N° d'ordre : 09 / PGS-2004/ MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE
USTHB - BAB EZZEOUAR – ALGER

FACULTE DE MATHEMATIQUES

Mémoire de Post-Graduation Spécialisée en Mathématiques

Spécialité : Cryptologie

Présenté par :

Monsieur : SAIDANI Kamal

Sujet :

Le Test Statistique Universel d'Ueli Maurer

Implémentation et Application

Soutenu le 24 / 03 / 2004

Devant le jury composé de :

M^r : KESSI. A ,	**Professeur, USTHB.**	**Président**
M^r : AISSANI Amar,	**Professeur, USTHB.**	**Directeur de Mémoire**
M^r : ZITOUNI Mohamed,	**Professeur, USTHB.**	**Examinateur**
M^r : KHALECHE Slimane,	**D.G.S.C.T.**	**Examinateur**

2004

Sommaire

Notations et abréviations

Symbole	Signification
s	La séquence binaire testée.
n	Nombre de bits de la séquence testée.
s_i	Le i^{eme} bit de la séquence s.
M	Le nombre de bits du bloc.
α	Le niveau de signification.
$log_2(x)$	$log_2(x) = \frac{ln(x)}{ln(2)}$.
σ^2	La variance de la variable aléatoire.
χ^2	La distribution de Khi-deux.
p-valeur	la probabilité qu'un générateur de nombres aléatoires parfait aurait produit une séquence moins aléatoire que la séquence qui a été évaluée.
erfc	La fonction complémentaire d'erreur qui est définie par ***erfc*** $(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-u^2} du$.
igamc	La fonction Gamma incomplète notée par ***igamc*** et définie par $Q(a, x) = \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} dt$, où $\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt$ (la fonction Gamma), $Q(a, 0) = 1$ et $Q(a, \infty) = 0$.

.

Abréviations	Définitions
ANSI	(**A**merican **N**ational **S**tandards **I**nstitute) [Institut National de Standards Américain]. Développe des standards via divers "comités de standardisation accrédités" [Accredited Standards Institute] (ASC). Le comité X9 s'intéresse particulièrement aux standards de sécurité pour l'industrie des services financiers.
X9.17	Spécification ANSI qui détaille la méthodologie de génération de nombres aléatoires et pseudo-aléatoires.
FIPS	(**F**ederal **I**nformation **P**rocessing **S**tandard) [Standard de traitement de données fédéral]. Standard gouvernemental américain publié par le NIST.
NIST	(**N**ational **I**nstitute for **S**tandards and **T**echnology) [institut national pour les standards et la technologie]. Division du département américain du commerce qui publie des standards ouverts d'interopérabilité appelés FIPS.

Chapitre 1

INTRODUCTION

1. INTRODUCTION

Aujourd'hui, la cryptographie est devenue une panacée et un vrai remède pour les problèmes de sécurité. La cryptographie qui est l'un des deux aspects de la cryptologie avec la cryptanalyse, étudie les techniques de camoufler les données, et la cryptanalyse étudie les manières de déjouer ces techniques**.**

L'analyse des protocoles cryptographiques repose grandement sur la théorie de l'information et la théorie des probabilités.

La sécurité de beaucoup de systèmes cryptographiques dépend de la génération des nombres aléatoires qui est une composante importante pour des algorithmes et des protocoles cryptographiques, elle est primordiale dans la génération des clefs de chiffrement comme la clé secrète dans l'algorithme de chiffrement DES (**D**ata **E**ncryption **S**ystem), les deux nombres premiers p et q dans l'algorithme de chiffrement RSA (du nom des personnes à qui on en a attribué le développement : Ronald **R**ivest, Adi **S**hamir et Leonard **A**dleman) et d'autres paramètres d'algorithmes cryptographiques comme la production de signatures digitales.

Les générateurs appropriés pour les applications cryptographiques nécessitent des exigences plus fortes que pour d'autres applications. En particulier les séquences produites doivent être d'une taille suffisante et aléatoires. Les tests statistiques permettent de s'assurer qu'un générateur produit en effet des nombres aléatoires.

Le but de ce mémoire est l'étude du test statistique universel de Maurer[1] qui est l'un des tests retenus par l'Institut National des Standards et de la Technologie (NIST[2]) et qui est capable de détecter une large gamme de défauts statistiques. Il détermine si la séquence testée peut être significativement compressée sans perte d'information et évalue la distribution des distances moyennes entre les formes occurrentes dans la séquence.

