UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

ECOLE DOCTORALE DE MATHEMATIQUES ET INFORMATIQUE

Année : 2015-2016 **N° d'ordre :**

THESE DE DOCTORAT UNIQUE EN MATHEMATIQUES

Mention : Mathématiques et Modélisation

(Spécialité : Codage-Cryptographie-Algèbre-Applications)

Présentée par : Babacar Alassane NDAW

Titre

Générateurs d'Aléa et Sécurité : Loi des Présences – Conception d'un Distingueur d'Aléa.

Soutenue publiquement le 03 Décembre 2016 devant le jury :

Président : Cheikh Thiécoumba GUÈYE Professeur, UCAD

Rapporteurs :
- Robert ROLLAND Chercheur, Institut de Mathématiques, Université d'Aix Marseille (France)
-Abdelmalek AZIZI Professeur, Université Mohamed Premier (Maroc)

Examinateurs :
- Oumar DIANKHA Professeur, UCAD
- Salimata guèye DIAGNE Professeur, UCAD

Directeur : Mamadou SANGHARE, Professeur, UCAD (Sénégal)

Remerciements

Je voudrais d'abord exprimer ma reconnaissance aux Autorités de la Présidence de la République du Sénégal qui ont marqué leur accord pour l'établissement et le renforcement d'un partenariat en matière de « recherche-développement », dans le domaine de la cryptologie, entre la Commission Nationale de Cryptologie-CNC), et le Laboratoire d'Algèbre, de Cryptographie, de Géométrie algébrique et Applications (LACGAA) de la Faculté des Sciences et Techniques.

J'adresse mes chaleureux remerciements, en premier lieu, au Professeur Mamadou SANGHARE, Directeur Général de l'Enseignement Supérieur, et Directeur du Laboratoire d'Algèbre, de Cryptographie, de Géométrie algébrique et Applications (LACGAA), mon Directeur de Thèse qui m'a guidé, appuyé et soutenu tout au long de mes travaux de recherches et de publication. J'ai pu avancer et terminer grâce à son très généreux et précieux concours.

Il m'est agréable de remercier également le Professeur Joseph SARR, Doyen de la Faculté des Sciences et Techniques, le Professeur Mamadou BARRY, Chef du Département de Mathématiques et Informatique ainsi que le Professeur Hamidou DATHE, Directeur de l'Ecole doctorale de mathématiques et Informatique qui m'a constamment encouragé et poussé à chaque fois que nous nous rencontrions.

Je remercie aussi les Professeurs Robert Rolland (Université d'Aix Marseille en France) et Abdelmalek Aziz (Université Mohamed Premier au Maroc) pour les remarques pertinentes ainsi que les très utiles conseils qu'ils ont formulés, lors de l'examen de la thèse et de la rédaction de leurs rapports.

Je tiens à remercier particulièrement le Professeur Cheikh Thiécoumba GUEYE pour sa constante sollicitude, son attention critique ainsi que ses observations et recommandations sur les travaux de publication et l'intégralité de la thèse.

Je n'oublie pas les Professeurs Salimata Gueye Diagne et Omar DIANKHA ainsi que l'ensemble de leurs collègues et amis du laboratoire (LACGAA) qui m'ont aimablement accueilli et facilité la tâche.

Table des matières

Liste des tableaux

Liste des figures

Avant-propos

A la fin du XIXème siècle, Auguste Kerckhoffs a posé dans son traité « La cryptographie militaire-1883 » des principes dits « principes de Kerckhoffs » qui stipulent notamment que la sécurité d'un système de chiffrement ne doit reposer que sur le secret de la clé partagée par les correspondants.

Aujourd'hui, les principes de Kerckhoffs sont compris comme des principes qui, non seulement préconisent que la sécurité ne doit pas dépendre du secret des algorithmes utilisés, mais, également, exigent que ces algorithmes soient rendus publics, ce qui permet d'établir des normes internationales.

La production de clés de chiffrement, attachées à des quantités aléatoires imprévisibles (suite de nombres, de bits aléatoires ou pseudo-aléatoires) est la base de la plupart des crypto systèmes cryptographiquement sûrs.

Ainsi, l'aléa occupe une place majeure en cryptographie, pour chiffrer et déchiffrer les messages, et pour faire également des attaques sur les crypto systèmes. Ce besoin d'aléa s'applique aussi à plusieurs applications informatiques et à internet.

Généralement, les suites utilisées doivent être produites par des générateurs de bits aléatoires (RBG-Random Bits Generators), non déterministes, des générateurs de bits pseudo-aléatoires (PRBG-Pseudo Random Bit Generators), déterministes, et des générateurs hybrides, qui doivent tous posséder de bonnes propriétés statistiques permettant de statuer sur leur degré de sécurité cryptographique.

La présente thèse a justement pour objectifs :

- d'étudier les générateurs de bits pseudo-aléatoires et leur sécurité,
- d'étudier et d'établir une loi de probabilité que nous avons appelée « *Loi des présences* » ainsi que « *Le Théorème des présences* », dans l'hypothèse où nous disposons d'un générateur qui délivre une suite uniforme, aléatoire ; le nombre de symboles présents dans la suite, après suppression des répétitions, étant une variable aléatoire ;
- et de concevoir un distingueur (ou différenciateur) d'aléa basé sur la loi des présences, qui permet de statuer sur l'uniformité ou non des suites chiffrantes, et partant d'évaluer la qualité cryptographique des générateurs d'aléas qui les produisent. Il s'agit d'une bonne méthode originale de test statistique d'aléa pour distinguer une distribution donnée d'une distribution aléatoire (ou pseudo aléatoire).

Chapitre 1

Introduction

Sommaire

1.1 Rappels historiques : la cryptologie d'hier et d'aujourd'hui [1] [2]

1.1.1 L'antiquité

La nécessité de correspondre de façon secrète est aussi ancienne que l'homme. En effet, c'est au troisième millénaire avant J.C. que naquirent à peu près en même temps que les différentes civilisations- chinoises, égyptiennes, phéniciennes, sumériennes - des écritures symboliques, alphabétiques ou hiéroglyphiques.

Ces écritures permettaient à ceux qui en étaient initiés d'emmagasiner leurs connaissances et de correspondre.
Les moyens les plus répandus :

- le sens, par exemple, de certains dessins primitifs - objets ou animaux - n'était connu que des seuls utilisateurs ;

- les écritures invisibles obtenues par l'emploi d'encres sympathiques.
- les écritures dissimulées qui camouflent le secret dans des textes anodins, incapables d'éveiller le moindre soupçon.

Quelques points repères :

- Plutarque - écrivain grec (vers 50 av J.C.) nous a signalé dans la vie de Lysandre, général spartiate (vers 395 av. J.C), des documents concernant l'utilisation d'un moyen de chiffrement appelé « la scytale des lacédémoniens » :

Figure 1.1 : La scytale des lacédémoniens

De là naquit le premier principe de chiffrement : « le principe de la transposition », qui évoque l'idée de mélange des lettres du texte clair.

- Suétone- historien en 69 après J.C. - faisait remarquer qu'en 56 avant J.C., Jules César correspondait avec ses partisans en utilisant un moyen de chiffrement simple qui consistait en un décalage constant des lettres de l'alphabet normal.

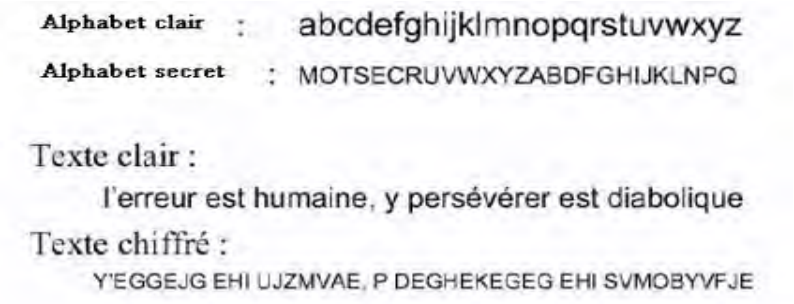

Figure 1.2 : Chiffrement de César

De là naquit le deuxième principe de chiffrement : « le principe de la substitution » qui évoque l'idée de remplacement de chaque lettre du texte clair par une lettre secrète.

1.1.2 La renaissance

L'usage du chiffre se répand ensuite dans toutes les chancelleries européennes et jusqu'au XVIIIème siècle de nouveaux procédés voient le jour (Ex : Le Cadran d'Alberti- architecte florentin, mort en 1472 ; le Carré de Vigenère- Diplomate français en 1560 ; la Grille de Cardan -savant italien vers 1550.

1.1.3 Le XVIIIème siècle :

Le XVIIIème siècle est une période sans éclat particulier, mais il y'a eu quelques inventions comme le cylindre du Secrétaire d'Etat Thomas Jefferson utilisé par l'armée américaine (1790-1800).

1.1.4 Le XIXème siècle

- Le " chiffre des francs-maçons" est utilisé à partir de 1815 même s'il est loin d'être original.
- En 1857, l'Amiral anglais Sir Francis Beaufort imagine un procédé de chiffrement qui porte encore son nom et qui est une variante simple du mode d'utilisation du « Carré de Vigenère ».
- La fin du XIXème siècle est très féconde. l'expérience acquise par les spécialistes à l'occasion des guerres de sécession (1860 -1865), franco-allemande (1870 - 1871) et russo-turque (1877 - 1878) semble avoir été un aliment. Pendant ces campagnes, le chiffre a eu, en effet, un rôle considérable.
- A Partir de 1880, la cryptologie est officiellement enseignée dans toutes les écoles militaires. Saint Cyr fait connaître la "réglette de Saint-Cyr" qui est à la fois une modernisation du « Jules César » et une simplification ingénieuse du « carré de Vigenère ».
- Le chiffre devient, par la force même des choses, un auxiliaire indispensable de l'art militaire ; il est et restera longtemps l'apanage des militaires.

1.1.5 Les XX et XXIème siècles

- Pendant la première guerre mondiale, on assiste à une véritable bataille sur le plan technique.
- Georges Painvin (Professeur de Paléontologie et Capitaine de réserve d'artillerie de l'Armée française) décrypte le Chiffre Allemand : ADFGX ; ADFGVX), ce qui contribuera à la victoire des alliés en 1914-1918
- Dès 1918 se sont multipliés les travaux sérieux, articles et ouvrages.
- Vers 1925, s'est annoncé l'ère des machines à chiffrer avec le dépôt de brevet (HAGELIN en Suède).
- Pendant la seconde guerre mondiale, les Allemands utilisèrent la machine à chiffrer Enigma pour chiffrer leurs messages.
- Le développement du Chiffre s'est intensifié après la deuxième guerre mondiale mettant à profit l'électricité, l'électronique, l'informatique, la linguistique, les mathématiques, ce qui a contribué, vers 1960, à l'épanouissement de la cryptologie.
- IBM lança un programme de recherche sur le chiffrement des données informatiques, et en 1975, le résultat est proposé aux banques : il s'agit du système de Chiffrement à clés secrètes DES (Data Encryption Standard algorithme de chiffrement des données par bloc de 64 bits).
- En 1976, deux chercheurs américains, Whitfield Diffie et Martin Hellman, de l'Université de Stanford, membres de l'IEEE, proposent dans un article « *New Directions in cryptography* », un protocole révolutionnaire d'échange de clefs de chiffrement entre correspondants par la seule connaissance de leurs données publiques. C'est ainsi qu'en 1979 naquit le premier système de chiffrement à clés publiques RSA, au Weizmann Institute en Israël, du nom de ses inventeurs (Ronald Rivest, Adi Shamir et Léonard Adleman), précurseur de la cryptologie non symétrique, encore appelée cryptologie à clés publiques, qui marque ainsi véritablement la naissance de la cryptologie moderne.
- De 1979 à nos jours, les recherches se sont intensifiées et la cryptologie a connu un bond majeur (avec notamment la formalisation des notions de sécurité, la cryptographie homomorphique, la cryptographie embarquée, la cryptographie post-quantique : cryptographie basée sur les codes correcteurs, cryptographie sur les réseaux arithmétiques, cryptographie multivariable,…) en offrant des services de sécurité permettant de faire face aux nouvelles menaces liées au développement des réseaux informatiques et à la dématérialisation des échanges.

1.2 Les services de sécurité :

Les services de sécurité regroupent la confidentialité, l'intégrité, l'authentification, la non-répudiation et la signature numérique.

1.2.1 **La confidentialité** : C'est un service qui permet d'assurer le secret des informations afin qu'elles ne soient ni rendues accessibles, ni divulguées à un utilisateur, une entité ou un processus non autorisé. (ex : chiffrement des données et des canaux de transmission).

1.2.2 **L'intégrité :** C'est un service qui permet de s'assurer que l'information n'a été ni altérée ni modifiée, par des personnes non autorisées, pendant sa transmission ou son stockage. Elle ne requiert pas le chiffrement.

1.2.3 **L'authentification** : c'est un service qui permet de s'assurer de l'identité de l'expéditeur de l'information, et par conséquent, de confirmer son identité.

1.2.4 **La non-répudiation** : c'est un service qui permet d'obtenir la preuve de l'émission ou de la réception d'une information ; l'expéditeur et le destinataire ne peuvent ainsi en nier l'envoi ou la réception. Il empêche donc le reniement d'actions ou de messages.

1.2.5 **La signature numérique (ou signature électronique) :** permet de vérifier l'authenticité de l'expéditeur ainsi que l'intégrité du message reçu, et d'en assurer la non répudiation (Services d'authentification, d'intégrité et de non répudiation).

1.3 Principes fondamentaux des crypto systèmes.

1.3.1 Les concepts cryptographiques

1.3.1.1 : La cryptologie :

La cryptologie est la science du secret. C'est une science pure, qui émet les idées et les principes qui servent de base à la technique du Chiffre dont l'objectif est de transformer une information claire en une information inintelligible à toute personne non qualifiée à la connaître, et de faire la transformation inverse en utilisant les clés, les matériels et les logiciels destinés à cet effet.

La cryptologie regroupe :

- la cryptographie
- et, la cryptanalyse

La cryptographie est une science appliquée, qui date, comme nous l'avons rappelé, de l'antiquité ; elle provient des mots grecs (Kruptos= caché et graphein = écrit ; écriture cachée, secrète). Elle est devenue un terme générique utilisé pour décrire la conception et l'analyse des mécanismes basés sur des techniques mathématiques qui fournissent des services de sécurité (Authentification, Intégrité, confidentialité des données, Non répudiation et Signature électronique).

Alors que la cryptographie, qui permet de concevoir de tels mécanismes, protège le secret des données, la cryptanalyse (ou le décryptement ou l'analyse cryptographique), qui permet d'analyser ces mécanismes, cherche à percer le secret des données sans connaissance initiale de la clé utilisée lors du chiffrement.

La cryptologie est ainsi l'étude scientifique de la cryptographie (la conception de ces mécanismes) et de la cryptanalyse (l'analyse desdits mécanismes).

1.3.1.2 Primitive cryptographique

Une primitive cryptographique est un processus cryptographique qui fournit un certain nombre de services de sécurité spécifiés. Les primitives sont des briques de construction des crypto systèmes (ex : Algorithmes par blocs, en continu ou par flux, codes d'authentification de message, fonctions de hachage, schémas de signature numérique). Dans la pratique, il est conseillé d'utiliser des primitives incluses dans une norme afin d'avoir une sécurité suffisante (ISO, ANSI, FIPS, IEEE, NIST, AFNOR…)

1.3.1.3 Algorithme de chiffrement :

Un algorithme de chiffrement est la spécification particulière d'une primitive cryptographique. C'est un ensemble de règles mathématiques (logiques) utilisées pour résoudre des problèmes de chiffrement et de déchiffrement. On distingue :

- *Les algorithmes symétriques ou à clefs secrètes* (algorithmes conventionnels, à clé secrète, ou à clé unique ; les clés de chiffrement et de déchiffrement sont identiques ou peuvent être facilement calculées l'une à partir de l'autre. Deux sous catégories existent – algorithmes par bloc (Ex : DES, 3DES, IDEA, AES, BLOWFISH….) - ou par flux/en continu : Ex : RC4, A5).
- *Les algorithmes non symétriques (asymétriques) ou à clefs publiques* (algorithmes de chiffrement utilisant deux clefs : l'une est publiée (dans un annuaire par exemple), l'autre privée ou secrète et elle n'est jamais transmise (Ex : Diffie-Hellman, RSA, Rabin, ElGamal, Elliptic Curve Cryptography-ECC, McEliece et Niederreiter, NTRU)
- *Les algorithmes hybrides* (mélange des deux précédents).

1.3.1.4 Schéma de chiffrement [3]

Un schéma de chiffrement noté II= (G, E_k, D_k) est composé de trois algorithmes ayant les fonctionnalités suivantes :

1. L'algorithme probabiliste de génération de clés (G), qui prend en entrée le paramètre de sécurité (1^n) et génère une clé (k) ; nous écrivons que : (k)←G (1^n) (ce qui signifie que G est un algorithme randomisé). Nous supposerons aussi que toute clé (k) générée par G (1^n) satisfait la relation $|k| \geq n$.
2. L'algorithme de chiffrement E_k (à clé secrète, ce qui nous intéresse dans cette thèse) qui prend en entrée une clé (k), choisie uniformément au hasard, un message clair $x \in \{0, 1\}^*$, et sort un cryptogramme (y).
   - S'il est probabiliste, on note $y \leftarrow E_k(x)$
   - S'il est déterministe, on note $y = E_k(x)$

3. L'algorithme de déchiffrement D_k prend en entrée une clé (k) et le cryptogramme (y) et sort le message clair (x) tel que $D_k(y) = D_k(E_k(x)) = x$.

L'espace des clefs K est l'ensemble de toutes les clés possibles générées par l'algorithme G.

L'espace des messages X est l'ensemble de tous les messages clairs (c'est-à-dire ceux pris en charge par l'algorithme de chiffrement) avec | X | > 1. K et X définissent ensemble tous les cryptogrammes possibles de l'espace des cryptogrammes noté Y.

Notons que pour le chiffrement à clé secrète, il est requis que ∀ (n), ∀ (k) générée par G (1^n), et ∀ x ∈ {0, 1}*, alors Dk (Ek (x)) =x.

Si (G, Ek, Dk) est telle que ∀ *k* générée par G (1^n), l'algorithme E_k est seulement défini pour les messages x ∈{0, 1} $^{\ell(n)}$, alors nous disons que (G, E_k, D_k) *est un schéma de chiffrement à clé secrète de longueur fixée pour les messages de longueur* $\ell(n)$.

Nous remarquons que G (1^n) choisit $k \leftarrow \{0, 1\}^n$ toujours uniformément au hasard.

1.3.1.5 Protocole cryptographique

Un protocole cryptographique est un ensemble de règles ou de conventions définissant un échange de messages, généralement chiffrés, entre des correspondants (Ex : SSL/TLS, SSH, IPSEC, PGP et GPG pour sécuriser l'internet). Si les primitives cryptographiques sont des outils pour la cryptographie, le protocole de chiffrement est la mise en œuvre de ces outils et leur utilisation de manière spécifique afin d'atteindre des objectifs de sécurité plus complexes.

1.3.1.6 Crypto système ou système de chiffrement

Un crypto système ou système de chiffrement est la mise en œuvre des primitives cryptographiques et leur infrastructure d'accompagnement, avec toutes les clefs possibles et la gestion de ces clefs.

C'est la donnée du quintuplet :

$C = \{X, Y, K, E, D\}, C \neq \emptyset$ avec :

X =ensemble fini des messages clairs

$= \{X_1, X_2, X_3, \ldots\ldots Xi \ldots, Xn\}$

Y = ensemble fini des messages chiffrés

$= \{Y_1, Y_2, Y_3, \ldots Yi \ldots, Ym\}$

K = ensemble des clefs utilisées

$= \{K_1, K_2, K_3, \ldots\ldots Ki \ldots\ldots, Kp\}$

E = Ensemble des fonctions de chiffrement

$= \{E_{K1}, E_{K2}, E_{K3}, \ldots\ldots E_{Ki} \ldots\ldots E_{Kp}\}$

D = Ensemble des fonctions de déchiffrement

$= \{D_{K1}, D_{K2}, D_{K3}, \ldots\ldots D_{Ki} \ldots\ldots\ldots D_{Kp}\}$

$\forall Ki \in K, \exists E_{Ki} \in E$ et $D_{Ki} \in D$ tel que :

$E_{Ki} : X \mapsto Y$

$D_{Ki} : Y \mapsto X$

avec $D_{Ki}(E_{Ki}(Xi)) = Xi, \forall Xi \in X$;

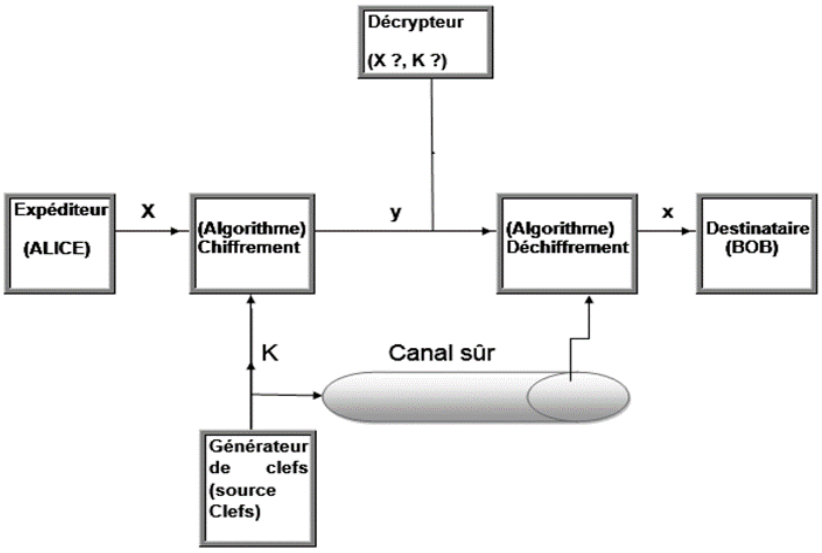

Figure 1.3 : Modèle de base de crypto système

1.3.2 Les différents crypto systèmes

1.3.2.1 Tableau de classement des crypto-systèmes [4]

Mettant en œuvre les primitives précitées, les crypto systèmes sont subdivisés en trois (3) catégories ainsi qu'il suit :

Crypto-systèmes à clefs secrètes	Crypto-systèmes à clefs publiques	Crypto-systèmes sans clefs (Unkeyed Cryptosystem)
▪ Chiffrement par blocs (3DES, IDEA, CAST, Blowfish, RC5, AES, ….etc) ▪ Chiffrement en continu /flux (RC4, SEAL, Sosemanuk,Trivium….etc) ▪ MAC (Code d'authentification de message : CBC MAC, HMAC, UMAC) ▪ PRBG (Générateurs de bits pseudo-aléatoires : Blum-Micali PRBG, RSA PRBG, Blum Blum Shub, PRBG….etc) ▪ PRF (Fonctions pseudo aléatoires)	▪ Chiffrement (RSA, ElGamal, ECC, McEliece…..etc). ▪ Protocoles d'échange de clefs- Key agreement (DH) ▪ Signature numérique (DSS : DSA, RSA, ECDSA) ▪ Protocoles d'authentification (Needham-Schroeder, , Kerberos, SSL/TLS, Ipsec, SSH…etc)	▪ Fonctions à sens unique (One-Way-function : OWF) ▪ Fonctions de hachage cryptographiques (MD4, MD5, Famille SHA….etc) ▪ RBG (Générateurs de bits aléatoires)

Tableau 1.1 : Classement des crypto-systèmes

1.3.2.2 Cartographie des primitives cryptographiques pour les services de sécurité

La cartographie des primitives cryptographiques pour les services de sécurité peut être résumée par les tableaux suivants :

	Confidentialité	Intégrité	Auth. Données	Non Répudiation	Auth. Entité
Chiffrement	Oui	Non	Non	Non	Non
Fonction de Hachage	Non	Souvent	Non	Non	Non
MAC	Non	Oui	Oui	Souvent	Non
Signature	Non	Oui	Oui	Oui	Non

Tableau 1.2 : Cartographie des primitives utilisées seules pour fournir des services de sécurité

	Confidentialité	Intégrité	Auth. Données	Non Répudiation	Auth. Entité
Chiffrement	Oui	Oui	Oui	Oui	Oui
Fonction de Hachage	Oui	Oui	Oui	Oui	Oui
MAC	Non	Oui	Oui	Oui	Oui
Signature	Non	Oui	Oui	Oui	Oui

Tableau 1.3 : Cartographie des primitives qui peuvent aider à fournir des services de sécurité

1.4 Objectifs, plan de la thèse et contribution

1.4.1 Le contexte

A la fin du XIXème siècle, Auguste Kerckhoffs a posé dans son traité « La cryptographie militaire-1883 » [5] des principes dits principes de Kerckhoffs dont l'un des plus importants est énoncé comme suit (cf. Chapitre 2, paragraphe 2.1) :

« *Le procédé de chiffrement ne doit pas être obligatoirement secret, et il doit pouvoir être en mesure de tomber sans inconvénient entre les mains de l'ennemi* ».

Autrement dit le schéma de chiffrement ne devrait pas être conservé secret, seule la clé devant constituer l'information secrète partagée par les correspondants du réseau de chiffrement : « *La sécurité d'un crypto système ne doit reposer que sur le secret de la clef* ». En effet, si la valeur cryptographique d'un système de

chiffrement exigeait le secret du procédé employé, la sécurité serait faible, à la merci de la moindre indiscrétion [6].

On ne peut donc espérer garder secret un crypto système (il faut toujours partir du principe que l'algorithme est connu du décrypteur) ; la difficulté en cryptanalyse ne doit pas dépendre du secret des algorithmes mais du secret des clés :

- les clefs doivent être robustes et *choisies de façon aléatoire* ;
- les clés doivent être changées fréquemment et il faut chiffrer chaque message avec une clef différente qui est ensuite détruite (crypto système à clef-une-fois ou One-Time-Pad du nom de son inventeur Gilbert Vernam -1917) [7] ;
- l'espace de clefs doit être très grand.

Si ces critères précités sont strictement respectés, le crypto système offre une sécurité théorique absolue selon Claude Shannon (1949) [8] et le décryptement est impossible.

La production de clés de chiffrement, attachées à des quantités aléatoires imprévisibles (suite de nombres, de bits aléatoires ou pseudo-aléatoires) est la base de la plupart des crypto systèmes cryptographiquement sûrs :

- *En cryptographie symétrique* : il est requis des quantités aléatoires utilisables comme clés secrètes ; plusieurs algorithmes de « chiffrement de flux additif » reposent sur les générateurs de bits pseudo-aléatoires. Par exemple, le chiffrement de flux additif pratiquement efficace et largement déployé ARCFOUR (RC4) représente un PRBG.
- *En cryptographie à clés publiques* : il est également requis des quantités aléatoires pour générer des paires de clés (publiques/privées).

Dans ces deux cas, si les quantités aléatoires sont générées à chaque utilisation du crypto système, celui-ci est alors probabiliste.

Ainsi, l'aléa occupe une place majeure en cryptographie, pour chiffrer et déchiffrer les messages, et pour faire également des attaques sur les crypto systèmes. Ce besoin d'aléa s'applique aussi à plusieurs applications informatiques et à internet comme :

- Les codes d'accès confidentiels liés aux cartes bancaires
- Les protocoles d'authentification tels que SSL/TLS, la divulgation nulle de connaissance (Zero-Knowledge)
- Les paramètres de signature électronique (RSA, DSA, ECDSA…)
- Les mots de passe
- Les simulations numériques
- La loterie, les jeux de hasard, les roulettes casino…etc.

Dans son sens le plus général, *l'aléa* désigne la mesure de la probabilité d'un évènement (*aléa*=mot latin qui signifie ''*jeu de dés, hasard*''). Il y'a plusieurs longues années Laplace disait déjà que « *Le hasard, le plus souvent, consiste dans l'ignorance du nombre et de l'importance des causes complexes et difficilement évaluables de certains évènements* » [9].

Le caractère aléatoire d'une suite est lié à la prédictibilité de cette suite. Pour que l'aléa soit optimal (donc sa prédictibilité, minimale), il faut que son entropie soit maximale.

Mathématiquement, l'entropie H (*x*) d'un évènement discret et aléatoire (*x*), qui peut être dans (*n*) états ($x_1, x_2, x_3, \ldots, x_n$) avec des probabilités d'apparition ($p_1\ p_2\ p_3\ \ldots\ldots p_n$), est calculée par la quantité suivante :

$$H(x) = \sum_{i=1}^{n} p_i \ log_2 \left(\frac{1}{p_i}\right) = -\sum_{i=1}^{n} p_i \ log_2 p_i$$

H (x) est la mesure du degré d'incertitude de cet évènement ; cette entropie est maximale lorsqu'il y'a équiprobabilité des différents états. *La suite est alors aléatoire*.

Généralement, les suites utilisées doivent être produites par des générateurs de bits aléatoires (RBG-Random Bits Generators), non déterministes, des générateurs de bits pseudo-aléatoires (PRBG-Pseudo Random Bit Generators), déterministes, et des générateurs hybrides, qui doivent tous posséder de bonnes propriétés statistiques permettant de statuer sur leur degré de sécurité cryptographique.

S'agissant particulièrement des PRBG, les suites produites doivent être tel que l'on ne peut pas les distinguer facilement de suites parfaitement aléatoires, même si toutes les caractéristiques du générateur utilisé sont connues de l'attaquant, à l'exception de la clé secrète. Il doit être impossible à l'attaquant, même s'il connaît quelques bits du générateur, de prédire la valeur des bits suivants.

En conséquence, les générateurs doivent résister aux attaques visant notamment à rétablir la clé, l'état initial, à prédire le bit suivant et à distinguer la suite produite d'une suite parfaitement aléatoire.

1.4.2 Objectifs et plan de la thèse

La présente thèse a justement pour objectifs :

- d'étudier les générateurs de bits pseudo-aléatoires et leur sécurité,
- d'étudier et d'établir une loi de probabilité que nous appellerons « *Loi des présences* » ainsi que « *Le Théorème des présences* », dans l'hypothèse où nous disposons d'un générateur qui délivre une suite uniforme, aléatoire ; le nombre de symboles présents dans la suite, après suppression des répétitions, étant une variable aléatoire ;
- de concevoir un distingueur d'aléa basé sur la loi des présences.

La thèse comprend deux (2) grandes parties qui traitent des sujets suivants :

- Première partie : Les générateurs d'aléas – La sécurité cryptographique
- Deuxième partie : La présentation de nos travaux

Elle débute par un premier chapitre (Introduction) qui retrace succinctement l'évolution de la cryptologie, de l'antiquité à nos jours, définit les différents services de sécurité offerts par la cryptologie, pose les principes fondamentaux des crypto systèmes basés notamment sur les différents concepts cryptographiques ainsi que sur les primitives cryptographiques, et présente les objectifs ainsi que le plan du travail réalisé.

Dans la première partie, après avoir rappelé dans un chapitre 2, les principes d'Auguste Kerckhoff, posés vers la fin du XIXème siècle, qui requièrent notamment que la sécurité cryptographique repose uniquement sur le secret de la clé partagée (l'algorithme doit être public), ainsi que les fondamentaux de la cryptographie moderne, la thèse aborde *les notions de sécurité cryptographique* qui sont indispensables pour l'étude, d'une manière générale, des générateurs d'aléas :

- *Le secret parfait* (ou *la sécurité inconditionnelle*), introduit par Claude Shannon (1940), plus de 20 ans après l'invention du crypto système « *One-Time-pad (**OTP**) ou chiffrement à clé-une-fois »* par Gilbert Vernam (1917), et qui est extrêmement fort, avec des critères contraignants, pour la construction des crypto systèmes cryptographiquement sûrs, et qui font qu'il n'est pas adaptée à des applications pratiques ;
- *La sécurité calculatoire*, appelée aussi *la sécurité pratique*, qui est moins forte et plus accessible que le secret parfait, sachant que la puissance de l'attaquant est illimitée. La grande majorité des crypto systèmes symétriques et des crypto systèmes à clés publiques sont concernés par cette notion de sécurité calculatoire.
- La sécurité des crypto systèmes à clés secrètes, catégorie à laquelle nous avons classé les générateurs de bits pseudo aléatoires (PRBG) ; A ce niveau, nous allons présenter *deux*

définitions équivalentes de la sécurité de ces cryptos systèmes, qui sont *la sécurité sémantique et l'indistinguabilité des chiffrements.*

Le chapitre 3 suivant, traite de la construction et de la sécurité des générateurs d'aléas. Il met particulièrement l'accent sur les concepts de fonction à sens unique, de permutations à sens unique, de prédicats difficiles, et montre :

- que les fonctions à sens unique permettent de construire des générateurs pseudo-aléatoires.
- qu'à partir de n'importe quelle permutation à sens unique, on peut construire des générateurs pseudo-aléatoires.
- que le prédicat difficile d'une permutation à sens unique peut être utilisé pour construire un générateur pseudo-aléatoire

Il insiste ensuite sur :

- les générateurs de bits aléatoires (RBG), modèles idéaux pour générer des suites binaires parfaitement aléatoires, les conditions de leur réalisation et de leur implantation matérielle et logicielle ainsi que les tests statistiques y afférents ;

- les générateurs de bits pseudo- aléatoires (PRBG), modèles utilisés pour générer des suites de grande longueur qui semblent aléatoires, les conditions de leur réalisation et de leur implémentation, avec notamment, comme base, les registres à décalage (cf. travaux présentés à la 2ème partie), ainsi que les tests y relatifs.

- les générateurs hybrides qui sont, généralement, des dispositifs regroupant les deux précédents.

- La sécurité des générateurs d'aléas (les différentes attaques, les générateurs de bits pseudo aléatoires cryptographiquement sûrs et les tests effectués sur les générateurs d'aléa pour déterminer s'ils possèdent ou non des faiblesses).

Enfin, la deuxième et dernière partie de la thèse est consacrée aux travaux que nous avons réalisés :

- Nous ferons un survol des travaux de recherche qui ont réalisés sur le plan de la cryptologie classique, et qui ont abouti à l'établissement du « *Théorème de la réciprocité* » qui énonce les conditions nécessaires et suffisantes pour que des matrices de Delastelle (damiers bigrammatiques de Delastelle) soient réciproques afin que le chiffrement et le déchiffrement puissent se faire dans le même sens en utilisant une clé aléatoire [10] fournie par un générateur d'aléas. Ces travaux présentent un intérêt didactique, parce que le thème peut être enseigné dans le cadre du cours de cryptographie classique, au même titre que le procédé de Playfair et le chiffrement matriciel de Lester Hill (chiffrement polygraphique). Ce procédé de chiffrement réciproque, qui est d'un très bon niveau de sécurité, peut aussi, dans la pratique, être utilisé en chiffrement classique, pour protéger de courts messages (la clé ne devant être utilisée qu'une et une seule fois).

- Nous examinerons ensuite les travaux de réflexion et de recherche qui ont été menés sur « *La construction des Registres à Décalage à Rétro action Linéaire* (LFSR) *bouclés à période maximale (Polynômes primitifs et suites récurrentes linéaires)-générateurs pseudo aléatoires basés sur les registres à décalage* » [11]. Les registres à décalage à réinjection constituent généralement l'élément de base des générateurs pseudo-aléatoires utilisés pour générer des chaînes cryptographiques ou ensemble de suites de clés de chiffrement. Ces générateurs sont beaucoup utilisés dans les systèmes de chiffrement en continu (stream cipher) et les systèmes de communication. Les travaux ont permis de proposer une méthode

de détermination mathématique, à partir du polynôme primitif, de la relation récurrente linéaire générant le LFSR bouclé à période maximale (et vice versa) et qui facilite ainsi sa construction ; cette méthode aide également à établir le polynôme primitif réciproque correspondant qui donne la possibilité de construire un autre LFSR aussi bon que le premier. Nous avons là une conception et un choix judicieux des LFSRs bouclés à période maximale, à utiliser, sur la base des polynômes primitifs, des polynômes réciproques et des récurrences linéaires associées, ce que ne montrent pas les méthodes utilisées jusqu'ici où les récurrences sont établies, sans plus de précisions, à partir des polynômes primitifs pour décrire les LFSRs. Nous avons étudié également, par la même occasion, les questions liées à la sécurité cryptographique des LFSR et indiqué certains problèmes ouverts dans le domaine des générateurs pseudo-aléatoires basés sur les registres à décalage à rétroaction non linéaires (NLFSR).

- Enfin, nous exposerons les travaux qui ont abouti :
  - d'une part, à l'établissement d'une loi de probabilité que nous avons dénommée « *Loi des présences* » ainsi que du « *Théorème des présences* ». A cet effet, nous nous sommes placés dans l'hypothèse où nous disposons d'un générateur aléatoire qui délivre des symboles pris dans un alphabet de (n) symboles selon une loi statistique uniforme, et avec lequel nous voulons fabriquer des suites chiffrantes. Pour cela, dans une tranche de (N) symboles fournis par le générateur, si nous supprimons les répétitions, nous obtenons une suite de (k) symboles ($1 \leq k \leq n$). Ce nombre (k) de symboles présents est une variable aléatoire dont nous avons établi la loi de probabilité.

  - et d'autre part, à la conception d'un distingueur (ou différenciateur) d'aléa basé sur cette loi de probabilité qui permet de statuer sur l'uniformité ou non des suites chiffrantes, et partant d'évaluer la qualité cryptographique des générateurs d'aléas qui les produisent. Il s'agit d'une bonne méthode originale de test statistique d'aléa pour distinguer une distribution donnée d'une distribution aléatoire (ou pseudo aléatoire).

1.4.3 Contribution

A titre de contribution, les résultats des travaux annoncés aux chapitres 4 et 5 de la deuxième et dernière partie de la thèse, ont fait l'objet d'articles [10] [11] publiés dans les journaux spécialisés « CRYPTOLOGIA » et « BRITISH JOURNAL OF MATHEMATICS AND COMPUTER SCIENCE » :

- Babacar A. NDAW, Amadou D. SARR, The problem of reciprocity in a Delastelle bigraphic substitution, CRYPTOLOGIA, vol 7, n°2, pages 170-179, April 1983, jun 2010.

  Compte tenu de son niveau scientifique, cette publication est indexée actuellement par ISI (International Scientific Indexing), Scopus (Sciverse Scopus Elle a été aussi indexée et abstractée "Zentralblatt Math, et beaucoup d'autres bases de données".

- Babacar A. NDAW, Djiby SOW, Mamadou SANGHARE, Construction of Maximum Period Linear Feedback Shift Registers (LFSR) (Primitive Polynomials and Linear Recurring Relations), British Journal of Mathematics and Computer Science, 11(4): 1-24, 2015, Article n°BJMCS.19442, SCIENCEDOMAIN *International*.

Les résultats contenus aux chapitres 6 et 7 sur « *La Loi des présences* », « *Le Théorème des présences* » et « *La conception du distingueur d'aléa* » sont en voie de publication. Ces travaux originaux pourraient, s'ils sont retenus, être soumis aux organismes internationaux de standardisation comme le NIST (National Institut of Standards and Technology), afin que soit étudiée la possibilité de standardiser le test que nous avons conçu, en vue de réaliser des applications cryptographiques destinées particulièrement à évaluer la qualité des générateurs d'aléas et à produire des clés de chiffrement.

REFERENCES

[1] Edmond Lerville, Les cahiers secrets de la cryptographie : le chiffre dans l'histoire des histoires du chiffre, Editions du Rocher Science 1972.

