{(p-1)(q-1)+4}{8}} \pmod{n}\,.$$

– choisir un x_0, résidu quadratique modulo n ;

– construire une suite $(x_j)_j$ telle que $x_j = x_{j-1}^2 \pmod{n}$;

– sortir la suite binaire $b = b_0 b_1 b_2 \cdots$, définie par

$$b_j = x_j \pmod{2}\,.$$

Florilège de générateurs

→ Générateurs reposant sur la théorie des codes correcteurs, . . .

• Ils sont, comme le BBS, trop lents pour être utilisés dans de nombreuses applications pratiques (cartes à puces).

→ Générateurs basés sur les algorithmes de chiffrement par bloc : DES, AES, fonctions de Hachage,. . .

→ Multiples combinaisons, littérature abondante.

Nécessité de protéger le code de l'algorithme ou ses implémentations hard contre les *attaques par canaux cachés* qui analysent de possibles fuites d'information au cours du fonctionnement (attaques par consommation de puissance, par injections de fautes, sur transfert de bits locaux,. . .)

Diabolisation. Les algorithmes de chiffrement doivent aussi être protégés : masquage des données traitées par l'algorithme au moyen d'un "bon" générateur qui doit lui-même être protégé par ?...

Partie 3

Générateurs de nombres aléatoires pour Cryptographes seulement

Partie 4

Applications

3. Générateurs de nombres aléatoires pour Cryptographes seulement

La source "idéale" est une suite de générateurs aléatoires qui émet des suites de symboles de S ($\#S = q$), de longueur finie n, avec la distribution uniforme U_n.

Générateur cryptographiquement sûr

Rappel : *Un générateur aléatoire de type $h(\cdot)$ sur S est une application $G : S^* \to S^*$ telle que la restriction $G_{|S^n}$ de G à S^n, notée G_n, soit à valeurs dans $G^{h(n)}$ pour tout $n \in \mathbf{N}$.*

Ici $n \mapsto h(n)$ est supposée être une suite strictement monotone d'entiers, à croissance polynomiale et $h(1) > 1$.

$\to$ Définition de "générateur pseudo-aléatoire cryptographiquement sûr" d'après M. Blum et S. Micali (1982) :

Algorithme qui émet des suites de symboles dont la distribution est *indistinguable* de la distribution uniforme au moyen de tout algorithme polynomial probabiliste.

Précisons "distribution indistinguable" :

Soit $(\ell(n))_n$ une suite strictement croissante d'entiers naturels.

Définition. *Une suite de probabilités $(\mu_n)_n$ sur $S^{\ell(n)}$ est dite indistinguable (polynomialement) si pour tout algorithme probabiliste polynomial A (sortant 0 ou 1) et pour tout $t \geq 1$ on a*

$$|\mu_n(\{A(\cdot) = 1\}) - U_{\ell(n)}(\{A(\cdot) = 1\})| < \frac{1}{n^t}$$

pour tout n assez grand.

Par comparaisons avec la q-normalité caractérisée par la non prédiction par automates (cours (I)) on a

Caractérisation.

$(\mu_n)_n$ est **indistinguable** de la distribution uniforme si pour tout algorithme probabiliste A à valeurs dans S ($\#S = q$), il n'existe pas d'entier $t \geq 1$ tel que

$$\mu_k(\{A(s_1 \cdots s_{\ell(k)-1}) = s_{\ell(k)}\}) > \frac{1}{q} + \frac{1}{\ell(k)^t}$$

pour une infinité de k

Définition d'un générateur pseudo-aléatoire de nombres (en cryptographie) :

Algorithme polynomial déterministe $G : S^* \to S^*$ tel que pour toute entrée w de longueur $|w|$, la sortie $G(w)$ est de longueur

$$|G(w)| = h(|w|)$$

et de plus,

$\to$ *la suite des distributions μ_n déterminées par G sur $S^{\ell(n)}$ est indistinguable de la distribution uniforme.*

• En 1982, Yao matérialise cette notion dans le cas binaire ($S = \{0, 1\}$). Pour cela il se restreint aux familles $\mathcal{C}$ de tests statistiques booléens C_ℓ, $\ell = 1, 2, 3, \ldots$

$\to$ C_ℓ est une fonction booléenne construite à partir de $\wedge$, $\vee$, $\neg$ et de ℓ variables booléennes, de sorte que le nombre de signes opératoires soit polynomial en ℓ ($\sim$ nombre de portes en version câblée).