Ce mémoire s'attache à montrer l'utilité de la génération aléatoire dans les protocoles cryptographiques au chapitre 2, en présentant les différents types de générateurs

[1] Ueli Maurer qui est né en 1960, est un professeur en informatique et le chef de groupe de recherche en cryptographie et la sécurité de l'information à l'Institut Suisse Fédéral de Technologie (ETH), à Zurich.

[2] NIST (National Institute of Standards and Technology) : Institut de standardisation du gouvernement américain.

aléatoires et pseudo-aléatoires. Le chapitre 3 est consacré au principe des tests statistiques utilisés pour la validation des générateurs binaires à des fins cryptographiques, ce sont les tests figurant dans la plupart des normes de sécurité tels que NIST, FIPS…etc.

Dans le chapitre 4, nous présentons le test statistique universel d'Ueli Maurer avec une application au chapitre 5.

Chapitre 2

GENERATION ALEATOIRE DANS LES PROTOCOLES CRYPTOGRAPHIQUES

2. GÉNÉRATION ALÉATOIRE DANS LES PROTOCOLES CRYPTOGRAPHIQUES

La cryptographie a souvent recours à des nombres aléatoires. Ainsi, lorsqu'une personne génère une clef secrète ou privée, elle doit faire intervenir le hasard de façon à empêcher un adversaire de deviner cette clef.

L'outil de base de la génération de nombres aléatoires en système décimal en général consiste en une séquence de nombres aléatoires *indépendants* et *identiquement distribués* selon la loi uniforme sur l'intervalle [0,1] : $g_i \rightarrow U(0,1)$.

En pratique, on utilise des procédés physiques (mécaniques) ou algorithmiques permettant de produire des suites aléatoires ou pseudo-aléatoires.

Des tests sont mis à l'épreuve pour tester l'incohérence des générateurs de nombres aléatoires et pseudo-aléatoires qui peuvent être employés pour beaucoup de buts incluant la cryptographie, la modélisation et les applications de simulation. Le NIST suggère que ces procédures sont utiles dans la détection de l'incohérence des déviations d'une séquence binaire. Cependant, on doit noter que des déviations apparentes de l'incohérence peuvent être dues à des générateurs mal conçus ou aux anomalies qui apparaissent dans la séquence binaire qui est évaluée (c'est-à-dire, on attend un certain nombre d'échecs dans des séquences aléatoires produites par un générateur particulier).

Dans ce chapitre, nous décrivons les principes de base de la génération aléatoire.

2.1 Générateurs aléatoires

2.1.1 Définition

Un *générateur aléatoire* est un dispositif qui est conçu pour la production d'une séquence de variables binaires, aléatoires, statistiquement indépendantes et equidistribuées.

2.1.2 Exemples de générateurs aléatoires

Les nombres aléatoires sont crées avec des phénomènes physiques difficiles à prévoir comme :

- L'exemple le plus simple : utiliser une pièce de monnaie pour produire une suite de 0 ou de 1. Une séquence de bits aléatoires pourrait être interprétée comme le résultat du jet d'une pièce de monnaie parfaite avec les côtés qui sont étiquetés "0" et "1", et à chaque lancement on a une probabilité d'exactement ½ de production "de 0" ou "1". En outre, les lancements sont indépendants l'un de l'autre : le résultat de chaque lancement de la pièce de monnaie précédent n'affecte pas le lancement suivant de la pièce. La pièce de monnaie parfaite est ainsi le parfait générateur de suite de bits aléatoires, puisque les valeurs "0" et "1" seront aléatoirement et uniformément distribuées. Tous les éléments de la séquence sont produits indépendamment les uns des autres et la valeur de l'élément suivant dans la séquence ne peut pas être prévu, indépendamment de la façon où les autres éléments ont déjà été produits.
  Évidemment, l'utilisation de pièces de monnaie parfaites pour des buts cryptographiques est peu pratique.
- Utiliser une source radioactive : on compte le nombre de particules radioactives (à l'aide d'un compteur) détectées durant un temps Δt, si le compteur indique un nombre pair on pose $Y = 1$, si ce nombre est impair on pose $Y = 0$.
  Des expérimentations physiques montrent que la probabilité de détection d'un nombre pair de particules suit une loi de Poisson (de paramètre disons λ) ; on montre que :

$$\mathrm{P}(Y=1) = \sum_{k=0}^{\infty}\left[\frac{(\lambda\,\Delta t)^{2k}}{(2k)!}\right]e^{-\lambda\,\Delta t} = \frac{\left[1+e^{-2\lambda\,\Delta t}\right]}{2}$$