[2] W. Diffie, M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, vol 22, n° 6, 1972.

[3] Jonathan Katz, Yehuda Lindell, Introduction to Modern Cryptography, Chapman & Hall/CRC Taylor & Francis Group, 2007

[4] Rolf Oppliger, Contemporary Cryptography, Artech House, Computer security series, pp 25, 219-329.

[5] Auguste Kerckhoffs, "La cryptographie militaire", *Journal des sciences militaires*, vol. IX, p. 5– 38, janvier 1883, p. 161–191, févr. 1883.

[6] Eyraud, C. 1953. Précis de cryptographie moderne. Paris: Editions Raoul Tari.

[7] Vernam, Gilbert S. (1926), "Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications", Journal of the IEEE **55**: 109–115.

[8] C.E. Shannon. Communication Theory of Secrecy Systems. Bell Sys. Tech. Jour. , Vol.28, pages 656-715, 1949.

[9] G. Cullmann, M. Denis Papin, A. Kaufmann, Cours de Calcul Informationnel Appliqué, Edition Albin Michel, 1970.

[10] Babacar A. NDAW, Amadou D. SARR, The problem of reciprocity in a Delastelle biographic substitution, CRYPTOLOGIA, vol 7, n°2, pages 170-179, April 1983, Jun 2010.

[11] Babacar A. NDAW, Djiby SOW, Mamadou SANGHARE, Construction of Maximum Period Linear Feedback Shift Registers (LFSR) (Primitive Polynomials and Linear Recurring Relations), British Journal of Mathematics and Computer Science, 11(4): 1-24, 2015, Article n°BJMCS.19442, SCIENCEDOMAIN *International*.

Première partie
Générateurs d'aléas-
Sécurité cryptographique

Chapitre 2

Sécurité cryptographique

Sommaire

D'une manière générale, la sécurité cryptographique avait toujours été vue sous l'angle du degré de résistance d'un crypto système aux attaques du décrypteur, contrairement à la sûreté cryptographique qui désigne le degré de confiance accordée au crypto système. Il s'agit là de définitions floues d'autant qu'il est difficile de quantifier ces degrés de résistance et de confiance liés aux crypto systèmes, et particulièrement aux primitives cryptographiques y afférentes.
La notion de sécurité cryptographique doit être définie et formalisée de façon claire et précise. Nous allons donc passer en revue, dans ce chapitre, les différentes notions de sécurité sur lesquelles nous nous appuierons pour nos travaux.

2.1 Les principes de Kerckhoffs. [1]

À la fin du XIXe siècle, Auguste Kerckhoffs a posé des principes dont les plus importants sont les suivants :

1) *« Le procédé de chiffrement ne doit pas être obligatoirement secret, et il doit pouvoir être en mesure de tomber entre les mains de l'ennemi sans inconvénient ».*

Autrement dit, ce principe de Kerckhoffs requiert que la sécurité repose uniquement sur le secret de la clé partagée ; l'algorithme doit être public, ce qui permet d'établir des normes internationales :

- Il est beaucoup plus facile pour les correspondants d'un réseau de garder le secret d'une clé courte que celui d'un algorithme.
- Il est plus facile de changer la clé, en cas de compromission, que de remplacer l'algorithme.
- Dans un réseau, il est plus facile aux correspondants d'utiliser le même algorithme, avec plusieurs clefs, plutôt que d'employer, chacun en ce qui le concerne, un algorithme différent.

2) *« Un procédé de chiffrement doit être pratiquement, sinon mathématiquement indécryptable »*

Autrement dit, « *il n'est pas nécessaire d'utiliser un schéma de chiffrement parfaitement secret, mais il suffit, à la place, d'utiliser un schéma de chiffrement qui ne peut pas être cassé dans un délai raisonnable avec une probabilité raisonnable de succès* ».

Aujourd'hui, les principes de Kerckhoffs sont compris comme des principes qui, non seulement préconisent que la sécurité ne doit pas dépendre du secret des algorithmes utilisés, mais, également, exigent que ces algorithmes soient rendus publics.

2.2 Les principes fondamentaux de la cryptographie moderne.

Selon [2], trois principes fondamentaux sont établis :

1) *La formalisation de la sécurité* : Toute définition de la sécurité doit prendre la forme générale suivante :

   « Un schéma cryptographique est sûr si aucun attaquant, disposant d'une puissance spécifiée, ne peut réaliser une cassure spécifiée ».

   De ce point de vue, un schéma de chiffrement est sûr si tout attaquant qui dispose d'un texte chiffré ne parvient pas à retrouver la clé secrète, ne peut obtenir aucune information utile sur le texte clair à partir du texte chiffré, et ne peut calculer une quelconque fonction du texte clair à partir du texte chiffré.

2) *L'énoncé précis et la validation des hypothèses* : Si la sécurité inconditionnelle d'un schéma ne peut pas être prouvée et doit s'appuyer sur une hypothèse, une démonstration mathématique que « *la construction est sûre si l'hypothèse est vraie* » peut être fournie seulement s'il y a un exposé précis de ce qu'est cette hypothèse.

3) *L'établissement de preuves rigoureuses de sécurité pour les actions à entreprendre* : pour la simple raison que la sécurité d'un schéma cryptographique ne peut être contrôlée de la même manière que le logiciel.

   L**'approche de la réduction cryptographique :** Il faut noter que la plupart des preuves de sécurité en cryptographie moderne utilisent ce qu'on appelle « *l'approche réductionniste* » :

   "Étant donné qu'une hypothèse X est vraie, la construction Y est sûre conformément à la définition donnée ".

   La preuve de sécurité (***appelée aussi sécurité prouvée ou sécurité par réduction***) montre comment réduire un problème donné par l'hypothèse X au problème qui consiste à casser la construction Y. Plus précisément, la preuve de sécurité, en général, consiste à démontrer que s'il existe un attaquant

qui est capable de casser un système donné, et d'en compromettre ainsi la sécurité, alors grâce à cet attaquant on peut construire un autre attaquant capable de casser un système réputé cryptographiquement sûr. Nous considérons l'attaquant comme un algorithme efficace, appelé (*la réduction*), qui sort un résultat relatif au problème de sécurité posé. Il simule l'environnement de l'attaquant, et à l'aide d'un ou de plusieurs appels à l'attaquant, résout le problème algorithmique difficile.

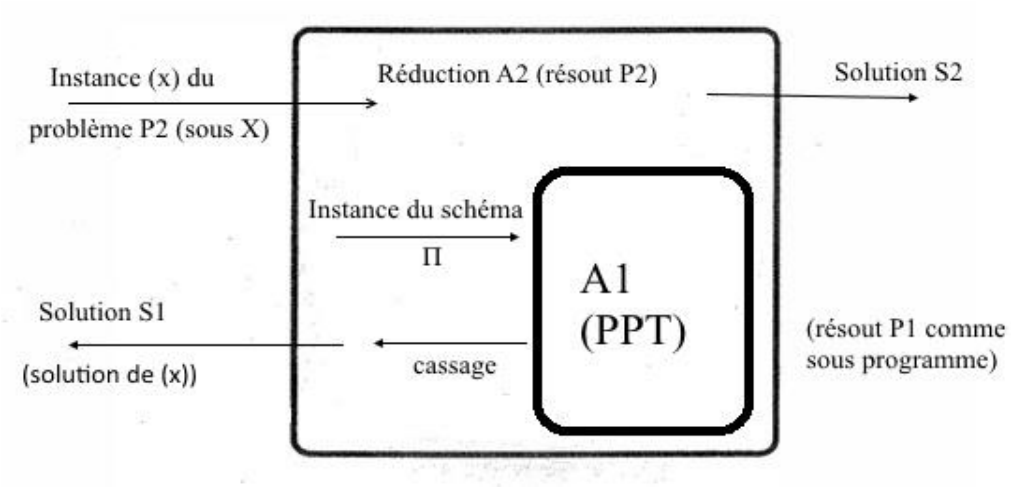

Figure 2.1 : Preuve de sécurité par la réduction

Dans cette figure, à partir d'un attaquant A_1 qui résout un problème P_1, en temps (t_1) avec une probabilité (ε_1), on construit un attaquant A_2 qui résout le problème P_2 en temps (t_2) avec une probabilité (ε_2), et on réduit le problème P_2 au problème P_1 :

- Problème P_1 = « casser la construction Y »
- Problème P_2=« problème difficile (sous hypothèse X : factorisation, log discret en cryptographie à clés publiques) »
- ∃ un attaquant A_1 ⇒ ∃ un algorithme de résolution de P_2
- -∄ algorithme de résolution de P_2 ⇒ ∄ attaquant A1
- La réduction A_2 simule parfois les oracles auxquels l'attaquant a accès (ex : oracle de déchiffrement)

2.3 Le Secret parfait (Sécurité inconditionnelle) [2-5]

2.3.1 Définition

« *Un schéma de chiffrement II = (G, E_k, D_k), sur un espace des messages (X), est parfaitement secret si pour toute distribution de probabilités sur (X), tout message x ϵ X et tout texte chiffré y ϵ Y avec Pr [Y =y) > 0, alors* :

$$\Pr [X= x \mid Y= y] = \Pr [X = x] \text{ ».} \qquad (1)$$

Autrement dit, les distributions sur les messages clairs et les textes chiffrés sont indépendantes.
Cette définition du *secret parfait* (c'est-à-dire de la sécurité inconditionnelle) peut être formulée d'une manière différente comme suit :

2.3.1.1 Indistinguabilité parfaite [2] :

« *Un schéma de chiffrement II = (G, E_k, D_k) sur l'espace des messages (X) est parfaitement secret si et seulement si pour toute distribution de probabilité sur (X), tout couple de messages clairs (x_0, x_1) ϵ X et tout message chiffré y ϵY :*

$$P_r [Y=y \mid X=x_0] = P_r [Y=y \mid X=x_1] \text{ ».} \qquad (2)$$

Selon cette nouvelle formulation, pour deux messages clairs (x_0, x_1) ϵ X, les distributions Y(x_0) et Y (x_1) sont identiques : *le texte chiffré ne fournit aucune information sur le message clair*.

Cette propriété est appelée « *indistinguabilité parfaite* » parce qu'il est impossible de distinguer un chiffré de x_0 d'un chiffré de x_1, du fait que la distribution sur le texte chiffré est la même dans chaque cas.

2.3.1.2 Indistinguabilité adverse : [2]

L'Indistinguabilité adverse formalise l'incapacité de l'attaquant (A) à distinguer le chiffrement d'un message clair donné du chiffrement d'un autre message clair, l'attaquant pouvant être considéré comme un algorithme réalisable qui sort un résultat relatif au problème de sécurité posé.

Soient un schéma de chiffrement II = (G, E_k, D_k), une expérience notée SK impliquant la clé secrète et un attaquant (A) qui ne reçoit qu'un texte chiffré (y), et qui tente de rétablir le message clair correspondant ainsi que l'exécution de l'expérience notée $SK_{II, A}$. La sortie de l'expérience est égale à ''1 '' ou ''0'' : ce qui veut dire que si $SK_{II, A} = 1$, (A) a réussi, sinon il a échoué.

Il est alors noté : *« qu'un schéma de chiffrement II = (G, E_k, D_k) sur un espace de messages X est parfaitement secret si ∀ l'attaquant A :*

$$P_r [SK_{II, A} = 1] = ½ \text{ »} \qquad (3)$$

2.3.2 Le One-Time-Pad : un système inconditionnellement sûr [3-4]

Appelé aussi crypto système de Vernam , du nom de son inventeur Gilbert Vernam (1917), qui fut Ingénieur à l'American Telephone and Telegraph Compagny, *le One-Time-pad (**OTP**) ou Chiffrement à clé-une-fois*, a été amélioré (1920) par Joseph O. Mauborgne, Chef du Service des Chiffres de l'Armée Américaine qui y rajouta la notion de clé aléatoire.
Le One-Time-pad ***(OTP)*** *offre un secret parfait :* il utilise les procédés de substitution poly alphabétique (Substitution à double clé type ou genre Vigenère) avec des clés principales apériodiques qui doivent remplir les trois critères suivants :

- Les clés doivent être au moins aussi longues que les messages à chiffrer.
- Les caractères composant chaque clé doivent être choisis de façon totalement aléatoire (utilisation d'un générateur aléatoire ou pseudo aléatoire)
- Toutes les clés sont équiprobables et chaque clé, ne doit être utilisée qu'une et une seule fois, après quoi elle est détruite (*masque jetable*).

Si ces critères sont strictement respectés, le système permet d'atteindre un niveau de sécurité théorique absolue selon Claude Shannon (1949) et le décryptement est impossible.

THEOREME DE SHANNON : Soit $\Pi = (G, E_k, D_k)$ un schéma de chiffrement sur un espace de messages X, avec $|X| = |K| = |Y|$. Ce schéma est parfaitement secret si et seulement si :

1. Toute clé $k \in K$ est choisie avec une probabilité égale $\frac{1}{|K|}$ par algorithme (G)

2. $\forall x \in X$ et $\forall y \in Y, \exists ! k \in K$ tel que $y = E_k(x)$.

On prouve que la probabilité de connaître le message clair étant donné le texte chiffré est égale à la probabilité de connaître le clair. En utilisant l'entropie de Shannon :

$$H(X/Y) = H(X) \qquad (X \text{ et } Y \text{ sont indépendants}).$$

où H(X) représente l'entropie c.-à-d. le degré d'incertitude devant lequel se trouve le décrypteur, par rapport au clair, lorsqu'il détient le cryptogramme :

$$H(X) = -\sum_x P(X = x) \log_2 P(X = x) \text{ et}$$

$$H(X/Y) = H(X, Y) - H(Y) \leq H(X) \text{ avec}$$

$$H(X, Y) = -\sum_{x,y} P(X = x, Y = y) \log_2 P(X = x, Y = y) \qquad (4)$$

THEOREME : Si un système cryptographique est parfaitement secret, alors $H(K) \geq H(X)$.

Ce qui veut dire qu'il y'a au moins autant d'incertitude sur les clés que sur les messages clairs. ***Le système est inconditionnellemnt sûr.*** *(Cela signifie que même si l'attaquant disposait d'une puissance de calcul infinie, il ne pourrait jamais décrypter le message. Le mot* ***"sécurité inconditionnelle"*** *prend alors tout son sens).*

2.4 La sécurité calculatoire (ou la sécurité pratique) [2, 6-9 ,10]

[2] signale que *la notion de secret parfait*, introduite par Claude Shannon, 25 ans après l'invention de Vernam, est extrêmement forte ; les critères sont contraignants, pour la construction des crypto systèmes cryptographiquement sûrs, et cette notion n'est donc pas adaptée à des applications pratiques, notamment commerciales. Les exigences ont donc été revues, ce qui a permis d'introduire *la notion de sécurité calculatoire*, appelée aussi *la sécurité pratique* (moins forte et plus accessible que le secret parfait) sachant que la puissance de l'attaquant est illimitée. La plupart des crypto systèmes symétriques (DES, 3DES, IDEA, AES…etc) et des crypto systèmes à clés publiques (RSA, ElGamal, …etc.) sont concernés par la sécurité calculatoire.

Cette *approche calculatoire* intègre deux assouplissements de *la notion de secret parfait* :

a) la sécurité n'est assurée que contre les attaquants efficaces qui travaillent dans un temps raisonnable (le calcul se fait par un algorithme polynomial c.-à-d. un algorithme dont le nombre moyens d'opérations s'écrit $p(n)$ où (p) est un polynôme et (n) la taille de la chaîne binaire initiale) ;

b) et, ces attaquants peuvent potentiellement réussir à casser le schéma avec une très faible probabilité (ce qui certainement ne se produira jamais)

Deux approches communes de la sécurité calculatoire ont été établies pour obtenir une théorie cohérente : *l'approche concrète et l'approche asymptotique.*

2.4.1 L'approche concrète

L'approche concrète quantifie la sécurité d'un schéma de chiffrement donné en limitant explicitement la probabilité de succès maximale de n'importe quel attaquant pendant au plus une certaine durée de temps spécifiée.
En d'autres termes, si (t, ε) sont deux constantes positives avec $\varepsilon \leq 1$, alors concrètement la sécurité est définie comme suit :

« Un schéma est **(t, ε)-sûr** *si tout attaquant qui dispose d'un temps d'exécution égal au plus à (t) réussit à le casser avec une probabilité égale au plus* à (ε) ».

Compte tenu de cette définition, les Schémas de chiffrement à clefs secrètes modernes offrent une sécurité optimale dans le cas suivant :

« *Si la longueur de la clé est* (*n*), *un attaquant qui dispose d'un temps d'attaque* (*t*) *peut réussir à casser le schéma avec une probabilité égale au plus à* $(t / 2^n)$ ».

2.4.2 L'approche asymptotique (sécurité asymptotique)

2.4.2.1 Définition :

La sécurité asymptotique est définie comme suit :

« Un schéma de chiffrement est sûr si tout attaquant PPT (Probabilistic Polynomial Time - Algorithme en temps polynomial probabiliste.), *réussit à le casser avec seulement une probabilité de succès négligeable ».*

Cette approche repose sur *la théorie de la complexité* : le temps de décryptement dont dispose l'attaquant ainsi que sa probabilité de succès sont des fonctions de certains paramètres plutôt que des nombres concrets. En générant, par exemple, des clés lors de l'utilisation d'un schéma de chiffrement, les utilisateurs choisissent une valeur *(n)* comme paramètre de sécurité, qu'ils supposent connue de tout attaquant du système. *Le temps de décryptement dont dispose l'attaquant, le temps de chiffrement/déchiffrement des utilisateurs et la probabilité de succès de l'attaquant sont donc tous considérés comme des fonctions de (n).*

2.4.2.2. Calcul efficace :

Le calcul efficace est un calcul qui peut être réalisé en temps polynomial probabiliste (PPT). Un algorithme probabiliste est celui qui a la capacité de réaliser des résultats de « tirage au sort, au hasard » ; il peut être vu comme un algorithme qui a accès à une source d'aléas qui délivre des bits aléatoires, uniformément répartis avec une probabilité d'apparition de $\frac{1}{2}$.

2.4.2.3 Probabilité de succès négligeable

En sécurité cryptographique, la notion de « *petite probabilité de succès* », est assimilée à une probabilité de succès aussi petite que n'importe quel polynôme inverse en (*n)*. Une fonction qui croît plus lentement que tout polynôme inverse est appelée « fonction négligeable ».

DEFINITION : Une fonction (*f*) est négligeable si pour tout polynôme p (·), il existe un entier N tel que pour tous les entiers $n > N$, alors $f(n) < \frac{1}{p(n)}$.

On note généralement une fonction arbitraire négligeable par (*negl*).

- Ainsi, si un attaquant peut réussir à casser un schéma de chiffrement avec une probabilité égale à $\frac{1}{p(n)}$ pour certains polynômes (positifs) *(p),* alors le schéma n'est pas sûr.

- Cependant, si la probabilité pour que le schéma de chiffrement puisse être cassé est asymptotiquement plus petite que $\frac{1}{p(n)}$ quel que soit le polynôme *(p*), alors le schéma est sûr. Cela est dû au fait que la probabilité de succès de l'attaquant est tellement petite qu'elle est considérée comme insignifiante.

Cette probabilité sont appelée « *probabilités de succès négligeable »*.

2.5 La sécurité des crypto systèmes à clés secrètes [2, 4-5, 6-9,10-11] (Sécurité sémantique – Indistinguabilité)

Nous consacrons ce paragraphe à la sécurité des crypto systèmes à clés secrètes, catégorie à laquelle nous avons classé les générateurs de bits pseudo aléatoires (PRBG), et nous allons présenter *deux définitions équivalentes de la sécurité* de ces cryptos systèmes : ***la sécurité sémantique et l'indistinguabilité des chiffrements.***

2.5.1 La sécurité sémantique

La définition de « *la sécurité sémantique* » est analogue à celle du « *secret parfait »* de Shannon : *elle énonce qu'il est calculatoirement impossible à l'attaquant d'obtenir des informations sur le texte en clair à partir du texte chiffré.*
Il est impossible à l'attaquant d'extraire le moindre bit d'information sur le clair à partir du chiffré ; le texte chiffré n'apporte aucune information sur le texte en clair.

2.5.1.1. Définition (Sécurité sémantique) : Un schéma de chiffrement à clef secrète

$\Pi = (G, E_k, D_k)$ est sémantiquement sûr si pour tout attaquant (A) en temps polynomial probabiliste, pour toute fonction *(f)* sur l'espace des messages (*X*) et pour toute distribution de probabilité (X_n), l'avantage de l'attaquant :

$$Adv^A = \left| Pr\left[A\left(E_k(x)\right) = f(x)\right] - Pr\left[A\left(E_k(x')\right) = f(x)\right] \right| \quad (5)$$

est une fonction négligeable (*negl(n)*).

Nous signalons que :

- *L'attaquant (A) du schéma de chiffrement à clé secrète $\Pi = (G, E_k, D_k)$, contre la sécurité sémantique, est un algorithme qui doit retrouver une information f(x) sur le message clair (x) à partir du cryptogramme $E_k(x)$.*

- $X = (X_1, X_2...)$ est une distribution telle que pour le paramètre de sécurité (*n*), le message clair est choisi en fonction de la distribution *Xn.*

- Pour tout *(n),* toutes les chaînes binaires dans *Xn* ont la même longueur.

- $G(1^n)$ choisit la clé $k \leftarrow \{0, 1\}^n$ toujours uniformément au hasard.

Dans la définition, l'expression de l'avantage exprime qu'il n'y a pas de différence de probabilité de succès notable de l'algorithme (A) pour retrouver l'information *f(x)* qu'on lui fournisse le message chiffré $E_k(x)$ correspondant au clair (x) ou le message chiffré $E_k(x')$ correspondant à un autre message clair *(x')*. Le cryptogramme $E_k(x)$ ne fournit aucune information sur la valeur de *f (x)*.
La sécurité sémantique est atteinte si l'avantage de l'attaquant est négligeable pour toute distribution de probabilité sur les messages clairs.

2.5.1.2 Conclusion

Malheureusement, la définition de la sécurité sémantique est complexe et difficile à appliquer.
Cependant, il y'a heureusement une définition équivalente qui est ***« l'indistinguabilité »,*** et qui est beaucoup plus simple et accessible. Les définitions étant équivalentes, nous pouvons adopter celle liée à « *l'indistinguabilité* », étant données que les garanties de sécurité sont celles offertes par *la sécurité sémantique***.**

2.5.2 L'indistinguabilité

2.5.2.1 Rappels

Dans la formalisation de l'indistinguabilté énoncée aux paragraphes 2.3.1.1 et 2.3.1.2, nous avons considéré une expérience d'indistinguabilité $SK_{II, A}$ dans laquelle l'attaquant (A) sort deux messages x_o et x_1 et le chiffrement de l'un de ces messages, choisi au hasard, en utilisant une clé générée de façon aléatoire. La formalisation 2.3.1.2 exprime qu'un schéma de chiffrement II est sûr si aucun attaquant (A) ne peut déterminer lequel des messages clairs (x_0, x_1) a été chiffré, avec une probabilité de succès meilleure que 1/2 (probabilité de tirage au sort).

En d'autres termes, un schéma de chiffrement à clé secrète possède une propriété d'indistinguabilité si un attaquant (A) est incapable de dire si le texte chiffré qu'on lui présente provient de l'un ou de l'autre message clair, sachant que ce texte chiffré est établi à partir de l'un de ces deux messages clairs.

2.5.2.2 Définition (Indistinguabilité) : Un schéma de chiffrement à clef secrète

$\Pi = (G, E_k, D_k)$ possède la propriété d'indistinguabilité si pour tout attaquant (A) en temps polynomial probabiliste et pour tout couple de messages (x_0, x_1), il vient :

$$Adv^A = \left| Pr[A(E_k(x_0)) = 1] - Pr[A(E_k(x_1)) = 1] \right| \quad (6)$$

est une fonction négligeable ((*negl(n*)).

Nous signalons que :

- $G(1^n)$ choisit toujours la clé $k \leftarrow \{0, 1\}^n$ uniformément au hasard

- *L'attaquant (A) d'un schéma de chiffrement à clé secrète Π = (G, E k, D k) contre la propriété de l'indistinguabilité est un algorithme à qui on fournit le cryptogramme E k (xi) d'un message clair (xi) choisi aléatoirement dans un ensemble de deux messages distincts de même longueur { x0, x1} et qui doit fournir à la sortie l'indice i ∈{ 0, 1}.*

  Un des deux messages clairs est chiffré et le message chiffré est donné à l'attaquant. Il doit déterminer lequel des deux messages clairs a été chiffré. Si l'algorithme polynomial probabiliste exécuté ne permet pas de répondre avec une probabilité de succès significativement plus grande que le tirage au sort (½), ***il n'a aucun avantage***. *Et dans ce cas, on dit que le système est indistinguable vis à vis d'une attaque du niveau de puissance considéré.*

La propriété de l'indistinguabilité énonce que tout attaquant échoue pour tout couple de messages clairs *(x0, x1)*.

2.5.2.3. Définition équivalente (Indistinguabilité) : Un schéma de chiffrement à clé secrète Π = (G, Ek, Dk) produit des chiffrements indistinguables, si pour tout attaquant (A), en temps polynomial probabiliste, il existe une fonction négligeable « *negl*» telle que :

$$P_r[SK_{\Pi, A}(n) = 1] \leq \frac{1}{2} + \text{negl}(n) \qquad (7)$$

lorsque la probabilité concerne les éléments aléatoires utilisés par A ainsi que les éléments aléatoires utilisés pour l'expérience (choix de la clé et de toute autre variable aléatoire utilisée dans le processus de chiffrement).

En conclusion, l'indistinguabilité s'exprime de la façon suivante : « *Etant donnés deux messages clairs distincts et de même longueur, et le message chiffré de l'un d'eux, l'Attaquant ne peut déterminer en temps raisonnable, avec une probabilité de succès significativement plus grande que le tirage au sort (1/2), de quel message clair provient le message chiffré* ». [10]

2.5.3 Equivalence de la sécurité sémantique et de l'indistinguabilité

Compte tenu de la conclusion 2.5.1.2, il est techniquement plus facile de travailler avec la définition de l'indistinguabilité (par exemple, pour prouver qu'un schéma de chiffrement donné est sécurisé), plutôt qu'avec la notion de sécurité sémantique. Heureusement, il a été montré [11] que les définitions sont équivalentes :

THEOREME (**Equivalence de la sécurité sémantique et de l'Indistinguabilité**) : Un schéma de chiffrement à clé secrète est sémantiquement sûr, si et seulement si, il a la propriété d'indistinguabilité.

En d'autres termes, il fournit des chiffrements indistinguables en présence d'un attaquant (A), si et seulement si, il est sémantiquement sûr en présence de cet attaquant.

2.6 Résumé schématique des notions de sécurité cryptographique

Les notions de sécurité cryptographique, détaillées ci-dessus, peuvent être résumées schématiquement comme suit :

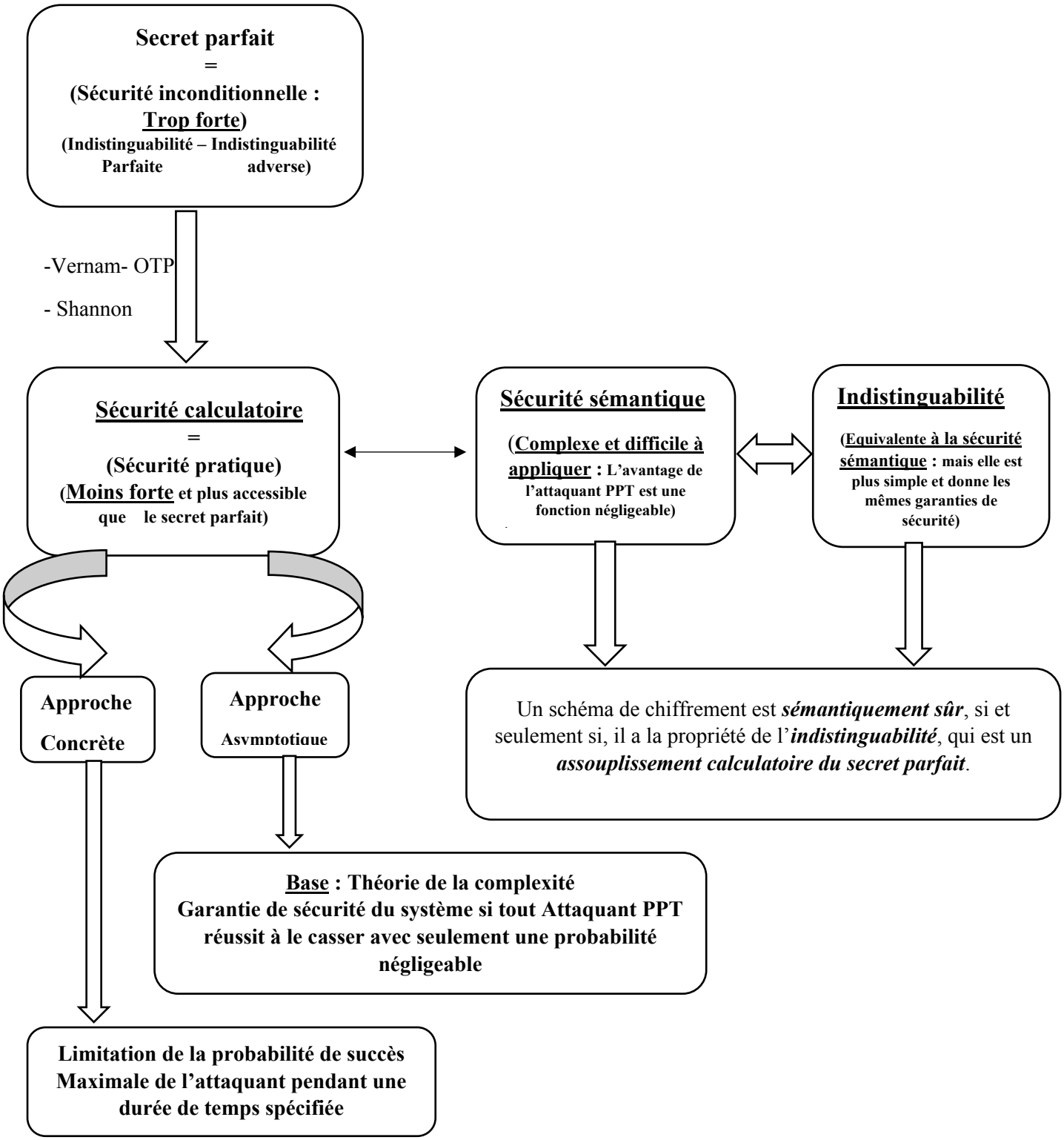

Figure 2.2 : Résumé des notions de sécurité cryptographique

Nous allons donc utiliser cette notion d'indistinguabilité dans nos travaux relatifs aux générateurs pseudo-aléatoires exposés dans la seconde partie de cette thèse, tout en étant assuré que les garanties de sécurité obtenues sont celles de la sécurité sémantique.

REFERENCES

[1] Auguste Kerckhoffs, "La cryptographie militaire", *Journal des sciences militaires*, vol. IX, p. 5– 38, janvier 1883, p. 161–191, févr. 1883.

[2] Jonathan Katz, Yehuda Lindell, Introduction to Modern Cryptography, Chapman & Hall/CRC Taylor & Francis Group, 2007.

[3] Vernam, Gilbert S. (1926), "Cipher Printing Telegraph Systems For Secret Wire and Radio Telegraphic Communications", Journal of the IEEE **55**: 109–115.

[4] C. E. Shannon. A mathematical theory of communication. Bell System. Tech. Journal, Vol. 27, pages 623-656, 1948.

[5] C.E. Shannon. Communication Theory of Secrecy Systems. Bell System. Tech. Journal, Vol. 28, pages 656-715, 1949

[6] Oded Goldreich, Modern cryptography, Probabilistic proof and pseudorandomness, springer 1999.

[7] Oded Goldreich. A Note on Computational Indistinguishability. Information Processing Letters, Vol. 34, pages 277-281, May 1 990

[8] Oded Goldreich. Foundations of Cryptography: Vol 1. Cambridge University Press, 2001.

[9] Oded Goldreich. Foundations of Cryptography: Vol 2. Cambridge University Press 2004

[10] Pierre Barthélémy, Robert Rolland, Pascal Veron, Cryptography : Principes et mise en œuvre, Ed. Lavoisier, 2005-2012.

[11] Hieu Phan, Philippe Guillot, Preuves de sécurité des schémas cryptographiques, Paris 8, oct.2013.

.

Chapitre 3

Les générateurs d'aléas

Sommaire

Après avoir défini les notions de sécurité cryptographique pour notamment les crypto système à clés secrètes, il nous est nécessaire d'aborder dans ce chapitre « la construction et la sécurité des générateurs d'aléa », afin de pouvoir exposer, à la deuxième partie de la thèse, nos travaux y afférents.

La notion « d'aléa » c'est-à-dire de production d'informations secrètes imprévisibles (suites de bits aléatoires ou pseudo aléatoires) est la base fondamentale de la sécurité des crypto systèmes :

- En cryptographie à clé secrète, il faut disposer de quantités de données aléatoires ou pseudo aléatoires qui peuvent être utilisées comme clés secrètes : un générateur d'aléa est utilisé pour générer et établir les clés secrètes partagées entre les correspondants dans les réseaux.

- En cryptographie à clé publique, il faut avoir aussi ce type de données pour générer des paires de clés publiques : un générateur d'aléa est également utilisé pour générer les paires de clés publiques/privées.

Les crypto systèmes utilisés, dans ces deux catégories, peuvent être probabilistes, et la probabilité de n'importe quelle valeur particulière qui est sélectionnée doit être suffisamment petite pour empêcher un attaquant d'avoir de l'avantage grâce à l'optimisation d'une stratégie de recherche basée sur cette probabilité.

D'autre part, il importe de préciser qu'une suite binaire pseudo aléatoire ressemble à une suite binaire uniformément répartie, tant que l'entité qui l'observe fonctionne en temps polynomial. Dans la mesure où « ***l'indistinguabilité*** » est considérée comme « ***un assouplissement calculatoire du secret parfait*** », « ***le pseudo-aléatoire*** » est également considéré comme « ***un assouplissement calculatoire du vrai aléa*** ». [1-6].

L'avantage d'utiliser, en chiffrement à clef secrète (surtout en chiffrement en continu), une suite binaire pseudo-aléatoire à la place d'une suite binaire vraiment aléatoire, résulte du fait qu'une longue suite binaire pseudo-aléatoire peut être générée à partir d'un *germe* (*seed*) aléatoire de longueur relativement courte (*une clé courte*). Une courte clé peut ainsi être vue comme pouvant être utilisée pour chiffrer un long message, ce qui est impossible lorsque le secret parfait est requis. [1]

3.1 Rappels

[4] nous signale que trois théories de l'aléa ont été développées :

- *Première théorie* [7] : Initiée par Shannon [8], elle s'enracine dans la théorie des probabilités. Selon Shannon, l'aléa parfait, associé à une distribution unique, est le cas extrême dans lequel il n'y a pas de redondance de l'information. Par définition, on ne peut donc pas générer de suites binaires aléatoires parfaites à partir de germes courts et aléatoires.

- *Deuxième théorie* [9-10] : Initiée par Solomonov [11], Kolmogorov [12] et Chait [13], elle s'enracine dans la théorie de la complexité. Après analyse, on ne peut pas aussi, à ce niveau, générer des suites binaires de haute complexité, en termes de germes courts et aléatoires.

- *Troisième théorie* : Elle est due à Blum, Goldwasser, Micali et Yao [14-16], et elle s'enracine dans la théorie de la complexité. Cette théorie définit la notion d'aléa parfait qui permet de générer efficacement des suites binaires parfaitement aléatoires à partir de germes aléatoires courtes. Selon l'étude, une distribution qui ne peut être ***distinguée*** de manière efficace de la distribution uniforme est considérée comme étant aléatoire (ou *elle est plutôt pseudo-aléatoire*).
  Il en résulte que l'aléa n'est pas une propriété inhérente aux distributions, mais plutôt à un observateur et à sa capacité de calcul *: « une distribution est pseudo-aléatoire si aucune procédure efficace ne peut la distinguer de la distribution uniforme, où des procédures efficaces sont associées à des algorithmes polynomiaux (probabilistes).* Cette notion de pseudo-aléatoire (l'indistinguabilité) est la plus appropriée, et nous allons donc retenir cette troisième théorie tout au long de nos travaux.

Nous allons nous intéresser auparavant aux concepts de fonctions à sens unique, de permutations à sens unique et étudier comment ces fonctions peuvent être utilisées pour construire les générateurs pseudo-aléatoires.

3.1.1 Fonctions à sens unique (One-Way Functions=OWF)

Une fonction f : $\{0, 1\}^* \mapsto \{0, 1\}^*$ est dite à sens unique, si :

1) Elle est facile à calculer dans un sens : il existe un algorithme (A) qui, étant donnée une entrée *(x)*, sort *f(x) ;*

2) Elle est difficile à calculer dans l'autre sens : Etant donnée *f (x),* il est impossible de trouver, avec une probabilité non négligeable, une pré-image de *f(x).* Tout algorithme en temps polynomial probabiliste, qui tente d'inverser *(f)* ne peut réussir qu'avec une probabilité négligeable,

la probabilité tenant compte du choix de ($x \leftarrow \{0, 1\}^n$) et de la valeur de *(n)* pour le paramètre de sécurité :

$$\Pr_{x \leftarrow \{0,1\}^n}\left[A\,(f(x)) \in f^{-1}(f(x))\right] \leq negl\,(n), \qquad (1)$$

3.1.2 Permutations à sens unique (One-Way Permutation=OWP)

DEFINITION : Soit $f : \{0, 1\}^* \mapsto \{0, 1\}^*$ une fonction de maintien de la longueur -*length-preserving*- (c-à-d $|f(x)| = |x| \Leftrightarrow |x|=|y|, \forall x$). et soit f_n la restriction de f au domaine $\{0, 1\}^n$ (c.-à-d. que f_n est définie seulement pour $x \in \{0, 1\}^n$, auquel cas $f_n(x) = f(x)$), alors :

« Une fonction (f) à sens unique est appelée une permutation à sens unique si pour tout (n), la fonction (f_n) est une bijection »

Les permutations à sens unique possèdent une propriété intéressante : *toute valeur (y) détermine uniquement la pré-image $x = f^{-1}(y)$. Même si (y) détermine entièrement (x), il est toujours difficile de trouver (x) en temps polynomial.*

3.1.3 Hard-Core Predicates (Prédicat difficile=Bit difficile)

Une fonction à sens unique est difficile à inverser, de sorte que, calculatoirement, aucune information ne peut être déterminée en temps polynomial sur *(x)* à partir de *f (x)*. Un bit difficile (*hard-core prédicat (prédicat difficile ou bit difficile)* d'une fonction est une information binaire sur l'entrée qu'il est difficile de retrouver à partir de la valeur de la fonction.

Une fonction booléenne $B : \{0, 1\}^* \mapsto \{0, 1\}$ est **un bit difficile pour une fonction** $f : \{0, 1\}^* \mapsto \{0, 1\}^*$, si elle a la propriété suivante :

« *Etant donnée f* (*x*), *il est impossible pour tout algorithme en temps polynomial de déterminer correctement B* (*x*) *avec une probabilité significativement meilleure que* $\frac{1}{2}$. (*Il est toujours possible de calculer B* (*x*) *avec une probabilité égale à* $\frac{1}{2}$, *en devinant au hasard* ».