Définition (Yao). *Le générateur G de type h est dit pseudo-aléatoire polynomial (non-uniforme) si :*

(1) il existe une algorithme qui calcule $G_k(x)$ pour (x, k) donné en entrée ($x \in \{0, 1\}^k$) ;

(2) pour toute famille $\mathcal{C}$ de circuits booléens, la suite des probabilités $\mu_n = U_n \circ G_n^{-1}$ est indistinguable, aux moyens des circuits de $\mathcal{C}$, de la distribution uniforme $U_{h(n)}$: pour tout $t \geq 1$,

$$|\mu_n(C_n = 1) - U_{h(n)}(C_n = 1)| < n^{-t}$$

pour tout n assez grand.

Construire des générateurs de nombres cryptographiquement sûr ?

Revient, en pratique, à construire des applications à sens-unique

$$G_n : S^n \to S^{h(n)}$$

au sens algorithmique :

$\to$ étant donnée une valeur image $y = G_n(x)$ (aléatoire) la valeur de x ne peut pas être calculée au moyen d'un algorithme polynomial en la longueur de y.

Exemple 1 : le générateur RSA

Initialisation

- Choisir deux grands nombres premiers p et q et calculer n (grand signifie ≥ 1024 digits) et $\phi = (p-1)(q-1)$.

- Choisir e, $1 < e < \phi$ (par exemple $e = 3$ pour faciliter les calculs d'exponentiation).

Générateur

DÉBUT

1. Choisir (au hasard) x_0, $1 \leq x_0 < n$

2. Calculer successivement

$$x_{i+1} = x_i^e \pmod{n} \quad (1 \leq i \leq L)$$

3. sortir $z_1 = x_1 \pmod 2, \ldots, z_L = x_L \pmod 2$.

FIN

La sécurité du générateur repose sur l'impossibilité actuelle de factoriser n en un temps polynomial en $\log_2 n$.

Exemple 2 : le BBS (déjà vu)

C'est un RSA avec :

- $e = 2$,
- p et q sont congrus à 3 modulo 4.
- x_2 est un carré.

BBS est Considéré comme cryptographiquement sûr, mais, comme le RSA :

- il est lent (même en utilisant le théorème des restes chinois),
- certaines valeurs initiales sont à éliminer (comme par exemple $x_0 = 1$).
- La longueur de période $T(x_0)$ des sorties $z_i = x_i \pmod 2$ est un diviseur de $\lambda(\lambda(n))$ où $\lambda(\cdot)$ est la fonction de Carmichael.

$$\lambda(2^e p_1^{e_1} \dots p_k^{e_k}) = \text{ppcm}\{(\mathbf{Z}/2^e\mathbf{Z})^*, (\mathbf{Z}/p_1^{e_1}\mathbf{Z})^*, \dots, (\mathbf{Z}/p_k^{e_k}\mathbf{Z})^*\}.$$

- Si 2 est d'ordre $\lambda(\lambda(n))$ modulo $\lambda(n)/2$ et si x_0 est d'ordre $\lambda(n)/2$ modulo n, alors

$$T(x_0) = \lambda(\lambda(n))\,.$$

Alternatives aux algorithmes

• Utiliser des propriétés physiques, notamment les sources d'aléas produites par des architectures électroniques sur silicium ou les états instables des données d'entrée et de communications internes dans un ordinateur.

→ But : extraire de l'entropie à partir de bruitages (agitation thermique, flicker,. . .)

Exemple : modélisation sur silicium

Système équivalent à un circuit CR intégrant

– un générateur de tension $V(t,\omega)$ supposé être une version continue d'un bruit blanc,

– une tension d'entrée V_e,

– une tension de sortie V_s.

L'état du circuit est classiquement donné par le système d'équations :

$$V_e - V_s = Ri - V(t,\omega), \quad V_s = q/C\,, \quad i = \frac{dq}{dt}$$

Solution :

$$\boxed{q(t,\omega) = q_0 e^{-\frac{T}{RC}} + \frac{1}{R}\int_0^t \exp\Big(-\frac{t-u}{RC}\Big)(V_e + V(u,\omega))du.}$$

La partie stochastique

$$B(t,\omega) = \frac{1}{R}\int_0^t \exp\Big(-\frac{t-u}{RC}\Big)V(u,\omega)du$$

représente la contribution des sources de bruit.

$\rightarrow$ Nécessité de traiter le signal de sortie (débiaisage) et de valider à la volée le bon fonctionnement du générateur “hard” (détection des pannes).