- Le bruit électronique : on utilise le niveau de bruit d'une valve électronique ; la valeur de voltage $u(t)$ est un processus aléatoire qu'on observe à des temps discrets $t_1, t_2, \ldots, t_q$. On choisit un niveau de section c de manière " adéquate " tel que $\mathrm{P}(Y=1)$ soit la plus proche possible de ½.

$$\text{On pose } Y = \begin{cases} 0 & si\ u(t) \le c \\ 1 & si\ u(t) > c \end{cases}$$

- Ecoulement de fluides (lava-lamps, particules dans un liquide en circulation, vent et tourbillons).

- Bruit blanc généré par un pont de diodes.
- Emission de particules issues de la décomposition de matières radioactives.

- D'autres générateurs utilisent les ressources logicielles : horloge du système, entrée de l'usager (e-mail), les statistiques du réseau ou du système d'exploitation.

- Techniques d'élimination du biais (de-skewing) : On peut utiliser ces techniques lorsqu'une source naturelle de chiffres binaires produit des chiffres biaisés (la probabilité que la source émette un 1 n'est pas égale à ½) ou corrélés (la probabilité d'émission d'un 1 dépend du chiffre émis précédemment).

2.2 Générateurs pseudo-aléatoires

2.2.1 Définitions

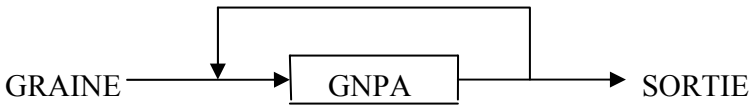
Une séquence de nombres $x_0, x_1, x_2, ...x_n$ est dite *pseudo-aléatoire* si elle est générée de façon *déterministe* mais semble avoir été produite de façon purement aléatoire (passe avec succès certains tests statistiques). Le terme x_n ($n > 0$) est le résultat d'un calcul (à définir) sur le ou les termes précédents. Le premier terme x_0 est appelé la *graine* ou le germe. Les opérations à faire pour calculer les termes de la séquence, ainsi que les différents paramètres de départ forment ce qu'on appelle un *générateur de nombres pseudo-aléatoires*.

Remarque : Dans des contextes pour lesquels l'imprévisibilité est nécessaire tels que la cryptographie, la graine elle-même doit être aléatoire (peut-être générée par un dispositif matériel) et imprévisible, car elle est essentiellement la clef pour les applications cryptographiques. Puisque dans de nombreux cas, l'algorithme de génération est publiquement disponible, la graine doit être tenue secrète et ne doit pas être dérivable de la séquence pseudo-aléatoire qu'il produit.

Les sorties d'un générateur de nombres pseudo-aléatoires sont des fonctions typiquement déterminées de la graine (voir la figure ci-dessous); c'est-à-dire, toute la vraie incohérence est limitée à la génération de la graine. La nature déterministe du processus mène au

terme"pseudo-aléatoire" puisque chaque élément d'une séquence pseudo-aléatoire est reproductible de sa graine.

PRINCIPE GENERAL



GNPA : générateur de nombres pseudo-aléatoires

2.2.2 Qualités requises d'un générateur pseudo-aléatoire

Un générateur pseudo-aléatoire doit présenter les caractéristiques suivantes :

- Le générateur doit calculer chaque nombre de la séquence rapidement, en utilisant un minimum possible de mémoire.
- La période de la suite produite par le générateur doit être suffisamment grande pour que les sous suites finies utilisées avec l'algorithme ou le protocole cryptographique ne soient pas périodiques.
- Ces sous suites doivent, sur le plan statistique, sembler aléatoires.
- Le générateur doit être imprévisible, au sens où il doit être impossible de prédire le prochain bit à partir des bits précédents.
- la méthode ne doit pas souffrir de faille grave : l'histoire des générateurs pseudo-aléatoires est pleine d'algorithmes qui se « coincent » lorsqu'ils arrivent sur un nombre particulier.

Remarque : On dit qu'un générateur pseudo-aléatoire est acceptable s'il passe avec succès toute une batterie de tests statistiques appropriés

2.2.3 Exemples de générateurs pseudo-aléatoires

- **ANSI[3] X9.17 :** utilisé surtout pour la génération de clefs et l'initialisation de vecteurs pour un usage dans DES.