Formellement, nous pouvons retenir la définition suivante :

DEFINITION : Etant donné une fonction $f : \{0, 1\}^* \mapsto \{0, 1\}^*$, une fonction booléenne $B : \{0, 1\}^* \mapsto \{0, 1\}$ est un bit difficile pour la fonction f si :

1) La fonction (*B)* peut être calculée en temps polynomial, et

2) Tout algorithme polynômial (A), ne peut calculer *B(x)* à partir de *f(x)* qu'avec une probabilité négligeable ; c'est-à-dire qu'il existe une fonction (*négl*) telle que :

$$\Pr_{x \leftarrow \{0,1\}^n}\left[A\,(f\,(x)) = B(x)\right] \leq \frac{1}{2} + negl\,(n), \qquad (2)$$

Remarque :

- *Les fonctions à sens unique permettent de construire des générateurs pseudo-aléatoires.*
- *A partir de n'importe quelle permutation à sens unique, on peut construire des générateurs pseudo-aléatoires.*
- *De même, le prédicat difficile d'une permutation à sens unique peut être utilisé pour construire un générateur pseudo-aléatoire*

3.1.4 Bit difficile pour toute fonction à sens unique

Etant donnée une fonction à sens unique (f), Oded Goldreich et L.A Levin ont démontré [17] qu'on peut construire parallèlement une autre fonction à sens unique (g) qui lui est similaire et qui a un bit difficile.

THEOREME de GOLDREICH-LEVIN : Soit (f) une fonction à sens unique et (g) une fonction telle que

$g : (x, r) \mapsto (f(x), r)$, avec $|x| = |r|$, alors la fonction booléenne $(x, r) \mapsto x.r$ est un bit difficile pour (g) ; où le produit scalaire $x.r = \sum_{i=1}^{n} x_i \, r_i = \bigoplus_{i=1}^{n} x_i \cdot r_i$, avec $(x = x_1 \cdots x_n)$ et $(r = r_1 \cdots r_n)$.(3)

Remarques : La fonction $g(x, \cdot)$ donne le résultat par OU EXCLUSIF des bits d'un sous-ensemble aléatoire de (x). Cela est dû au fait que (r) peut être considéré comme la sélection d'un sous-ensemble aléatoire de $\{1, 2, \ldots, n\}$ (à savoir si $r_i = 1$, le bit (x_i) est inclus dans le XOR, et sinon, il ne l'est pas), et (r) est uniformément répartie.
Ainsi, le théorème indique essentiellement que si (f) est une fonction arbitraire à sens unique, alors $f(x)$ cache le OU exclusif d'un sous-ensemble aléatoire des bits de (x).

Un bit difficile d'une fonction à sens unique est important en cryptographie comme mentionné ci-dessus. En outre, il permet de satisfaire la propriété de discrétion ; l'existence d'une fonction à sens unique implique par exemple l'existence de preuve sans divulgation de la connaissance d'une solution de problème NP (ZK=Zero-Knowledge proof).

3.2 Les générateurs de bits aléatoires (RBG-Random Bits Generators-catégorie : crypto systèmes sans clés) [18]

3.2.1 Définitions

Avant d'aborder le sujet, il importe de rappeler qu'on appelle :

- « ***Algorithme déterministe*** : *un algorithme qui effectue une suite d'opérations qui ne dépend que des données d'entrée, et non des résultats de tirages aléatoires, à partir des mêmes données initiales, il donne toujours le même résultat* »

- ***« Algorithme non-déterministe ou probabiliste*** : *un algorithme qui effectue des tirages aléatoires lors de son exécution ; à partir des données initiales, il produit des résultats différents, chacun de ces résultats ayant une certaine probabilité d'apparition* ».

1ère DEFINITION *: Un générateur de bits aléatoires est* ***un dispositif*** *ou* ***un algorithme non-déterministe*** *qui génère des suites de bits statistiquement indépendants et imprévisibles.*

C'est dire que les bits doivent toujours se produire avec la même probabilité d'apparition :

$$P_r[0] = P_r[1] = 1/2 \qquad (4)$$

et tous les (2^n) n-uplets possibles doivent aussi se produire approximativement de façon égale avec la probabilité $P_r = \frac{1}{2^n}$, $\forall\ n \in \mathbb{N}$.

2ème DEFINITION : *Alternativement, un générateur de bits aléatoires est parfois également défini comme un modèle idéalisé d'un dispositif ou d'un algorithme qui génère des suites de bits statistiquement indépendants et imprévisibles.*

Dans les deux cas, il est important de noter qu'un *générateur de bits aléatoires* n'a aucune entrée, et qu'il génère seulement une sortie de bits aléatoires.

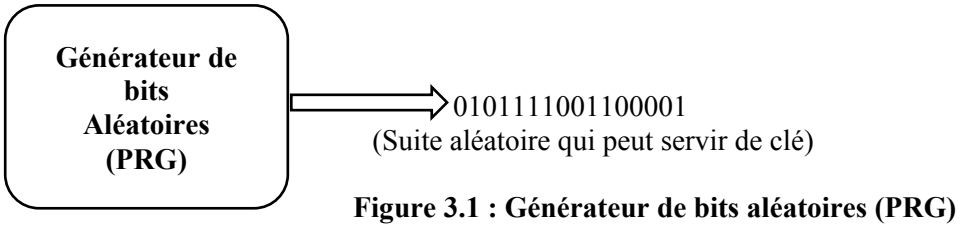

Figure 3.1 : Générateur de bits aléatoires (PRG)

Il existe des procédés (matériels/logiciels) de génération de bits aléatoires ainsi que des tests statistiques qui peuvent être utilisés pour vérifier les propriétés (aléatoires) d'un générateur de bits aléatoires donné.

3.2.2 Générateurs de bits aléatoires (matériels) [19-21]

Les générateurs de bits aléatoires non-déterministes, de conception matérielle, utilisent des sources physiques imprévisibles, tels que :

- L'émission, par un atome, d'un rayonnement radioactif ;
- Les mouvements de l'utilisateur d'un clavier et d'une souris d'un ordinateur ;
- Le bruit thermique d'une diode à semi-conducteur ou d'une résistance ;
- Les statistiques de l'activité d'un disque dur d'un ordinateur ;
- L'instabilité de la fréquence d'un oscillateur de course libre ;
- Le son d'un microphone ou l'entrée vidéo d'une caméra.

Ces différents éléments peuvent être combinés. De même, la liste n'est pas exhaustive : d'autres phénomènes et processus physiques peuvent être employés dans la conception ainsi que dans la réalisation matérielle des générateurs de bits aléatoires, et ils pourraient être intégrés dans les équipements informatiques.

3.2.3 Générateurs de bits aléatoires (logiciels) [22-23]

La conception logicielle d'un générateur de bits aléatoires est plus difficile à réaliser que celle matérielle. [18] nous signale qu'il existe de nombreux processus qui peuvent être utilisés pour la conception des générateurs de bits aléatoires basés sur des logiciels [22-23].

3.3 Les générateurs de bits pseudo aléatoires (PRBG-Pseudo Random Bits Generators) [1, 2, 4-6, 17-18, 25-30]

3.3.1 Définitions

Dire qu'une distribution sur des suites binaires de longueur (*T)* est pseudo-aléatoire, c'est dire que cette distribution est indistinguable d'une distribution uniforme sur ces mêmes suites binaires de longueur (*T*).

1ère DEFINITION *: Un générateur pseudo-aléatoire est* ***un algorithme déterministe*** *qui reçoit un germe parfaitement aléatoire (seed) et fournit une longue suite binaire qui ressemble à une suite de bits aléatoires. Cette suite binaire de sortie est appelée suite pseudo- aléatoire.*

L'architecture générale d'un PRBG comprend (cf. Figure 3.3) :

- Un registre d'état
- Une fonction (*f*) de changement d'état
- Une fonction (*g*) de sortie.

Le registre d'état est initialisé avec un germe (s_0), et la fonction de changement d'état (*f*) calcule (s_{i+1}) à partir de (s_i) pour $i \geq 0$. Pour chaque (s_i), la fonction (*g*) calcule une valeur binaire (x_i) (un bit ou une séquence de bits) de sortie pour le PRBG.

Par conséquent, la PRBG génère et délivre à la sortie une suite binaire : $(x_i)_{i\geq 1} = x_1, x_2, x_3, \ldots$

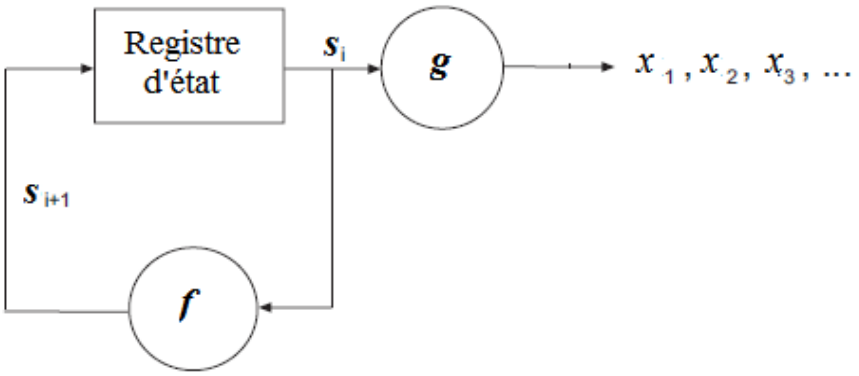

Figure 3.3 : Architecture d'un PRBG

De façon formelle, nous pouvons retenir :

2ème DEFINITION *:* Soit *T (·)* un polynôme et soit (*G*) un algorithme déterministe en temps polynomial telle que pour toute entrée (*s) ϵ {0, 1}* ^{n}, l'algorithme (G) fournit une chaîne de longueur *T (n).* On dit que G est un générateur pseudo-aléatoire si les deux conditions suivantes sont vérifiées :

1) (L'expansion) *: ∀ n, T(n)> n* ; (La fonction *T (·)* est appelé le facteur d'expansion de (*G*).

2) (Le pseudo-aléatoire) : *∀ (D)* un algorithme en temps polynômial probabiliste, ∃ une fonction négligeable (*negl*), telle que :

$$P_r [D(r) = 1] - P_r [D (G(s)) = 1] | \leq negl (n) \quad (5)$$

où *(r)* est choisi uniformément au hasard dans *{0, 1}* $^{T(n)}$, le germe *(s)* est choisi uniformément au hasard dans *{0, l}* ^{n}, et les probabilités sont calculées sur les variables aléatoires utilisées par *(D)* et le choix de *(r)* et *(s).*

3.3.2 Les PRBG construits à partir des permutations à sens unique.

3.3.2.1 La construction avec expansion d'un bit

Le bit difficile d'une permutation à sens unique peut être utilisé pour construire un générateur pseudo-aléatoire grâce au théorème suivant.

THEOREME : Soit *(f)* une permutation à sens unique et soit *B* : {0, 1} * ------> {0, 1} un bit difficile de *(f),* alors la fonction :

$$G : \{0, 1\}^{n} \mapsto \{0, 1\}^{n+1}, \quad (6)$$
$$(s) \mapsto (f(s), B(s))$$

constitue un générateur pseudo aléatoire avec une fonction d'expansion *T(n)= n+1*.

En effet, toute la sortie de (G) est pseudo-aléatoire, puisque :

- les (*n)* premiers bits sortis de G (s) sont vraiment aléatoires (bits de *f(s)*) étant donné que (*f*) est une permutation à sens unique.
- Le bit difficile *B* (*s*) pour la fonction *f* (*s*), est *pseudo aléatoire*.

Mais, ce générateur est très simple. Aussi, pour des applications qui sont notamment liées au chiffrement et au déchiffrement de longs messages, il faut, pour des raisons de sécurité, disposer de générateurs pseudo-aléatoires avec une fonction d'expansion plus grande (objet du paragraphe ci-dessous).

3.3.2.2 La construction avec expansion polynomiale.

- Soient *(f)* une permutation à sens unique et *B* : {0, 1} * ↦ {0, 1} un bit difficile de *(f)* ;
- Soit *(F)* un générateur pseudo-aléatoire telle que :

$$F : \{0, 1\}^n \mapsto \{0, 1\}^{n+1}, \quad (7)$$
$$(s) \mapsto (f(s), B(s))$$

- Soit $p(n)$ un polynôme tel que $\forall n \in \mathbb{N}, p(n) > n$ et soit $T(n)=p(n)$
  Soit (G) une fonction définie sur $\{0, 1\}^*$ dont la valeur sur les vecteurs binaires (s) de dimension (n) est : $G(s) = \sigma_1\sigma_2 \ldots \sigma_{p(n)}$ telles que $s_0=s$ et $F(s_{i-1}) = (s_i, \sigma_i)$ pour $i=1$ à $p(n)$.

$$\Rightarrow \quad G: \{0, 1\}^n \mapsto \{0, 1\}^{p(n)},$$
$$(s) \mapsto G(s) = \sigma_1\sigma_2 \ldots \sigma_{p(n)}$$

- Pour $i=1$ à $p(n)$, $F(s_0) = (f(s_0), B(s_0)) = (s_1, \sigma_1)$
  $F(s_1) = (f(s_1), B(s_1)) = (s_2, \sigma_2)$
  $F(s_2) = (f(s_2), B(s_2)) = (s_3, \sigma_3)$
  . . . .
  . . . .
  . . . .
  $F(s_{p(n)-1}) = (f(s_{p(n)-1}), B(s_{p(n)-1})) = (s_{p(n)}, \sigma_{p(n)})$

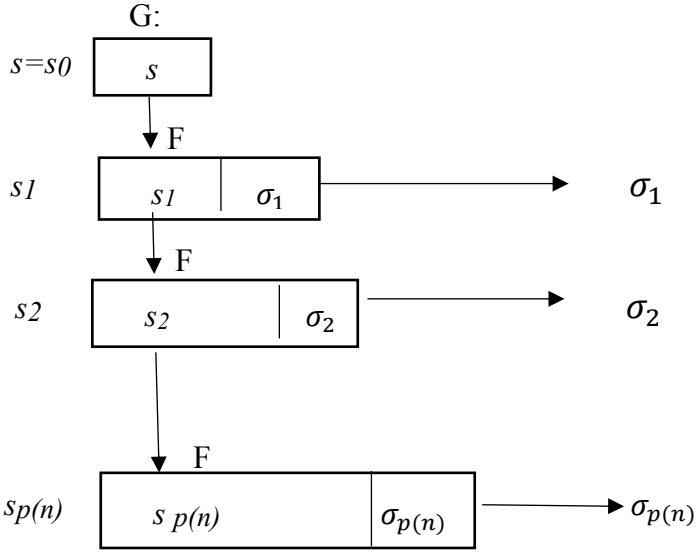

Figure 3.4 : PRBG avec expansion polynomiale

Il vient ; $G(s) = \sigma_1\sigma_2 \ldots \sigma_{p(n)} = B(s)\, B(s_1)\, B(s_2) \ldots\ldots B(s_{p(n)-1})$

$$G(s) = B(s)\, B(f(s))\, B(f^2(s)) \ldots B(f^{p(n)-1}(s)) \quad (8)$$

qui est un générateur pseudo-aléatoire.

THEOREME : S'il existe un générateur pseudo-aléatoire (*F*) avec une fonction d'expansion *T (n) = n + 1*, alors pour tout polynôme *p (n)> n*, il existe un générateur pseudo-aléatoire (*G*) avec une fonction d'expansion polynomiale *T (n) = p (n)*.

THEOREME Soit $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ une permutation à sens unique avec expansion de longueur avec un bit difficile $B : \{0, 1\}^* \rightarrow \{0, 1\}$. Si $p(\cdot)$ est un polynôme, alors :

$G : \{0, 1\}^n \mapsto \{0, 1\}^{p(n)}$, défini par :

$$G(s) = B(s)\, B(f(s))\, B(f^2(s)) \ldots B(f^{p(n)-1}(s))$$

est un générateur pseudo-aléatoire.

En conclusion, les générateurs pseudo-aléatoires avec fonction d'expansion polynomiale peuvent être construits à partir de toute permutation à sens unique.

3.3.2.3 Parenthèse sur la construction de fonctions et de permutations pseudo-aléatoires à partir des générateurs pseudo-aléatoires.

Il est également démontré qu'on peut construire des fonctions pseudo-aléatoires et des permutations pseudo-aléatoires à partir des générateurs pseudo-aléatoires pour la réalisation notamment des schémas de chiffrement par blocs (qui ne nous intéressent pas ici) [1-2 ; 4-6 ; 27-35]. *En fait, tous les schémas de chiffrement à clés secrètes peuvent être construits à partir des générateurs pseudo-aléatoires et des fonctions pseudo-aléatoires* :

1. S'il existe une permutation à sens unique, alors il existe un Générateur pseudo-aléatoire.
2. S'il existe un générateur pseudo-aléatoire, alors il existe une fonction pseudo-aléatoire.
3. S'il existe une fonction pseudo-aléatoire, alors il existe une (forte) permutation pseudo-aléatoire

THEOREME : S'il existe des fonctions à sens unique, alors il existe des générateurs pseudo-aléatoires, des fonctions pseudo-aléatoires, et les permutations pseudo-aléatoires fortes.

Hastad, Impagliazzo, Levin et Luby ont effectivement montré dans [33] qu'un générateur pseudo-aléatoire existe si et seulement si une fonction à sens-unique existe.

Dans [27] Goldreich, Goldwasser et Micali ont indiqué comment construire des fonctions pseudo-aléatoires à partir d'un générateur pseudo-aléatoire.

De même dans [30] Luby et Rackoff ont montré comment construire des permutations pseudo-aléatoires fortes à partir de fonctions pseudo-aléatoires (à condition qu'il existe des fonctions à sens-unique).

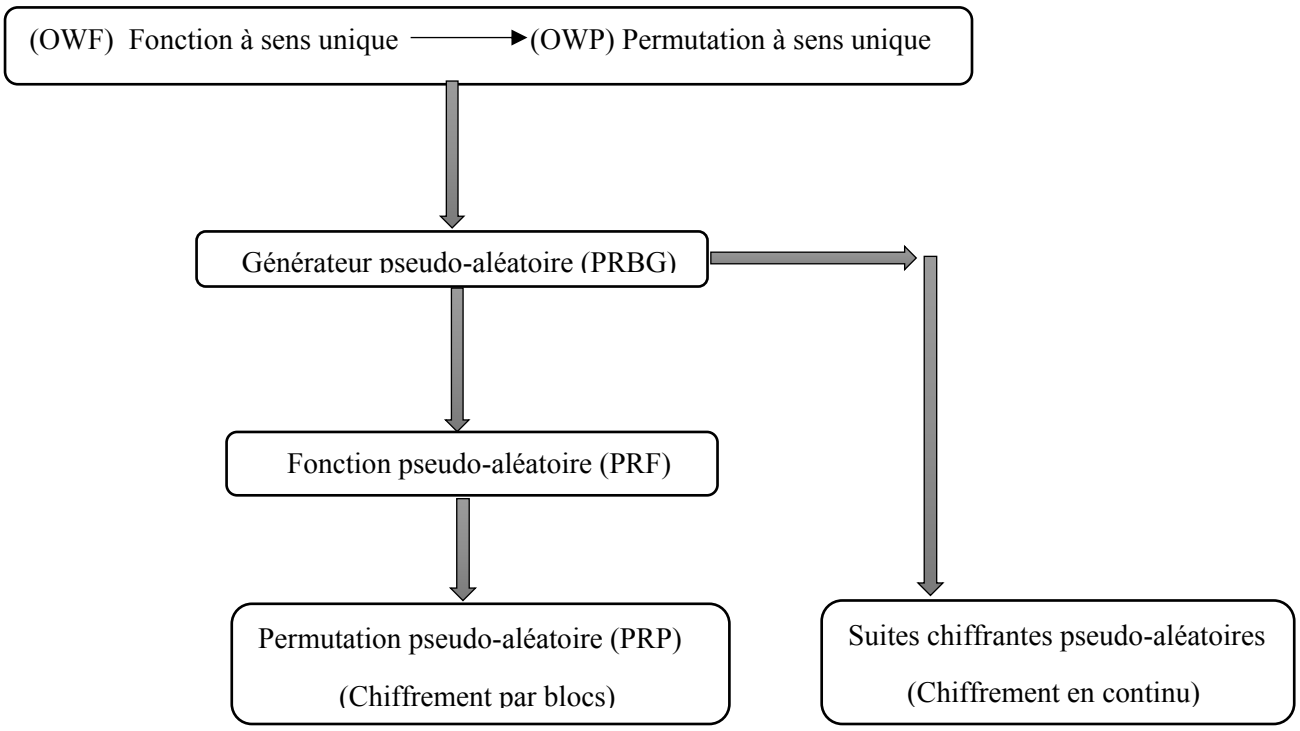

Figure 3.5 : Construction des schémas de chiffrement à clés secrètes à partir des PRBG, des fonctions pseudo aléatoires et des permutations pseudo aléatoires

3.3.3 Autres types de PRBG

D'autres générateurs de bits pseudo-aléatoires avaient, auparavant, été développés : Générateurs middle square (John van Neumann-1946), Générateurs congruentiels linéaires (G.H. Lehmar-1948, G.J. Mitchell et D.P Moore-1958), Générateur Mersenne Twister (Makoto Matsumoto et Takuji Nishimura-1997).

Les Générateurs de bits pseudo aléatoires à base de registres à décalage (*LFSR/NLFSR*), de construction plus simple et qui sont utilisés pour générer des suites pseudo-aléatoires, sont largement détaillés dans notre article [36].

Il faut signaler aussi ***le générateur pseudo aléatoires hybrides*** construits à partir des fonctions de hachage ou des algorithmes de chiffrement par blocs, tels que :

- ANSI X-9.17 PRBG [46] : l'algorithme prend en entrée un germe (s) aléatoire (et secret), une clé (k) de 3DES en mode CBC et un entier (n). Il produit une suite de (n) chaînes de 64 bits pseudo aléatoire.

$$(s, k, n)$$

$$I \leftarrow E_k(D)$$
for $i = 1$ *to* n *do*
$$x_i \leftarrow E_k(I \oplus s)$$
$$s \leftarrow E_k(x_i \oplus I)$$
Output x_i

Figure 3.6 : Algorithme ANSI X9.17

(D=représentation interne sur 64 bits de la date et de l'heure ; I= valeur intermédiaire dans une étape d'initialisation de l'algorithme).

- Le PRBG hybride Yarrow-160 décrit dans [47] en est un autre exemple ; il utilise SHA1 et le triple-DES. Il a été amélioré par la suite par le PRBG Fortuna [48].

- Il faut noter que FIPS 186-2 [49] définit également un PRBG basé sur l'algorithme de signature DSA et d'autres méthodes de construction à partir notamment de SHA-1 et DES.

Remarque :

Dans la pratique, il est surtout conseillé de générer des suites aléatoires, à l'aide d'un PRG, et de les utiliser comme germes (seeds/clés courtes, aléatoires et imprévisibles) pour amorcer le PRBG. Le PRBG est ensuite utilisé pour générer une suite potentiellement infinie de bits pseudo-aléatoires qui peut être utilisée comme suite chiffrante dans le chiffrement en continu. (C'est le cas ***des générateurs dits hybrides*** où du germe est de temps en temps introduit dans l'état interne du générateur pseudo-aléatoire pour le rafraîchir afin d'éviter que l'état interne ne soit prévisible s'il arrive qu'il soit compromis).

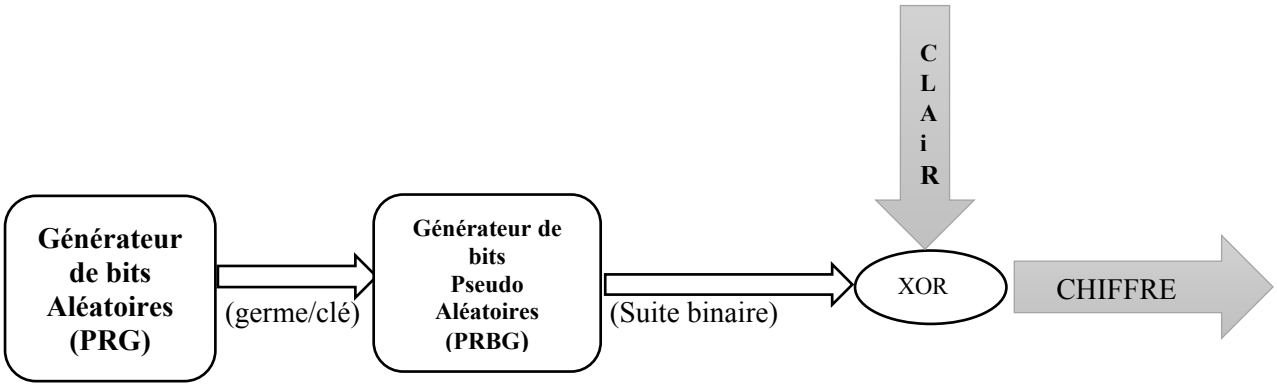

Figure 3.7 : Amorçage d'un PRBG à l'aide d'un PRG

Ainsi, une courte clé peut être utilisée pour générer une longue suite binaire pseudo aléatoire en vue du chiffrement un long message, alors qu'il est impossible d'utiliser la courte clé pour le chiffrer lorsque le secret parfait est requis.

3.4 Sécurité des générateurs pseudo-aléatoires

3.4.1 Les attaques sur les générateurs

La sécurité des systèmes de chiffrement en continu reposent entièrement sur la sécurité des générateurs pseudo-aléatoires mis en œuvre. Ces générateurs doivent résister à quatre (4) d'attaques :

1) ***Les attaques par recouvrement de clé*** : Elles consistent à rétablir la clé secrète utilisée par le PRBG à partir de la connaissance d'un certain nombre de bits de sa sortie.

2) ***Les attaques par recouvrement de l'état initial*** : Elles consistent à retrouver l'initialisation du PRBG, ce qui permet à l'attaquant de calculer autant de bits qu'il veut à partir de cet état initial. Elles ne lui permettent pas pour autant de retrouver la clé secrète, même si la connaissance de celle-ci peut aider à rétablir l'état initial du PRBG. Ces attaques sont donc moins puissantes que celles décrites en (1).

3) ***Les attaques par prédiction du bit suivant*** (Next-bit test) : Elles consistent, à partir de la connaissance des (*n*) premiers bits de la suite produite par le PRBG, à prédire le bit suivant.
L'attaquant est considéré comme « *un prédicteur » c.-à-d. un algorithme en temps polynomial probabiliste P qui tente de prédire le prochain bit d'un générateur de bits G (x) compte tenu de ses bits initiaux.* Tout bon générateur devrait être imprévisible : par conséquent, avec un test du prochain bit (next-bit test), il doit échouer près de la moitié du temps.

4) ***Les attaques par distingueur*** : Elles permettent de déterminer si une suite de (*n*) bits correspond à la sortie du générateur pseudo-aléatoire considéré ou s'il s'agit d'une suite vraiment aléatoire.
On peut montrer que l'existence d'attaques par distingueur de complexité polynomiale est strictement équivalente à celle d'attaques polynomiales par prédiction du bit suivant.

Les PRBG qui résistent à ces types attaques sont dits « ***PRBG pratiquement forts*** » ou ***PRBG cryptographiquement sûrs*** [18, 37, 39]. Un PRBG pratiquement fort est sûr contre un ensemble spécifique d'attaques connues à un moment donné. Ils sont généralement très efficaces et peuvent être implémentées entièrement dans des logiciels.

3.4.2 Les générateurs de bits pseudo- aléatoires cryptographiquement sûrs [15-16, 18, 34, 37, 39-40, 52].

3.4.2.1 Quelques rappels utiles

1°) Le concept de PRBG cryptographquement sûr a été formalisée dès 1984, par Manuel Blum et Silvio Micali [15]. « *Un PRBG cryptographiquement sûr »* est un PRBG basé sur le Problème du Logarithme Discret (DLP), et pour lequel l'attaquant ne peut pas deviner le prochain bit dans une séquence de sortie. Il en résulte que tout attaquant qui peut résoudre le DLP peut compromettre la sécurité du PRBG de Blum-Micali).

2°) Leonore Blum, Manuel Blum et Michael Shub ont ensuite proposé en 1986, un autre PRBG, appelé le BBS (Blum-Blum-Shub), qui est sûr et dont la mise en œuvre est simple [39]. En 1988, un PRBG similaire est proposé, avec la fonction de chiffrement RSA [40]. Ce PRBG RSA est cryptographiquement sûr en ce sens que tout attaquant qui ne peut résoudre le problème de la factorisation ne réussira pas compromettre sa sécurité.

3°) Andrew C. Yao [16] a montré que ces PRBG sont tous parfaits en ce sens qu'aucun algorithme PPT ne peut permettre de déterminer avec une probabilité supérieure à 1/2 si une suite binaire de longueur (*n*), à l'entrée, a été choisie au hasard dans l'ensemble des suites possibles $\{0, 1\}^{n}$ ou si elle a été générée par ces PRBG. Cet important résultat de Yao peut être reformulé comme suit : « *un PRBG qui réussit le test du prochain bit de Blum-Micali est parfait s'il satisfait tous les tests statistiques en temps polynomial* ».

Les PRBG de Blum-Micali et BBS ainsi que la preuve de Yao et ses extensions qui ont été publiées dans [52] par Ballet Stephane et Robert Rolland, constituent une avancée majeure dans la conception des PRBG cryptographiquement sûrs.

4°) Il a été, en outre, prouvé que l'existence d'une fonction à sens unique est équivalente à l'existence d'un PRBG cryptographiquement sûr (c'est-à-dire que le PRBG satisfait tous les tests statistiques en temps polynomial) [34] [cf. paragraphe précédent].

5°) De nombreux générateurs similaires à ceux de Blum-Micali et BBS ont été proposés ; mais, généralement, ils sont tous lents pour des applications cryptographiques, contrairement aux générateurs pseudo aléatoires basés sur les registres à décalage que nous aborderont dans nos travaux (deuxième partie de la thèse).

3.4.2.2 Définitions

DEFINITION : Un PRBG est cryptographiquement sûr si aucun algorithme PPT ne peut distinguer la suite binaire qu'il fournit d'une vraie séquence de bits aléatoires de même longueur.

Soient :

$$X = \{X_n\} = \{X_{n \in \mathbb{N}}\} = \{X_1, X_2, X_3, ...\} \text{ et } Y = \{Y_n\} = \{Y_{n \in \mathbb{N}}\} = \{Y_1, Y_2, Y_3, ...\}$$

deux espaces de probabilité. $\forall\, n \in \mathbb{N}^+$, X_n et Y_n renvoient aux distributions de probabilité sur $\{0, 1\}^n$. Dire que $x \in X_n$ ($x \in Y_n$), c'est dire que $x \in \{0, 1\}^n$ et (x) est choisi en fonction des espaces de probabilité X_n (Y_n).

On dit alors que (X) est indistinguable en temps polynomial de (Y) si pour tout algorithme (D) en temps polynomial probabiliste (PPT) et tout polynôme (p), il existe un entier $N \in \mathbb{N}^+$, tel que

$\forall\, n > N$:

$$\left| \Pr_{x \in X_n} [D(x)=1] - \Pr_{x \in Y_n} [D(x)=1] \right| \leq \text{negl}(n) \quad (9)$$

(une fonction négligeable telle que définie en 2.4.2.3)

Autrement dit, $\forall\, (x)$ suffisamment grand, aucun algorithme PPT, (D), ne peut permettre d'affirmer s'il a été échantillonné à partir de X_n ou Y_n. L'algorithme PPT, (D), est appelé ***« test statistique en temps polynômial (Polynomial-time statistical test) » ou distingueur (distinguisher)***. Néanmoins, cette définition n'a pas de sens pour une seule suite binaire (x), étant donné qu'elle peut être tirée de chacune des distributions.

Dans la théorie de la complexité, il est admis une hypothèse générale d'indistinguabilité qui suppose qu'il existe des ensembles de probabilité indistinguables calculatoirement. *Par conséquent, nous admettons que $\{X_n\}$ est pseudo-aléatoire s'il est indistinguable en temps polynomial de $\{U_n\}$, où U_n désigne la distribution de probabilité uniforme sur $\{0, 1\}^n$*. Ce qui veut dire que pour tout algorithme PPT (D), et tout polynôme (p), il existe un entier $N \in \mathbb{N}^+$, tel que $\forall\, n > N$:

$$\left| \Pr_{x \in X_n} [D(x)=1] - \Pr_{x \in U_n} [D(x)=1] \right| \leq \text{negl}(n) \quad (10)$$

Ce qui permet de définir formellement la notion de PRBG cryptographiquement sûr :

DEFINITION :

Soit (G) un PRBG avec une fonction d'expansion $T : \mathbb{N} \rightarrow \mathbb{N}$ telle que $T(n) > n$, $\forall n \in \mathbb{N}$.

(G) est cryptographiquement sûr si elle satisfait les deux conditions suivantes :

1 °) $\left| G(s) \right| = T(|s|), \forall s \in \{0,1\}^*$;

2 °) $\{G(U_n)\}$ est Pseudo-aléatoire c.-à-d. qu'il est indistinguable de $\{U_{T(n)}\}$ en temps polynomial)

La première condition est liée à la propriété d'expansion du PRBG (sa sortie est plus grande que son entrée).

La deuxième condition concerne la propriété afférente au fait qu'il est mathématiquement impossible de distinguer la suite binaire pseudo-aléatoire générée par le PRBG d'une suite binaire aléatoire vraie.

3.4.2.3 Conclusion :

Les deux conditions réunies permettent d'affirmer qu'un *« **PRBG cryptographiquement sûr**, et qui est par conséquent utilisable à des fins cryptographiques, est un algorithme déterministe efficace qui est capable de procéder à l'expansion d'une suite binaire d'entrée en une suite binaire de sortie plus longue, et pour lequel aucun algorithme PPT n'est capable de distinguer la suite binaire de sortie du PRBG de la suite binaire de sortie, de même longueur, d'un générateur de bits aléatoires vrai, avec une probabilité de succès qui est non négligeable, de plus de 1/2 ».*

3.4.3 Tests statistiques sur les Générateurs de bits pseudo aléatoires

La qualité d'un générateur pseudo aléatoire est également évaluée en utilisant des tests statistiques, qui sont effectivement ***des distingueurs***, qui permettent de statuer sur le caractère pseudo-aléatoire, l'uniformité ou non des suites chiffrantes de sortie, et partant si le générateur possède ou non des faiblesses (*c'est l'objet de nos travauxs exposés à la deuxième partie de la thèse*).

3.4.3.1. Définition [41 ; 50-51]

Soit (D) un test statistique sur l'ensemble des suites binaires finies de longueur *(n)* muni d'une probabilité uniforme. Alors *(D)* est une variable aléatoire telle que :

$$D : \{0,1\}^n \rightarrow \mathbb{R}$$

Elle détermine une loi de probabilité sur $\mathbb{R}$. De ce fait, si nous voulons tester une suite aléatoire finie $x \in \{0,1\}^n$, celle-ci est vue comme un échantillon issu d'une suite de variables aléatoires indépendantes X_i avec $(i = 1, \ldots n)$, qui prennent leurs valeurs dans $\{0, 1\}$ et qui suivent une loi uniforme.
Il s'agit alors de fixer les conditions pour que la valeur $D(x_1, \ldots x_n)$ appartienne à un Intervalle validant le test (*hypothèse nulle* H_0) : la suite testée est alors déclarée aléatoire pour le test, sinon elle est rejetée (*hypothèse alternative* H_1). En fait, H_0 et H_1 nous indiquent respectivement si la suite, et partant le générateur, est aléatoire ou non.

Par conséquent, pour chaque essai, nous prenons une décision ou nous tirons une conclusion qui nous conduit à accepter ou à rejeter l'hypothèse nulle, à nous prononcer sur la question de savoir si le générateur présente ou non des faiblesses liées au caractère aléatoire de la suite produite.

Un paramètre important du test, dont il faut tenir compte, est « *l'erreur de première espèce* $\alpha \in [0.001, 0.05]$ » qui désigne la probabilité de se tromper en rejetant l'hypothèse H_0 à tort, c'est-à-dire la probabilité de déclarer que le générateur n'est pas aléatoire sachant qu'il est aléatoire :

$$\alpha = \Pr[H_0 = \text{rejetée} / H_0 \text{ est vraie}] \qquad (11)$$

3.4.3.2 Aperçu sur quelques tests existants [18 ; 41-45 ; 50-51] :

Il importe d'abord de rappeler que les postulats de S.W. Golomb [42] énoncent des critères d'uniformité des suites pseudo-aléatoires, mais ils ne sont pas utilisés, par manque de flexibilité.

Comme nous l'avons vu au chapitre 3, Paragraphe 3.4.2, le concept de générateurs pseudo-aléatoires formalisé par Blum, Micali et Yao impose également des conditions fortes et contraignantes à appliquer. En effet, dire qu'une suite binaire est indistinguable en temps polynomial, c'est dire qu'aucun algorithme probabiliste ne peut permettre de la distinguer d'une suite binaire aléatoire vraie, et qu'il faut à ce propos lui appliquer tous les algorithmes probabilistes.

Selon [18] [43], il existe beaucoup de tests statistiques pour déterminer si une suite est aléatoire ou non, mais ***aucune batterie de tests spécifiques n'est jugé complète***. En outre, les résultats des tests statistiques doivent être interprétés avec une certaine prudence et de la précaution pour éviter des conclusions erronées sur un générateur spécifique.

Auparavant, nous rappelons ci-après quelques-uns des quinze (15) tests statistiques classiques recommandés par NIST (National Institut of Standards and Technology) et dont la description détaillée figure dans [43], un document de 163 pages (NIST SP 800-22 de 2010) :

- *Test de fréquence (Monobit)* : l'écart entre le nombre de bits '0' et le nombre de bits '1' ne doit pas être significatif.

- *Test de fréquence par bloc* : les nombres d'occurrences des *2k* blocs de longueur (*k*) doivent être proches.

- *Test de rang de la matrice binaire* : construction, à partir des suites binaires, de matrices 32x32 dont il faut vérifier si les rangs suivent la loi du KHI-2.

- *Test spectral* (basé sur le calcul de la transformée de Fourier discrète).

- *Test de la complexité linéaire* (appliqué surtout aux LFSR).

- *Test statistique universel de Maurer*, qu'Ueli Maurer a proposé dans [24] : « *L'idée de base du test est qu'il ne doit pas être possible de compresser nettement (sans perte d'informations) la séquence de sortie d'un générateur de bits aléatoires. Autrement dit, si un échantillon de la séquence de sortie du générateur peut être significativement compressé, alors ce générateur doit être rejeté parce qu'il présente des défauts. Au lieu de compression de séquence, le test statistique universel Maurer peut être utilisé pour calculer une quantité qui est liée à la longueur de la séquence compressée. L'universalité du test statistique Maurer est que celui-ci est capable de détecter de possibles défauts du générateur de bits* ».

La plupart des tests statistiques exigent 20.000 bits ou 2500 octets, et les résultats obtenus sont interprétés en fonction des intervalles auxquels ils appartiennent.

D'autres bibliothèques de tests d'aléas sont développées par des laboratoires universitaires, comme les tests DIEHARD, d'emploi plus complexe, recommandés par Dr Georges Marsaglia de l'Université de Floride. [44] [45]

RÉFÉRENCES

[1] Jonathan Katz, Yehuda Lindell, Introduction to Modern Cryptography, Chapman & Hall/CRC Taylor & Francis Group, 2007.

[2] Oded Goldreich, Modern cryptography, Probabilistic proof and pseudo randomness, springer 1999.