$\rightarrow$ **Techniques d’avenir.**

Dans les applications

A) On dispose habituellement d'un générateur binaire natif ;

Problème : dans de nombreux cas pratiques, on recherche un générateur sur un nombre q de symboles qui peut ne pas être une puissance de deux.

Solutions de deux types :

1) Transformer un générateur (pseudo-aléatoire) binaire en un générateur (pseudo-aléatoire) q-aire aux moyens d'algorithmes type *Las Vegas* ;

2) Approcher arbitrairement un générateur (pseudo-aléatoire) q-aire par un générateur (pseudo-aléatoire) binaire aux moyens de chaînes de Markov.

Ou troisième solution : utiliser une dynamique non inversible d'entropie strictement positive (machine fractale).

Une quatrième ?

B) Dans l'implantation d'un processus cryptographique : nécessité de mettre en place des contre-mesures pour contrer les attaques par canaux cachés.

• • • ***Comme applications, nous allons examiner deux cas "typiques".***

4. Applications

Architecture client/serveurs et marches aléatoires sur groupes

Le problème client/serveur

• Le serveur, seul, est un point faible :

– peut être corrompu, victime d'attaque par déni de service ;

– problèmes de pannes diverses (hardware ou software).

• Comment se prémunir de ces échecs ?

– une solution simple : utiliser plusieurs serveurs (disons n) ;

– renforcer la sécurité en faisant intervenir aléatoirement k serveurs parmi les n.

• Réalisation ?

– Par la construction d'un générateur aléatoire "k parmi n", au moyen d'un générateur binaire.

Solutions possibles

Nombreuses, par exemple :

– en choisir un, puis un autre,. . .

– algorithme probabiliste basé sur l'algorithme de mélange de Fisher-Yates ;

– algorithme RANKSB de Nijenhuis and Wilf (1978).

→ Ces algorithmes conduisent à un biaisage important vis-à-vis de la distribution uniforme.

Solution proposée (A. Bonnecaze et P. Liardet (2011))

– Basée sur certains codes correcteurs.

– Essentiellement pour tous codes admettant des designs (voir plus loin dans le cas de systèmes de Steiner associés).

– La méthode fait usage d'une marche aléatoire sur le groupe des automorphismes du code.

Un exemple explicite

• Le code de service : code binaire de Golay étendu $\mathcal{G}_{24}$: c'est un code parfait, de longueur 24, de dimension 12, de poids minimal 8 (poids des mots : 0, 8, 12 ou 24),

• pour construire un générateur "5 parmi 24".

Rappels brefs sur les codes et designs

– Un code binaire linéaire $\mathcal{C}$ de longueur n est un sous-espace vectoriel de $\mathbf{F}_2^n$.

– Les paramètres de $\mathcal{C}$ sont $[n, k, d]$ où k est la dimension vectorielle et d le plus petit poids de Hamming non nul.

– Soit Y un v-ensemble (ensemble de v éléments).

• Un $t-(v, k, \lambda)$ design est un ensemble de k-sous-ensembles (les blocs) de Y tel que tout t-sous-ensemble de Y est recouvert par exactement λ blocs.

– Système de Steiner $S(t, k, v)$: c'est un $t-(v, k, 1)$ design.

Exemple d'un Système de Steiner :

Basé sur le code binaire de Golay $\mathcal{G}_{24}$:

– les octades (blocs) sont les mots du code de poids 8 (les 8-sous-ensembles) ;

– Tout vecteur binaire de longueur 24 ($v = 24$) de poids 5 (5-sous-ensemble) est recouvert par exactement une octade de $\mathcal{G}_{24}$.

Le code de Golay construit donc un système de Steiner $S(5, 8, 24)$.

Ici

$Y = \{1, \ldots 24\}$ et les blocs sont les parties de Y construites à partir des positions (indices) des 1 dans les octades.

Illustration :

Un 5-ensemble : (0100000100101000000000100)

L'octade le recouvrant : (0100100100101100010000100)

Groupes des automorphismes de $\mathcal{G}_{24}$

C'est le groupe de Mathieu M_{24} ; $\#M_{24} = 210.33.23.11.7.5$. Il est 5-transitif sur les octades.

Il est engendré par 4 permutations opérant sur les 24 coordonnées du vecteur-code indexée sur l'espace projectif $\mathbf{F}_{23} \cup \{\infty\}$:

$S : i \mapsto i + 1$

$V : i \mapsto 2i$

$U : i \mapsto -1/i$

$W(\infty) = 0$,

$W(0) = \infty$,

$W(i) = -(i/2)^2$ si i est un résidu quadratique modulo 23,

$W(i) = (2i)^2$ sinon.