  Algorithme :

  Entrée : une racine s à 64 bits aléatoire (et secrète), un entier m et une clef de codage k de type DES E-D-E.

  Sortie : m séries pseudo-aléatoires à 64 bits : $x_1, x_2, \ldots, x_m$.

  1. Calculer la valeur intermédiaire : $I = E_k(D)$, où D est une représentation à 64 bits du vecteur date/temps jusqu'à ce qu'une résolution fine soit disponible.

  2. Pour i =1 à m faire :

  2.1 $x_i \leftarrow E_k(I \oplus s)$.

  2.2 $s \leftarrow E_k(x_i \oplus I)$.

  3. Retour à $(x_1, x_2, \ldots, x_m)$.

- **Blum-Blum-Shub:** on l'appelle encore générateur BBS, générateur x^2 mod n ou générateur à résidu quadratique. Il est cryptographiquement sûr sous l'hypothèse de complexité du problème de factorisation entière. De plus il est à la base du schéma probabiliste de cryptage à clef publique de Blum-Goldwasser. Il génère une séquence de bits $z_1, z_2, \ldots, z_l$ de longueur l.

  1. Générer deux grands entiers premiers aléatoires (et distincts) p et q congrus à 3 mod 4. Calculer $n = pq$.
  2. Sélectionner un entier aléatoire s (la graine) dans l'intervalle [1, n -1] tel que PGCD (s, n) =1 et calculer $x_0 \leftarrow s^2 \bmod n$.
  3. Pour i =1 à l faire :

  3.1 $x_i \leftarrow x_{i-1}^2 \bmod n$.

  3.2 $z_i \leftarrow$ le bit le moins significatif de x_i.

  4. La séquence de sortie est $z_1, z_2, \ldots, z_l$.

[3] ANSI (American National Standards Institute) [Institut National de Standards Américain]. Le comité X9 s'intéresse particulièrement aux standards de sécurité pour l'industrie des services financiers.

Remarque (efficacité de BBS) **:** Il a été montré qu'à moins de factoriser n, on ne peut trouver les sorties présentes, passées ou futures du générateur. Sa sécurité dépend de la difficulté à factoriser n et non pas à sa conception algorithmique. Chaque bit pseudo-aléatoire z_i nécessite une élévation au carré.

- **Générateurs (linéaires) congruentiels (multiplicatifs, additifs)** : ce sont les plus courants (en simulation) où il n'y a pas de correspondance évidente entre les « graines » et les sorties, $x_n = (a\ x_{n-1} + c) \bmod m$; a, c, m des entiers qui dépendent en général de la machine utilisée.

  En général, la période $k \leq m$. Pour les générateurs multiplicatifs ($c = 0$), la période maximale est m-1. On a pour habitude de choisir m de telle manière que l'opération modulo soit efficace, et donc de choisir a et c pour rendre la période la plus longue possible.

  Remarque : Un générateur congruentiel a une période égale à m si et seulement si :

  I. PGCD(c, m) =1.
  II. $a \equiv 1 \bmod p$ pour chaque facteur premier p de m.
  III. $a \equiv 1 \bmod 4$ si 4 divise m.

- **Registres à décalage** : les registres à décalage (*shift registers* en anglais) et, en particulier, les registres à décalage à rétroaction linéaire (*Linear Feedback Shift Registers, LFSR)* sont souvent utilisés pour construire des générateurs pseudo-aléatoires, car il s'agit de composants très rapides et faciles à implémenter en hardware. L'idée de ces derniers registres est de commencer avec un registre rempli arbitrairement avec des 0 et des 1, puis de décaler cette suite d'un cran vers la droite. On remplira la position tout à gauche par la somme (modulo 2) du contenu des deux registres les plus à droite (avant le décalage). Ces registres présentent l'inconvénient de générer des suites linéaires, si bien que des grands nombres générés à partir de sous-suites sont fortement corrélés. C'est pourquoi les générateurs pseudo-aléatoires sont généralement construits en combinant, à l'aide d'une fonction non linéaire, plusieurs registres à décalage de tailles différentes.

  Pour cela il est nécessaire que la fonction de combinaison soit une ***fonction résiliente*** (i.e. une fonction sans-corrélation dont la sortie est uniformément distribuée). En effet,

s'il existe une corrélation entre la sortie et l'un des registres, il devient possible d'attaquer les différents registres séparément (attaque de Siegenthaler). Cette technique de cryptanalyse, introduite par Siegenthaler, est de type "diviser pour mieux régner" : elle consiste à retrouver l'initialisation de chacun des registres indépendamment des autres. Pour cela, on exploite l'existence d'une éventuelle corrélation entre la sortie du générateur pseudo-aléatoire et celle d'un des registres utilisés. Ce type de générateur est très utilisé par les algorithmes de chiffrement en continu.