[3] Oded Goldreich. A Note on Computational Indistinguishability. Information Processing Letters, Vol. 34, pages 277-281, May 1 990

[4] Oded Goldreich. Foundations of Cryptography: Vol 1. Cambridge University Press, 2001.

[5] Oded Goldreich. Foundations of Cryptography: Vol 2. Cambridge University Press 2004.

[6] Oded Goldreich, On the foundation of modern cryptography, LNCS 1294, Advances in cryptology, CRYPTO 97, Santa Barbara California, USA, August 17-21, 1997, Springer Verlag pp 56-84.

[7] T. M. Cover and G.A. Thomas. Elements of Information Theory. John Wiley& Sons, Inc., New- York, 1991.

[8] C. E. Shannon. A mathematical theory of communication. Bell Sys. Tech. Journal, Vol. 27, pages 623-656, 1948.

[9] L. A. Levin. Randomness Conservation Inequalities: Information and Independence in Mathematical Theories. Inform. and Control, Vol. 6 1 , pages 1 5-37, 1984

[10] M. Li and P. Vitanyi. An n Introduction to Kolmogorov Complexity and its Applications. Springer Verlag, August 1993.

[11] R.J. Solomonoff. A Formal Theory of Inductive Inference. Inform. and Control, Vol. 7/ 1, pages 1-22, 1964.

[12] A. Kolmogorov. Three Approaches to the Concept of "The Amount of Information" Probl. of Inform. Transm. , Vol. 1 / 1, 1965.

[13] G. J. Chaitin . On the Length of Programs for Computing Finite Binary Sequences. Journal of the A CM, Vol. 13, pages 547-570, 1966.

[14] S. Goldwasser and S. Micali. Probabilistic Encryption. Journal of Computer and System Science, Vol. 28, No. 2, pages 27G--299, 1984. Preliminary version in 14th A CM Symposium on the Theory of Computing, 1982.

[15] M. Blum and S . Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits . SIAM Journal on Computing, Vol. 13, pages 850-864, 1984. Preliminary version in 23rd IEEE Symposium on Foundations of Computer Science, 1982.

[16] A. C. Yao. Theory and Application of Trapdoor Functions. In 23rd IEEE Symposium on Foundations of Computer Science, pages 80-9 1 , 1982.

[17] Oded Goldreich and L. Levin, A hard-core predicate for all one-way functions, In Proc. 21st Annual ACM Symposium on Theory of Computing, pages 25-32. ACM, 1989.

[18] Rolf OPPLIGER, Contemporary Cryptography, Artech House Computer security sciences (pp: 25-219-329)

[19] Eastlake, D., S. Crocker, and J. Schiller, "Randomness Recommendations for Security," Request for Comments (RFC) 1750, December 1994.

[20] Fairfield, R.C., R.L.Mortenson, and K.B. Koulhart, "An LSI Random Number Generator (RNG)," Proceedings of CRYPTO '84, 1984, pp. 203–230 .

[21] Agnew, G.B., "Random Sources for Cryptographic Systems," Proceedings of EUROCRYPT '87, Springer-Verlag, LNCS 304, 1988, pp. 77–81.

[22] Menezes, A., P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography. CRC Press, Boca Raton, FL, 1996.

[23] Lacy, J.B., D.P.Mitchell, and W.M. Schell, "CryptoLib: Cryptography in Software," Proceedings of the USENIX Security Symposium IV, USENIX Association, October 1993, pp. 1–17.

[24] Maurer, U.M., "A Universal Statistical Test for Random Bit Generators," Journal of Cryptology, Vol. 5, 1992, pp. 89–105.

[25] Pierre Barthélémy, Robert Rolland, Pascal Veron, Cryptography : Principes et mise en œuvre, Ed. Lavoisier, 2005-2012.

[26] Hieu Phan, Philippe Guillot, Preuves de sécurité des schémas cryptographiques, Paris 8, oct.2013.

[27] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions Journal of the ACM, Vol. 33, No. 4, pages 792-807, 1986.

[28] O. Goldreich, S . Goldwasser, and S . Micali. On the Cryptographic Applications of Random Functions. In Crypto84, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 276-288, 1985.

[29] O. Goldreich. Two Remarks Concerning the GMR Signature Scheme. In Crypto86, Springer-Verlag Lecture Notes in Computer Science (Vol. 263) , pages:104-1 10, 1 987

[30] M. Luby, C.Rackoff, How to construct pseudorandom permutations from pseudorandom functions, SIAM Journal on computing, Vol. 17, n°2, pp. 373-386, April 1988.

[31] Oded Goldreich, Shafi Goldwasser, Silvio Micali. How to Construct Random Functions. FOCS 1984.

[32] Abhishek Banerjee, Chris Peikert, Alon Rosen. Pseudorandom Functions and Lattices. Eurocrypt 2012.

[33] I. Impagliazzo, L. Levin, and M. Luby. Pseudo-Random Generator from One-Way Functions. In Proc. of the 21st STOC, pages 12–24. ACM Press, New York, 1989.

[34] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A Pseudorandom Generator from any One-Way Function. SIAM Journal of Computing, 28(4):1364–1396, 1999.

[35] R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In 21st Annual ACM Symposium on Theory of Computing, pages 44–61, Seattle, Washington, USA, May 15–17, 1989. ACM Press.

[36] Babacar A. Ndaw, Djiby Sow, Mamadou Sanghare, Construction of Maximum Period Linear Feedback Shift Registers (LFSR) (Primitive Polynomials and Linear Recurring Relations), British Journal of Mathematics and Computer Science, 11(4): 1-24, 2015, Article n°BJMCS.19442, SCIENCEDOMAIN International.

[37] Gutmann, P., Software Generation of practically strong random Numbers, proceedings of the Seventh USENIX Security Symposium, June 1998, pp. 243-255.

[38] Kelsey, J., B, Schneier, and N. Ferguson, Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number generator, proceeding of the 6th Annual workshop on selected Areas in Cryptography, Springer Verlag, August 1999.

[39] Blum, L., M, Blum and M. Shub, A Simple Unpredictable Pseudo- Random Number Generator, SIAM Journal of Computing, Vol.15, May 1986, pp.364-383.

[40] W. Alexi, B.-Z. Chor, O. Goldreich, C.P. Schnorr, RSA and Rabin Functions: Certains Parts are as Hard as The Whole, SIAM Journal of Computing, Vol17, N°2, April 1988, pp 194-209.

[41] P. Liardet, A. Bonnecaze, Randomness and cryptography with a dynamical point of view

juin 2013 (pierre.liardet/alexis.bonnecaze @univ-amu.fr).

[42] S.W. Golomb et L.R. Welch, Shift register sequences, Aegean Park Press Laguna Hills, CA, 1967, 1982.

[43] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo: A Statistical Test Suite for random and pseudo-random number generators for cryptographic applications, NIST April 2010.

[44] Benjamin Pousse, Design et cryptanalyse de chiffrement à flot, Thèse Université de Limoges, Décembre 2010,

[45] G., Marsaglia: DIEHARD: A battery of tests of randomness, 1996
http://stat.fsu.edu/~geo/diehard.html

[46] American national Standards Institut, American National standard X9.17: Financial Institution Key Management, Washington. Dc, 1985.

[47] J. Kelsey, N. Fergusson and B Schneier : Yarrow-160, Notes on The Design and Analysis of the Yarrow Cryptographic Pseudo random Generator, 1999.
http://www.schneier.com/paper_yarrow.html.

[48] N. Ferguson, B. Schneier, T. Kohno : Cryptography Engineering Design Principles and Practical Application, Wiley Publishing, Inc, 2010.

[49] U.S National Institute of Standard and Technology (NIST), Digital Signature Standard (DSS), FIPS PUB 186, May 1994.

[50] R. Santoro, vers des générateurs de nombres aléatoires uniformes et gaussiens à haut débit, Thèse 11 Janvier 2010, Université de rennes 1.

[51] P. Lacharme, Générateurs vraiment aléatoires dans un composant sécurisé, Thèse, Université de Toulon et du Var.

[52] Ballet Stephane and Robert Rolland, A note on a Yao's theorem about pseudo random generators. Cryptography and Communications, Vol.3 N.4 (2011), Page 189-206 doi: 10.1007/s12095-011-0047-1 Springer, 2011.

Deuxième partie
Présentation des travaux

Chapitre 4

Le problème de la réciprocité dans la substitution bigrammatique de Delastelle [1-4]

Sommaire

Compte tenu de son niveau scientifique, ce travail de recherche en cryptologie avait fait l'objet d'une publication dans le journal ''CRYPTOLOGIA'', qui l'a édité électroniquement le 04 juin 2010 [4]. Ce journal est indexé ISI, Scopus, Zentralblatt Math ainsi que beaucoup d'autres bases de données,.

Ce travail présente un intérêt didactique et épistémologique ; il peut être enseigné dans le cadre du cours de cryptographie classique, au même titre que le procédé de Playfair et le chiffrement matriciel de Lester Hill (chiffrement polygraphique). Il s'agit également d'un procédé de chiffrement réciproque d'un très bon niveau de sécurité, qui, dans la pratique, peut être utilisé, en chiffrement classique, pour protéger de courts messages (si la clé associée, et qui est fournie par un générateur d'aléas, est utilisée une et une seule fois).

4.1 Définition

Une substitution simple bigrammatique est une substitution dans laquelle les lettres du texte clair sont remplacées, couple par couple, par les lettres du texte chiffré. Par conséquent, les unités claires et les unités chiffrées sont des bigrammes, d'où le nom d'une telle substitution.

Il y'a beaucoup de procédés de substitutions simples bigrammatiques, parmi lesquels la substitution de Delastelle qui est souvent utilisée en raison de sa simplicité [1-3]. Pour utiliser cette substitution, nous avons besoin d'un alphabet de 25 lettres et de quatre matrices de chiffrement 5x5 (A, B, C, D), dont deux pour les lettres du texte clair et deux pour les lettres du texte chiffré (Fig. 1).

Pour traduire un bigramme clair (par exemple : ta), on repère (t) aux positions *(i, j) de* A et (a) aux positions (*k, h)* de B respectivement. Les lettres du texte chiffré F et Z sont situées à l'intersection de la ligne *(i)* et de la colonne *(h) de* C et à l'intersection de la ligne *(k)* et de la colonne (j) de D respectivement. (Voir Fig. 1.).
Exemple : Texte clair :

cr	yp	to	lo	gy
RU	MA	VT	EB	IV

Nous remarquons que si le chiffrement par la recherche des lettres claires dans les matrices A et B est facile, les lettres du texte chiffré sont mélangées dans les matrices C et D. Ainsi, des matrices permettant les substitutions réciproques sont utilisées par commodité opérationnelle. La réciprocité nous permet de chiffrer et de déchiffrer de la même façon. Aussi, nous allons déterminer les conditions nécessaires et suffisantes pour rendre réciproques les matrices de Delastelle.

4.2 Théorie mathématique de la substitution bigrammatique : Théorème de la réciprocité

4.2.1 Chiffrement et déchiffrement

Soit A, B, C, et D quatre matrices bigrammatiques (*n x n*), (soit quatre matrices 5 x 5 pour Delastelle)

$$A = (a_{ij}) \quad ; \quad B = (b_{kh}) \quad ; \quad C = (c_{ih}) \quad ; \quad D = (d_{kj}) \qquad (1)$$

Où i, j, k, h ∈ {*1, 2, 3, ...n*}.

A

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	x	y	z

C

J	D	R	B	S
Y	K	F	I	N
C	O	A	Q	E
Z	H	U	L	V
M	X	G	P	T

D

Z	G	S	Y	F
E	L	O	C	K
X	B	I	R	T
Q	N	U	A	M
J	V	D	P	H

B

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	u	x	y	z

Figure 4.1 : Matrices de Delastelle

La fonction de chiffrement (s) est définie comme suit :

$$s: A \times B \longrightarrow C \times D$$

$$(a_{ij}, b_{kh}) \longrightarrow (c_{ih}, d_{kj}) \qquad (2)$$

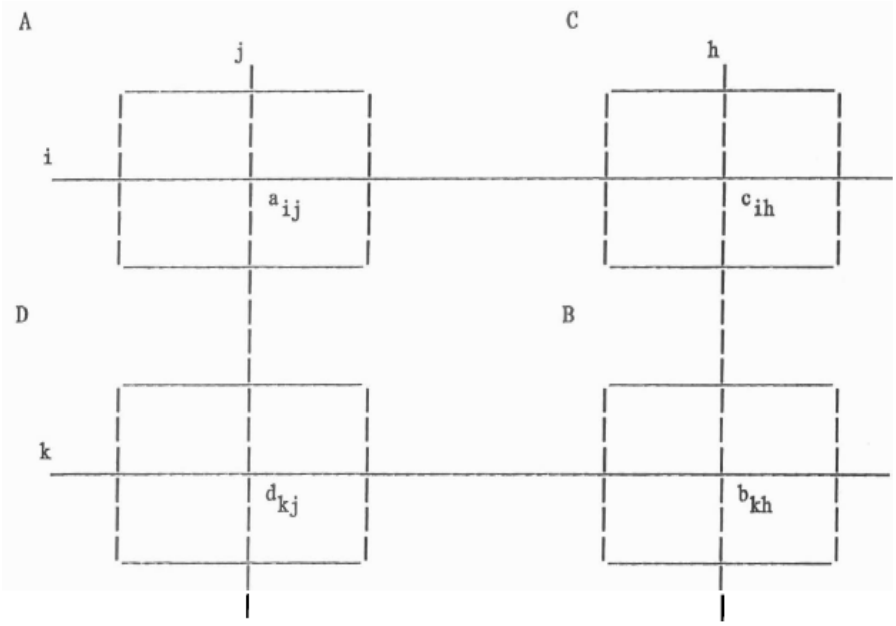

Figure 4.2 : Schéma de chiffrement avec les matrices de Delastelle

Le problème est de trouver une fonction *(s)* qui puisse être utilisé à la fois pour le chiffrement et le déchiffrement :

$$\text{Chiffrement} \quad s(a_{ij},b_{kh}) = (c_{ih},d_{kj})$$
$$\text{Déchiffrement} \quad s(c_{ih},d_{kj}) = (a_{ij},d_{kh}) \qquad (3)$$

Il vient de (2) et (3) :

$$s(c_{ih},d_{kj}) = s[s(a_{ij},d_{kh})] = s^2(a_{ij},b_{kh}) = (a_{ij},b_{kh})$$

Alors $s=s^{-1}$ (4)

(s) est une involution et une fonction réciproque et, à ce titre, elle définit les deux opérations de chiffrement et déchiffrement.

$$\text{Chiffrement} \quad s(a_{ij},b_{kh}) = (c_{ih},d_{kj})$$
$$\text{Déchiffrement} \quad s^{-1}(c_{ih},d_{kj}) = s(c_{ih},d_{kj}) = (a_{ij},d_{kh})$$

4.2.2 Réciprocité

Etant données les matrices A et B, le problème consiste à trouver les matrices C et D de sorte que la fonction *(s)* soit réciproque. Pour ce faire, il faut déterminer deux transformations *(g) et (h)* qui permutent les lignes et les colonnes de A et B de manière à obtenir C et D, respectivement.

$$g(a_{ij}) = c_{ih} \quad \text{and} \quad h(b_{kh}) = d_{kj} \qquad (5)$$

Pour définir ces transformations, désignons (L) et (L') les permutations des lignes i = 1, 2, 3, 4, 5 de A et B, respectivement de telle sorte que :

$$L(i) = L_i \quad \text{et} \quad L'(i) = L'_i$$
$$i \in \{1,2,3,\ldots,n\} \quad \text{et} \quad L_i, L'_i \in \{1,2,3,\ldots,n\} \qquad (6)$$

Exemple : Pour les matrices 5x5 avec les lignes i=1, 2, 3, 4, 5

$$L = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ L_1 & L_2 & L_3 & L_4 & L_5 \end{pmatrix} \quad L' = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ L'_1 & L'_2 & L'_3 & L'_4 & L'_5 \end{pmatrix}$$

Soient (M) et (M') les permutations des colonnes de A et B pour obtenir les colonnes de C et D respectivement. Elles sont définies de la même manière que (L) et (L') :

$$M(j) = M_j \quad \text{et} \quad M'(j) = M'_j$$
$$j \in \{1,2,3,\ldots,n\} \quad \text{et} \quad M_j, M'_j \in \{1,2,3,\ldots,n\} \qquad (7)$$

Pour une explication plus claire, nous pouvons définir des matrices intermédiaires (A') et (B'), entre A, B et C, D (figure 3) où :

- (A') est obtenue par permutation des lignes de (A), et une permutation des colonnes de (A') donne (C).
- (B') est obtenue par permutation des lignes de (B), et une permutation des colonnes de (B') donne (D).

$$A \xrightarrow{L} A' \xrightarrow{M} C$$
$$B \xrightarrow{L'} B' \xrightarrow{M'} D$$

Procédons comme suit : Soient (L) et (L') les permutations effectuées sur les lignes des matrices A et B pour obtenir A 'et B', respectivement, et soient (M) et (M') les permutations effectuées sur les colonnes des matrices (A') et (B') pour obtenir C et (D) respectivement (figure 3)

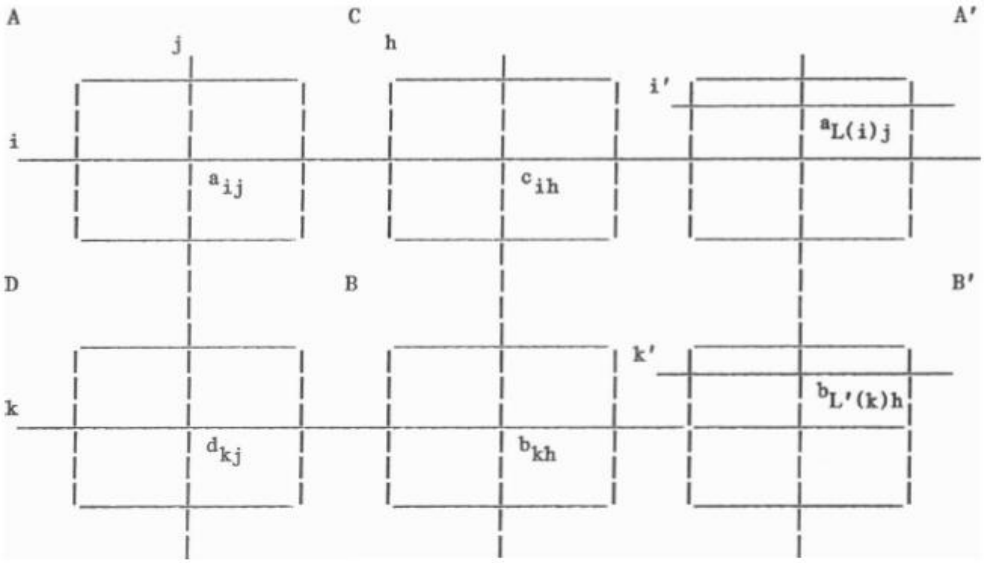

Figure 4.3 : Chiffrement avec les matrices intermédiaires

L'élément c_{ih} de la matrice C était à la position $a_{L(i)j}$ dans la matrice A' et l'élément d_{kj} était à la position $b_{L'(k)h}$ dans la matrice B'

1. Permutation ligne : $L: A \longrightarrow A'$

$$a_{ij} \longrightarrow L(a_{ij}) = a_{L(i),j} \qquad (8)$$

2. Permuatation ligne : $L': B \longrightarrow B'$

$$b_{kh} \longrightarrow L'(b_{kh}) = b_{L(k),h} \qquad (9)$$

3. Permuatation colonnne : $M: A' \longrightarrow C$

$$a_{L(i),j} \longrightarrow c_{ih} = M[a_{L(i),j}] = a_{L(i),M(j)} \qquad (10)$$

4. Permutation colonne : $M': B' \longrightarrow D$

$$b_{L'(k),h} \longrightarrow d_{kj} = M'[b_{L'(k),h}] = b_{L'(k),M'(h)} \qquad (11)$$

Ainsi, les relations entre les matrices A, B, C, et D sont les suivantes :

$$c_{ih} = a_{L(i),M(j)} \quad \text{et} \quad d_{kj} = b_{L'(k),M'(h)} \qquad (12)$$

En résumé :

1. Equation de chiffrement :

$$s(a_{ij}, b_{kh}) = (c_{ih}, d_{kj})$$

$$s^{-1}(c_{ih}, d_{kj}) = (a_{ij}, b_{kh}) = s(c_{ih}, d_{kj}) \qquad (13)$$

2. Formules pour aller d'une matrice à une autre :

$$c_{ih} = a_{L(i),M(j)} \quad \text{et} \quad d_{kj} = b_{L'(k),M'(h)} \qquad (14)$$

Alors nous pouvons écrire :

$$s(a_{ij}, b_{kh}) = (c_{ih}, d_{kj}) = [a_{L(i),M(j)}, b_{L'(k),M'(h)}] \qquad (15)$$

Procédons au déchiffrement maintenant :

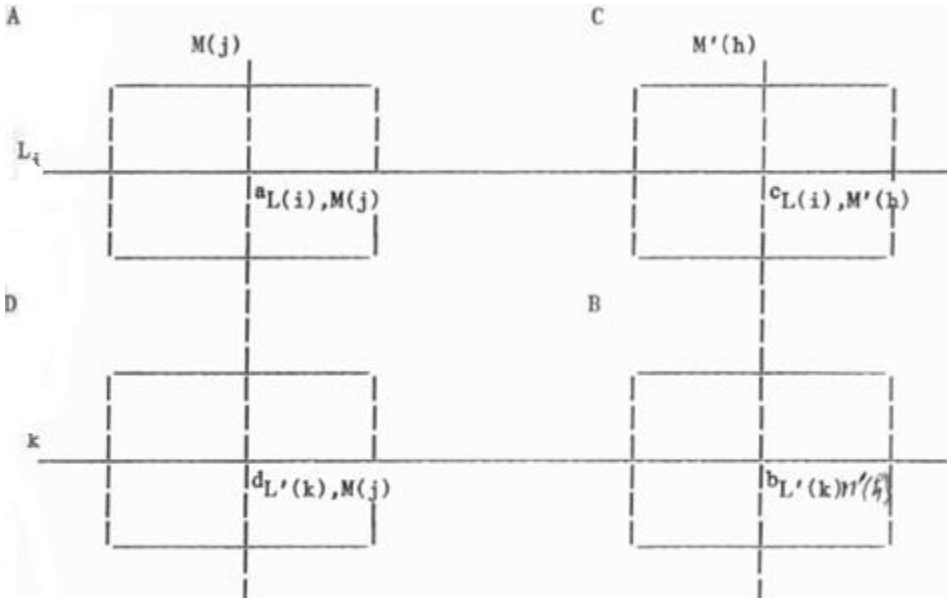

Figure 4.4 : Schéma de déchiffrement avec les matrices

Le chiffrement est :

$$s[a_{L(i),M(j)}, b_{L'(k),M'(h)}] = [c_{L'(i),M'(h)}, d_{L'(k),M(j)}] \quad (16$$

Il vient avec la relation (15) :

$$a_{ij} = c_{L(i),M'(h)} \quad \text{and} \quad b_{L'(i),M(j)} \quad (17)$$

En utilisant la formule (14), on a :

1. $a_{ij} = c_{L(i),M'(h)} = A_{L^2(i),MM'(h)}$

Akors $L^2(i) = I$ et $MM(h)' = I$ et il vient

$$L^2 = I \quad \text{et} \quad MM' = I \quad (18)$$

où I est la matrice identité.

2. $b_{kh} = b_{L'^2(i),M(j)'}$ à partir de la formule (14) et nous avons:

$$b_{kh} = b_{L'^2(i),MM'(k)}$$

Alors $L'^2(i) = i$ et $MM'(k) = k$ et il vient :

$$L'^2 = I \quad \text{et} \quad MM' = I \quad (19)$$

Ces résultats prouvent le théorème suivant énonçant les conditions de réciprocité des matrices de Delastelle.

THEOREME DE LA RECIPROCITE : Les conditions nécessaires et suffisantes pour que des matrices de Delastelle soient réciproques sont :

1. $L^2 = I$ avec $L = L^{-1}$
2. $L'^2 = I$ avec $L' = L'^{-1}$
3. $M'M = I$ avec $M' = M^{-1}$

$$A \xrightarrow{L} A' \xrightarrow{M} C \quad \text{et} \quad B \xrightarrow{L'} B' \xrightarrow{M'} D.$$

Où L, M et L', M' sont des permutations à effectuer sur les lignes et les colonnes respectivement de la première matrice (A) et de la deuxième matrice (B) pour obtenir la troisième matrice (C) et la quatrième matrice (D).

Exemple : Déterminons C et D pour que les matrices de Delastelle ci-dessous soient réciproques :

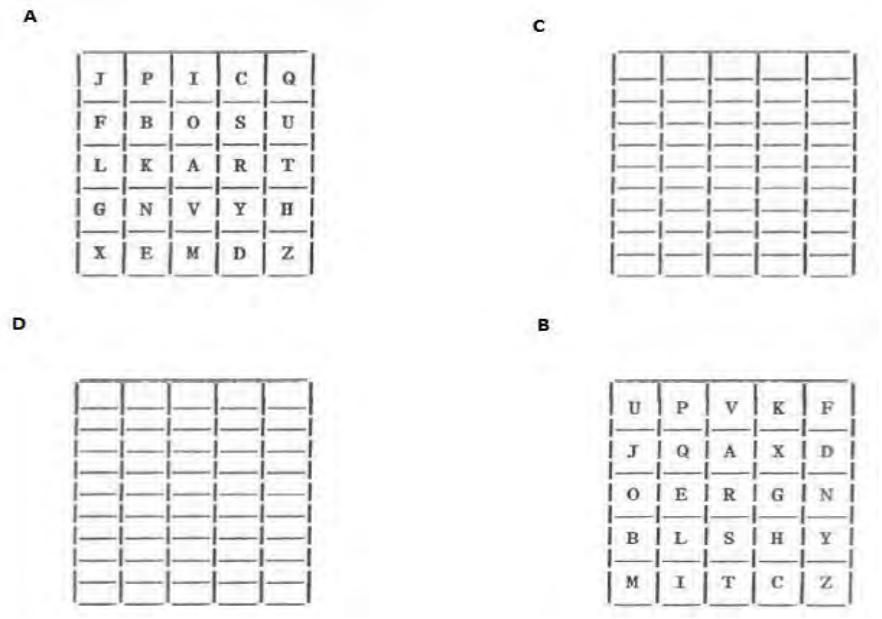

Figure 4.5 : Détermination des matrices C et D

Soient (L) et (L') les permutations effectuées sur les lignes de (A) et (B), respectivement :

$$L = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 4 & 2 \end{pmatrix} = L^{-1}$$

$$L' = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 2 & 5 & 4 \end{pmatrix} = L'^{-1}$$

Permutons les colonnes de (A) et (B) par :

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 2 & 5 & 3 \end{pmatrix} \quad ; \qquad M' = M^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 5 & 1 & 4 \end{pmatrix}$$

Il vient alors :

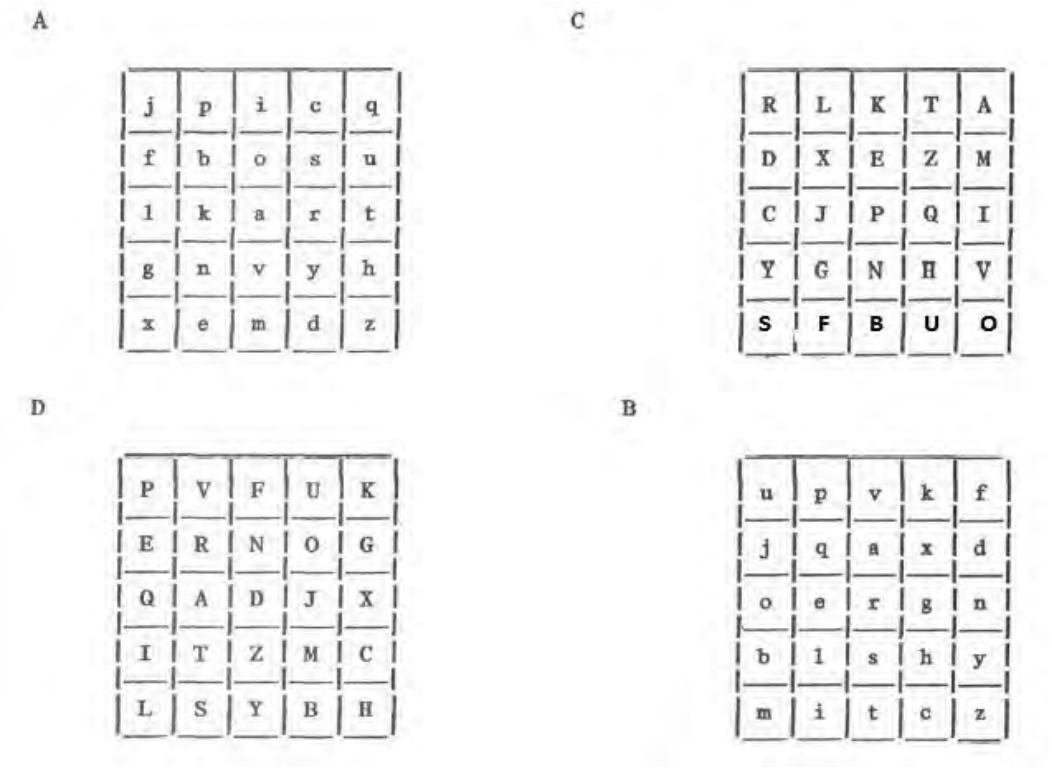

Figure 4.6 : Matrices de chiffrement réciproque

Exemple :	Bigrammes clairs	Bigrammes chiffrés
	(st)	(EB)
	(eb)	(ST)

4.3 Matrices équivalentes- Espace des clés générées

Pour une configuration donnée, on peut déterminer les matrices équivalentes en procédant comme suit :

- La même permutation des lignes de (A) et (C)
- La même permutation (indépendante de la précédente) des lignes de (B) et (D)
- La même permutation (indépendante de la précédente) des colonnes (C) et (D)
- la même permutation (indépendante de la précédente) des colonnes de (A) et (D)

- Il existe $(25\ !)^2$ choix possibles pour les matrices (A) et (B) : elles peuvent être fournies par ***un générateur d'aléas*** qui délivre des lettres prises dans un alphabet de 25 avec des probabilités d'apparition égales à $\frac{1}{25}$.
- Il y'a aussi (26) permutations réciproques possibles (L) et (L')
- Et (5 !) possibilités pour (M) (avec M' = M^{-1}).

Par conséquent, la taille de l'espace des clés de chiffrement pouvant être générées est :

$$\frac{(25!)^2\ 26^2\ 5!}{(5!)^4} = \frac{1}{5!}\left(\frac{26!}{5!}\right)^2 = 9{,}4113 \times 10^{46}$$

REFERENCES

[1] Delastelle, F.M. 1902. Traité élémentaire de cryptographie Paris : Gauthier Villars.

[2] Eyraud, C. 1953. Précis de cryptographie moderne. Paris : Editions Raoul Tari.

[3] Muller, A. 1971. Les Ecritures secrètes. Paris : Presses Universitaires de France

[4] Babacar A. NDAW, Amadou D. SARR, The Problem of Reciprocity in a Delastelle Digraphic Substitution, CRYPTOLOGIA, vol 7, n°2, pages 170-179, April 1983, Jun 2010.

Chapitre 5

Construction des registres à décalage à rétroaction linéaire (LFSR) bouclés à période maximale (Polynômes primitifs et relations de récurrences linéaires)- PRBG basés sur les registres à décalage.

Sommaire

Ce travail de recherche original a fait l'objet d'une publication dans ‘’British Journal of Mathematics and Computer Science” (Article n°BJMCS.19442, SCIENCEDOMAIN *International,*11(4) ; 1-24, 2015).

Le registre à décalage à rétroaction ou réinjection (Feedback Shift Register -FSR) est généralement l'élément de base des générateurs pseudo-aléatoires utilisés pour générer des chaînes cryptographiques ou ensemble de suites de clés de chiffrement. Ce type de générateur est beaucoup utilisé dans les systèmes de chiffrement en continu ou par flot (stream cipher) et les systèmes de communication tels que C.D.M.A (Code Division Multiple Access), les systèmes de communication mobiles, les systèmes de radars et de navigation, les systèmes de communication à large spectre

L'objectif de l'article publié est alors de proposer une méthode de détermination des suites récurrentes linéaires engendrant les registres à décalage à rétroaction linéaire (LFSR) à partir des polynômes primitifs (et vice-versa). Les suites récurrentes linéaires facilitent la construction des LFSR bouclés à période maximale. L'article insiste, dans la dernière partie, sur la sécurité cryptographique des LFSR et indique certains problèmes ouverts dans le domaine des générateurs pseudo-aléatoires basés sur les registres à décalage à rétroaction non linéaires (NLFSR).

5.1 Introduction

Contrairement aux algorithmes de chiffrement par blocs, les algorithmes de chiffrement en continu (Stream Cipher), opèrent sur chaque unité de chiffrement du clair (chiffrement d'un bit/caractère à la fois, chiffrement bit à bit ou caractère par caractère) ; les bits sont chiffrés individuellement. Ils sont généralement plus rapides que ceux par blocs, et ont des circuits moins complexes [1-18].

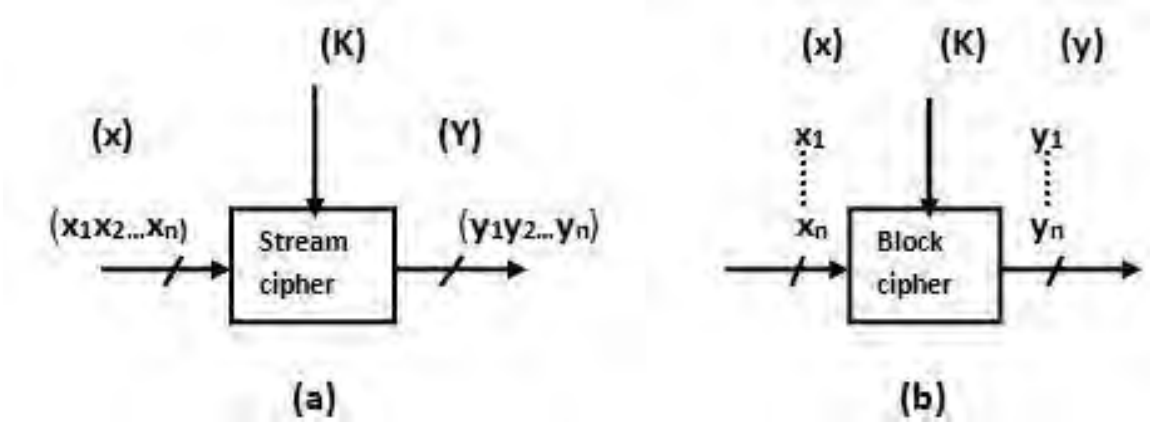

Figure 5.1 : (a) Chiffrement en continu et (b) Chiffrement par blocs

Le chiffrement en continu repose sur un générateur de clefs qui engendre un flux de bits (Key Stream) c.-à-d. une séquence de clefs : $K = (k_1, k_{2,...} k_n)$ qui combiné (par XOR exclusif) aux bits du clair $X = (x_1, x_2,....,x_n)$ fournit le crypto $Y = (y_{1, y2},.....,y_n)$.

- Equation de chiffrement : $y_i = E_{k\,i}(x_i) = x_i \oplus k_i$
- Equation de déchiffrement : $x_i = D_{ki}(y_i) = y_i \oplus k_i$

(E_{ki}=fonction de chiffrement et D_{ki}=fonction de déchiffrement)

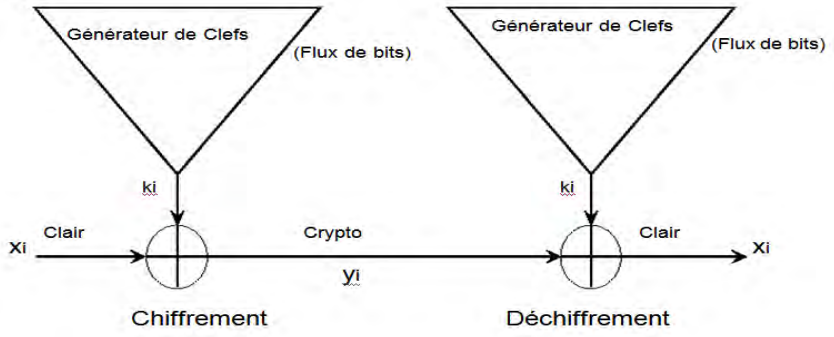

Figure 5.2 : Exemple de schéma de chiffrement en continu

Le chiffrement est réciproque : on chiffre comme on déchiffre. Le générateur de clefs est similaire à une machine à états finis :

- Une erreur dans y_i n'affecte qu'un bit de x_i
- La perte ou l'ajout d'un bit de y_i affecte tous les bits suivants de (X) après chiffrement.
- if $k_i = 0$, $\forall i$, $X = Y$

- Si la suite de clés (k_i) est infinie et complétement aléatoire, on obtient un système de chiffrement à clef-une-fois (One-Time-Pad) de Vernam [2][5][6][9][10][17][19-21] qui est inconditionnellement sûr, comme nous l'avions signalé dans le chapitre 2, contre les attaques à texte chiffré connu (cipher text only attack), le cryptogramme ne fournissant aucune information sur le texte clair.

Généralement, le chiffrement en continu repose sur le même principe que le « One-Time-Pad » avec la seule différence que celui-ci exige une vraie suite aléatoire, qui ne peut être produite à moins de connaître toute la suite.

A défaut, on utilise donc des suites de clefs ou suites chiffrantes pseudo-aléatoires engendrées par un générateur pseudo-aléatoire [12] [22-25] [cf. chapitre 3]. Une bonne suite pseudo aléatoire est celle pour laquelle, connaissant une portion de la suite, il est extrêmement difficile, dans la pratique, de déterminer le reste de la suite [26]. Une méthode classique pour générer une suite pseudo aléatoire [27] [28] est d'utiliser un registre à décalage à rétroaction [cf. paragraphe II].

Les flux de bits (Key Stream) ou suites de clefs produits par le générateur de clefs constituent une chaîne cryptographique.

Le chiffrement en continu est classé en deux (2) catégories :

- **<u>le Chiffrement Synchrone en continu</u> :**

Dans un système de chiffrement synchrone en continu, le flux des bits de la clef est généré indépendamment du flux des bits du message clair et du flux des bits du crypto. L'expéditeur et le destinataire doivent être synchronisés c.-à-d. utiliser la suite chiffrante et être au même état pour que le déchiffrement puisse se faire.