Marche aléatoire sur un groupe fini G

Elle est donnée par

- un probabilité $Q := \{Q(g)\,;\ g \in G\}$;
- Une chaîne de Markov homogène :
  - espace des états : G ;
  - probabilité initiale : Q,
  - matrice de transition $T_{g,h} = Q(gh^{-1})$.

La marche M est construite au moyen d'une suites de variables aléatoires i.i.d. à valeurs dans G, de loi Q :

$$M_{n+1} = X_n M_n$$

Faits :

– Si $\{g\,;\ Q(g) > 0\}$ engendre G, la chaîne est irréductible et

– sa distribution stationnaire est la probabilité uniforme $U(G)$ sur G ;

– si la chaîne est irréductible et apériodique (i.e. mélangeante), la loi de distribution de M_n converge (pour la variation totale) vers $U(G)$ à vitesse exponentielle.

Problèmes

– Trouver une majoration effective de cette rapidité de convergence.

– Si le groupe est "gros" avec un système E de générateurs "petit" et s'il opère sur un ensemble $\mathcal{M}$ "pas trop gros", alors on peut généralement répondre de manière satisfaisante au problème précédent...

$\rightarrow$ en effectuant une marche markovienne homogène sur un ensemble fini $\mathcal{M}$ avec un ensemble E de bijections Y de $\mathcal{M}$. La marche est symétrique si E est stable en passant aux bijections réciproques. Posons

$$\mu := \#\mathcal{M}\,, \qquad \chi := \#E\,.$$

La matrice de transition (stochastique symétrique) est :

$$T = \frac{1}{\chi} \sum_{Y \in E} \mathbf{Y}$$

où $\mathbf{Y}$ est la matrice représentant la bijection Y sur la base canonique de $\mathbf{R}^{\mathcal{M}}$.

• La matrice T a μ valeurs propres

$$-1 < \lambda_{\mu-1} \leq \cdots \leq \lambda_1 = \rho < \lambda_0 = 1\,.$$

Théorème. *On a*

$$-1+\frac{2}{\chi}\leq\lambda_{\mu-1}\quad et\quad \rho<1-\frac{2}{\kappa(\kappa+1)\chi}$$

où κ est le plus petit nombre de pas nécessaire pour aller de tout état (dans $\mathcal{M}$) à tout autre état.

La première inégalité est facile, la seconde est laborieuse à établir.

Mode d'emploi

Introduisons pour deux probabilités p, p' sur $\mathcal{M}$ la distance (variation totale)

$$d(p,p')=\frac{1}{2}\sum_{m\in\mathcal{M}}|p(m)-p'(m)|\,.$$

Soit $U(\mathcal{M})$ la probabilité uniforme sur $\mathcal{M}$ et $P_m^{(n)}$ la distribution de la marche issue de m après n pas et posons

$$d(n)=\max_{m\in\mathcal{M}}d(P_m^{(n)},U(\mathcal{M}))\,.$$

Alors

$$\boxed{d(n)\leq\tfrac{\sqrt{\mu}}{2}\left(1-\tfrac{4}{\kappa(\kappa+1)\chi}\right)^n}$$

Ou encore, pour mettre en évidence la vitesse exponentielle :

$$\boxed{d(\lceil ab + b\gamma \rceil) \leq e^{-\gamma}}$$

avec

$$a = \frac{1}{2} \log \mu - \log 2$$

et

$$b = -\frac{1}{\log\left(1 - \frac{4}{\kappa(1+\kappa)\chi}\right)} .$$

Cas du Golay

– Il conduit à l'approximation de la distribution uniforme de "5 parmi 24"

– Le groupe M_{24} est trop gros ;

– On remplace la marche sur le groupe par son image sur les blocs du système de Steiner $S(5, 8, 24)$;

– La marche markovienne s'effectue sur les octades de manière symétrique en utilisant l'action des automorphismes pris aléatoirement dans

$$E := \{I, S, S^{-1}, U, V, V^{-1}, W, W^{-1}\}\,.$$

Paramètres : $\mu = 759$ (octades), $\chi = 8$ $(= \#E)$, $\kappa = 7$. D'où, explicitement,

$$\boxed{d(292 + 111\gamma) \leq e^{-\gamma}}$$

Algorithme G5-24(N)