On peut conclure à la fin de ce chapitre que les suites produites par des procédés physiques ont été longtemps (et sont toujours) acceptées comme aléatoires. Cependant ils sont coûteux et les suites obtenues ainsi présentent des biais et des dépendances. Bien que produisant également des nombres pseudo-aléatoires, les générateurs algorithmiques ont l'avantage d'être simples à réaliser sur ordinateur.

Seules les expérimentations peuvent confirmer l'aspect aléatoire des générateurs car des insuffisances sont détectées dans les générateurs algorithmiques, ainsi que des défaillances dans les procédés physiques. La nature de la sécurité peut faire opter pour ces derniers, mais c'est l'aspect économique qui explique la popularité des générateurs algorithmiques.

Chapitre 3

TESTS STATISTIQUES DE LA QUALITE DE GENERATEURS ALEATOIRES

3. TESTS STATISTIQUES DE LA QUALITE DE GÉNÉRATEURS ALÉATOIRES

Dans ce chapitre nous présentons les tests statistiques utilisés pour valider les générateurs binaires utilisés à des fins cryptographiques. Nous présentons d'abord le principe d'un test statistique d'hypothèse, ensuite nous décrivons les normes du NIST et FIPS.

3.1 Test d'hypothése

Définition 3.1.1: Un test d'hypothèse en statistique est utilisé comme une règle de décision entre deux hypothèses (*une hypothèse* est une conjecture ou supposition). Il sous-entend une inférence c'est à dire un raisonnement par lequel on admet une proposition en vertu de sa liaison avec d'autres propositions déjà tenues pour vraies. .

Ce test statistique est formulé pour évaluer une hypothèse nulle H_0 spécifique. Pour notre part nous considérons que l'hypothèse nulle est que la séquence évaluée est aléatoire. On associe à cette hypothèse nulle, l'hypothèse alternative (ou la contre hypothèse) (H_a), est que la séquence n'est pas aléatoire.

Pour chaque test appliqué, une décision ou une conclusion est tirée, on accepte ou on rejette l'hypothèse nulle, c'est-à-dire, si le générateur produit (ou ne produit pas) de valeurs aléatoires, basée sur la séquence qui a été produite.

Pour chaque test, une incohérence appropriée statistique doit être choisie et employée pour déterminer l'acceptation ou le rejet de l'hypothèse nulle. Sous une supposition d'incohérence, une telle statistique a une distribution de valeurs possibles. Une distribution de référence théorique de cette statistique sous l'hypothèse nulle est déterminée par des méthodes mathématiques.

De cette distribution de référence, une valeur critique est déterminée. Pendant un test, une valeur de test statistique est calculée sur les données (la séquence étant évaluée). Cette valeur du test statistique est comparée à la valeur critique. Si la valeur statistique évaluée excède la valeur critique, l'hypothèse nulle pour l'incohérence est rejetée. Autrement,

l'hypothèse nulle (l'hypothèse d'incohérence) n'est pas rejetée (c'est-à-dire, l'hypothèse est acceptée).

Donc, quand la valeur calculée du test statistique excède la valeur critique, la conclusion est faite que la supposition originale de l'incohérence est suspecte ou défectueuse. Dans ce cas, l'hypothèse statistique évaluée rend les conclusions suivantes: rejeter (l'incohérence) H_0 et accepter H_a (la non incohérence).

LA VRAIE SITUATION	**CONCLUSION**	
	Accepter H_0	Accepter H_a (rejeter H_0)
Les données sont aléatoires (H_0 est vraie)	Aucun Type d'erreur	Erreur Type 1
Les données ne sont pas aléatoires (H_a est vraie)	Erreur Type 2	Aucun Type d'erreur

Définition 3.1.2 : Si les données sont, en vérité, aléatoires, alors la conclusion de rejeter l'hypothèse nulle (c'est-à-dire, conclure que les données sont non aléatoires) est appelée *erreur Type 1*.

Définition 3.1.3: Si les données sont, en vérité, non aléatoires, alors une conclusion pour accepter l'hypothèse nulle (c'est-à-dire, conclure que les données sont en réalité aléatoires) est appelée *erreur Type 2*. La probabilité d'une erreur Type 2 est notée β.