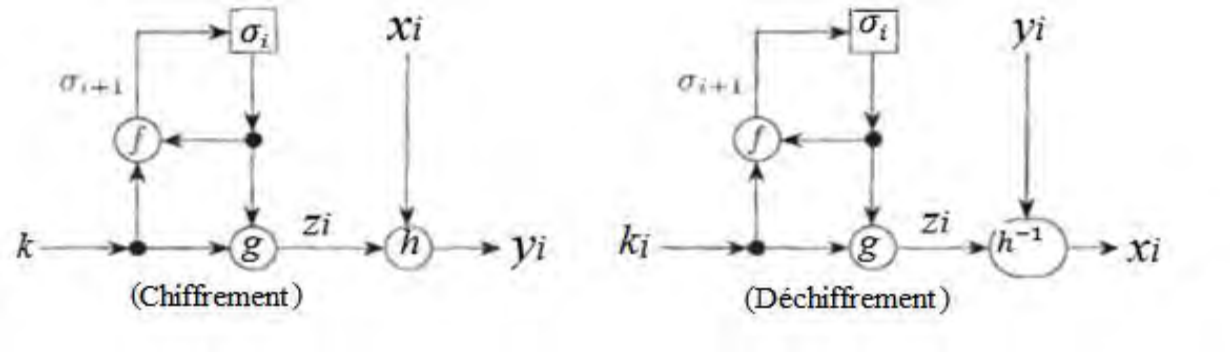

Figure 5.3 : Schéma général de Chiffrement synchrone en continu

<u>Équations de chiffrement</u>

$$\sigma_{i+1} = f(\sigma_i, k)$$
$$z_i = g(\sigma_i, k)$$
$$y_i = h(z_i, x_i)$$

<u>Équations de déchiffrement</u>

$$\sigma_{i+1} = f(\sigma_i, k)$$
$$z_i = g(\sigma_i, k)$$
$$x_i = h^{-1}(z_i, y_i)$$

σ_0 = l'état initial. Il peut être déterminé à partir de la clef k
f = une fonction de passage à l'état suivant
g = la fonction qui engendre le flux de clefs z_i
h = la fonction de sortie qui combine le flux des bits des clefs et le flux des bits clairs pour donner le crypto y

S'il y'a une perte ou un rajout de bits, le déchiffrement échoue. En revanche, la modification d'un bit lors de la transmission ne perturbe pas le déchiffrement des bits suivants.

Exemples [2] [14] : le mode de chiffrement OFB (Output Feedback mode) des systèmes de chiffrement par blocs et le mode CTR (ConTeR Mode) sont des exemples de chiffrement synchrone en continu.

- **le Chiffrement Auto synchrone ou Asynchrone en continu :**

Dans un système de chiffrement auto synchrone en continu (ou Asynchrone) chaque bit du flux de bits engendré par le générateur de clef est une fonction d'un nombre fixe de bits (*t-bits*) du crypto qui précède. Dans ce procédé, les générateurs se synchronisent automatiquement.

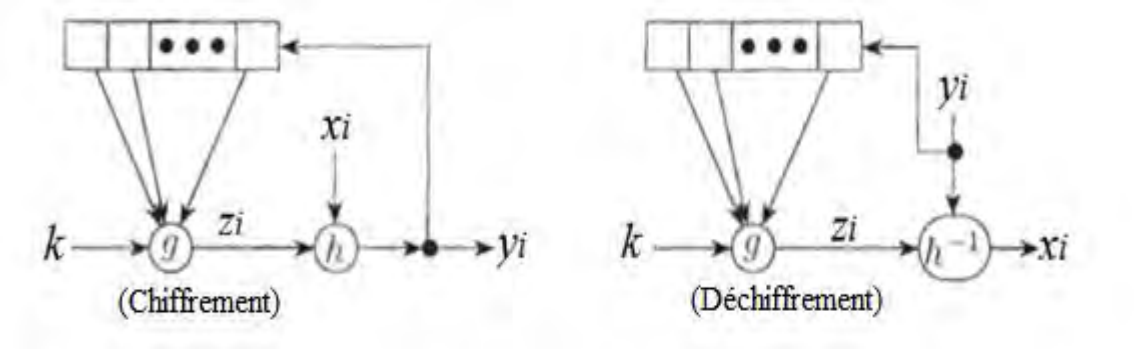

Figure 5.4 : Schéma général de Chiffrement auto synchrone en continu

La fonction de chiffrement est décrite par les équations suivantes :

$\sigma_i = (y_{i-t}, y_{i-t+1}, \dots\dots\dots y_{i-1})$

$z_i = g(\sigma_i, k)$

$y_i = h(z_i, x_i)$

Avec $\sigma_0 = (y_{-t}, y_{-t+1}, \dots\dots\dots y_{-1})$

Pour le déchiffrement, il suffit de remplacer $y_i = h(z_i, x_i)$ par $x_i = h^{-1}(z_i, y_i)$.

Si quelques bits chiffrés sont perdus ou ajoutés dans le crypto, l'auto synchronisation est toujours possible. Cependant, le système est sujet à la propagation d'erreurs ; de même, une modification du crypto par un décrypteur conduit à un déchiffrement incorrect de plusieurs bits.

Exemple [14] : Le mode de chiffrement à rétroaction (CFB) transforme le chiffrement par bloc en chiffrement auto synchrone en continu.

Les deux (2) procédés de chiffrement en continu précités sont décrits en détail notamment dans [1] [2] [6] [10] [12] [14] [19] [29] [30].

5.2 LFSR (Linear Feedback Shift Register- Registres à décalage à rétroaction linéaire)

Un exemple de ce type de générateur est le FSR (Feedback Shift Register : Registre à Décalage à Rétroaction= Registres à Décalage + une fonction de rétroaction ou de réinjection).

5.2.1 Définition 1 : [14] [27] [31-33]

- Une bascule est un dispositif électronique capable d'emmagasiner une information binaire (bit 0 et 1).

- Un Registre à décalage de longueur (n) est constitué de n-bascules interconnectées, de façon que l'état binaire de la bascule de rang (i) soit transmis à la bascule de rang $(i+1)$ quand un signal d'horloge est appliqué à l'ensemble des bascules. L'information binaire de la dernière bascule est toujours accessible physiquement.

Un Registre à décalage est donc constitué :

- d'une entrée qui, en mode décalage, fera progresser le bit d'une bascule à la bascule suivante ;

- de (n) bascules constituant les étages du registre ;

- et d'une sortie.

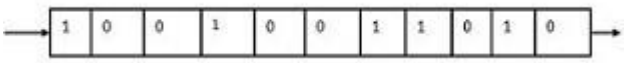

Figure 5.5 : Exemple d'un registre à 11 étages

La plupart des générateurs pseudo-aléatoires sont construits en utilisant des registres à décalage (Exemple : le projet eSTREAM qui s'est déroulé de 2004 à 2008 pour le choix de nouveaux algorithmes de chiffrement en continu standards : Sosemanuk, Grain, Mickey, Trivium) [27] [31] [34-39].
Les Registres à Décalage à Rétroaction Linéaires constituent l'élément de base des générateurs pseudo-aléatoires utilisés pour la génération des clefs de chiffrement. Ce type de générateur est très utilisé dans les systèmes de chiffrement en continu.

Un Feedback Shift Register (FSR) de taille (n) est un automate construit à l'aide d'une fonction booléenne (f) (Réf : définition 2.2) et d'une fonction (F), toutes les deux à (n) variables sur un corps fini $GF(p)$ telle que :

$F : \{0,1\}^n \rightarrow \{0,1\}^n$ (souvent $p = 2$ pour le corps binaire où $p = 2^w$ pour quelques extensions du corps binaire).

$$F(x_1, x_2,..., x_n) = (x_2, x_3,..., x_n, f(x_1, x_2,..., x_n)) \quad (1)$$

- F qui est la fonction de l'état suivant, donne le nouvel état du FSR à partir de l'état antérieur ;
- et (f) qui est la fonction de rétroaction calcule le n ième terme de l'état suivant ;
- Si $(x_1, x_2,.....,x_n)$ est l'état Initial, alors l'application de (f) et (F) donne la séquence d'état:

$$F(x_1, x_2,...,x_n) = (x_2,x_3,...,x_n,x_{n+1}); \qquad x_{n+1} = f(x_1,x_2,...,x_n)$$
$$F(x_2, x_3,...,x_{n+1}) = (x_3,x_4,...,x_{n+1},x_{n+2}); \qquad x_{n+2} = f(x_2,x_3,...,x_{n+1})$$
$$F(x_3,x_4,...,x_{n+2}) = (x_4,x_5,...,x_n,x_{n+1},x_{n+2},x_{n+3}); \qquad x_{n+3} = f(x_3,x_4,...,x_{n+2})$$
$$... \qquad ...$$

La séquence de sortie générée par le FSR:

$$(x_i)_{i \in N} = (x_1,x_2,...,x_n,x_{n+1},...) \quad (2)$$

satisfait la relation de récurrence :

$$x_{i+n} = f(x_i, x_{i+1}, x_{i+2},..., x_{i+n-1}) \quad (3)$$

Si la fonction de réinjection (*f*) est linéaire (ref: 2.2 Définition), le FSR est appelé Registre à Décalage à Rétroaction (ou réinjection) Linéaire - Linear Feedback Shift Registers (LFSR).Sinon, il est appelé Registre à Décalage à Rétroaction Non linéaire- Nonlinear Feedback Shift Register (NLFSR) [40].

5.2.2 Définition 2 : [1] [2] [4] [9] [14] [15] [17] [20] [27] [28] [30-34] [41-48]

• Un registre à décalage à rétroaction linéaire de longueur (*n*) bit (LFSR n-bits) est composé de deux parties :

- Un registre à décalage contenant une séquence de *n* bits ($x_1,....,x_n$) placés de la gauche vers la droite qui constitue l'état initial du registre;
- Et, une fonction de réinjection (ou de rétroaction) linéaire $f(x_1,x_2,....,x_n)$

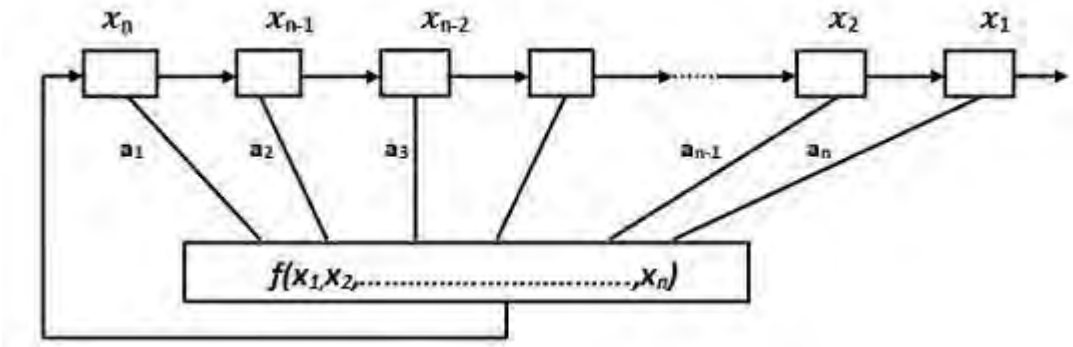

Figure 5.6 : Schéma général d'un Registre à Décalage à Rétroaction Linéaire

- Le registre est appelé par son acronyme anglais : LFSR (Linear Feedback Shift Register).
- A des intervalles périodiques déterminés par l'horloge, le contenu de l'étage *(i)* est transféré dans l'étage *(i+1)* : chaque fois un bit est requis, et tous les bits dans le registre subissent un décalage d'un bit à droite.
- Le nouveau bit le plus à gauche (front du registre) est calculé en fonction des autres bits dans le registre avec la fonction de réinjection $f(x_1, x_2,.....,x_n)$;
- La sortie du registre est 1 bit ; la séquence générée est appelée séquence de dérivation (ou flux de sortie)
- La période du registre est la longueur de la séquence produite avant qu'elle ne se répète. (Ref. Definition 2.1).
- La fonction de réinjection $f(x_1, x_2,..., x_n)$ est telle que :
-

$$f(x_1, x_2,..., x_n) = a_n x_1 + a_{n-1} x_2 + ... + a_1 x_n$$

$$= \sum_{i=1}^{n} a_i x_{n-i+1} \qquad (4)$$

où $a_i = 0$ or $1, \forall i\ 1 \le i \le n$, et l'addition (opération XOR) se fait dans GF(2).

- $f: GF(2^n) \rightarrow GF(2)$
- f est une fonction booléenne (n) variables [2][30][60-64];
- Il y'a 2^{2^n} fonctions booléennes différentes pour (n) variables données.

- La séquence produite par le LFSR satisfait la relation de récurrence linéaire :

$$a_{n+j} = \sum_{i=1}^{n} a_i x_{n+j-1} \Leftrightarrow x_{n+j} = \sum_{i=0}^{n-1} a_{i+1}\ x_{n+j-i-1} \qquad (5)$$

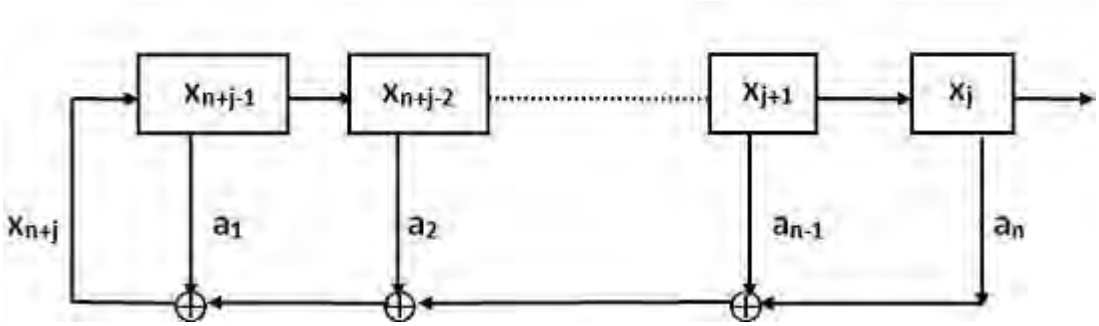

Figure 5.7 : Schéma du LFSR

Et la matrice (A) associée à l'application linéaire est :

$$[GF(2)]^n \rightarrow [GF(2)]^n$$

$$\begin{bmatrix} x_j \\ x_{j+1} \\ . \\ . \\ x_{n+j-2} \\ x_{n+j-1} \end{bmatrix} \rightarrow \begin{bmatrix} x_{j+1} \\ x_{j+2} \\ . \\ . \\ x_{n+j-1} \\ x_{n+j} \end{bmatrix} \qquad (6)$$

$$A = \begin{pmatrix} 0 & 1 & 0 \ldots . . 0 & 0 \\ 0 & 0 & 1 \ldots \ldots 0 & 0 \\ & \ldots & \ldots \ldots \ldots . . & \\ 0 & 0 & 0 \ldots \ldots 0 & 1 \\ a_n & a_{n-1} & \ldots\ldots & a_2\ a_1 \end{pmatrix} \qquad (7)$$

• Une telle configuration, à laquelle nous allons nous intéresser, est appelée **« Configuration de FIBONACCI (Générateur de Fibonacci)»** [2] [14] [28] [30] [45] [49]. Elle est efficace en hardware puisqu'elle ne requiert qu'un seul LFSR de *n*-bits et des opérations XOR bien qu'étant peu efficace en implémentation software (LFSR en mode Fibonnaci ou External-XOR LFSR) contrairement à son autre homologue dite « Configuration de GALOIS » (LFSR en mode Galois ou Internal-XOR LFSR) " traitée dans [2] [14] [28] [30].

- Le générateur de FIBONACCI ou External-XOR LFSR est basé sur les suites de Fibonacci modulo la valeur maximale désirée :

$$x_n = (x_{n-1} + x_{n-2}) \pmod{m} \text{ avec } x_0 \text{ and } x_1 \text{ en entrée} \qquad (8)$$

Alternativement, nous pouvons utiliser cette formule dite formule des “suites de Fibonacci généralisées "

pour générer des suites pseudo aléatoires [49-51] :

$$x_n = \pm (x_{n-s} \pm x_{n-k}) \pmod{m} \text{ avec } x_0 \ldots . x_{k-1} \text{ en entrée} \qquad (9)$$

comme des mots de (w) bits ; généralement $m = 2^w$.

La qualité du générateur dépend des coefficients *(k), (s)* qui doivent être choisis avec soin et des valeurs utilisées dans l'état initial du générateur. Ce générateur est par contre très simple à implémenter et consomme peu de ressources.

Figure 5.8 : LFSR en mode Fibonacci

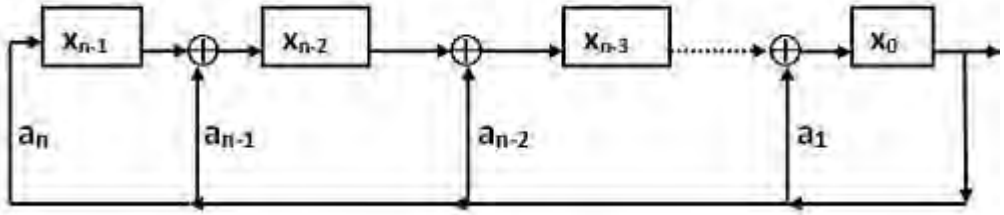

Figure 5.9 : LFSR en mode Galois

- Il faut le différencier du G**énérateur congruentiel linéaire** [9] [14] [17] [30][45] [52][53] qui produit des suites pseudo-aléatoires de la forme :

$$x_n = (ax_{n-1} + b) \pmod{m}, \text{ où } a, b \text{ et } m = \text{entiers} \qquad (10)$$

x_n est le n^{ième} bit de la séquence ; x_0= la semence (germe ou graine) ; la période du générateur ≤ ***m***

Si *a, b, m* sont judicieusement choisis, alors le générateur sera dit « *à période maximale* ***m*** »

(c-à-d si : b ∧ m= 1 ; b et m étant premiers entre eux) ; si b=0, le générateur est dit congruentiel multiplicatif homogène [45][52].

Les générateurs congruentiels linéaires sont rapides et requièrent peu d’opérations par bit, mais il a été démontré qu’ils ne peuvent pas être utilisés en cryptographie. En effet, ils peuvent être prédits et sont donc décryptables. Il en est de même :

- des générateurs quadratiques :

$$x_n = (ax_{n-1}^2 + bx_{n-1} + c) \pmod m \tag{11}$$

- des générateurs cubiques :

$$x_n = (ax_{n-1}^3 + bx_{n-1}^2 + cx_{n-1} + d) \pmod m \tag{12}$$

évoqués dans [14] qui fait remarquer que la combinaison de générateurs congruentiels linéaires offrant de longues périodes n'était pas aussi prouvée cryptographiquement sûre.

• Les LFSR sont largement utilisées dans le chiffrement en continu car ils sont facilement mis en œuvre en hardware aussi bien qu'en software. Si on se réfère aux définitions ci-dessus, il est possible de généraliser les LFSR, dans n'importe quel corps fini *GF(p)*. Du côté software, il est utilisé un corps fini de la forme *GF* (2^n) avec $n = 8, 32, 64...$

5.2.3 Exemples

Exemple 1 : Registre LFSR-n bits bouclé à période maximale

Un LFSR n-bits bouclé à période maximale sur *GF* (2) avec un période maximale $T = 2^n - 1$ est un registre qui peut théoriquement générer une séquence pseudo-aléatoire de longueur $T = 2^n - 1$ bits avant la répétition (et non 2^n à cause de la séquences nulle 0000…00). " La séquence de sortie obtenue est appelée une "m-séquence".

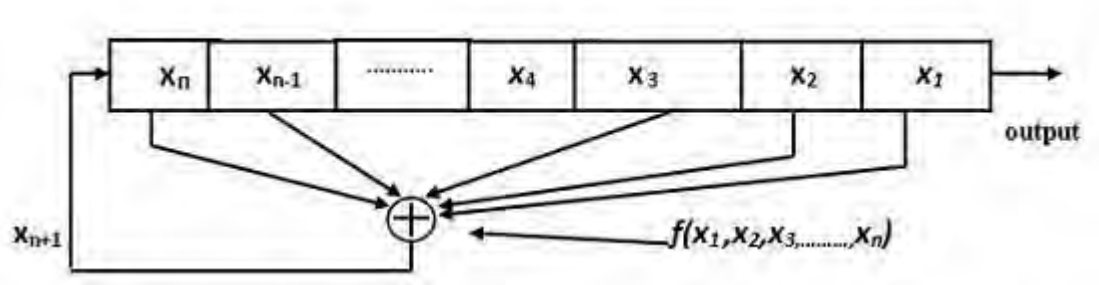

Figure 5.10 : Schéma d'un LFSR n-bits

Exemple 2 : LFSR 3-bits bouclé à période maximale. Il y'a deux boucles possibles :

- Les étages 1 and 3 : $x_{n+1} = x_n + x_{n-2} \pmod 2$ (équation de récurrence)
- Les étages 2 and 3 : $x_{n+1} = x_{n-1} + x_{n-2} \pmod 2$ (équation de récurrence)
- Les étages 1 and 2 sont interdits : Le LFSR boucle après deux étapes comme montré à la figure 10 (pas de période maximale)

La boucle 1 et 3 : Période maximale. La période maximale est : $T = 2^3 - 1 = 8 - 1 = 7$.

L'équation de récurrence est : $x_{n+1} = x_n + x_{n-2} \pmod 2$.

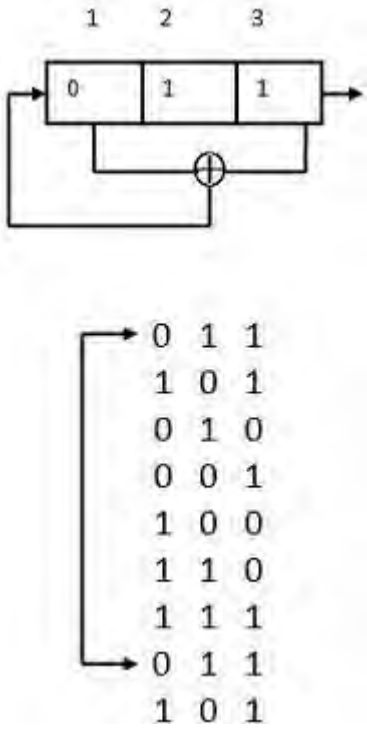

Figure 5.11 : LFSR 3-bits

La Suite clef : 1101001

La boucle 2 -3 : Période maximale

L'équation de récurrence est : $x_{n+1} = x_{n-1} + x_{n-2} \pmod 2$

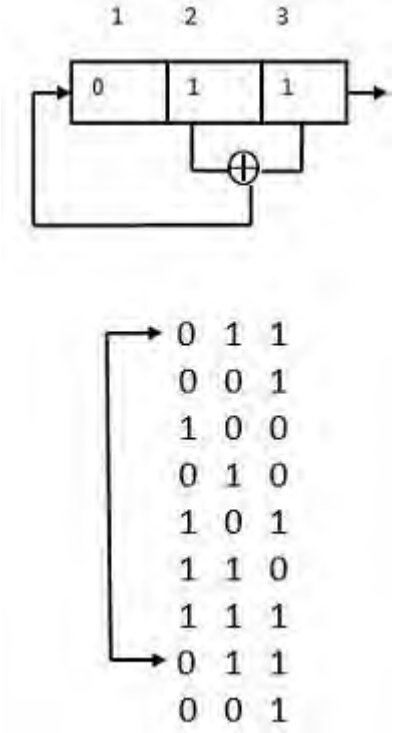

Figure 5.12 : LFSR 3-bits

La suite clef : 1100101
La boucle 1 et2 :

L'équation de récurrence est : $x_{n+1} = x_n + x_{n-1} \pmod 2$

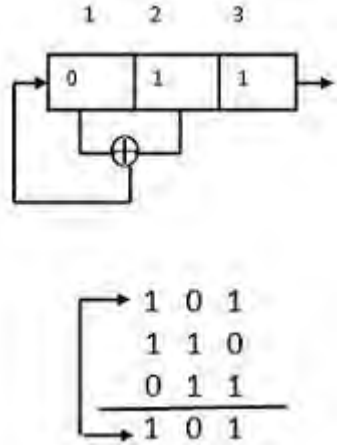

Figure 5.13 : LFSR 3-bits

Exemple 3 : LFSR 4-bits bouclé à période maximale
La période maximale est : $T = 2^4 - 1 = 15$ avec :
La boucle 1 -4 : Période maximale

L'équation de récurrence est : $x_{n+1} = x_n + x_{n-3} \pmod 2$

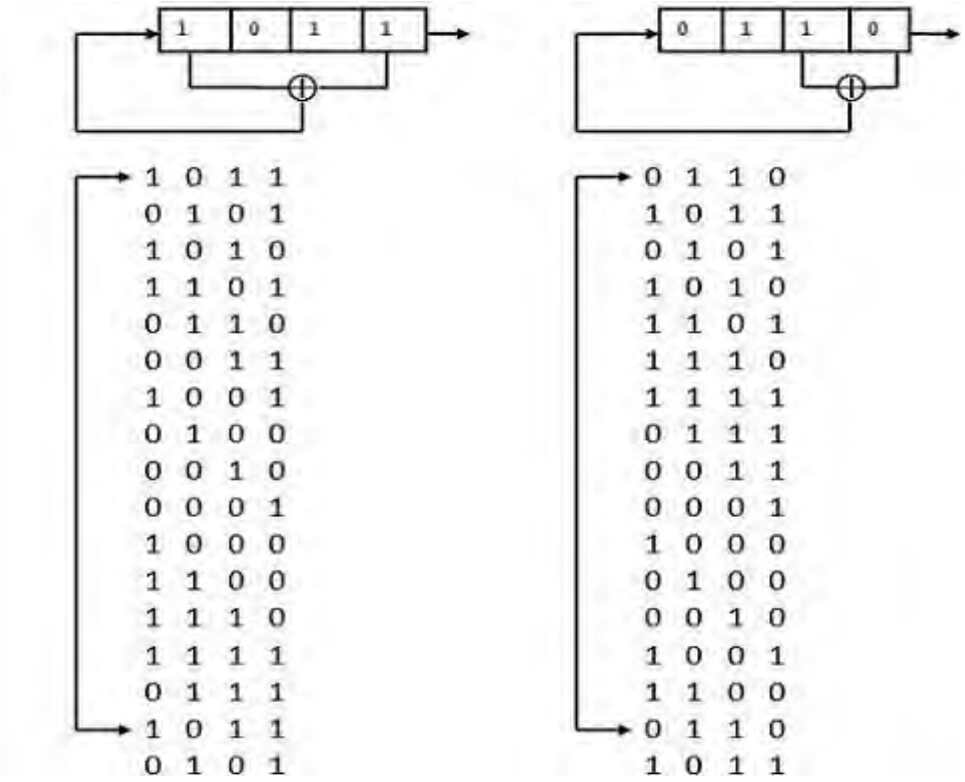

Figure 5.14 : LFSR 4-bits

Les suites clés : 110101100100011 and 011010111100010
La boucle 3 – 4 : Période maximale

L'équation de récurrence est : $x_{n+1} = x_{n-2} + x_{n-3} \pmod 2$

Exemple 4 : LFSR 6-bits bouclé à période maximale ; $T = 2^6 - 1 = 63$

- La boucle 1 et 6 : $x_{n+1} = x_n + x_{n-5}$;
- La boucle 5 et 6 : $x_{n+1} = x_{n-4} + x_{n-5}$;

(Boucle 5 et 6 : Période maximale)

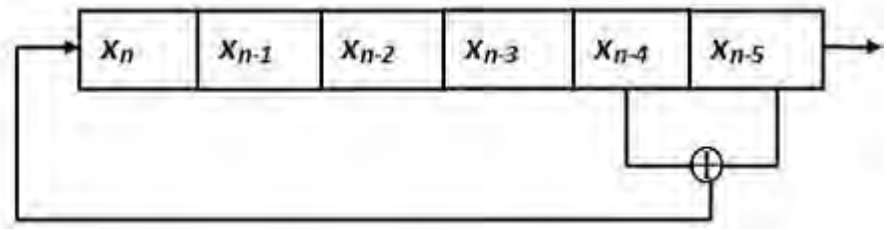

Figure 5.15 : LFSR 6-bits

Exemple 5 : LFSR 31-bits bouclée à période maximale ; $T = 2^{31} - 1 = 2.147.483.647$

- La boucle 3 – 31 : $x_{n+1} = x_{n-2} + x_{n-30} \pmod 2$;
- La boucle 28 – 31 : $x_{n+1} = x_{n-27} + x_{n-30} \pmod 2$;

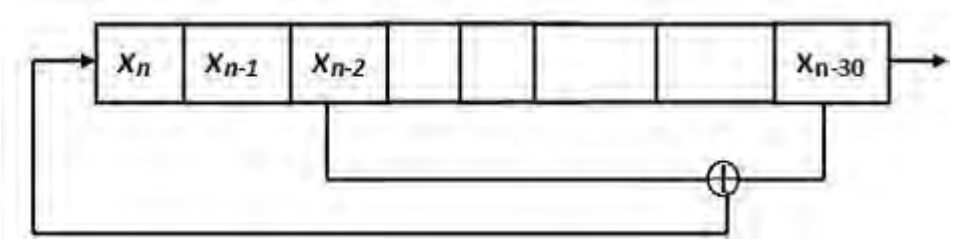

Figure 5.16 : LFSR 31-bits

5.2.4 Théorème : [4] [14] [28] [31-34] [47]

Pour qu'un LFSR-n bits soit de période maximale, il faut que le polynôme formé à partir de la séquence de dérivation soit un polynôme primitif de degré *(n)*, sur *GF(2)* :

- On associe donc à un LFSR-n bits, un polynôme primitif générateur :

$$f(x) = 1 + \sum_{i=1}^{n} a_i x^i = 1 + a_1 x + a_2 x^2 + \ldots\ldots + a_n x^n \quad (13)$$

- Un polynôme primitif de degré (*n*) est un polynôme irréductible de degré *(n)* qui divise $(x^{2^n-1} + 1)$.

Exemple : Le polynôme $f(x) = x^3 + x + 1$ de degré 3 est primitif sur $GF(2)$, il divise $x^7 + 1$,
$(T = 2^n - 1 = 2^3 - 1) \longrightarrow x^7 + 1 = (x^3 + x + 1)(x^4 + x^2 + x + 1)$.

- Si tout polynôme *f(x)* associé à un LFSR est primitif sur *GF (2)* alors tout état initial non nul produit une suite de période maximale $T = 2^n - 1$.

5.2.5 Définition 3

Soit f un polynôme de F_2[X]. Son ordre, noté *ord(f),* est le plus petit entier *(t)* tel que $x^t \equiv 1 \bmod f(x)$.

5.2.6 Définition 4

Soit *f(x)* un polynôme irréductible de degré (*n)* dans F_2[X]. Il est dit primitif s'il est d'ordre $(2^n - 1)$.
Ainsi, si l'on veut construire un LFSR-*n* bits optimal (au regard de la période de la suite produite) de longueur *n*, il faut s'assurer que le polynôme de réinjection choisi est de degré *n* et qu'il est primitif.
On sera alors assuré d'obtenir une période maximale, mais en prenant la précaution de partir d'un état initial non nul.
Un autre intérêt des polynômes de réinjection primitifs est la qualité statistique des suites produites.

5.2.7 Definition 5 [31]

Soit $f(x)$ un polynôme irréductible de degré n sur $([G\,F\,(2^n)])^*$. Il est dit primitif si l'une de ses racines engendre le sous-groupe multiplicatif $[GF\,(2^n)]$, avec $[GF\,(2^n)] = F_2[X]/f(x)$ (les polynômes réduits modulo *f(x)*).
Rappelons d'abord que, le sous-groupe multiplicatif d'un corps fini est cyclique. Autrement dit,
$\forall\, \alpha \in [G\,F\,(2^n)]$, we have $\alpha^{2^n-1} = 1$.

Soit (α) une racine de (f), alors nous avons $f(\alpha) = 0$. Si α génère le groupe multiplicatif, les éléments $\alpha, \alpha^2, \alpha^3 \ldots \alpha^{2^n-1}$ correspondent à tous les éléments non nuls du corps, et il y'a $2^n - 1$ éléments distincts ($\alpha^{2^n-1} = 1$ puisque $\alpha \in ([G\,F\,(2^n)])^*$ qui est cyclique).

5.2.8 Definition 6 [31] [47]

En fait, on peut définir l'ordre d'un élément α comme le plus petit *(t)* tel que $\alpha^t = 1$
Ce que l'on cherche à savoir, c'est si l'ordre de α est égal à $2^n - 1$ ($f(x)$ =polynôme primitif) ou non ($f(x)$ = polynôme non-primitif) ; → il faut se rappeler que pour un LFSR n-bit, $T = (2^n - 1)$ et $f(x)$ divise $(x^{2^n-1} + 1)$.

Exemple : Soit $f(x) = x^3 + x + 1$ irréductible, et soit (α) tel que $f(\alpha) = 0 \Rightarrow \alpha^3 + \alpha + 1 = 0 \Rightarrow \alpha^3 = \alpha + 1$.
Déterminons alors les puissances de α :

$\alpha^1 = \alpha$

$\alpha^2 = \alpha^2$

$\alpha^3 = \alpha + 1$

$\alpha^5 = \alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1$

$\alpha^6 = \alpha^3 + \alpha^2 + \alpha = \alpha^2 + 1$

$\alpha^7 = \alpha^3 + \alpha = 1 \Rightarrow \alpha^{2^3} - 1$ avec $T = 7$.

Nous pouvons conclure que le polynôme $f(x) = x^3 + x + 1$ est primitif.

5.3 Polynômes de réinjection primitifs et Récurrences linéaires pour la construction des LFSR bouclés à période maximale

Grâce au polynôme primitif, on peut identifier l'équation de récurrence linéaire associée au registre LFSR-n bits et vice versa.

Soit le registre suivant et le polynôme primitif défini au paragraphe 5.2.4 (13) :

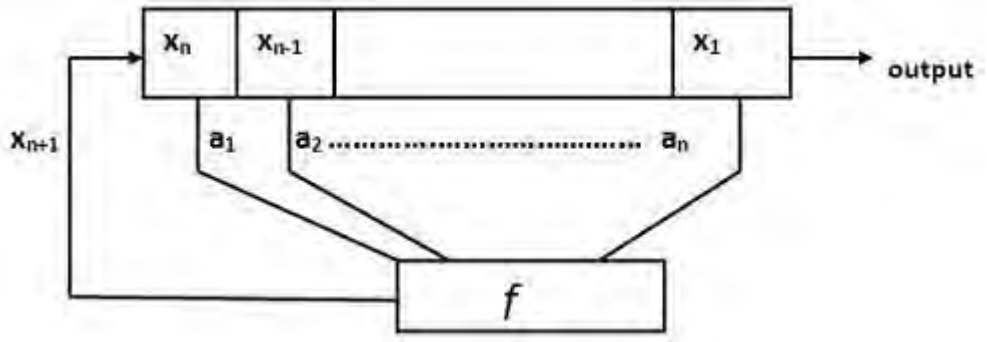

Figure 5.17 : Le LFSR n-bits

Alors, notre équation de récurrence peut être exprimée sous cette forme :

$$\begin{array}{lll} x_{n+1} & = & a_1x_n + a_2x_{n-1} + ... + a_nx_1 = \sum_{k=1}^{n-1} a_kx_{n-k+1} \\ \vdots & & \vdots \\ x_{n+i} & = a_1x_{n+i-1} + a_2x_{n+i-2} + ... + a_nx_i = \sum_{k=1}^{n} a_kx_{n-k+i} \end{array}$$

Nous considérerons dans notre étude, la forme simple à une seule équation telle que :

$$x_{n+1} = f(x_1, x_2, ..., x_n) = \sum_{k=1}^{n} a_kx_{n-k+1} \tag{14}$$

5.3.1 Période maximale

Dans le cas où le polynôme générateur est primitif, pour tout état initial de n bits non nuls, la période T est maximale : $T = 2^n - 1$.

5.3.2 L'équation d'identification

En utilisant notre polynôme primitif et l'équation de récurrence linéaire, nous avons une équation d'identification sous la forme :

$$f(x) = 1 + \sum_{k=1}^{n} a_kx^k = 1 + a_1x + a_2x^2 + ... + a_nx^n$$

$$x_{n+1} = \sum_{k=1}^{n} a_kx_{n-k+1} \tag{15}$$

L'équation d'identification permet de déterminer les coefficients a_k , et par conséquent l'équation de récurrence à partir du polynôme (et vice versa) :

$$x_{n+1} \Leftrightarrow f(x) \tag{16}$$

1. Application à un LFSR 4-bits :

Soit $f(x) = x^4 + x + 1$

La période maximale : $n = 4$ et $T = 2^n - 1 = 2^4 - 1 = 15$.

L'équation d'identification :

$$\begin{aligned} f(x) &= x^4 + x + 1 \\ &= 1 + \sum_{k=1}^{4} a_k x^k \\ &= 1 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 \end{aligned} \tag{17}$$

Nous pouvons identifier alors : $a_3 = a_2 = 0$ et $a_4 = a_1 = 1$.

Mais $x_{n+1} = \sum_{k=1}^{4} a_k x_{n-k+1} = a_1 x_n + a_2 x_{n-1} + a_3 x_{n-2} + a_4 x_{n-3}$, alors l'équation de récurrence est :

$$x_{n+1} = a_1 x_n + a_4 x_{n-3} \pmod 2$$

$$x_{n+1} = x_n + x_{n-3} \pmod 2 \tag{18}$$

Nous obtenons le LFSR 4 bits bouclé à période maximale suivant :

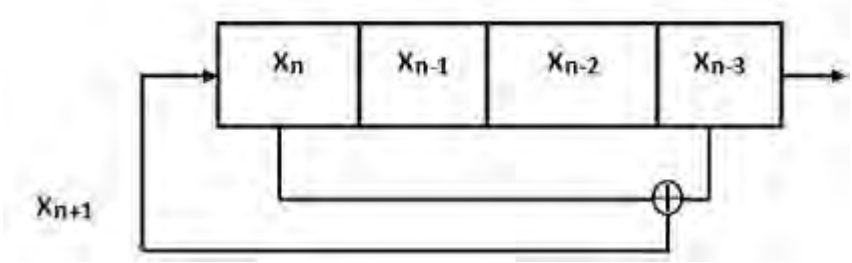

Figure 5.18 : LFSR 4-bits

Remarque : Si nous faisons ce changement de variable $x = \frac{1}{x}$, nous avons un autre registre possible. En effet, si $f(x)$ est un polynôme primitif, $g(x) = x^n f(\frac{1}{x})$ est aussi primitif. Si nous supposons

$f(x) = x^n + x^s + 1) \Rightarrow g(x)) = x^n f(\frac{1}{x}) = x^n(\frac{1}{x^n} + \frac{1}{x^s} + 1) = x^n + x^{n-s} + 1$ qui est aussi primitive (le polynôme réciproque).