INPUT : N

OUTPUT : un vecteur binaire de poids 5, de longueur 24

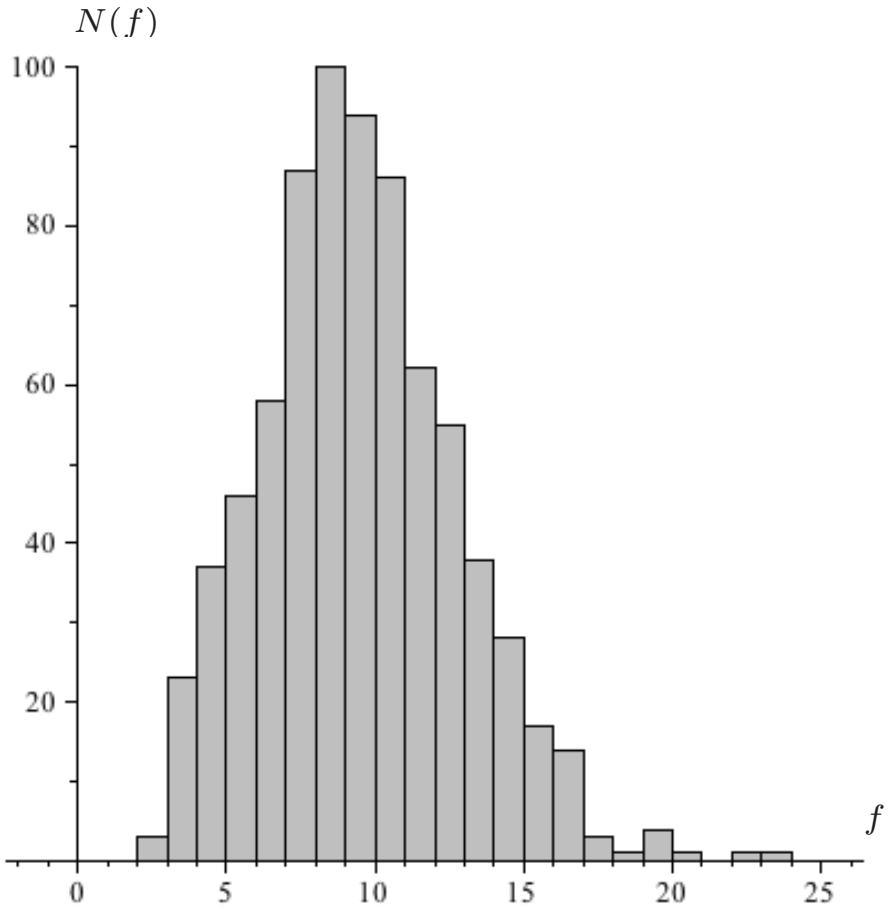
(a) choisir une octade m

(b) remplacer m par m' par mise à 0 des 3 premiers 1

(c) faire agir aléatoirement sur m' les éléments de E

(d) répéter N fois et sortir le mot m'.

Pour la mis en oeuvre de l'algorithme, on a utilisé le générateur BBS.



Le diagramme suivant représente les résultats statistiques obtenus avec une marche très courte : $N(f)$ est le nombre d'octades obtenues f fois au cours de 7590 marches de longueur 11.

Autres exemples

Générateur “5 parmi 12”

On utilise le code de Golay ternaire, code sur $GF(3)$ de paramètre $[12, 6, 6]$. Il conduit à un $5-(12, 6, 1)$ design permettant de construire, par la méthode décrite, un générateur “5 parmi 12”. L'étude de l'action du groupe des automorphismes donne $\chi = 8$, $\kappa = 6$, la vitesse de convergence vers la distribution uniforme est donnée par :

$$d(146 + 83\gamma) \leq e^{-\gamma}\,.$$

Générateur “2 parmi 31”

Ce générateur est basé sur un $2-(31, 7, 7)$ design et utilise un générateur “2 parmi 7” grâce à un $2-(7, 3, 1)$ désign.

Générateur “3 parmi 16”

Il utilise un $3-(16, 12, 55)$ design. Dans ce cas, $\chi = 8$ et $\kappa = 6$ et

$$d(148 + 83\gamma) \leq e^{-\gamma}\,.$$

Contre-mesure pour l'AES

L'AES (Advanced Encryption Standard)

– Lauréat 2000 du concours "AES"

– Algorithme symétrique de chiffrement proposé par J. Daemon et V. Rijmen ;

– input/output : blocs de 128 bits ;

– clefs de 128, 192 ou 256 bits ;

– à ce jour, attaque exhaustive hors d'atteinte ;

– utilisé dans de nombreux systèmes embarqués ;

– nombreux travaux sur les contre-mesures pour les attaques par canaux cachés (side-channel attacks) ;

– Les méthodes classiques de contre-mesures consistent à randomiser les résultats intermédiaires.