Remarque 1: Les conclusions : accepter H_0 quand les données sont vraiment aléatoires et rejeter H_0 quand les données sont non aléatoires, sont correctes.

Définition 3.1.4 : La probabilité d'une erreur Type 1 est souvent appelée *le niveau de signification du test*. Cette probabilité peut être mise avant un test et elle est notée α. Typiquement α est choisi dans la gamme (0.001, 0.01]. Pour le test, α est la probabilité que le test indiquera que la séquence n'est pas aléatoire quand elle est vraiment aléatoire. C'est-à-dire une séquence semble avoir des propriétés non aléatoires même quand un " bon" générateur a produit cette séquence.

Remarque 2: Les valeurs communes de α dans la cryptographie sont environ 0.01.

Un des buts primaires des tests suivants doit réduire au minimum la probabilité d'une erreur de Type 2 , c'est-à-dire, pour réduire au minimum la probabilité d'accepter une séquence étant produite par un bon générateur quand le générateur était en réalité mauvais. Les praticiens choisissent d'habitude une taille type n et une valeur pour α (la probabilité d'une erreur Type 1 - le niveau de signification). Alors un point critique pour une donnée statistique est choisi, qui produira le plus petit β (la probabilité d'une erreur Type 2). La limite pour l'acceptabilité est choisie tel que la probabilité pour accepter faussement une séquence comme aléatoire a la plus petite valeur possible.

Chaque test est basé sur une valeur calculée du test statistique, qui est une fonction de données. Si la valeur statistique du test est S et la valeur critique est t, alors la probabilité de l'erreur de Type 1 est P ($S > t$ || H_0 est vraie) = P (rejeter H_0 | H_0 est vraie) et la probabilité d'erreur Type 2 est P ($S \leq t$ || H_0 est fausse) = P (accepter H_0 | H_0 est fausse).

Définition 3.1.5: Le test statistique est employé pour calculer une *p-valeur* qui récapitule la force de la preuve contre l'hypothèse nulle. Pour ces tests, chaque p-valeur est la probabilité qu'un générateur de nombres aléatoires parfait aurait produit une séquence moins aléatoire que la séquence qui a été évaluée.

Remarque 3: Si une p-valeur pour un test est déterminée pour être égale à 1, donc la séquence semble avoir une parfaite incohérence. Une p-valeur de zéro indique que la séquence semble être complètement non aléatoire.

Remarque 4: Si p-valeur $\geq \alpha$, donc l'hypothèse nulle est acceptée; c'est-à-dire, la séquence semble être aléatoire.
Si p-valeur $< \alpha$, donc l'hypothèse nulle est rejetée; c'est-à-dire, la séquence semble être non aléatoire.

3.2 Les distributions normale et khi-deux

Les distributions normale et khi-deux sont largement employées dans des applications statistiques.

Définition : Une fonction de densité de probabilité d'une variable aléatoire continue X est la fonction $f(x)$, intégrable et qui vérifie :

(1) $f(x) \geq 0$, pour tout $x \in R$.

(2) $\int f(x)\,dx = 1$.

(3) Pour tout $a, b \in R$, $P(a \leq x \leq b) = \int_a^b f(x)dx$.

3.2.1 La distribution normale

Définition : Une variable aléatoire est dite *normale* N (μ, σ) si sa fonction de densité est donnée par : $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$.

- Si X est N (0, 1), on dit que X est une distribution normale standard.
- La loi normale N (0, 1) est une distribution d'une loi continue qui est symétrique.

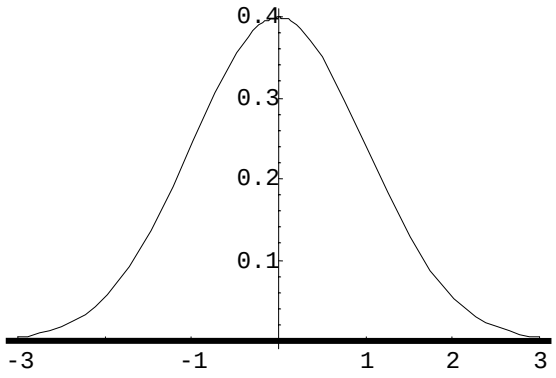


Figure 1 : graphe de la distribution normale N (0, 1)

α	0.1	0.05	0.025	0.01	0.005	0.0025	0.001	0.0005
x	1.2816	1.6449	1.9600	2.3263	2.5758	2.8070	3.0902	3.2905

Table 1 : sélection de quelques valeurs de la distribution normale N (0, 1). Si X est une variable aléatoire qui suit une loi normale N (0 ,1), alors $P(X > x) = \alpha$.