Les deux polynômes peuvent être utilisés pour construire des registres pour des applications. (Plus généralement $f(x) = 1 + \sum_{k=1}^{n} a_k x^k$ et $g(x) = x^n + \sum_{k=1}^{n} a_k x^{n-k}$)

Avec le polynôme $f(x) = x^4 + x + 1$, nous avons $g(x) = x^4 + x^3 + 1 = 1 + \sum_{k=1}^{n} a_k x^k$

$$= 1 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4.$$

En utilisant l'équation d'identification, nous avons :

- $a_4 = a_3 = 1$ and $a_2 = a_1 = 0$
- $x_{n+1} = \sum_{k=1}^{4} a_k x_{n-k+1} = a_1 x_n + a_2 x_{n-1} + a_3 x_{n-2} + a_4 x_{n-3}$
- $\Rightarrow x_n = x_{n-2} + x_{n-3}$

Nous obtenons alors le LFSR - bits bouclé à période maximale suivant :

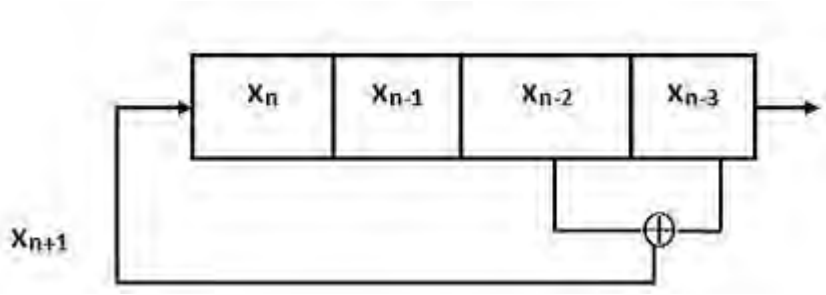

Figure 5.19 : LFSR 4-bits

2. Application à un LFSR-35 bits :

Si $f(x) = x^{35} + x^2 + 1$

Période maximale : $T = 2^n - 1 = 2^{35} - 1$

L'équation d'identification :

$$f(x) = x^{35} + x^2 + 1 = 1 + \sum_{k=1}^{n} a_k x_k$$
$$= 1 + a_1 x + a_2 x^2 + ... + a_{34} x^{34} + a_{35} x^{35} \quad (19)$$

Tous les $a_k = 0$ exceptés $a_2 = a_{35} = 1$

- $x_{n+1} = \sum_{k=1}^{35} a_k x_{n-k+1} = a_1 x_n + a_2 x_{n-1} + ... + a_{35} x_{n-34}$

En identifiant, il vient :

$$x_{n+1} = a_2 x_{n-1} + a_{35} x_{n-34} \quad (20)$$

$$x_{n+1} = x_{n-1} + x_{n-34} \quad (21)$$

Nous avons ce LFSR 35- bits bouclé à période maximale :

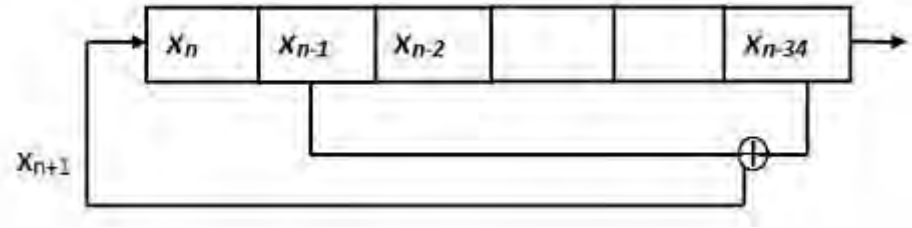

Figure 5.20 : LFSR 35- bits

Avec le changement de variable habituel, il vient alors :

$f(x) = x^{35} + x^2 + 1 \Rightarrow g(x) = x^{35} + x^{33} + 1$ est aussi primitif.

L'équation d'identification est :

$g(x) = x^{35} + x^{33} + 1 = 1 + \sum_{k=1}^{35} a_k x^k = 1 + a_1 x_+ a_2 x^2 + ... + a_{33} x^{33} + a_{34} x^{34} + a_{35} x^{35}$

Tous les a $_k = 0$ excepté $a_{35} = 1$ et $a_{33} = 1$

$$x_{n+1} = \sum_{k=1}^{35} a_k x_{n-k+1}$$

$$= a_1 x_n + a_2 x_{n-1} + ... + a_{33} x_{n-32} + a_{34} x_{n-33} + a_{35} x_{n-34}$$

$$x_{n+1} = a_{33} x_{n-32} + a_{35} x_{n-34}$$

$$x_{n+1} = x_{n-32} + x_{n-34}$$

Le LFSR 35-bits bouclé à période maximale est le suivant :

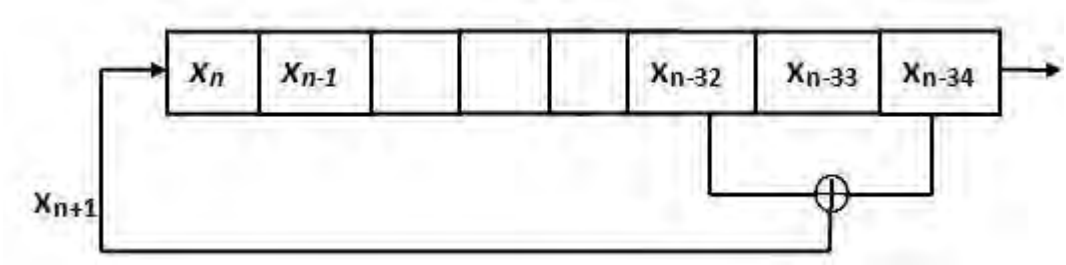

Figure 5.21 : LFSR 35-bits

Enfin, il convient de noter que [54-56] [45] rappellent quelques travaux qui ont été faits sur les polynômes primitifs binaires et donnent des tableaux des polynômes disponibles. Toutefois, pour la réalisation des LFSRs, il est fortement recommandé d'utiliser des polynômes primitifs avec des coefficients non-nuls (polynômes primitifs denses) plutôt que des polynômes dont la plupart des coefficients sont nuls et qui sont cryptographiquement faibles [14].

5.4 Sécurité Cryptographique

5.4.1 Complexité linéaire

Au plan de la sûreté cryptographique, l'utilisation d'un seul registre LFSR n'est pas sûr, par ce que ce LFSR est prédictible :

- Si on connaît (*n)* bits consécutifs produits par un LFSR- n bits et le polynôme primitif qui lui est associé, on peut en déduire le *(n+1)* ^{ème} bit produit par le registre ;

- Et, si on connaît également *(2n)* bits consécutifs produits par un LFSR- *n* bits dont on ne connaît pas le polynôme primitif qui lui est associé, on peut retrouver ce polynôme par l'algorithme de Berlekamp-Massey [57] [28] [30] [44] [47] [48] [58].

Cet algorithme permet de déterminer la complexité linéaire d'une suite aléatoire c.-à-d. la longueur du plus petit LFSR qui permet de l'engendrer (appelé aussi « linear span ») [30] [59]. En 1969, James L. Massey [44] a montré, en effet, que l'algorithme proposé en 1967 par Edwyn Ralph Berlekamp pour le décodage des codes BCH [57] permet également de retrouver le plus petit LFSR engendrant une suite donnée. [39] livre un éventail de résultats sur la complexité linéaire des suites aléatoires.

5.4.2 Recommandations-Perspectives

Les générateurs pseudo-aléatoires à base de FSR utilisés pour générer des clefs doivent avoir les caractéristiques suivantes [12] :

- Une longue période
- Une grande complexité linéaire
- De bonnes propriétés statistiques

Comme nous l'avons indiqué plus haut, l'avantage majeur des LFSR est la facilité de mise en œuvre matérielle et logicielle associée à leur bonne conception mathématique. Cependant, les LFSRs utilisés seuls, ne sont pas sûrs compte rendu de leur linéarité qui est exploitée pour concevoir des méthodes de cryptanalyse, dont le premier et le plus connu est l'algorithme de Berlekamp-Massey.
Pour renforcer la sécurité cryptographique des générateurs LFSR, nous utilisons des générateurs LFSR plus complexes à l'aide des fonctions booléennes non linéaires (fonctions booléennes cryptographiques [2]) [30] [60-64] qui doivent avoir certaines propriétés (haut degré algébrique, grande complexité linéaire, non-linéarité élevée et forte immunité aux corrélations) qui peuvent détruire la linéarité :

- Registres LFSR combinés non-linéaires [12] [65-68]: un générateur pseudo-aléatoire de clefs de chiffrement dans lequel les sorties de plusieurs LFSR sont combinées par une fonction linéaire [12] [61] [63]

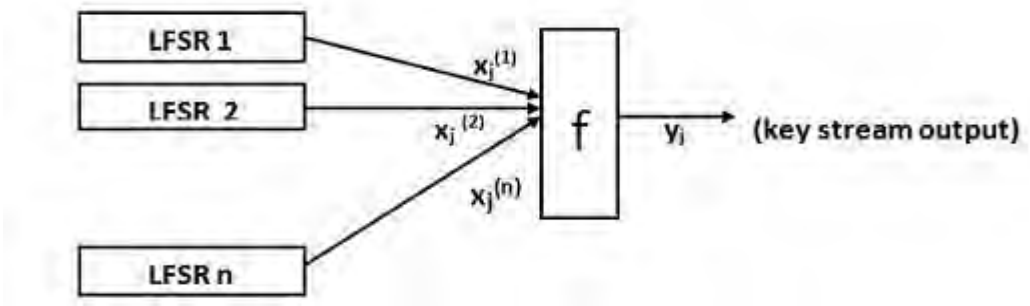

Figure 5.22 : LFSR combinés non-linéaires

- Registre filtré non-linéaire [12] [69] [70] : un générateur pseudo-aléatoire de clefs de chiffrement qui est composé d'un LFSR unique et d'une fonction non-linéaire (appelée aussi fonction de filtrage non linéaire) qui est appliquée à certains étages du registre à décalage pour produire la sortie du générateur.

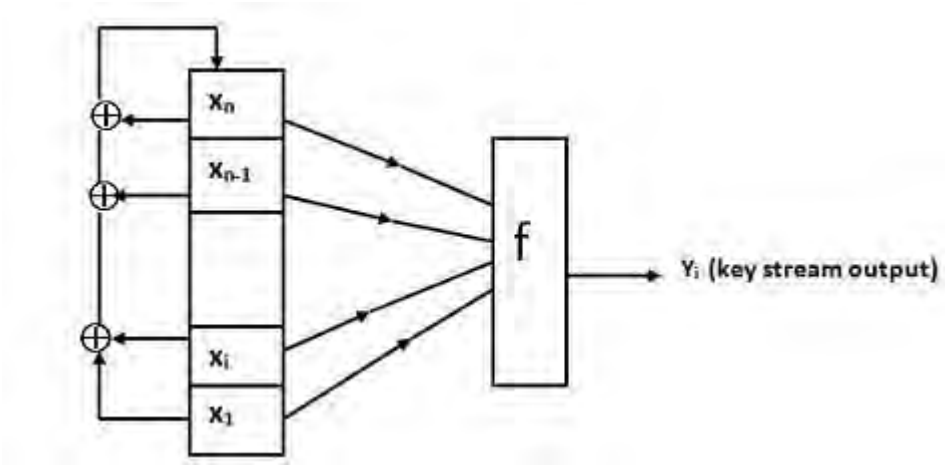

Figure 5.23 : Générateur filtré Non linéaire

- Générateur avec contrôle d'horloge (stop-and-go generator ou générateur à signal d'arrêt) [12] [71] [72] : un générateur pseudo-aléatoire de clefs de chiffrement dans lequel la sortie d'un premier LFSR est utilisée pour contrôler l'horloge d'un second LFSR. Le second registre ne change d'état à l'instant *(t)* que si la sortie du premier est égale à *(1)* à l'instant *(t-1)*.

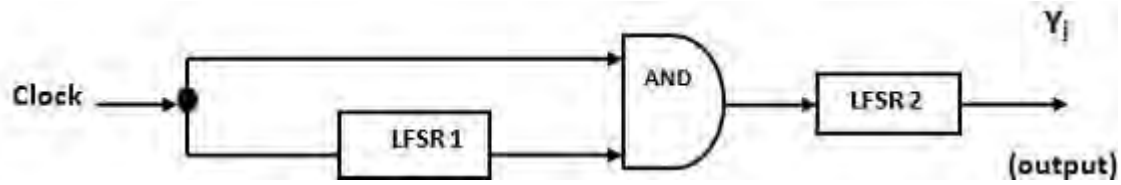

Figure 5.24 : Générateur avec contrôle d'horloge

- Il est également utile de mentionner les constructions concernant :

- Le « shrinking generator » (ou générateur rétrécissant) inventé en 1993, par D. Coopersmith, H. Krawczys et Y. Mansour [73] [74].
- Le « self-shrinking generator » (ou générateur auto-rétrécissant) inventé en 1994 par W. Meier et O. Staffelback [75].

Mais les mesures de sécurité supplémentaires n'ont pas permis jusqu'à présent de se mettre à l'abri des attaques :

- Attaques par force brute ;
- Attaques par compromise temps-mémoire [76-78]
- Attaques par corrélation [12] [23] [38] [78-83]
- Attaques algébriques [38] [78] [80] [84-88]
- Attaques par canaux cachés [38] [89-91]
- Attaques par distingueurs [38] [92]

À l'heure actuelle, les recherches sont orientées vers de nouvelles classes de registres à décalage basés notamment sur les anneaux algébriques, ce qui implique de nouvelles méthodes de cryptanalyse et de nouvelles mesures de sécurité [28] [59] [93] [94]. En fait, des études importantes ont été menées dans le domaine des registres à décalage à rétroaction non linéaire (NLFSR), aussi bien du point de vue de la conception que des attaques, et qui ont conduit :

- Depuis 1993 (par Andrews Klapper et Mark Goresky), aux FCSR (Feedback with Carry Shift Registers-Registres à décalage avec retenue) [28] [93-95] en mode de Fibonacci ou Galois, avec comme base mathématique, l'anneau des entiers de N−*Adic* au lieu de l'anneau des séries formelles utilisé pour les LFSR.

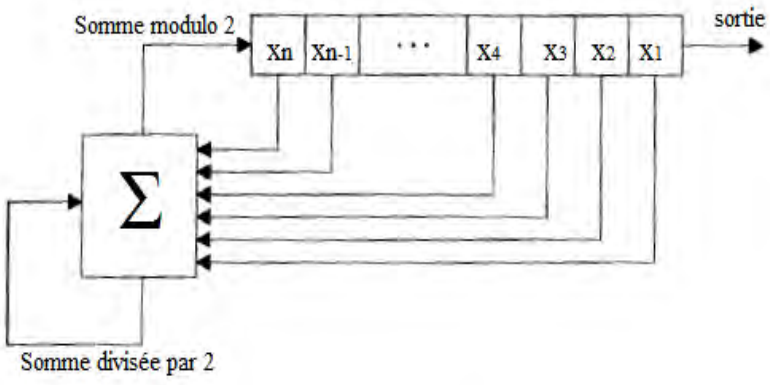

Figure 5.25: Feedback with Carry Shift Registers (FCSR)

Cependant, en dépit de leur grande complexité linéaire, ils sont sensibles aux attaques de " *l'algorithme d'approximation rationnelle*" [94] qui est similaire à celui de Berlekamp-Massey.

Cependant, ils pourraient être couplés aux LFSR dans la conception des générateurs pseudo-aléatoires.

- Aux registres à décalage avec retenue vectoriels (VFCSR), conception vectorielle des FCSR dont l'analyse a été étendue aux corps finis $GF(p^n)$.[96]

- Aux FCSR filtrés (F-FCSR), conception des FCSRs pour contrer l'attaque par l'algorithme d'approximation rationnelle. [94] [97] [98].

- Aux registres à décalage algébriques (AFSR) lorsque la base mathématique est l'anneau des entiers π−*Adic* (non spécifié, étant donné que les entiers π−*Adic* sont des généralisations des séries formelles et des entiers N−*Adic*). Les LFSR et les FCSR sont des cas particuliers des AFSR. [28]

Il est utile de prendre connaissance de la recherche sur la théorie de la stabilité des crypto systèmes en continu c'est-à-dire de la résistance de ces systèmes aux petites variations de certains de leurs paramètres en ce qui concerne en particulier la complexité linéaire et les fonctions booléennes non linéaires utilisées [3] [59] [99] :

- Pour le chiffrement synchrone additif, il existe déjà des techniques de contrôle de la stabilité de la complexité linéaire. Mais, le problème de la stabilité de la complexité linéaire locale semble pour l'instant difficile à résoudre (il s'agit d'un problème ouvert).

- Pour les registres combinés non-linéaires et les registres filtrés non-linéaires, des résultats partiels ont été obtenus sur certains aspects de la théorie de la stabilité, mais les travaux doivent se poursuivre davantage : un domaine prometteur de la recherche.

D’autres voies de recherche pourraient être explorées dans le domaine des études réalisées, en particulier, sur les espaces métriques et les séries [100] [101] [102].

Enfin, des recherches sont également menées sur les RNLUs (Registers with Nonlinear Update-Registres avec mise à jour non linéaire) qui sont une généralisation des NLFSRs dont l'étude est théorique et devrait être encore affinée. [103] [104]

5. 5 Conclusion

Comme indiqué au début de l'article publié [105], la méthode proposée détermine mathématiquement, à partir du polynôme primitif, la relation récurrente linéaire générant le LFSR (et vice versa) et facilite ainsi sa construction ; elle aide également d’établir le polynôme primitif réciproque correspondant qui donne la possibilité de construire un autre LFSR aussi bon que le premier.

Nous avons une conception et un choix judicieux des LFSRs bouclés à période longueur maximale, à utiliser, sur la base des polynômes primitifs, des polynômes primitifs réciproques et des relations de récurrence linéaire associées, ce que ne montrent pas les méthodes utilisées jusqu'ici où les récurrences sont établies, sans plus de précisions, à partir des polynômes primitifs pour dessiner les LFSRs.

D’autre part, il a semblé important de passer en revue les LFSRs, d’insister sur leur sécurité cryptographique ainsi que les recommandations figurant dans le paragraphe ci-dessus et de mettre en évidence les possibilités de recherches dans ce domaine.

REFERENCES:

[1] Canteaut, A. Stream ciphers systems. Encyclopedia of Cryptology and Security 2005, Springer Verlag (2005).

[2] Dawson, E., and Simpson, L. Analysis and Design Issues for Synchronous Stream cipher. Queensland University of Technology, Australia.

[3] Ding, C., Xiao, G., and Shan, W. The stability theory of stream ciphers. Department of Applied mathematics and Institute for Information security, Xidian, China, Springer Verlag, 1991.

[4] Konheim, Alan, G. Computer security and Cryptography. John Wiley and Sons, Inc, 2007.

[5] Mao, W. Modern Cryptography, Theory and Practice. Hewlett Packard Company-Prentice Hall PTR.

[6] Menezes Alfred J., Oorschot Paul, CV., Vanstone A. Handbook of Applied Cryptography CRC Press ,Inc, 1997.

[7] Meyer, Carl, H., and Matyas, Stephen, M. Cryptography :A new dimension in Computer data security, A guide for design and implementation. John Wiley and Sons Inc, 1982.

[8] Oppliger R. Contemporary cryptography. Artech House, 2005.

[9] Paar C. Applied cryptography and Data security. 2005.

[10] Paar C., Pelzl J. Understanding Cryptography: A Textbook for Students and Practitioners. Springer Verlag, 2010.

[11] Rothe J. Complexity Theory and Cryptography. Springer Verlag, 2005.

[12] Rueppel, Rainer A. Stream ciphers. Contemporary Cryptology, The Science of Information Integrity, Sindia National Laboratories, IEEE Press (1992), 65-134.

[13] Schmeh K. Cryptography and Public Key Infrastructure on Internet. Wiley, 2001.

[14] Schneier B. Cryptographie appliquée, Protocoles et Codes sources en C. Vuibert, 2001.

[15] Smart N. Cryptography: An Introduction. McGraw Hill.

[16] Stallings W. Cryptography and Network security: Principles and Practices. Prentice Hall, 2011.

[17] Trappe W. Introduction to Cryptography with Coding theory. Prentice Hall, 2001.

[18] Vergnaux D. Exercices et Problèmes de cryptographie. Dunod, 2012.

[19] QUH. Stream cipher and Linear complexity. University of Maryland, 2007.

[20] Talbot J, Welsh D. Complexity and Cryptography: An introduction. Cambridge University Press, 2006.

[21] Yuen PK. Practical cryptology and Web security. Pearson Education, 2006.

[22] Dumas, Jean G., Roch JL., Varrette S. Théorie des Codes. Dunod, 2007.

[23] Ebrahimi T., Leprevost F., and Warusfel B. Cryptographie et Sécurité des Systèmes et Réseaux, Lavoisier, 2007.

[24] Hersehey John E. Cryptography demystified. Mc Graw-Hill Telecom.

[25] Zemor G. Cours de cryptographie. Université de Bordeaux, 2000.

[26] Blum M., and Micali S. How to generate cryptographically strong sequences of pseudorandom bits. SIAM J. Computer, 1984, 13(4).

[27] Golomb, Solomon W. Shift register sequence. Aegen park Press, laguna Hills, CA; 1982.

[28] Goresky M., Klapper A. Algebraic Shift Register Sequences. Cambridge University Press, 2012.

[29] Foursov M. Chiffrements de flux/flot (stream ciphers), Université de Rennes.

[30] Robshaw JB. Stream ciphers. RSA Laboratories Technical Report TR-701 (July 1995).

[31] Becker H., and Piper F. Cipher systems, the protection of communications. Northwood Publications .1982,

[32] Chasse G. Cryptographie mathématique. Techniques de l'Ingénieur.

[33] Mogollon M. Cryptography and Security services: Mechanisms and Applications. University of Dallas, USA, 2008.

[34] Details of the cryptographic principles of the MA4240-Racal Datacom limited, Milford industrial estates Tollgate Road Salisbury, Wiltshire Sp12jg (appendix 1).

[35] Babbage S., Borghoff J., Vosselin V. The eSTREAM portfolio. Available: http://www.ecrypt.eu.org/estream/

[36] Babbage S., Canniere CD, Canteaut A., Cid C, Gilbert H, Johansson T, Parker M., Preneel B., Rijmen V, and Robshaw JB. The eSTREAM portfolio. Available: http://www.ecrypt.eu.org/estream/

[37] Mehreen A, Kansar F, Massood A. Comparative analysis of the structures of Estream submitted stream cipher. ICET 06, 245-250.

[38] Meier W. Stream ciphers, a perspective. LNCS 7374, Africacrypt 2012,Springer Verlag , 2012.

[39] Rueppe Rainer, A. Analysis and Design of stream ciphers. Springer Verlag, 1986.

[40] Dubrovna E. Generation of cycles by a composition of NLFSRs. Designs, Codes and Cryptography, Springer Series, Business Media (March 2014), 469-486.

[41] Beckett B. Cipher systems, The Protection of Communications. Masson, 1990.

[42] Dawson E. Linear feedback shift registers and stream ciphers.

[43] Klove T. Linear recurring sequences in boolean rings. Math. Scand 1973, 83;5- 12.

[44] Massey James L. Shift register synthesis and BCH decoding. IEEE Transaction on Information Theory, 1969, IT-15.

[45] Maurin J. Simulation deterministe du hasard. Masson et Cie, Editeurs; 1975.

[46] Saluja Kewa K. Linear feedback shift register: theory and applications. University of Winconsin 1987; 1988; 1991.

[47] Song HY. Feedback shift register sequences. Yonsei University, Seoul.

[48] Van Tilborg, Henk CA. Fundamentals of Cryptology, A professional reference and Interactive Tutorial. Kluwee Academic Publishers, 2000.

[49] Anderson R., On Fibonacci keystream generators. Lecture Notes in Computer science, 1994; 1008, Fast Software Encryption (December), 346 -352.

[50] Brent Richard P. On the periods of generalized Fibonacci recurrences. Lecture Notes in Computer science, ,Fast software encryption,12th International Conference Workshop FSE. 1992; 3557.

[51] L'ecuyer P. Uniform random number generator, Annals of operations research.

[52] Downham DY, Roberts FDK. Multiplicative Congruential Pseudo-random Generators. Department of Computational and Statistical Science, The University, Liverpool3.

[53] Kurita Y. Une méthode pour choisir les paramètres d'un générateur de nombres aléatoires congruentiels. Laboratoire National de Recherches en Météorologie .1996, 104-113.

[54] Hansen T, Mullen Gary L. Primitive polynomials over finite fields. Mathematics of Computation. 1992;59 (200), 639-643.

[55] Zierler N. Primitive trinomials whose degree is a mersenne exponent. Information and Control 1969;15; 67-69.

[56] Zivkovic M. A table of primitive binary polynomials. Mathematics of Computation; 1994;69.

[57] Berlekamp Elwy R. Algebraic Coding Theory. Mc Graw-Hill, 1967.

[58] Stamp M., Low Richard M. Applied cryptanalysis, breaking cipher in the real world. John Wiley and Sons, 2007.

[59] Cusick TW., Ding C, Renvall A. Stream ciphers and Number theory. North-Hollow Mathematical Library, Elsevier Sciences B.V; 1998.

[60] Braeken A. Cryptographic Properties of Boolean Functions and S-Boxes. PhD thesis, Department Electrical Engineering ESTA/COSIC, Universiteit Leuven. Belgium; 2006.

[61] Braeken A ,Semaev I. The ANF of the composition of addition and multiplication mod2n with a boolean function. LNCS, Fast software encryption, 12th International Conference Workshop FSE. 2005; 3557; 121-134.

[62] Deepak Kuma D, Gupta Kishan C, Maitra S. Cryptographically significant boolean functions: construction and analysis in terms of algebraic immunity. Lecture Notes in Computer Science. Fast software encryption, 12th International conference workshop, Paris. 2005; 3557; 107-120.

[63] Gupta Kishan C. Cryptographic and Combinatorial Properties of Boolean Functions and S-Boxes. PhD thesis, Applied Statistics Unit, Indian Statistical Institute; 2004.

[64] Preneel B, Logachev Oleg A. Boolean functions in cryptology and information security. Proceedings of the NATO Advanced Study Institute on Boolean Functions, IOS Press. 2007;18.

[65] Canteaut A. Combination generator. Encyclopedia of Cryptology and Security 2011, Springer Verlag 2011; 222-224.

[66] Geffe PR. How to protect data with ciphers are really hard to break. Electronics. 1973; 99-101.

[67] Golic Jovan D. Cryptanalysis of alledged A5 stream cipher. Advances in Cryptology, EUROCRYPT 2011, 97 (1233);239-255.

[68] Wei S. On generalization of Geffe's generator. International Journal of Computer Science and Network. Security, 2006; 6 n8A.

[69] Canteaut A. Filter generator. Encyclopedia of Cryptology and Security 2011, Springer Verlag. 2011, 458-460.

[70] Golic Jovan D. On the security of nonlinear filter generator. Fast software encryption LNCS, Springer Verlag. 1996; 1039: 173-188.

[71] Beth T, Piper F. The stop-and-go generator. LNCS 209; Advances in Cryptology-Eurocrypt 84, Springer Verlag. 1985-84; 88-92.

[72] Chambers WG, Gollman D. Clock-controlled shift registers. IEEE Journal on selected areas in Communications. 1989; 7,(4): 525-533

[73] Blöcher U., Dicht M. Fish: A Fast Software Stream Cipher, in Fast Software Encryption. Lecture Notes in Computer science. Springer 2000; 809; 56-63.

[74] Coppersmith D., Krawczys H., Yishay M. The shrinking generator. Lecture Notes in Computer Science. Advances in Cryptology, CRYPTO 93, Springer Verlag. 1994; 773(93) 22-39.

[75] Meier W.,Staffelback O. The self-shrinking generator. Lecture Notes in Computer Science, Advances in Cryptology, EUROCRYPT 94. Springer Verlag. 1994, 94:205 -214.

[76] Babbage S. Space Time-off in exhaustive search attacks on stream ciphers. European Convention on security and detection, IEEE Conference Publication .408:1995.

[77] Birynkov B., Shamir A. Cryptanalytic Time memory Data Trade-off for stream cipher. Lecture Notes in Computer science. Asiacrypt, Springer Verlag, 2000; 976: 1-13.

[78] Joux A. Cryptography and Network Security, Algorithmic Cryptanalysis. CRC Press Taylor and Francis Group; 2009.

[79] Golic Jovan D., Salmasizadeh M., Simpson L, Dawson E. Fast correlation attacks on nonlinear filter generator. Information Processing Letters. 1997; 64 (1): 37-42.

[80] Joux A., Berbain C., Gilbert H. Algebraic correlation attacks against linearly filtered nonlinear feedback shift registers. Lecture Notes in Computer Science. Springer Verlag . 2009; 5381: 184-198.

[81] Meier W, Staffelback O. Fast correlation attacks on certain stream ciphers. Journal of Cryptology. 1992;3:67- 86.

[82] Salmasizadeh M., simpson L, golic jova D, Dawson E. Fast correlation attacks and Multiple linear approximation. Information Security and Privacy, LNCS, Springer Verlag. 1997 1270;228 -239.

[83] Stengenthaler T. Decrypting a class of stream cipher using ciphertext only. IEEE Transactions on Computers. 1985; C-34, 81- 85.

[84] Courtois Nicolas, T. Fast algebraic attacks on stream cipher with linear feedback. Lecture Notes in Computer Science, CRYPTO , Springer Verlag. 2003; 2729: 176-194.

[85] Courtois Nicolas T, Meier W. Algebraic attacks on stream ciphers with linear feedback. Lecture Notes in Computer Science, EUROCRYPT. Springer Verlag. 2003; 2656: 345-350.

[86] Debraize B.,Goubin L. Guess-and-determine algebraic attacks on the self-shrinking generation Lecture Notes in Computer Science. Fast software encryption. Springer Verlag. 2008; 5086: 235- 252

[87] Lee Dong H. Algebraic attacks on stream ciphers (a survey). Trends in Mathematics, Information Center for Mathematical Science. 2005; 8: 133-143.

[88] Limniotis K. Algebraic attacks on stream ciphers-recent development and new results. Journal of Applied Mathematics and BioInformatics. Scienpress Ltd. 2013; 3(1): 57-5-81.

[89] Joux A, Delauney P. Galois LFSR embedded devices and side channel attacks weaknesses. Progress in Cryptology-INDOCRYPT. LNCS 4329. Springer Verlag. 2006; 436 451.

[90] Kocher Paul C. Timing attacks on implementation of Diffie Hellman, RSA, DSS and other systems. Crypto96, LNCS 1109, Springer Verlag. 1996; 104 -113.

[91] Strobel D. Side Channel Analysis attacks on Stream ciphers. PhD thesis, Universitat Bochum, 2009.

[92] Hell M., Johansson T, Brynielson L. An overview of distinguishing attacks on stream ciphers. Cryptography and Communication. Springer Verlag. 2008; 1(1) 71-94.

[93] Klapper A, Goresky M. 2-adic shift register. Fast software Encryption. LNCS 809, 174- 178.

[94] Klapper A.,Goresky M. Feedback shift registers, 2-adic span and combiners with memory. Journal of Cryptology. 1997; 10:2.

[95] Klapper A. A survey of feedback with cary shift register. SETA. LNCS 3486. Springer Verlag. 2005 ;56- 71.

[96] Allailou B. Conception et Evaluation des générateurs d'Alea. PhD thesis, Université de Paris 8 (Laboratoire d'Analyse, de Géométrie et Applications-LAGA), 2010.

[97] Arnault F, Berger TP. F-FCSR, design of new class of stream cipher. Lecture Notes in Computer science. Fast Software Encryption. 2005; 3557:83 -97.

[98] Arnault F, Berger TP, Lauradoux C. F-FCSR stream cipher. Lecture Notes in Computer science 4986, New stream cipher designs, the eSTREAM finalists. 2005; 170-178.

[99] Ding C. The stability theory of stream ciphers. Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, 2011.

[100] Mishra VN. Some Problems on Approximations of Functions in Banach Spaces. PhD thesis, Indian Institute of Technology, Roorkee- 247 667, Uttarakhand, India, 2007.

[101] Mishra VN, MISHRA LN. Trigonometric approximation of signals (functions) in Lp ($p \geq 1$) norm. International Journal of Contemporary Mathematical Sciences. 2012; 7(19) 909-918.

[102] Deepmala A Study on Fixed Point Theorems for Nonlinear Contractions and its Applications. PhD thesis, Pt. Ravishankar Shukla University, Raipur (Chhatisgarh) India 492 010, 2014.

[103] Dubrovna E, Li N. An algorithm for constructing a smallest register with nonlinear update generating a given sequence. Proceeding of IEEE International Symposium on Multiple-valued Logic (ISMVL 2014). 2014 ; 254-259.

[104] Rijmen V. Stream cipher and the Estream project. The ISC International journal of Information Security. 2010 ; 3 -11.

[105] Babacar A. Ndaw, Djiby Sow, Mamadou Sanghare, Construction of Maximum Period Linear Feedback Shift Registers (LFSR) (Primitive Polynomials and Linear Recurring Relations), British Journal of Mathematics and Computer Science, 11(4): 1-24, 2015, Article n°BJMCS.19442, SCIENCEDOMAIN *International*.

Chapitre 6

La loi des présences

Sommaire

6.1 Introduction

6.1.1 Enoncé du problème

Un générateur aléatoire délivre des symboles pris dans un alphabet de (n) symboles (exemples : les lettres de l'alphabet normal, des bits, des octets …etc) selon une loi statistique uniforme c-à-d. que les probabilités d'apparition des différents symboles sont égales à $(\frac{1}{n})$ et que ces apparitions sont toutes indépendantes.

Avec ce générateur, nous voulons fabriquer des suites chiffrantes. Pour cela, dans une tranche de (N) symboles fournis par le générateur, en supprimant les répétitions, nous obtenons une suite de (k) symboles ($1 \leq k \leq n$). Ce nombre (k) de symboles présents est une variable aléatoire dont nous pouvons établir la loi de probabilité [1-11].

Nous allons :

- étudier la loi de (k) que nous appelons « Loi des présences », dans l'hypothèse où le générateur délivre une suite uniforme ;
- Et concevoir, à l'aide de cette loi, un distingueur d'aléa.

6.1.2 Probabilité sur les suites

Soit A= {Ai/ i= 1, 2, 3, …, n} l'alphabet représentant les (n) symboles et $\Omega = \{X_N\} = \{X_{N \in \mathbb{N}}\} = \{X_1, X_2, X_3, \ldots\}$, l'ensemble des suites de longueur N (cf. Définitions 2.4.2.2).

Soit $\mathcal{P}(\Omega)$ l'ensemble de toutes les parties de Ω. Nous supposons que Ω est muni d'une algèbre de Boole, que le couple $(\Omega, \mathcal{P}(\Omega))$ est probabilisable et que nous pouvons, par conséquent, y définir une probabilité.

Il vient alors que nous avons un espace de probabilité $(\Omega, \mathcal{P}(\Omega), Pr)$ sur lequel nous pouvons calculer la probabilité d'une suite X_N :

$$\Pr(X_N) = \frac{1}{n^N} \qquad (1)$$

Notons que (n^N) est le nombre de suites possibles de longueur *(N)*, et que dans une suite X_N peuvent figurer (k) catégories de symboles ($1 \leq k \leq n$).

Supposons que nous effectuons un tirage de N symboles pour obtenir :

N_1 symboles de la première catégorie

N_2 symboles de la deuxième catégorie

……..

N_k symboles de la k ième catégorie

telle que $N = N_1 + N_2 + \ldots N_k$

Le nombre de façon d'obtenir N_1 symboles de la 1ère catégorie est $C_N^{N_1}$

Le nombre de façon d'obtenir N_2 symboles de la 2ème catégorie est $C_{N-N_1}^{N_2}$

…….....

Le nombre de façon d'obtenir N_{k-1} symboles de la (k-1) ème catégorie est $C_{N-N_1-N_2-\cdots\ldots N_{k-2}}^{N_{k-1}}$

Finalement, le nombre de façons d'obtenir X_N est :

$$C_N^{N_1} \cdot C_{N-N_1}^{N_2} \ldots C_{N-N_1-N_2-\cdots\ldots N_{k-2}}^{N_{k-1}} = \frac{N!}{N_1\,!N_2!\ldots\ldots\ldots N_k\,!} \qquad (2)$$

Si nous désignons par X_i la variable aléatoire qui représente le nombre N_i de symboles obtenus pour la i ème catégorie, alors :

$$\Pr[X_1 = N_1 \cap X_2 = N_2 \cap \ldots\ldots\ldots\ldots \cap X_i = N_i \cap \ldots \cap X_k = N_k] = \frac{N!}{N_1\,!N_2!\ldots\ldots\ldots N_k\,!} \cdot \frac{1}{n^N} \qquad (3)$$

représente la probabilité d'obtenir une suite (X_N) contenant N_1 symboles de la première catégorie, N_2 de la deuxième catégorie, ….....et N_k de la k ième catégorie.

6.2 La loi des présences

6.2.1 Variable aléatoire : nombre de symboles présents

Etudions maintenant le nombre de symboles présents dans la suite (X_N), après suppression des répétitions. Il est clair que ce nombre varie suivant les suites, car dans une suite donnée on peut avoir *1* ou *2* ou *3*…ou *n* symboles présents.

Soit Ω' l'ensemble des valeurs {*1, 2, 3, ……k, k+1, …n*} et (X) l'application suivante, qui, à une suite de Ω, fait correspondre une valeur dans Ω' :

$$X : \Omega \mapsto \Omega'$$

$$X_N \mapsto X(X_N)$$

X (X_N) est une variable aléatoire discrète.

Supposons que Ω' soit munie d'une famille de sous-ensembles fermée pour les mêmes opérations que $\mathcal{P}$ (Ω) ; nous pouvons alors, naturellement, mettre une probabilité sur Ω' [1-11].

Considérons B= {k, $k+1$} : pour définir la probabilité de B, nous remarquons que $X(X_N) \epsilon B$ si et seulement si (X_N) appartient au sous – ensemble de Ω contenant les suites à (k) ou ($k+1$) symboles.

Alors, ce sous-ensemble est $X^{-1}(k, k+1) = X^{-1}(B)$; nous pouvons alors affecter la même probabilité à B et à $X^{-1}(B)$ puisque les occurrences $X(X_N) \epsilon B$ et $(X_N) \epsilon X^{-1}(B)$ sont équivalentes :

$$X(X_N) \epsilon B \Leftrightarrow (X_N) \epsilon X^{-1}(B)$$

$$\Rightarrow \Pr[X(X_N) \epsilon B] = \Pr[(X_N) \epsilon X^{-1}(B)]$$

Et nous pouvons noter simplement :

$$\Pr[X(X_N) = k] = \Pr[X=k] \quad (4)$$

et déterminer, ensuite, la loi de probabilité (ou distribution de probabilité) qui est celle d'avoir (k) symboles présents dans la suite X_N. Nous l'appellerons « *Loi des présences* ».

6.2.2 Etablissement de la loi des présences – Le théorème des présences

Soit (B_k) le nombre de suites qui présentent exactement (k) symboles, alors :

$$\Pr[X=k] = \frac{B_k}{n^N} \quad (5)$$

Désignons par D_k le nombre de suites qui présentent (k) symboles donnés. Le nombre de façons de choisir ces (k) symboles pour obtenir les suites qui présentent exactement (k) symboles est C_n^k :

$$B_k = C_n^k . D_k \quad (6)$$

Exemple1 : Pour un alphabet de $n=3$ lettres {a, b, c}, le nombre de suite B_2 qui présentent exactement 2 lettres est :

$$B_2 = C_3^2 . D_2$$

Où D_2 représente le nombre de suites qui présentent 2 lettres données (*ab, bc ou ac*).

Tout ce que nous savons, c'est que le nombre de suites que nous pouvons obtenir avec 2 lettres est (2^N).
Représentons alors schématiquement les tableaux contenant toutes les suites qui présentent exactement 2 lettres données et ceux contenant toutes les suites qui présentent exactement une lettre donnée, de façon à obtenir toutes les suites possibles dont le nombre est (2^N).