Description rapide de l'AES

• Le chiffrement des 128 bits :

– Input/output : message découpé en blocs de 128 bits. Chaque bloc (état) est découpé en octets (8 bits) placés dans un tableau T de 4 lignes en 4 colonnes ; même découpage K pour la clef en 4 lignes de 4, 6 ou 8 colonnes selon la taille de la clef ;

– un octet correspond à un élément du $GF(2)$-espace vectoriel $GF(2^8) = GF(2)[X]/(X^8 + X^4 + X^3 + X + 1)$;

– L'AES effectue plusieurs tours (10, 12 ou 14 selon la taille de la clef) constitués de transformations élémentaires :

`SubBytes` (SB(T))

`ShiftRows` (SR(T))

`MixColumns` (MC(T))

`AddRoundKey` (ARK(T,K))

`KeysExpansion` (KE(K))

– Initiation : entrée du message en tableau T et de la clef en tableau K : exécution sur K de KE puis ARK sur (T, K) ;

– premier tour : SB $\to$ SR $\to$ MC $\to$ ARK ;

– répétions du premier tour jusqu'à l'avant dernier ;

– dernier tour : SB $\to$ SR $\to$ ARK

• La transformation `SubBytes` n'est pas linéaire, c'est elle qui nous intéresse, elle compose

– une fonction inversion g : chaque octet x (dans $GF(2^8)$) non nul de T est remplacé par son inverse $g(x)$, l'octet nul est conservé $(g(0) = 0)$;

– une fonction affine $f(x) = Ax + c$.

Résultat sur chaque octet $h : x \mapsto A(g(x)) + c$. En pratique courante, h est donnée par une table (Sbox).

• Les transformations `ShiftRows`, `MixColumns`, `AddRoundKey` sur T sont linéaires.

Petit inventaire des attaques par canaux cachés pour une carte à puce :

– ***Attaques passives***

- Consommation de courant (simple / différentielle à clair choisi ou connu)
- Émissions électromagnétiques, température de fonctionnement,...
- Temps de calcul

– ***Attaques actives***

- Invasives/ semi-invasive / non-invasives
- Injection de fautes

– ***Méthodes statistiques mises en oeuvre (traitement du signal)***

- Méthode différentielle (DPA, Kocher 1999)
- Par coefficients de corrélation (CPA, Brier 2004)
- Entropie : attaque par information mutuelle (MIA, Gierlichs 2008)

Procédures générales de masquage

BUT :

– masquer les calculs, unifier leur temps d'exécution,

– randomiser les valeurs intermédiaires v par une combinaison $v' = C(v, m)$ où m est une valeur aléatoire.

En général : $C(v, m) = c \oplus m$ (dans le corps $GF(2^8)$).

Problème : à l'issue des tours, revenir à la vraie valeur de sortie.

– Pas de difficulté pour une transformation linéaire F :

$$F(v \oplus m) \ominus F(m) = F(v)\,.$$

– Pour SB, les travaux rivalisent d'imagination.

Notre proposition pour masquer SB( ) (A. Bonnecaze, P. Liardet, A. Venelli (2012)) :

• Au lieu de représenter $GF(2^8)$ par $GF(2)(G)[X]/Q$, avec le polynôme $R = X^8 + X^4 + X^3 + X + 1$, on randomise la construction de $GF(2^8)$ en des tours d'extensions : une extension biquadratique (fixée) suivie d'extensions quadratiques :

$$GF(2^8) \simeq GF(2)(Q)[z]/(P(z))$$

– Il y a 3 polynômes biquadratiques irréductibles sur $GF(2)$ dont deux primitifs ;

– $P(z) = z^2 + \psi z + \lambda$ irréductible sur $GF(2)(Q)[z]$; il y en a 120.

Le calcul de l'inverse se fait en soft avec ou sans l'aide d'une table peu encombrante (inversion dans $GF(2)(Q)$) :

– le calcul de l'inverse de x fait intervenir le calcul de la norme de x dans l'extension $GF(2^8)/GF(2)(Q)$.

• La construction de $GF(2^8)$ revient à construire une base, écrite dans la représentation $GF(2)[X]/(R)$, sous la forme

$$L(\xi, \gamma) := [\xi^3\gamma, \xi^2\gamma, \xi\gamma, \gamma, \xi^3, \xi^2, \xi, 1]$$

• Les valeurs des normes sont mal distribuées dans cette randomisation, on les masque en remplaçant x par xm.