Remarque : Si la variable aléatoire *X* est N (μ, σ), alors la variable Z = $\frac{(X-\mu)}{\sigma}$ est N (0, 1).

3.2.2 La distribution de Khi-deux

On dit qu'une variable aléatoire *X* obéît à la loi de khi-deux (χ^2) si sa fonction de densité a la forme suivante : $f(x) = \begin{cases} 0 & si \quad x \leq 0 \\ c\, e^{-\frac{x}{2}}\, x^{\frac{n}{2}-1} & si \quad 0 < x < \infty \end{cases}$

- En général, *n* est compris entre 0 et 30.
- *n* désigne le nombre de degrés de liberté.
- On parle d'un χ^2 à *n* degrés de liberté et on note χ^2_n.
- *c* est une constante tq $\int_0^\infty f(x)\,dx = 1$.
- χ^2 n'est pas une distribution symétrique.

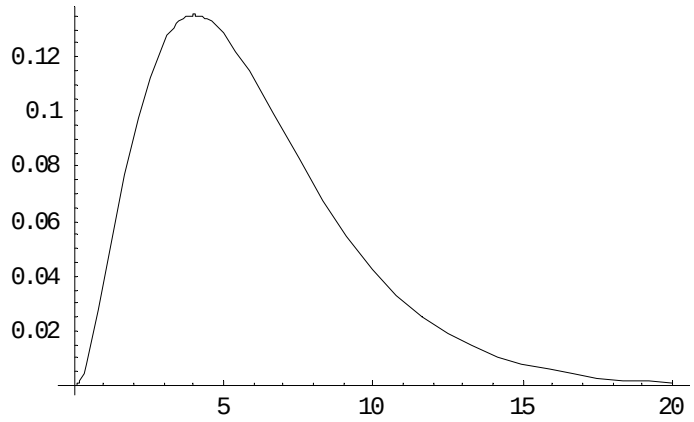


Figure 2 : graphe de la distribution de khi-deux à 6 degrés de liberté.

Remarques :

- la distribution normale résulte quand un grand nombre de variables aléatoires indépendantes et identiquement distribuées sont additionnées.
- Le χ^2_1 est obtenu comme carré d'une N (0, 1).
- Le χ^2_d est obtenu comme la somme de d carrés de N (0, 1) indépendantes.

3.3 Procédure d'évaluation pour une séquence binaire simple

Le processus	Commentaires
1- exposer l'hypothèse nulle	Supposer que la séquence binaire est aléatoire
2- effectuer le test statistique	Le test est effectué au niveau du bit
3- calculer la p-valeur	p-valeur $\in[0,1]$
4- comparer la p-valeur avec α.	Fixer α, si $\alpha \in$ (0.001 , 0.01] le succès est déclaré chaque fois que la p-valeur $\geq \alpha$ autrement l'échec est déclaré.

3.4 Tests de base

Considérons d'abord les tests classiques de base vus pour le cas de séquences uniformes réelles. Si $s= \{s_0, s_1, \ldots\}$ est une suite infinie, on notera $s^n = \{s_0, s_1, \ldots, s_{n-1}\}$ la sous suite finie.

Définition 3.4.1: La suite s est *N-periodique* si $s_i = s_{i+N}$ pour tout $i \geq 0$. Elle est dite périodique si elle est N-periodique pour un certain entier positif N.

La *période* est le plus petit entier positif pour lequel s est N-periodique.

Si s est une séquence périodique de période N, alors le *cycle* de s est la sous suite s^N.

Définition 3.4.2: Une série (run) de s est une sous suite de 0 consécutifs (ou de 1 consécutifs) qui n'est ni précédée ni suivie par le même symbole. Une série de 0 est appelée trou (gap) et une série de 1, un bloc.

Définition 3.4.3: La fonction d'autocorrelation d'une suite périodique de période N est la fonction à valeurs entières :

$$C(t) = \frac{1}{N}\sum_{i=0}^{N-1}(2s_i - 1)(2s_{i+t} - 1) \quad \text{pour } 0 \leq t \leq N\text{-}1.$$

Elle mesure la quantité de similarité entre la séquence s et une translation de s en t positions. Si s est une séquence aléatoire périodique de période N, alors on peut espérer que $|N.C(t)|$ soit assez petit pour toute valeur de t, $0 < t < N$.