I	a, b	a, c	b, c	Suites présentant exactement 2 lettres données (D_2)
II	a	a	b	Suites présentant exactement 1 lettre donnée (D_1)
III	b	c	c	Suites présentant exactement 1 lettre donnée (D_1)

Tableau 6.1 : Tableau des suites présentant exactement 2 lettres données

Il vient alors :

$$2^N = D_2 + D_1 + D_1 = D_2 + 2D_1$$

En tenant compte du fait que le nombre de suites D_1 que nous pouvons former avec une lettre donnée est (1), il vient :

$$D_2 = 2^N - 2 \quad \text{et} \quad B_2 = C_3^2 . D_2 = C_3^2 \ (2^N - 2)$$

Exemple2 : Pour un alphabet de 4 lettres {a, b, c, d}, le nombre de suites D3 qui représentent exactement 3 lettres données est :

$$B_3 = C_4^3 . D_3$$

Nous obtenons les tableaux qui suivent :

I	(a b c)	(a b d)	(b c d)	(c d e)	Suites présentant exactement 3 lettres données (D_3)
II	(a b)	(a b)	(b c)	(c d)	Suites présentant exactement 2 lettres données ($C_3^2 . D_2$)
	(a c)	(a d)	(b d)	(c d)	
	(b c)	(b d)	(c d)	(d a)	
III	a	a	b	c	Suites présentant exactement 1 lettre donnée ($C_3^1 . D_1$)
	b	b	c	d	
	c	d	d	a	

Tableau 6.2 : Tableau des suites présentant exactement 3 lettres données

Le nombre total de suites que nous pouvons former avec 3 lettres données étant (3^N), il vient :

$$3^N = D_3 + C_3^2 . D_2 + C_3^1 . D_1$$

$$\Rightarrow \quad D_3 = 3^N + C_3^2 . D_2 + C_3^1 . D_1$$

$$\Rightarrow \quad D_3 = 3^N - C_3^2 (2^N - 2) - C_3^1$$

$$\Rightarrow \quad B_3 = C_4^3 . D_3 = C_4^3 \ [3^N - C_3^2 (2^N - 2) - C_3^1]$$

Exemple 3 : Le même raisonnement peut être fait pour un alphabet de 5 lettres {a, b, c, d, e} et en calculant le nombre total de suites D_4 qui présentent 4 lettres données. Il nous suffit d'adjoindre à ces suites, celles qui contiennent 3 lettres données prises parmi les 4 lettres ; celles à 2 lettres données ; et celles à 1 lettre donnée, pour obtenir le nombre total de suites possibles qui contiennent 4 lettres c-à-d (4^N).

I	(a b c d)	(a b c e)	(b c d e)	(c d e a)	(d e a b)	Suites présentant exactement 4 symboles donnés (D_4)
II	(a b c)	(a b c)	(b c d)	(c d e)	(d e a)	Suites présentant exactement 3 symboles donnés $(C_4^3.D_3)$
	(a b d)	(a b e)	(b c e)	(c d a)	(d e b)	
	(b c d)	(b c e)	(c d e)	(d e a)	(e a b)	
	(c d a)	(c e a)	(d e b)	(e a c)	(a b d)	
III	(a b)	(a b)	(b e)	(c d)	(d e)	Suites présentant exactement 2 symboles donnés $(C_4^2.D_2)$
	(a c)	(a c)	(b d)	(c e)	(d a)	
	(b c)	(b c)	(c d)	(d e)	(e a)	
	(a d)	(a e)	(b e)	(c a)	(d b)	
	(b d)	(b e)	(c e)	(d a)	(e b)	
	(c d)	(c e)	(d e)	(e a)	(a b)	
IV	a	a	b	c	d	Suites présentant exactement 1 symbole donnée $(C_4^1.D_1)$
	b	b	c	d	e	
	c	c	d	e	a	
	d	e	e	a	b	

Tableau 6.3 : Tableau des suites présentant exactement 4 lettres données

Il vient donc que le nombre total de suites de longueur N que nous pouvons former avec 4 lettres est :

$$4^N = D_4 + C_4^3.D_3 + C_4^2.D_2 + C_4^1.D_1$$

$$\Rightarrow \quad D_4 = 4^N - C_4^3.D_3 - C_4^2.D_2 - C_4^1.D_1$$

$$\Rightarrow \quad D_4 = 4^N - C_4^3[3^N - C_3^2(2^N - 2) - C_3^1] - C_4^2(2^N - 2) - C_4^1$$

$$\Rightarrow \quad B_4 = C_5^4.D_4 = C_5^4[4^N - C_4^3.D_3 - C_4^2.D_2 - C_4^1.D_1]$$

Généralisation (récurrence):

Le nombre total de suites équiprobables de longueur N que nous pouvons former avec exactement (k) symboles donnés est (k^N).

En nous référant aux tableaux précédents, nous pouvons faire une partition de l'ensemble de ces suites en procédant de la manière suivante :

- Un premier tableau contiendra les suites qui présentent exactement *(k)* symboles donnés : soit (D_k) leur nombre ;
- A ce tableau pourront être adjoints d'autres tableaux qui contiendront les suites présentant (*k-1*) symboles choisis exactement parmi les *(k)* : ils sont au nombre de C_k^{k-1}. Si (D_{k-1}) est le nombre de suites qui présentent (*k-1*) symboles exactement, alors (C_k^{k-1}. D_{k-1}) est le nombre total de suites à (*k-1*) symboles et qui se trouvent dans les C_k^{k-1} tableaux.
- Le même raisonnement vaut pour les C_k^{k-2} tableaux de suites qui présentent (*k-2*) symboles et le nombre total de suites que contiennent ces tableaux est (C_k^{k-2}. D_{k-2}).

- Il en sera de même jusqu'aux tableaux qui contiennent des suites présentant un et un seul symbole pris parmi les (k) donnés et dont le nombre est C_k^1.

- En définitive, en additionnant toutes les suites obtenues, nous obtenons le nombre total de suites équiprobables (k^N) que nous pouvons former avec (k) symboles.

k symboles	Suites présentant exactement (k) symboles donnés (D_k)
(k-1) symboles	Suites présentant exactement (k-1) symboles donnés $(C_k^{k-1}.D_{k-1})$
(k-1) symboles	
……..	
(k-1) symboles	
(k-2) symboles	Suites présentant exactement (k-2) lettres données $(C_k^{k-2}.D_{k-2})$
(k-2) symboles	
……..	
……..	
……..	
(k-2) symboles	
1 symbole	Suites présentant exactement 1 lettre donnée $(C_k^1.D_1)$
1 symbole	
……..	
1 symbole	

Tableau 6.4 : Tableau des suites présentant exactement (k) lettres données

Il vient donc pour le nombre de suites que nous pouvons former avec (k) symboles donnés :

$$k^N = D_k + C_k^{k-1}D_{k-1} + C_k^{k-2}D_{k-2} + \ldots\ldots\ldots\ldots\ldots + C_k^1 D_1 \qquad (7)$$

$$\Rightarrow D_k = k^N - C_k^{k-1}D_{k-1} - C_k^{k-2}D_{k-2} - \ldots\ldots\ldots\ldots\ldots - C_k^1 D_1 \qquad (8)$$

<u>Extension aux suites présentant exactement</u> (k+1) <u>symboles donnés</u> :

Le nombre total de suites que nous pouvons former avec (k+1) symboles donnés est $(k+1)^N$. En faisant le même raisonnement que précédemment, il vient :

$$k^{N+1} = D_{k+1} + C_{k+1}^k D_k + C_{k+1}^{k-1}D_{k-1} + \ldots\ldots\ldots\ldots\ldots + C_{k+1}^1 D_1 \qquad (9)$$

$$\Rightarrow D_{k+1} = (k+1)^N - C_{k+1}^k D_k - \ldots\ldots\ldots\ldots\ldots - C_{k+1}^1 D_1 \qquad (10)$$

Finalement, nous pouvons établir la relation de récurrence suivante concernant le nombre de suites (B_k) qui présentent exactement (k) symboles donnés :

$$B_k = C_n^k D_k = C_n^k[k^N - C_k^{k-1}D_{k-1} - C_k^{k-2}D_{k-2} - \ldots\ldots\ldots\ldots\ldots - C_k^1 D_1] \qquad (11)$$

(X) étant la variable aléatoire désignant le nombre (k) de symboles présents dans la suite (X_N), il vient d'après (5) :

$$\Pr[X=k]=\frac{B_k}{n^N}=\frac{C_n^k D_k}{n^N}=\frac{C_n^k\left[k^N-C_k^{k-1}D_{k-1}-C_k^{k-2}D_{k-2}-\ldots\ldots\ldots\ldots-C_k^1D_1\right]}{n^N} \qquad (12)$$

Application : *n=5 ; k=1, 2, 3, 4, 5*.

Vérifions avec ces exemples simples que nous avons bien une loi de probabilité.

$D_1=1$

$D_2=2^N-2$

$D_3=3^N-C_3^2(2^N-2)-C_3^1$

$D_4=4^N-C_4^3[3^N-C_3^2(2^N-2)-C_3^1]-C_4^2(2^N-2)-C_4^1$

$D_5=5^N-C_5^4[4^N-C_4^3[3^N-C_3^2(2^N-2)-C_3^1]-C_4^2(2^N-2)-C_4^1]-C_5^3[3^N-C_3^2(2^N-2)-C_3^1]$

$-C_5^2(2^N-2)-C_5^1$

Nous pouvons calculer alors les probabilités suivantes :

$\Pr(X=1)=\frac{C_5^1D_1}{5^N}=\frac{C_5^1}{5^N}$

$\Pr(X=2)=\frac{C_5^2D_2}{5^N}=\frac{C_5^2(2N-2)}{5^N}$

$\Pr(X=3)=\frac{C_5^3D_3}{5^N}=\frac{C_5^3[3N-C_3^2(2N-2)-C_3^1]}{5^N}$

$\Pr(X=4)=\frac{C_5^4D_4}{5^N}=\frac{C_5^4[4N-C_4^3[3N-C_3^2(2N-2)-C_3^1]-C_4^2(2N-2)-C_4^1]}{5^N}$

$\Pr(X=5)=\frac{C_5^5D_5}{5^N}=\frac{5N-C_5^4[4N-C_4^3[3N-C_3^2(2N-2)-C_3^1]-C_4^2(2N-2)-C_4^1]-C_5^3[3N-C_3^2(2N-2)-C_3^1]-C_5^2(2N-2)-C_5^1}{5^N}$

En additionnant membre à membre, on vérifie bien que :

$\Pr(X=1)+\Pr(X=2)+\Pr(X=3)+\Pr(X=4)+\Pr(X=5)=1$

Et plus généralement pour différentes valeurs de *k=1, 2, 3, ... ,n* , on vérifie bien que $\sum_{k=1}^{n}\Pr(X=k)=1$

Nous avons donc bien une loi de probabilité (12) que nous pouvons simplifier d'avantage.

Simplification : De (12), il résulte que :

$$\Pr[X=k]=\frac{B_k}{n^N}=\frac{C_n^k\left[k^N-\sum_{i-1}^{k-1}C_k^{k-i}D_{k-i}\right]}{n^N} \qquad (13)$$

Où B_k est le nombre de suites qui représente exactement (k) symboles.

$$B_k = C_n^k D_k = C_n^k [k^N - \sum_{i=1}^{k-1} C_k^{k-i} D_{k-i}]$$

Mais, $B_{k-i} = C_n^{k-i} D_{k-i} \quad \Rightarrow \quad D_{k-i} = \frac{B_{k-i}}{C_n^{k-i}}$

En remplaçant dans l'expression de B_k, il vient : $B_k = C_n^k \left[k^N - \sum_{i=1}^{k-1} \frac{C_k^{k-i}}{C_n^{k-i}} B_{k-i}\right] = C_n^k k^N - \sum_{i=1}^{k-1} \frac{C_n^k C_k^{k-i}}{C_n^{k-i}} B_{k-i}$

$$\Rightarrow B_k = C_n^k k^N - \sum_{i=1}^{k-1} C_{n-k+i}^i B_{k-i} \qquad (14)$$

En posant dans cette expression $E_k = C_n^k k^N$, il vient :

$$B_k = E_k - \sum_{i=1}^{k-1} C_{n-k+i}^i B_{k-i}$$

Dès lors, calculons B_k pour différentes valeurs de *(k)* :

k=1 $\quad B_1 = E_1$

k=2 $\quad B_2 = E_2 - C_{n-1}^1 B_1 = E_2 - C_{n-1}^1 E_1$

k=3 $\quad B_3 = E_3 - C_{n-2}^1 B_2 - C_{n-1}^2 B_1$

$= E_3 - C_{n-2}^1 (E_2 - C_{n-1}^1 E_1) - C_{n-1}^2 E_1 = E_3 - C_{n-2}^1 E_2 + C_{n-1}^2 E_1$

k=4 $\quad B_4 = E_4 - C_{n-3}^1 B_3 - C_{n-2}^2 (E_2 - C_{n-1}^1) E_1 - C_{n-1}^3 E_3$

$= E_4 - C_{n-3}^1 [E_3 - C_{n-2}^1 E_2 + C_{n-1}^2 E_1] - C_{n-2}^2 [E_2 - C_{n-1}^1 E_1] - C_{n-1}^3 E_3$

$= E_4 - C_{n-3}^1 E_3 + C_{n-2}^2 E_2 - C_{n-1}^3 E_1$

En admettant le résultat jusqu'à l'ordre (*k*), il vient :

$B_k = C_{n-k}^0 E_k - C_{n-k+1}^1 E_{k-1} + C_{n-k+2}^2 E_{k-2} - \ldots (-1)^i C_{n-k+i}^i E_{k-i} \ldots (-1)^{k-1} C_{n-1}^{k-1} E_1$

$= \sum_{i=0}^{k-1} (-1)^i C_{n-k+i}^i E_{k-i}$

Mais, $E_{k-i} = C_n^{k-i} (k-i)^N$, il vient alors $B_k = \sum_{i=0}^{k-1} (-1)^i C_n^{k-i} C_{n-k+i}^i (k-i)^N$

En introduisant dans l'expression la valeur $C_n^{k-i} C_{n-k+i}^i = \frac{n!}{i!(n-k)!(k-i)!}$ il vient, après simplification :

$$B_k = C_n^k \sum_{i=0}^{k-1} (-1)^i C_k^i (k-i)^N \qquad (15)$$

On montre de la même manière que cette formule est vraie pour l'ordre (k+1) :

$$B_{k+1} = C_n^{k+1} \sum_{i=0}^{k-1} (-1)^i C_{k+1}^i (k-i)^N \qquad (16)$$

Puisque, d'après (5) :

$$\Pr[X=k] = \frac{B_k}{n^N} = \frac{Nombre\ de\ suites\ qui\ présentent\ exactement\ (k) symboles}{Nombre\ de\ suites\ possibles}$$

$$\Rightarrow \quad \Pr[X=k] = C_n^k \sum_{i=0}^{k-1} (-1)^i\ C_k^i\ \left(\frac{k-i}{n}\right)^N \qquad (17)$$

Ces résultats prouvent le théorème suivant énonçant **la loi des présences**.

THEOREME DES PRESENCES : Soit (G) un générateur aléatoire qui fournit des suites chiffrantes de (N) symboles pris dans un alphabet de (n) symboles avec des probabilités d'apparition égales à $\left(\frac{1}{n}\right)$. Alors le nombre de symboles (k) ($1 \leq k \leq n$) présents dans une suite chiffrante donnée, après suppression des répétitions des symboles, est une variable aléatoire (notée X) qui suit la loi de probabilité suivante appelée ***« la loi des présences » :***

$$\Pr[X=k] = C_n^k \sum_{i=0}^{k-1} (-1)^i\ C_k^i\ \left(\frac{k-i}{n}\right)^N$$

avec :
n = nombre de symboles de l'alphabet
N =longueur de la suite chiffrante (clé de chiffrement)
i = indice de l'addition
k = nombre de symboles présents dans la suite.

6.2.3 Tabulation

A présent, pour n et N fixés, nous pouvons tabuler la loi des présences aux fins particulièrement d'élaboration de tests statistiques ou de distingueurs, dans le chapitre 7 de la thèse (Exemple : champ d'application n= 8, 26, 27, 31, 32, 256 et N= 10, 20, 50, 100, 200, 1250, 2500). Les applications porteront sur des tableaux de répartition de la variable aléatoire (X) et conduiront, particulièrement à des calculs d'espérances mathématiques ou valeurs moyennes de (X), des variances de (X) pour caractériser la dispersion de ses valeurs au voisinage des espérances mathématiques, à tracer et à interpréter les diagrammes de distribution des valeurs des probabilités.

Exemple 1 : Suite de longueur N=10 obtenue à partir d'un alphabet de n=8 symboles

k	$P_r(X = k)$	$P_r(X \leqslant k)$
1	$7.450 \cdot 10^{-9}$	$7.450 \cdot 10^{-9}$
2	$2.665 \cdot 10^{-5}$	$2.665 \cdot 10^{-5}$
3	0.0029	$2.994 \cdot 10^{-3}$
4	0.053	$5.635 \cdot 10^{-2}$
5	0.2661	$3.224 \cdot 10^{-1}$
6	0.4285	$7.509 \cdot 10^{-1}$
7	0.2207	$9.717 \cdot 10^{-1}$
8	0.0281	0.9998

Tableau 6.5 : Tableau de répartition de X pour n=8 et N=10

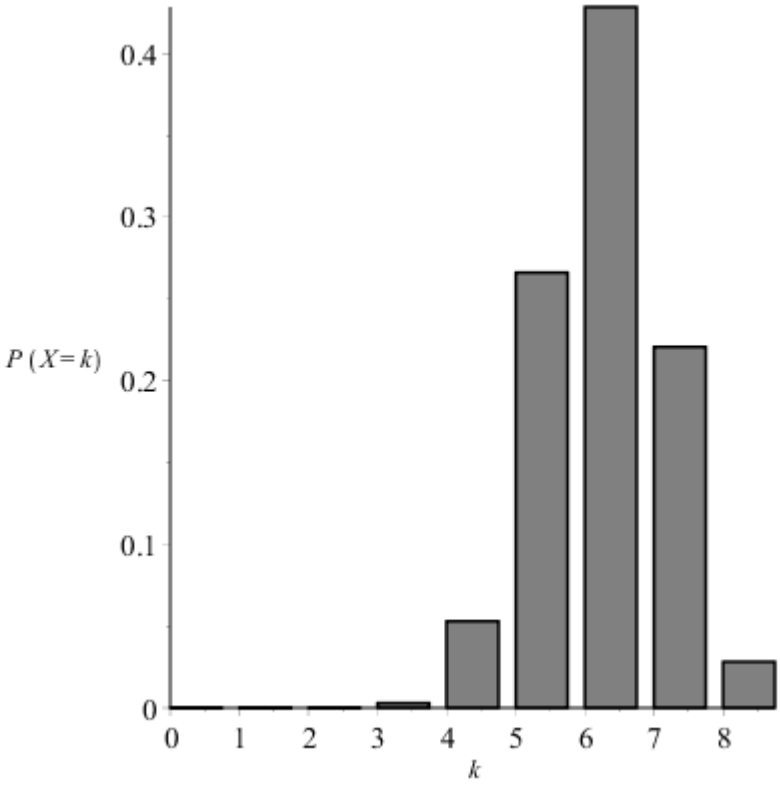

Figure 6.1 : Diagramme de répartition des valeurs (n=8 ; N=10)

L'espérance mathématique est : E(*X*)= $\sum_{k=1}^{8} kP_r\ (X = k)$ =5,89

et la variance V(X)=E(X^2) –[E(X)] ^{2} = 0.8386.

Nous pouvons prendre une moyenne de 6 symboles qui est la valeur la plus probable (cf. figure 6.1).

Exemple 2 : Suite de longueur *N*=20 obtenue à partir d'un alphabet de *n*=8 symboles

k	$P_r(X = k)$	$P_r(X \leqslant k)$
1	$6.938 \cdot 10^{-18}$	$6.938 \cdot 10^{-18}$
2	$2.546 \cdot 10^{-11}$	$2.546 \cdot 10^{-11}$
3	$1.698 \cdot 10^{-7}$	$1.698 \cdot 10^{-7}$
4	$6.591 \cdot 10^{-5}$	$6.608 \cdot 10^{-5}$
5	$4.366 \cdot 10^{-3}$	$4.4321 \cdot 10^{-3}$
6	$7.529 \cdot 10^{-2}$	$7.9722 \cdot 10^{-2}$
7	$3.897 \cdot 10^{-1}$	$4.6942 \cdot 10^{-1}$
8	$5.305 \cdot 10^{-1}$	0.9999

Tableau 6.6 : Tableau de répartition de X pour n=8 et N=20

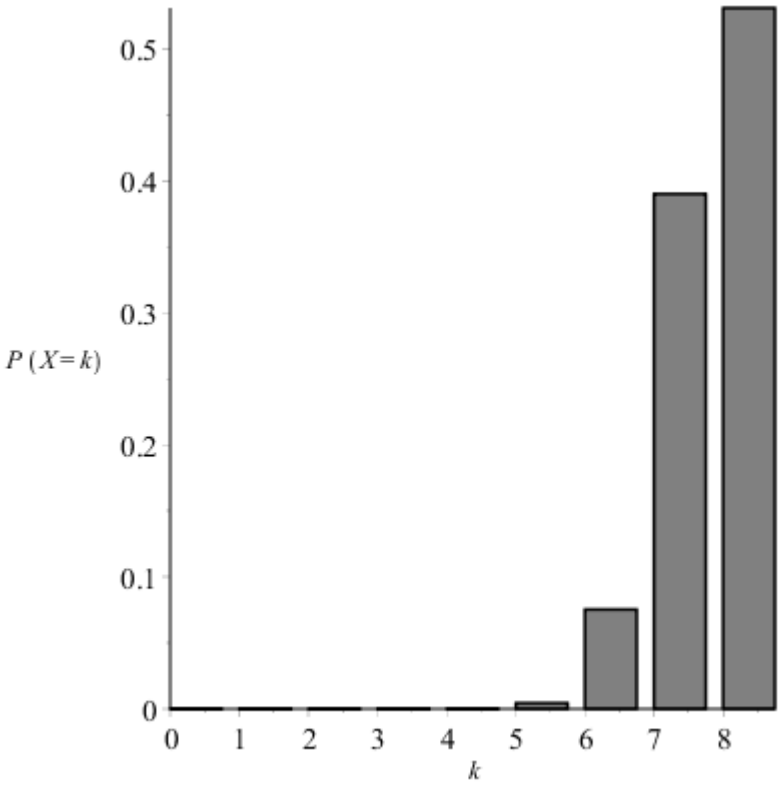

Figure 6.2 : Diagramme de répartition des valeurs (n=8 ; N=20)

L'espérance mathématique E(*X*) =7,4457. Soit approximativement une moyenne de (7) symboles (cf. figure 6.2).

La variance V(X) est estimée à 0,5144.

6.2.3.1 Interprétations des premiers résultats

Dans l'exemple 1 (*n*=8, *N*=10), le diagramme de distribution des probabilités est symétrique, sa forme montre que les valeurs sont très dispersées autour de la moyenne, et la variance (moyenne des carrés des déviations par rapport à la moyenne, et qui sert à caractériser la dispersion de la distribution autour de la moyenne ou espérance mathématique), qui est assez grande, est égale à 0,8386. Nous pouvons nous attendre à obtenir en moyenne 6 symboles.

Dans l'exemple 2 (*n*=8, *N*=20), la forme du diagramme de distribution des probabilités permet d'affirmer que la valeur à laquelle il faut s'attendre, se trouve du côté des grandes valeurs et ne s'écarte pas trop de la moyenne. La valeur de la variance est nettement plus petite (0,5144), et nous pouvons nous attendre, à obtenir en moyenne 7 symboles différents dans la suite aléatoire de longueur *N*=20.

En conséquence, plus la longueur (*N*) de la suite est petite, plus les écarts entre les différentes valeurs et la moyenne sont assez grands. Contrairement à une suite de plus grande longueur qui correspond à un diagramme de distribution qui a plutôt tendance à se réduire à une droite ; la variance y est nettement plus petite et la moyenne est plus grande. En effet, plus la variance est petite, plus les valeurs sont concentrées et proches les unes des autres ; plus les écarts entre les valeurs et la moyenne sont petits, plus ces valeurs sont concentrées autour de la moyenne. Plus la variance est élevée, plus les écarts entre les valeurs sont élevés.

Exemple 3 : Suite de longueur N=100 obtenue à partir d'un alphabet de n=26 symboles

E(X)= 25 V(X)= 0.5219

k	$P_r(X = k)$	$P_r(X \leqslant k)$
1	.	.
2	.	.
3	.	.
4	.	.
5	.	.
.	.	.
.	.	.
.	.	.
22	$6.703 \cdot 10^{-4}$	$6.703 \cdot 10^{-4}$
23	$9.321 \cdot 10^{-3}$	$9.991 \cdot 10^{-3}$
24	$7.626 \cdot 10^{-2}$	$8.625 \cdot 10^{-2}$
25	$3.314 \cdot 10^{-1}$	$4.176 \cdot 10^{-1}$
26	$5.822 \cdot 10^{-1}$	$9.998 \cdot 10^{-1}$

Tableau 6.7 : Tableau de répartition de X pour n=26 et N=100

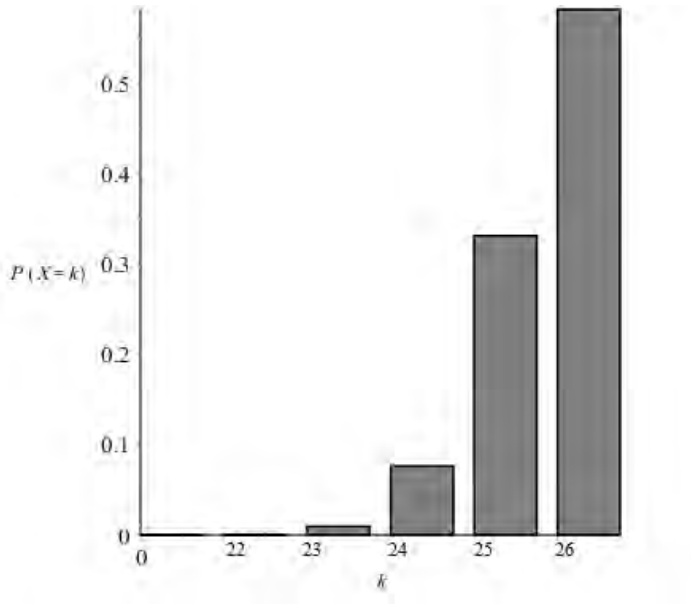

Figure 6.3 : Diagramme de répartition des valeurs (n=26 ; N=100)

Exemple 4 : Suite de longueur N=200 obtenue à partir d'un alphabet de n=26 symboles

E(X) =25.98 V(X) = 0,01

k	. $P_r(X = k)$	. $P_r(X \leq k)$
1	..	..
2	..	..
..	..	..
..	..	..
..	..	..
24	$3,6088.10^{-5}$	$3,6088.10^{-5}$
25	$1,0121.10^{-2}$	$1,0157.10^{-2}$
26	0,98984	1,0000

Tableau 6.8 : Tableau de répartition de X pour n=26 et N=200

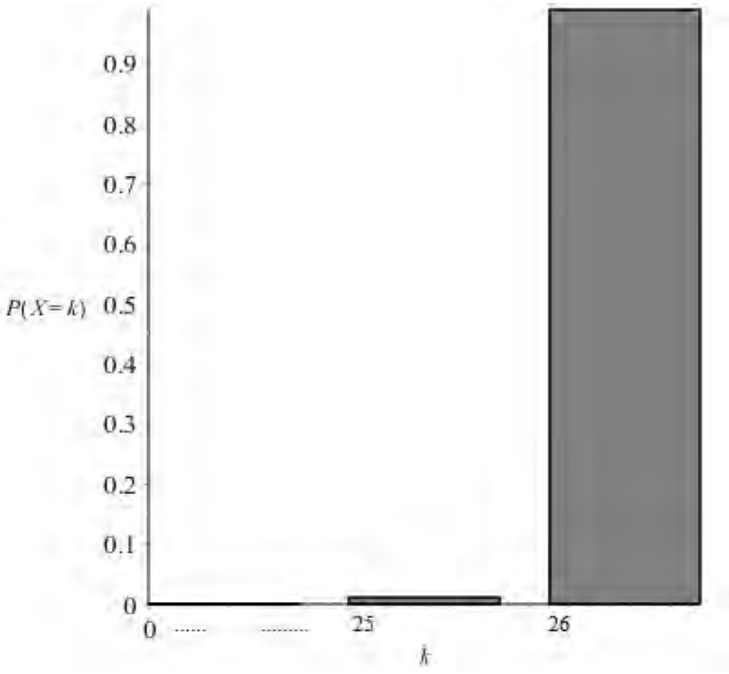

Figure 6.4 : Diagramme de répartition des valeurs (n=26 ; N=200)

Exemple 5 : Suite de longueur N=50 obtenue à partir d'un alphabet de n=27 symboles

E(X) = 22,91 V(X) = 2.27

k	$P_r(X = k)$	$P_r(X \leqslant k)$
1	.	.
.	.	.
.	.	.
16	$0.5 \cdot 10^{-4}$	$0.5 \cdot 10^{-4}$
17	0.00033	0.00038
18	0.00221	0.00259
19	0.01213	0.01472
20	0.04437	0.05909
21	0.11527	0.17436
22	0.20876	0.38312
23	0.25892	0.64204
24	0.21283	0.85487
25	0.10970	0.96457
26	0.03162	0.99619
27	0.0039	1.0001

Tableau 6.9 : Tableau de répartition de X pour n=27 et N=50

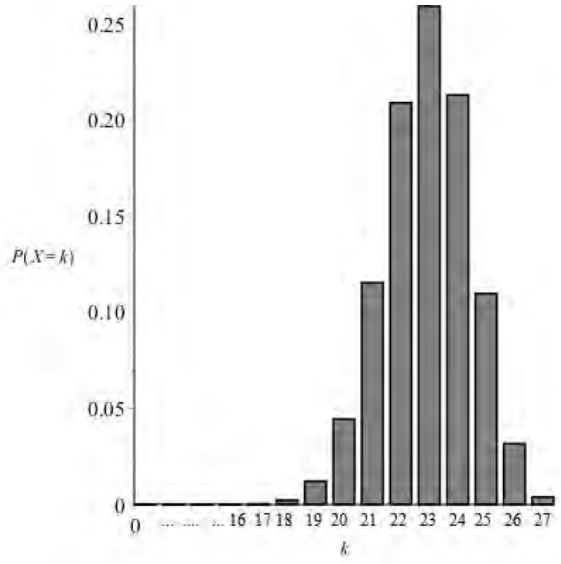

Figure 6.5 : Diagramme de répartition des valeurs (n=27 ; N=50)

Exemple 6 : Suite de longueur N=100 obtenue à partir d'un alphabet de n=31 symboles

E(X) = 29,823 V(X) = 1,31

k	$P_r(X = k)$	$P_r(X \leqslant k)$
1	.	.
2	.	.
3	.	.
.	.	.
.	.	.
23	$86 \cdot 10^{-8}$	$86 \cdot 10^{-8}$
24	$0.20 \cdot 10^{-4}$	$2.085 \cdot 10^{-5}$
25	0.00018	$2.008 \cdot 10^{-4}$
26	0.0023	$2.5009 \cdot 10^{-3}$
27	0.0160	$1.8501 \cdot 10^{-2}$
28	0.0778	$9.6301 \cdot 10^{-2}$
29	0.23251	$3.2881 \cdot 10^{-1}$
30	0.39101	$7.1982 \cdot 10^{-1}$
31	$2.7979 \cdot 10^{-1}$	$9.996 \cdot 10^{-1}$

Tableau 6.10 : Tableau de répartition de X pour n=31 et N=100

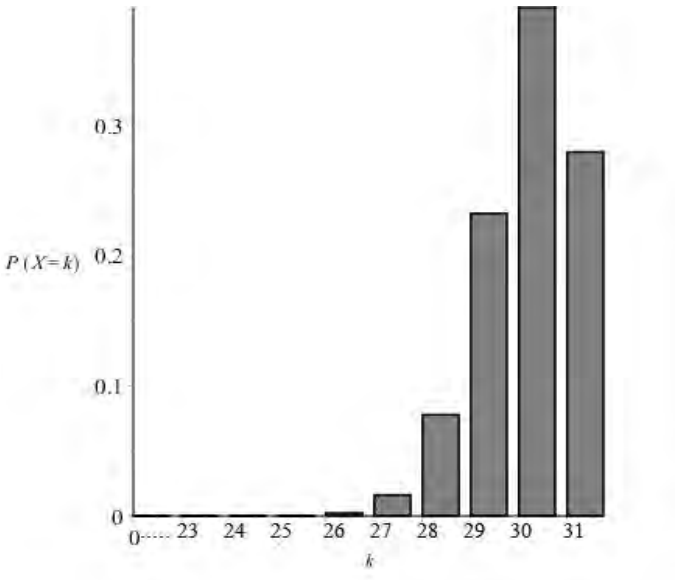

Figure 6.6 : Diagramme de répartition des valeurs (n=31 ; N=100)

Exemple 7 : Suite de longueur N=100 obtenue à partir d'un alphabet de n=32 symboles

E(X) = 30.65 V(X) = 1.38054

k	$P_r(X = k)$	$P_r(X \leqslant k)$
1	.	.
2	.	.
3	.	.
.	.	.
.	.	.
25	$0.6399 \cdot 10^{-4}$	$0.6399 \cdot 10^{-4}$
26	0.00042	$4.8399 \cdot 10^{-4}$
27	0.00435	$4.834 \cdot 10^{-3}$
28	0.02603	$3.0864 \cdot 10^{-2}$
29	0.10366	$1.3452 \cdot 10^{-1}$
30	0.26137	$3.9589 \cdot 10^{-1}$
31	0.3742	$7.7009 \cdot 10^{-1}$
32	0.2296	$9.9969 \cdot 10^{-1}$

Tableau 6.11 : Tableau de répartition de X pour n=32 et N=100

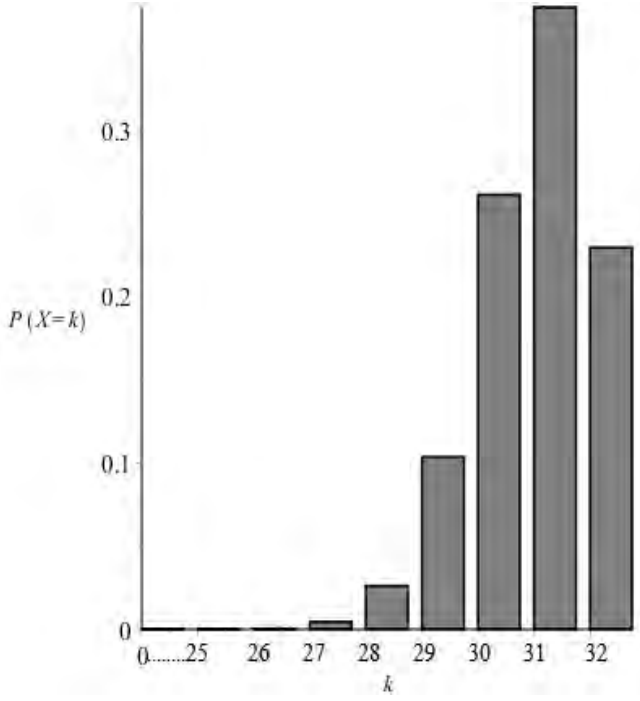

Figure 6.7 : Diagramme de répartition des valeurs (n=32 ; N=100)

Exemple 8 : Suite de longueur N=1250 obtenue à partir d'un alphabet de n=256 symboles

E(X) = 254.08 ($\simeq$ 254) V(X) = 41.0

k	$P_r(X = k)$	$P_r(X \leqslant k)$
1	.	.
2	.	.
.	.	.
.	.	.
240	2.4750×10^{-11}	2.4760×10^{-11}
241	2.8699×10^{-10}	3.1175×10^{-10}
242	3.0377×10^{-9}	3.3495×10^{-9}
243	2.9236×10^{-8}	3.2586×10^{-8}
244	2.5457×10^{-7}	2.8716×10^{-7}
245	1.9941×10^{-6}	2.2813×10^{-6}
246	1.3958×10^{-5}	4.2754×10^{-6}
247	8.6588×10^{-5}	9.0863×10^{-5}
248	4.7135×10^{-4}	5.6221×10^{-4}
249	2.2245×10^{-3}	2.7867×10^{-3}
250	8.9597×10^{-3}	1.1746×10^{-2}
251	3.0172×10^{-2}	4.1918×10^{-2}
252	8.2653×10^{-2}	0.12457
253	0.17674	0.30131
254	0.27665	0.57796
255	0.28183	0.85979
256	0.14018	0.99997

Tableau 6.12 : Tableau de répartition de X pour n=256 et N=1250

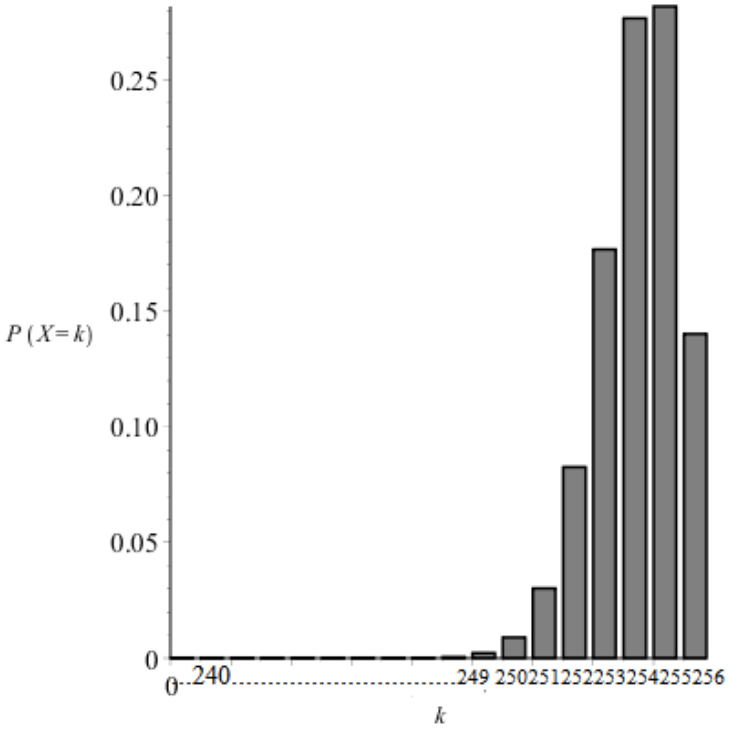

Figure 6.8 : Diagramme de répartition des valeurs (n=256 ; N=1250)

Exemple 9 : Suite de longueur *N*=2500 obtenue à partir d'un alphabet de *n*=256 symboles

E(*X*) = 255.99 (≃256) V(*X*) = 7.0

k	$P_r(X = k)$	$P_r(X \leqslant k)$
1	.	.
2	.	.
.	.	.
.	.	.
.	.	.
245	1.2774×10^{-29}	.
.	.	.
250	6.4824×10^{-15}	6.4824×10^{-15}
251	3.3442×10^{-12}	3.3507×10^{-12}
252	1.3762×10^{-9}	1.3796×10^{-9}
253	4.3384×10^{-7}	4.3522×10^{-7}
254	9.8259×10^{-5}	9.8694×10^{-5}
255	1.4216×10^{-2}	1.4315×10^{-2}
256	0.98569	1.0000

Tableau 6.13 : Tableau de répartition de X pour n=256 et N=2500

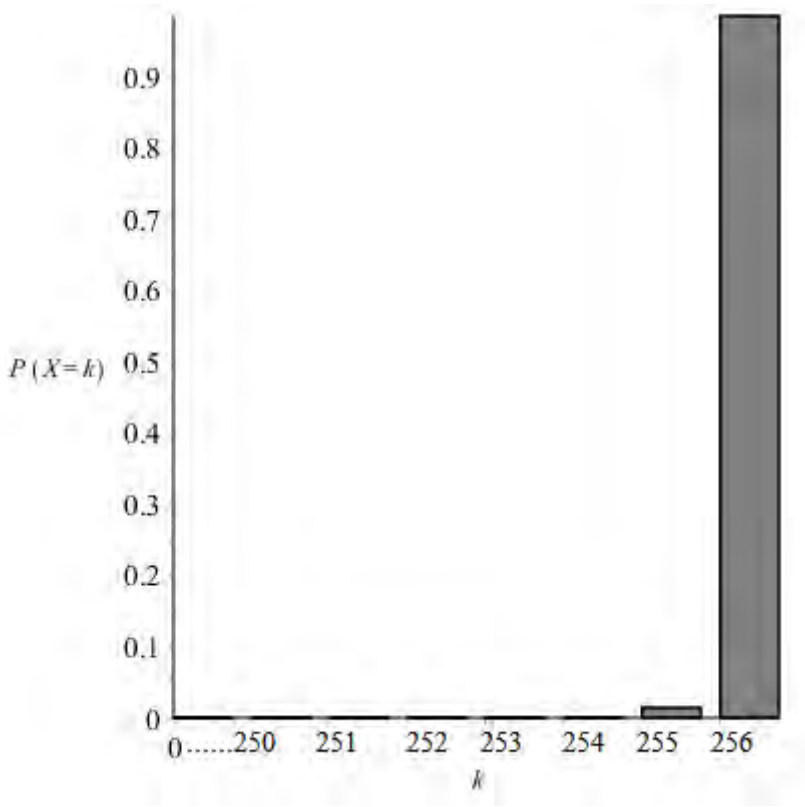

Figure 6.9 : Diagramme de répartition des valeurs (n=256 ; N=2500)

6.2.3.1 Interprétation générale des diagrammes obtenus-conclusions

- Distribution de probabilité découlant de l'étude des suites de longueurs (N), petites ou moyennes, obtenues à partir d'alphabets de symboles n = 26, 27, 31, 32, 256

Nous entendrons par suite de petites longueurs, des suites de longueurs N tel que (N<50). Les suites de longueurs moyennes sont des suites de longueurs N tel que ($50 \leq N \leq 100$).