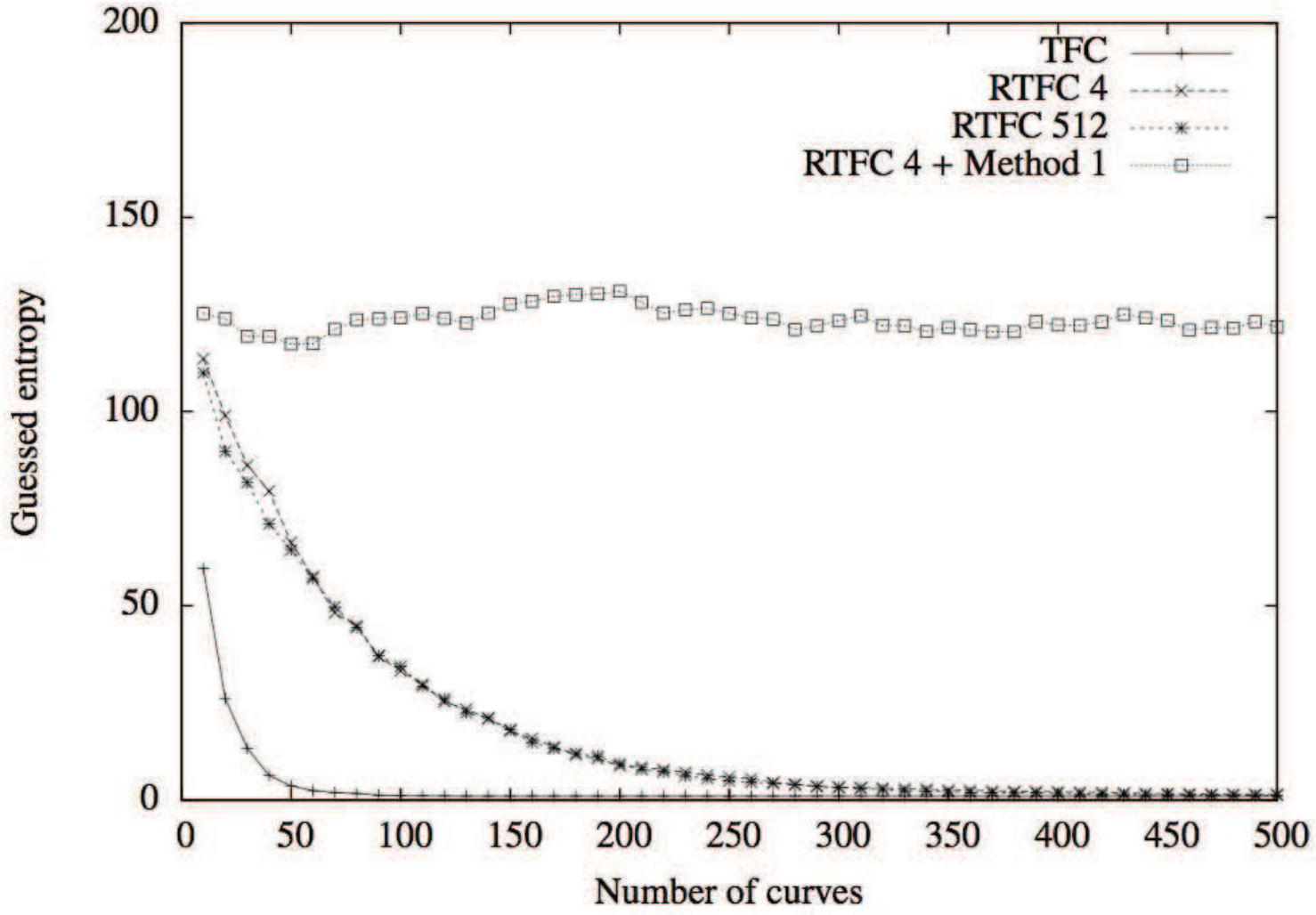
– méthode 1 : $m = uv$ sans couvrir tous les cas mais suffisamment ;

– méthode 2 : m est choisi aléatoirement dans un système complet de représentants des classes cyclotomiques (la norme est constante sur ces classes).