Les diagrammes obtenus sont symétriques et les valeurs moyennes sont très proches des valeurs les plus probables. Cependant, les valeurs des probabilités sont assez faibles, voire négligeables pour les valeurs extrêmes (à gauche et à droite) de (k).

Si on se place alors par rapport à la moyenne, le nombre de présences auquel on s'attend, ne s'en écarte pas. Les valeurs qui sont susceptibles d'être obtenues, peuvent être résumées comme suit :

Alphabet utilisé	26	27	31	32	256
Valeurs moyennes des présences	22	23	25	25	254
Valeurs les plus probables	22	23	25	26	255

- Distribution de probabilité découlant de l'étude des suites de grandes longueurs (*N*), obtenues à partir d'alphabets de de symboles n = 26, 27, 31, 32, 256.

Nous entendrons par suite de grandes longueurs, des suites de longueur N > 100 symboles (grammatiques ou binaires). Quel que soit le nombre de symboles de l'alphabet utilisé, il ressort des diagrammes obtenus que les probabilités des différentes présences (k = 1, 2, 3, ….. n) sont très petites, parfois pratiquement nulles pour les valeurs extrêmes gauches de k (nombre de présences) et assez élevées pour les grandes valeurs de (k). Avec les grandes valeurs, nous aurons donc tendance à obtenir tous les symboles de l'alphabet utilisé (cas par exemple de *N*=1250, 2500 octets). Il est noté une concentration des courbes pour les grandes valeurs de (k).

Les valeurs qui sont susceptibles d'être obtenues, peuvent être résumées comme suit :

Alphabet utilisé	26	27	31	32	256
Valeurs moyennes des présences	25	26	30	30	256
Valeurs les plus probables	26	27	30	31	256

REFERENCES

[1] J., Neveu, Bases mathématiques du calcul des probabilités, Masson et Cie, 1964

[2] C., Berge, Principes de combinatoire, DUNOD, 1968

[3] B., Vauquois, Probabilités, Hermann, 1969

[4] L., Comtet, Analyse combinatoire, Tomes 1 et 2, PUF 1970

[5] G. Cullmann, M. Denis-Papin, A.Kaufmann, Cours de calcul informationnel appliqué, Editions Albin Michel 1970

[6] Y., Rozanov, processus aléatoire, Edition Mir, Moscou, 1975

[7] H., Ventsel, Théorie des probabilités, Editions Mir, Moscou 1973, 1977

[8] R., Oppliger, Contemporary Cryptography, Artech House, Computer security sciences, Artech House, pp 103-123, 2005.

[9] J., Hoffstein, J., Piper, J., H., Silverman, n Introduction to mathematical cryptograhy, Springer, 2008, pp 203-257.

[10] C., E., Pfister, Théorie des probabilités, Presses Polytechniques Universitaires Romandes (PPUR) Fev 2012.

[11] R. C., Dalang, D., Conus, Introduction à la théorie des probabilités, PPUR, jan.2015

Chapitre 7

Conception d'un distingueur d'aléas basé sur la loi des présences

Sommaire

7.1 Rappels

D'après la théorie de Blum, Goldwasser, Micali et Yao (avec les extensions de Ballet Stéphane et Robert Rolland), signalée au chapitre 3 [1-4], *« une distribution est pseudo-aléatoire si aucune procédure efficace ne peut la distinguer de la distribution uniforme, où des procédures efficaces sont associées à des algorithmes polynomiaux (probabilistes) ».* Cette notion de pseudo-aléatoire (l'indistinguabilité) est la plus appropriée, et nous avions indiqué que nous la retenons tout au long de nos travaux.

En outre, d'après 3.4.2.2, un PRBG est cryptographiquement sûr si aucun algorithme PPT ne peut distinguer la suite binaire qu'il fournit d'une vraie séquence de bits aléatoires de même longueur.

En effet, étant donnés deux espaces de probabilité :

$X = \{X_n\} = \{X_{n \in \mathbb{N}}\} = \{X_1, X_2, X_3, ...\}$ et $Y = \{Y_n\} = \{Y_{n \in \mathbb{N}}\} = \{Y_1, Y_2, Y_3, ...\}$

(X) est indistinguable en temps polynomial de (Y) si pour tout algorithme (D) en temps polynomial probabiliste (PPT) et tout polynôme (p), il existe un entier $N \in \mathbb{N}^+$, tel que $\forall\, n > N$:

$$\left| \Pr_{x \in X_n} [D(x)=1] - \Pr_{x \in Y_n} [D(x)=1] \right| \leq \text{negl}(n)$$

(une fonction négligeable telle que définie en 2.4.2.3).

Autrement dit, $\forall x$ suffisamment grand, aucun algorithme PPT, (D), ne peut permettre d'affirmer s'il a été échantillonné à partir de X_n ou Y_n. L'algorithme PPT, (D), est appelé ***« test statistique en temps polynômial (Polynomial-time statistical test) » ou distingueur (distinguisher)***.

Enfin, d'après 3.4.3, la qualité d'un générateur pseudo aléatoire est également évaluée en utilisant des tests statistiques [5-12], ***des distingueurs***, qui permettent de statuer sur **l'uniformité ou non des suites chiffrantes de sortie,** et partant si le générateur possède ou non des faiblesses.

7.2 Critère des présences

7.2.1 Définition

L'exploitation de ces résultats et de ceux obtenus au chapitre 6, permettent d'établir la définition suivante :

DEFINITION : « Une suite de longueur (*N*) délivrée par un générateur d'aléa, à partir d'un alphabet de (*n*) symboles, est dite ***uniforme***, si le nombre de présences (*k*) ou nombre de symboles obtenus après suppression des répétitions des symboles dans la suite, est une variable aléatoire qui suit la loi de probabilité définie par le théorème des présences ».

Ce concept d'uniformité recouvre deux notions essentielles, la première impliquant la seconde sur laquelle nous allons nous appuyer pour établir un distingueur d'aléa :

- L'équiprobabilité et l'indépendance des symboles
- Le nombre de présences.

7.2.2 Le nombre de présences

Le nombre de présences est une variable aléatoire dont nous connaissons la loi de probabilité ainsi que toutes les caractéristiques et les grandeurs qui lui sont associées (espérance mathématique, variance, écart-type, mode et médiane).

Cette loi étant déterminée et tabulée pour *N* (longueur de la suite uniforme) et *n* (nombre de symbole de l'alphabet) donnés, il est à présent possible, étant donnée une suite, de nous prononcer sur son caractère uniforme.

A cet effet, nous allons nous baser sur les critères des présences et construire le distingueur d'aléa.

7.2.3 Critères des présences

Selon les courbes de distribution de la loi des présences, dans l'hypothèse d'une suite uniforme, le nombre de présences le plus probable ne s'écarte pas trop de la moyenne, qui se situe, en général, du côté des grandes valeurs :

- Pour les suites de longueurs (*N*), petites ou moyennes, obtenues à partir d'alphabets de symboles $n = 26, 27, 31, 32, 256$, les courbes de distribution présentent une parfaite symétrie et la valeur la plus probable est très proche de la valeur moyenne des présences.
  Autrement dit « *pour les suites uniformes et de longueurs (N) petites ou moyennes, le nombre de présences ne doit pas trop s'écarter de la valeur moyenne des présences* ».
- En revanche, pour les suites de grandes longueurs (*N*), obtenues à partir d'alphabets de de symboles $n = 26, 27, 31, 32, 256$, les courbes de distribution présentent une très forte concentration du côté des grandes valeurs.
  Autrement dit « *pour les suites uniformes et de longueurs (N) assez élevées, le nombre de présences doit se situer du côté des grandes valeurs de (k), et il doit être égal ou supérieur à la valeur moyenne des présences* ».

Quel que soit le cas, l'écart entre le nombre de présences (k) et la valeur (m) est assez faible :

$$| k - m| \leq \varepsilon_0 \qquad (1)$$

Où ε_0 majore l'écart, en valeur absolue, entre (k) et (m).
Toute suite uniforme doit donc respecter ces critères.

7.3 Test des présences – Distingueur d'aléa

Compte tenu des critères précités, un attaquant en présence d'une suite délivrée par un générateur d'aléa doit être en mesure de valider ou d'infirmer le caractère uniforme de cette suite.

Pour ce faire, il pourra tester une hypothèse nulle H_0 contre une hypothèse alternative H_1 [cf ; chapitre 3, paragraphe 3.4.3] pour se prononcer sur l'uniformité ou non de la suite, si elle provient d'un générateur aléatoire ou non [5-15] :

- H_0 (*Hypothèse nulle*) : « La suite est uniforme : le nombre de présences se situe du côté des grandes valeurs ; il ne s'écarte pas de façon significative de la moyenne».
- H_1 (*Hypothèse alternative*) : «La suite n'est pas uniforme : le nombre de présences se situe du côté des petites valeurs ; il s'écarte de façon significative de la moyenne et cet écart observé ne peut être attribué au hasard ».

Il en résulte que pour affirmer l'uniformité d'une suite donnée, et par conséquent la qualité cryptographique du générateur qui l'a délivrée, il faut que l'hypothèse H_0 soit vraie ; sa défaillance constitue une présomption défavorable sur le caractère aléatoire du générateur qui a produit la suite.

7.3.1 Distribution symétrique

Pour une distribution symétrique de la loi des présences, les probabilités d'obtenir les valeurs extérieures sont assez faibles. Mais, la courbe présente une forte concentration du côté des grandes valeurs, et il n'est pas impossible, pour les suites de longueurs (N) petites ou moyennes, d'obtenir les grandes valeurs extrêmes bien que leurs probabilités d'apparition soient assez faibles.

Nous pouvons alors nous fixer une probabilité (α), qui définit le seuil de signification ou degré de confiance, telle que la probabilité pour que le nombre de présences (k) se trouve à l'extérieur de la région des grandes valeurs (région d'acceptation de l'hypothèse H_0) soit inférieure ou égale à (α).

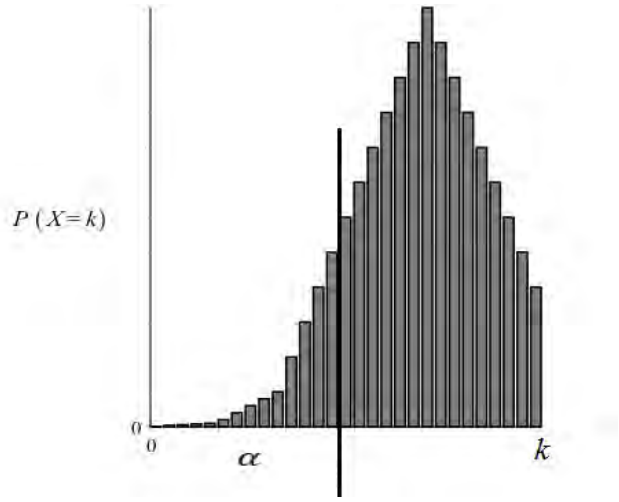

Figure 7.1 : Distribution symétrique

7.3.2 Distribution non-symétrique

Pour une distribution symétrique, la moyenne se situant du côté des grandes valeurs extrêmes, nous pouvons dès lors, nous fixer un seuil de signification (α) telle que la probabilité pour que le nombre de présences (k) se trouve dans la région des faibles valeurs soit inférieure ou égale à (α).

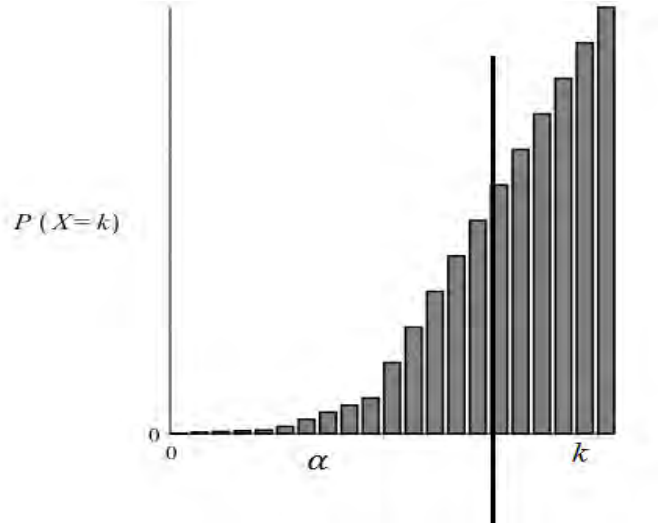

Figure 7.2 : Distribution non-symétrique

7.3.3 Régions d'acceptation et de rejet du test

Dans tous les cas, le problème reste le même aussi bien pour la distribution symétrique que pour la distribution non-symétrique : il s'agit de choisir sous l'Hypothèse nulle H_0 un domaine d'acceptation notée :

$$A_\alpha = \{X / X \geq k_0\} \qquad (2)$$

et qui est la région des grandes valeurs des présences ou l'ensemble des valeurs supérieures ou égales à la valeur (k_0) que l'on détermine en fixant α. En fixant α, A_α peut être déterminé grâce aux tables de probabilités pour N et n fixés.

Nous rejetterons H_0 pour les petites valeurs de X c.-à-d. à un seuil α dès que le nombre de présences n'appartient pas à la région des grandes valeurs.

Il s'agit alors de tester :

$$H_0 : X=k \geq k_0 \qquad \text{contre} \qquad H_1 : X=k < k_0 \qquad (3)$$

La région de rejet ou région critique du test, que nous notons R_α au seuil α, est telle que :

$$R_\alpha = \{X / X < k_0\} \qquad (4)$$

Avec la probabilité critique :

$$P_{r_{H_0}}(R_\alpha) \leq \alpha \qquad (5)$$

Nous rejetterons l'hypothèse nulle H_0 dès que la probabilité sous H_0 de la région de rejet est inférieure ou égale au seuil α. La région de rejet R_α doit être choisie de telle façon qu'elle ait une très faible probabilité de contenir *X*.

Les règles de décision sont alors établies comme suit :

- <u>*Décision* 1</u> :
  Décider H_1 si le nombre de présences appartient à R_α «*La suite n'est pas uniforme : le nombre de présences se situe des côté des petites valeurs ; il s'écarte de façon significative de la moyenne et cet écart observé ne peut être attribué au hasard* ».

- <u>*Décision* 2</u> :
  Décider H_0 si le nombre de présences appartient à A_α « *La suite est uniforme : le nombre de présences se situe du côté des grandes valeurs ; il ne s'écarte pas de façon significative de la moyenne* ».

7.3.4 Risques d'erreurs

7.3.4.1 Erreur de première espèce

Adopter la décision 1, revient à accepter l'hypothèse alternative H_1 selon laquelle la suite n'est pas uniforme, et à rejeter, par conséquent, l'hypothèse nulle H_0. Nous admettons que nous pouvons commettre une erreur dite « *erreur de première espèce* » si l'hypothèse H_0 est vraie ; nous l'avons notée α : c'est le seuil de signification ou la probabilité de rejeter l'hypothèse nulle H_0 lorsque celle-ci est vraie, de déclarer que la suite n'est pas uniforme alors qu'elle l'est.

$$P_r\,(H_0\ rejetée/H_0 \text{ vraie}) = P_{r_{H_0}}(R_\alpha) \leq \alpha \qquad (6)$$

7.3.4.2 Erreur de deuxième espèce

Adopter la décision 2, revient à rejeter l'hypothèse alternative H_1, et à accepter, par conséquent l'hypothèse nulle H_0 selon laquelle la suite est uniforme. Si la suite n'est pas uniforme, nous commettons, en ce moment, une erreur dite « *erreur de deuxième espèce* » car nous rejetons à tort ; nous la notons β.

$$\beta = P_r\ \{H_1 \text{rejetée} / H_1 \text{ vraie}\} \qquad (7)$$

$$\beta = P_r\ \{A_\alpha / H_1 \text{ vraie}\} = P_{r_{H_1}}(A_\alpha) \qquad (8)$$

Dans ce qui va suivre, nous verrons que pour différentes valeurs décroissantes de α (l'erreur de première espèce), l'intervalle d'acceptation A_α de l'uniformité augmentera et celui de rejet R_α diminuera ; par contre, l'erreur de deuxième espèce β augmentera. En cherchant alors à minimiser α, on maximisera β.

		H_0 (hypothèse nulle)	H_1 (hypothèse alternative)
Décisions du test	Rejet de H_0	α (Risque de 1ère espèce)	$1-\beta$ (Puissance du test)
	Acceptation de H_0	$1-\alpha$	β (Risque de 2ème espèce)

Tableau 7.1 : Résumé des risques d'erreurs

$1-\alpha$ / α / N	n	95% 5%	99% 1%
25	26	14-26	13-26
50	26	20-26	19-26
100	26	23-26	22-26
50	27	19-27	18-27
100	31	27-31	26-31
100	32	28-32	27-32
200	26	25-26	25-26
1250	256	251-256	250-256
2500	256	255-256	255-256

Tableau 7.2 : Quelques intervalles d'acceptation A_α

7.3.5 Construction du distingueur

Comme indiqué au paragraphe 7.1, notre distingueur ***D***, qui nous permet de statuer sur l'uniformité ou non des suites chiffrantes de sortie d'un générateur, et partant sur la qualité de ce générateur, est un test d'hypothèse c-à-d une règle qui, pour un échantillon de suite chiffrante donné ($X_1, X_2, \ldots, X_N$) nous permet de décider si nous acceptons ou rejetons H_0.

Et, pour la décision 1 et la décision 2, il est plus commode de noter respectivement :

D ($X_1, X_2, \ldots, X_N$) =1 si nous rejetons H_0 (9)

D ($X_1, X_2, \ldots, X_N$) =0 si nous acceptons H_0 (10)

Nous pouvons, dès lors, écrire l'algorithme qui suit :

1. Choisir un échantillon de suite chiffrante (X_N)= ($X_1, X_2, \ldots, X_N$) de sortie du générateur, à tester.
2. En fonction de la longueur (N) de la suite et de la taille (n) de l'alphabet utilisé, Choisir le tableau de répartition de la variable aléatoire « Tabulation de la loi des présences ».
3. Calculer le nombre de présences (k).
4. Fixer un seuil de signification $\alpha \in [5\%, 1\%]$
5. Déterminer la région d'acceptation $A_\alpha = \{X / X \geq k_0\}$ et, la région de rejet R_α tel que $P_{r_{H_0}}(R_\alpha) \leq \alpha$
6. Fixer les règles de décision d'acceptation ou de rejet du caractère uniforme de la suite chiffrante :
   - H_0 (*Hypothèse nulle)* : La suite est uniforme (*décision 2)*
   - H_1 (*Hypothèse alternative*) : La suite n'est pas uniforme (*décision 1)*
7. Si $X=k < k_0$, écrire ***D*** ($X_1, X_2, \ldots, X_N$) =1 : H_0 est rejetée
   Sinon, écrire ***D*** ($X_1, X_2, \ldots, X_N$) =0 : H_0 est acceptée (la suite est indistinguable d'une suite uniforme).

7.3.6 Exemples d'application

Exemple 1

1. Soit à tester l'uniformité de la suite suivante (X_N), délivrée par un générateur d'aléa :

D	V	I	A	L	A	V	L	X	M	M	A	L	I	M	R	N	N	U	N
I	T	A	M	N	R	U	O	N	R	S	T	U	S	S	S	H	V	E	N
O	N	E	H	T	D	D	F	M	V	M	M	I	R	Y	A	V	T	S	I
E	A	U	P	E	S	D	D	V	N	S	R	L	I	M	I	P	S	E	I
T	E	V	L	U	J	P	U	S	D	N	E	D	O	A	N	R	P	O	I

2. N= 100 et n=26 (la loi est déjà tabulée : cf. Tableau 6.7)
3. k=19
4. Seuil de signification α =5%
5. Région d'acceptation A_α= [23 -26] avec $P_{r_{H_0}}(k < 23)$ =6.703 10^{-4} < 5%
6. Règles de décision :
   - H_0 (*Hypothèse nulle) : X=k* $\geq$ 23
   - H_1 (*Hypothèse alternative*) : *X=k* < 23

7. k =19 < 23 $\Rightarrow$ ***D*** ($X_1, X_2, \ldots, X_N$) =1 : H_0 est rejetée au motif que la suite n'est pas uniforme. Le générateur l'ayant produite présente des faiblesses, et ne peut être considéré comme un générateur aléatoire.

Exemple 2

1. Etudions l'uniformité de cette autre suite (X_N) suivante :

E	W	U	X	L	P	O	L	Q	K	O	I	H	A	V	C	F	Z	J	O
P	M	B	Z	R	U	J	Q	Y	C	F	L	M	B	S	K	O	T	C	K
P	L	X	Z	U	R	A	T	Q	N	W	R	G	M	W	O	E	I	B	S
F	Q	X	E	R	H	G	Z	Y	H	G	M	S	I	F	V	R	D	P	G
U	V	U	N	Y	R	B	Z	Q	D	M	G	T	N	Y	Q	J	K	J	P
K	V	N	H	D	Q	Y	Y	Q	L	T	Z	C	T	B	E	A	H	C	O
P	J	R	T	H	R	U	A	I	V	D	X	X	A	X	N	E	W	B	U
W	V	L	Q	M	I	M	F	G	E	C	A	O	S	K	U	P	G	C	B
P	J	E	Z	T	L	E	R	J	C	I	N	D	C	O	K	D	H	Q	Y
M	B	A	F	S	N	V	A	I	N	S	D	T	X	O	L	I	G	F	W

2. N= 200 et n=26 (cf. Tableau 6.8)
3. k=26
4. Seuil de signification α =1%
5. Région d'acceptation A_α= [25-26] avec $P_{r_{H_0}}(k < 25)$ <1%
6. Règles de décision :
   - H_0 (*Hypothèse nulle) : X=k* $\geq$ 25
   - H_1 (*Hypothèse alternative*) : *X=k* < 25
7. Les calculs conduisent à conclure que ***D*** ($X_1, X_2, \ldots, X_N$) =0 : H_0 est acceptée.

Nous concluons alors que le nombre de présences tombe dans la région d'acceptation qui est celle des grandes valeurs, et rejetons l'hypothèse alternative de non-uniformité au profit de l'hypothèse nulle d'uniformité. Nous ne pouvons distinguer la suite chiffrante d'une suite uniforme, aléatoire.

Exemple 3 : D'une manière générale, une suite binaire (X_N), de longueur N, de sortie d'un générateur, peut être divisée en blocs de *u- bits* obtenus à partir d'un alphabet de $n= 2^u$ symboles (Si $n \nmid N$ les derniers bits de la suite étant oubliés). Le problème peut alors consister à étudier l'uniformité de la suite binaire (X_N) par rapport à la longueur (u) des blocs, et de se prononcer, par conséquent, sur le meilleur *u- bits* à choisir pour la qualité du générateur.

Etudions la suite binaire (X_N) suivante, en fixant par exemple : *u= 4, 5, 8*

0 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 0
0 1 1 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0

1er cas : *u=4*

1. N= 100 et $n= 2^4$ =16 (la suite est divisée en 25 blocs de 4 bits, sans perte de bits à la fin)
2. k= 11
3. Seuil de signification α =1%
4. Région d'acceptation A_α= [14-16] avec $P_{r_{H_0}}(k < 14) = 5{,}3593 \ \ 10^{-7} \ <1\%$
5. Règles de décision :
   - H_0 (*Hypothèse nulle) : X=k* ≥ 14
   - H_1 (*Hypothèse alternative*) : *X=k* < 14
6. *K=11* $< 14 \Rightarrow \boldsymbol{D}$ ($X_1, X_2, \ldots, X_N$) =1 : *La suite n'est pas uniforme* (aléatoire ou pseudo aléatoire), et le générateur a des faiblesses.

2ème cas : *u=5*

1. N= 100 et $n= 2^5$=32 (la suite est divisée en 20 blocs de 5 bits, sans perte de bits à la fin)
2. k= 16
3. Seuil de signification α =1%
4. Région d'acceptation A_α= [27-32] avec $P_{r_{H_0}}(k < 27) = 0{,}004834 \ <1\%$
5. Règles de décision :
   - H_0 (*Hypothèse nulle) : X=k* ≥ 27
   - H_1 (*Hypothèse alternative*) : *X=k* < 27
6. k =16 $< 27 \Rightarrow \boldsymbol{D}$ ($X_1, X_2, \ldots, X_N$) =1 : *La suite n'est pas uniforme*.

3ème cas : *u=8*

1. N= 100 et $n= 2^8$=256 (la suite est divisée en 12 blocs de 8 bits, avec une perte de 4 bits à la fin)
2. k= 12
3. La région d'acceptation est celle des grandes valeurs selon la taille de l'alphabet avec un seuil de signification de α =1%, et le nombre de présences est en dehors de la région d'acceptation A_α, ce qui conduit à affirmer que, là également, *la suite n'est pas uniforme*.

Dans les trois (3) cas, la suite binaire (X_N) n'est pas uniforme quelle que soit la longueur (u) des blocs choisie.

Exemple 4 : **Etude comparative du distingueur des présences et des tests de NIST**

La suite (X_N) suivante a passé le test Monobit et le test de fréquence par blocs de NIST [8] (cf. page 23), et a été déclarée aléatoire (au seuil de signification $\alpha = 1\%$) :

1 1 0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1 1
0 0 0 0 1 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1 1 1 0 0 0

Si nous utilisons notre distingueur basé sur la loi des présences, en considérant des blocs de *u*-bits de longueurs 3, 4, et 5, il vient les résultats suivants :

1er cas : *u=3*

1. *N*= 100 et *n*= 2^3 =8 (la suite est divisée en 33 blocs de 3 bits, avec perte d'un bit à la fin)
2. *k*= 8
3. Seuil de signification α =1%
4. Région d'acceptation A_α= [7-8] avec $P_{r_{H_0}}(k < 7) = 1{,}2702 \ \ 10^{-5} \ <1\%$
5. Règles de décision :
   - H_0 (*Hypothèse nulle) : X=k* $\geq$ 7
   - H_1 (*Hypothèse alternative*) : *X=k* < 7
6. $\Rightarrow$ ***D*** (*X1, X2, …, X N*) =0 : *La suite est déclarée uniforme*.

2ème cas : *u=4*

Si on se réfère aux calculs du 1er cas de l'exemple 3 ci-dessus, *la suite est déclarée non-uniforme* au seuil de signification 1% (nombre de présences *k=13*).

3ème cas : *u=5*

D'après le 2ème cas de l'exemple 3, *la suite est également déclarée non-uniforme* au seuil de signification 1% (nombre de présences *k=16*).

Dans ces trois derniers cas, il est d'ailleurs aisé de remarquer que les blocs ne sont pas équirépartis, certains blocs étant plus fréquents que d'autres.

Conclusion :

Il ressort de ce qui précède que ce test est indépendant de ceux du NIST qu'il est donc tout à fait possible qu'une suite binaire passe les tests du NIST et qu'elle soit rejetée par la loi des présences.

En effet, avec la loi des présences, la suite binaire (X_N) précitée est jugée bonne pour des blocs de longueur *u=3*, les résultats obtenus étant les mêmes que ceux issus des tests de NIST. Mais, pour les blocs de longueurs plus grandes (*u=4, 5*), la suite est jugée non-uniforme. Notre test d'aléa est, dans ces cas, plus sévère que ceux du NIST puisqu'il rejettera le plus souvent le caractère aléatoire ou pseudo aléatoire des suites, ce qui n'est pas mauvais si l'on veut être sûr de disposer de suites de suites chiffrantes de bonne qualités.

Tout dépend évidemment des longueurs de suites binaires testées :

- Pour des suites de longueur moyenne ($N \leq 100$), on peut faire le test en choisissant des tailles de blocs u=3 au seuil 1%, avec la possibilité d'obtenir des résultats qui ne s'écartent pas trop de ceux du NIST. En revanche, si on veut être plus sévère par rapport au caractère uniforme des suites, on peut choisir des tailles de blocs plus grandes (par exemple *u=4 ou 5*).

- Pour les suites binaires de grandes longueurs ($N > 100$; cf. Tableau 7.2) on peut faire le test en choisissant des tailles de blocs (par exemple $u=4, 5, 8$)

7.4 Conclusion

Sur la base de la loi de probabilité des présences (cf. Théorème des présences), nous avons établi un distingueur basé sur la loi des présences, qui nous permet de statuer sur l'uniformité ou non des suites chiffrantes, et partant d'évaluer la qualité cryptographique des générateurs d'aléas qui les produisent.

Il s'agit d'une bonne nouvelle méthode de test statistique d'aléa pour distinguer une distribution donnée d'une distribution aléatoire (ou pseudo aléatoire).

Dans la pratique, ce test peut être utilisé seul, de façon indépendante des tests du NIST [4-11] (comme il peut aussi être associé à ces tests).

REFERENCES

[1] S. Goldwasser and S. Micali. Probabilistic Encryption. Journal of Computer and System Science, Vol. 28, No. 2, pages 27G--299, 1984. Preliminary version in 14th A CM Symposium on the Theory of Computing, 1982.

[2] M. Blum and S . Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits . SIAM Journal on Computing, Vol. 13, pages 850-864, 1984. Preliminary version in 23rd IEEE Symposium on Foundations of Computer Science, 1982.

[3] A. C. Yao. Theory and Application of Trapdoor Functions. In 23rd IEEE Symposium on Foundations of Computer Science, pages 8Q-9 1, 1982.

[4] Ballet Stephane and Robert Rolland, A note on a Yao's theorem about pseudo random generators. Cryptography and Communications, Vol.3 N.4 (2011), Page 189-206 doi: 10.1007/s12095-011-0047-1 Springer, 2011.

[5] Rolf OPPLIGER, Contemporary Cryptography, Artech House Computer security sciences

(pp: 25-219-329)

[6] P. Liardet, A. Bonnecaze, Randomness and cryptography with a dynamical point of view

juin 2013 (pierre.liardet/alexis.bonnecaze @univ-amu.fr).

[7] S.W. Golomb et L.R. Welch, Shift register sequences, Aegean Park Press Laguna Hills, CA, 1967, 1982.

[8] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo: A Statistical Test Suite for random and pseudo-random number generators for cryptographic applications, NIST April 2010.

[9] Benjamin Pousse, Design et cryptanalyse de chiffrement à flot, Thèse Université de Limoges, Décembre 2010,

[10] G., Marsaglia: DIEHARD: A battery of tests of randomness, 1996 http://stat.fsu.edu/~geo/diehard.html

[11] R. Santoro, vers des générateurs de nombres aléatoires uniformes et gaussiens à haut débit, Thèse 11 Janvier 2010, Université de Rennes 1.

[12] P. Lacharme, Générateurs vraiment aléatoires dans un composant sécurisé, Thèse, Université de Toulon et du Var.

[13] A. Vessereau, La statistique, PUF, 1976

[14] L. Lebart, J.P. Fenelon, Statistique et Informatique appliquées, ISUP, Bordas, 1975.

[15] R. Cehessat, Exercices commentés de statistique et informatique appliquées, Bordas, 1976.

Chapitre 8

Conclusion-Perspectives

Nous avons d'abord examiné, dans la première partie de la thèse, la construction des générateurs d'aléas (RBG, PRBG, générateurs hybrides), les conditions de leur réalisation et de leur implantation matérielle et logicielle ainsi que les problèmes de sécurité cryptographique y afférents.

Dans la deuxième partie, nous avons présenté les travaux que nous avons réalisés :

I°) L'étude du problème de la réciprocité dans la substitution bigrammatique de Delastelle, qui a abouti à l'établissement du « *Théorème de la réciprocité* ».

Comme nous l'avons précisé au chapitre 4, ce travail de recherche en cryptologie, qui est d'un très bon niveau scientifique, présente notamment un intérêt didactique et épistémologique ;il peut être enseigné dans le cadre du cours de cryptographie classique, au même titre que le procédé de Playfair et le chiffrement matriciel de Lester Hill. Il s'agit également d'un procédé de chiffrement réciproque d'un très bon niveau de sécurité, qui, dans la pratique, peut être utilisé, en chiffrement classique, pour protéger de courts messages (si la clé associée, et qui est fournie par un générateur d'aléas, est utilisée une et une seule fois).

II°) La construction des registres à décalage à rétroaction linéaire (LFSR) bouclés à période maximale (Polynômes primitifs et relations de récurrences linéaires- PRBG basés sur les registres à décalage).

Les FSR constituent l'élément de base des générateurs pseudo-aléatoires utilisés pour générer les chaînes cryptographiques qui sont utilisées dans les systèmes de chiffrement en continu ou par flot (stream cipher).

Nous avons proposé une méthode qui permet de déterminer mathématiquement, à partir du polynôme primitif, la relation récurrente linéaire générant le LFSR bouclé à période maximale (et vice versa) et qui facilite ainsi sa construction ; elle aide également à établir le polynôme primitif réciproque correspondant qui donne la possibilité de construire un autre LFSR aussi bon que le premier.

Nous avons une conception et un choix judicieux des LFSRs bouclés à période longueur maximale, à utiliser, sur la base des polynômes primitifs, des polynômes primitifs réciproques et des relations de récurrence linéaire associées, ce que ne montrent pas les méthodes utilisées jusqu'ici où les récurrences sont établies, sans plus de précisions, à partir des polynômes primitifs pour dessiner les LFSRs.

D'autre part, il a semblé important de passer en revue les LFSRs, d'insister sur leur sécurité cryptographique et d'indiquer certains problèmes ouverts dans le domaine des générateurs pseudo-aléatoires basés sur les registres à décalage à rétroaction non linéaires (NLFSR).

Comme indiqué au chapitre 5, les recherches sont orientées, à l'heure actuelle, vers de nouvelles classes de registres à décalage basés notamment sur les anneaux algébriques, ce qui implique de nouvelles méthodes de cryptanalyse et de nouvelles mesures de sécurité. En fait, des études importantes ont été menées dans le domaine des registres à décalage à rétroaction non linéaire (NLFSR), aussi bien du point de vue de la conception que des attaques, et qui ont conduit :

- Depuis 1993, aux FCSR (Feedback with Carry Shift Registers- Registres à décalage avec retenue) en mode de Fibonacci ou Galois, avec comme base mathématique, l'anneau des entiers de N−*Adic* au lieu de l'anneau des séries formelles utilisé pour les LFSR.

Cependant, il faut préciser qu'en dépit de leur grande complexité linéaire, ils sont sensibles aux attaques de " *l'algorithme d'approximation rationnelle*" qui est similaire à celui de Berlekamp-Massey. Néanmoins, ils pourraient être couplés aux LFSR dans la conception des générateurs pseudo-aléatoires.

- Aux registres à décalage avec retenue vectoriels (VFCSR), conception vectorielle des FCSR dont l'analyse a été étendue aux corps finis $GF(p^n)$.
- Aux FCSR filtrés (F-FCSR), conception des FCSRs pour contrer l'attaque par l'algorithme d'approximation rationnelle.
- Aux registres à décalage algébriques (AFSR) lorsque la base mathématique est l'anneau des entiers π−*Adic* (non spécifié, étant donné que les entiers π−*Adic* sont des généralisations des séries formelles et des entiers N−*Adic*). Les LFSR et les FCSR sont des cas particuliers des AFSR.

Nous avons aussi signalé qu'il est utile de prendre connaissance de la recherche sur la théorie de la stabilité des crypto systèmes en continu c'est-à-dire de la résistance de ces systèmes aux petites variations de certains de leurs paramètres en ce qui concerne en particulier la complexité linéaire et les fonctions booléennes non linéaires utilisées :

- Pour le chiffrement synchrone additif, il existe déjà des techniques de contrôle de la stabilité de la complexité linéaire. Mais, le problème de la stabilité de la complexité linéaire locale semble pour l'instant difficile à résoudre (*il s'agit d'un problème ouvert*).
- Pour les registres combinés non-linéaires et les registres filtrés non-linéaires, des résultats partiels ont été obtenus sur certains aspects de la théorie de la stabilité, mais les travaux doivent se poursuivre davantage : un domaine prometteur de la recherche.

Nous avons également indiqué que d'autres voies de recherche pourraient être explorées dans le domaine des études réalisées, en particulier, sur les espaces métriques et les séries.

Enfin, des recherches sont également menées sur les RNLUs (Registers with Nonlinear Update-Registres avec mise à jour non linéaire) qui sont une généralisation des NLFSRs dont l'étude est théorique et devrait être encore affinée.

III°) L'étude et l'établissement de la loi de probabilité dite « *Loi des présences* », qui a permis d'énoncer « *Le Théorème des présences* », dans l'hypothèse où nous disposons d'un générateur d'aléas qui délivre des suites chiffrantes uniformes, ainsi que la conception à l'aide de cette loi d'un *« distingueur d'aléa »*.

Comme mentionné au chapitre 7, sur la base de la loi de probabilité des présences, nous avons pu disposer d'une très bonne méthode de test statistique d'aléas à côté des tests existants (NIST, DIEHARD…) pour distinguer une distribution donnée d'une distribution aléatoire (ou pseudo aléatoire), et partant d'évaluer la qualité cryptographique des générateurs d'aléas.

Dans la pratique, ce test peut être utilisé seul, de façon indépendante des tests du NIST [4-11] (comme il peut aussi être associé à ces tests).

Les résultats importants de ces travaux originaux sur ce test statistique d'aléas pour les RBG et PRBG pourraient, s'ils sont retenus, être soumis aux organismes internationaux de standardisation comme le NIST (National Institut of Standards and Technology), afin que soient étudiée la possibilité de standardiser ledit test pour des applications cryptographiques.