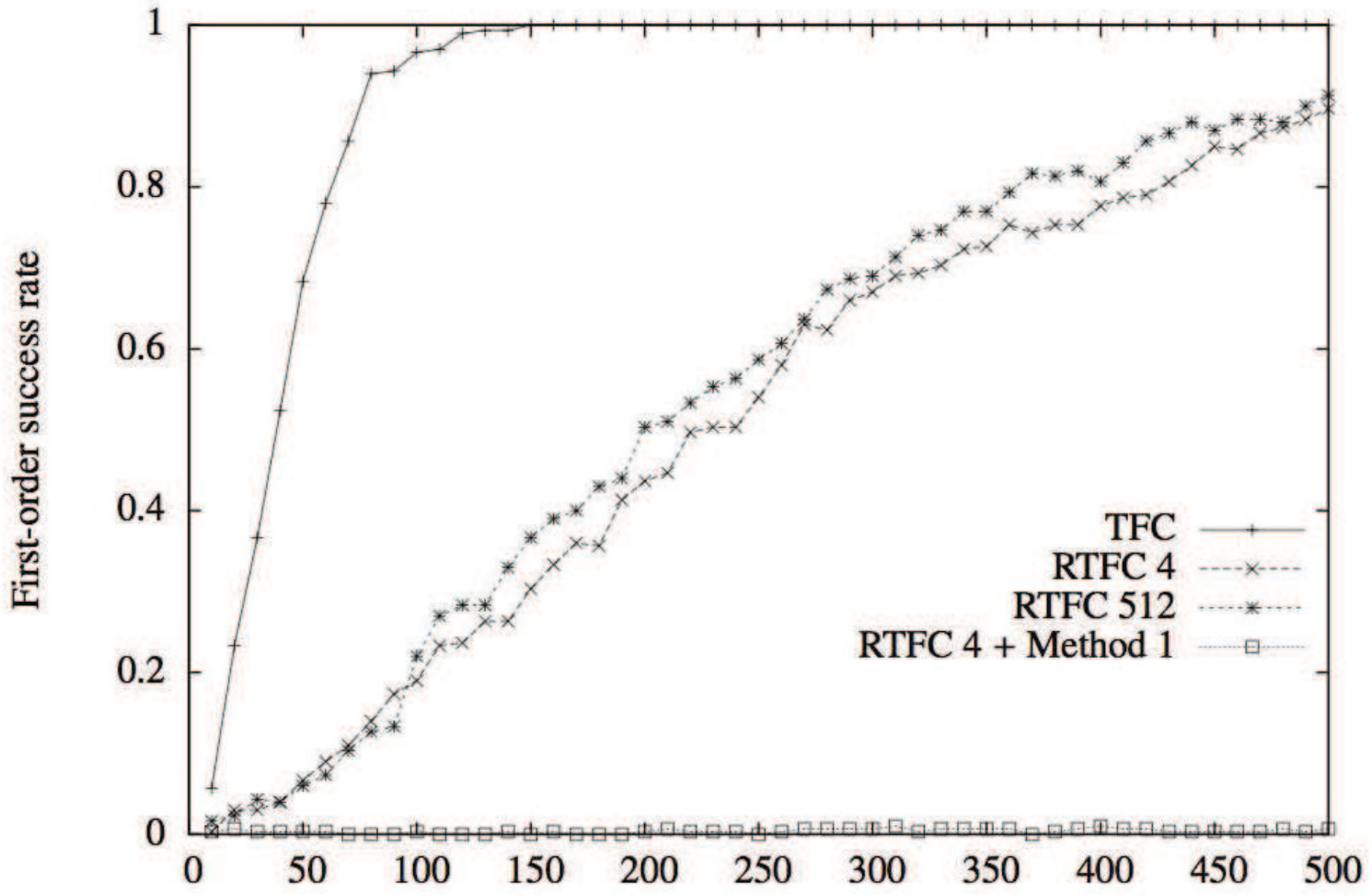
• La méthode a été implantée sur carte à puce et testée statistiquement pour deux types classiques d'attaque (par entropie, par corrélation) en comparant quatre mise en oeuvre des tours d'extensions.

Implémentations sur AVR 8 bits ATmega 256 processsor.

Resultats : comparaison entre : la méthode par une tour d'extensions (TFC), la randomisation par 4 tours d'extensions (RTFC 4), la randomisation avec 512 tours d'extensions (RTFC 512) et la randomisation 4 tours+masquage des normes type 1 (RTFC 4+ Method 1). Cette dernière est la plus performante.



Side-channel attacks using guessed entropy



Side-channel attacks using guessedfirst-order success rate metrics

RÉFERENCES :

A. Bonnecaze et P. Liardet : *Uniform Generators and Combiatorial Designs*, International J. on Advances in Networks and Services, vol 4, No 1 & 2 (2011),

A. Bonnecaze, P. Liardet et A. Vinelli : *AES Side-Channel Countermeasure using Random Tower Field Constructions*, Designs, Codes ans Cryptography (2012).