Postulats de Colomb :

Les postulats de Colomb, au nombre de trois, formulent les conditions nécessaires pour qu'une suite pseudo-aléatoire ressemble à une suite vraiment aléatoire :

1. Dans le cycle s^N de s, le nombre de 1 diffère du nombre de 0 au plus d'une unité.
2. Dans le cycle de s^N au moins la moitié des séries sont de longueur 1, au moins un quart de longueur 2, au moins un huitième de longueur 3, …tant que le nombre de séries ainsi indiqué excède 1. De plus, pour chacune de ces longueurs, il y'a presque autant de trous (gaps) que de blocs.
3. La fonction d'autocorrelation est de la forme :

$$N.C(t) = \sum_{i=0}^{N-1}(2s_i - 1)(2s_{i+t} - 1) = \begin{cases} N & si\ t = 0 \\ K & si\ 1 \leq t \leq N\text{-}1 \end{cases}$$

Exemple :

Soit la suite périodique de période $N = 20$ de cycle s^{20}= 10100100101110010110

Vérifions que cette suite satisfait les 3 postulats.

Postulat 1 : le nombre de 1 est 10, le nombre de 0 est 10.

Postulat 2 : la suite a 14 séries ; 9 séries de longueur 1, 4 séries de longueur 2, une série de longueur 3.

Postulat 3 : la fonction d'autocorrelation $C(t)$ prend deux valeurs : $C(0) = 1$et $C(t) = -0.35$ pour $1 \leq t \leq 14$.

Par conséquent cette suite satisfait les postulats de Colomb.

Une suite binaire qui satisfait les postulats de Colomb est dite pseudo-bruitée.

Ces types de suites se rencontrent en pratique comme séquences de sortie de longueur maximale des registres à décalage, et des générateurs aléatoires en général.

3.4.1 Test de fréquence (test mono bit)

Il teste si la suite possède presque autant de 0 que de 1, comme on pourrait s'y attendre dans une séquence aléatoire. Soit n_0 le nombre de 0, et n_1 le nombre de 1.

Le critère du test est basé sur la statistique :

$$\chi_1 = \frac{(n_0 - n_1)^2}{n}$$

qui suit approximativement une loi de khi-deux à un degré de liberté pour $n \geq 10$ (en pratique on recommande , comme d'ailleurs pour les tests suivants , de prendre une taille d'échantillon beaucoup plus grande par exemple 10 000).

Exemple :

On considère la séquence s de longueur $n = 100$

11001 00100 00111 11101 10101 01000 10001 00001 01101 00011 00001 00011 01001 10001 00110 00110 01100 01010 00101 11000

on a n_0=58, n_1=42 , $\chi_1 = \frac{(n_0 - n_1)^2}{n} = \frac{(58-42)^2}{100} =$ 2.56 qui est la valeur du critère ; la valeur théorique de cette statistique obtenue des tables de la loi de khi-deux pour un degré de liberté et un niveau $\alpha = 0.05$ est χ_1 théorique) = 3.8415 > 2.56 , par conséquent la séquence *s* passe ce test .

3.4.2 Test de séries (test à deux bits)

Ce test détermine si les nombres d'occurrences de 00, 01, 10, 11 en tant que sous suites de s sont approximativement égaux, comme on pourrait l'espérer d'une suite vraiment aléatoire. Soit n_0 le nombre de 0, et n_1 le nombre de 1, et n_{00}, n_{01}, n_{10}, n_{11} les nombres d'occurrences de 00, 01, 10, 11 respectivement. Notons que $n_{00} + n_{01} + n_{10} + n_{11} = n - 1$.

Le critère du test est basé sur la statistique :

$$\chi_2 = \frac{4}{n-1} \left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2 \right) - \frac{2}{n} \left(n_0^2 + n_1^2 \right) + 1$$

qui suit approximativement une loi de khi-deux à 2 degrés de liberté pour $n \geq 21$.

Exemple :

On considère la même séquence de l'exemple précédent :

On a $n_{00} = 32$, $n_{01} = 25$, $n_{10} = 26$, $n_{11} = 16$, et la valeur calculée $_2 = \frac{4}{n-1} (n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{n} (n_0^2 + n_1^2) + 1 = \frac{4}{99}(32^2+25^2+26^2+16^2) - \frac{2}{100}(58^2+42^2) + 1 =$ 2.72, alors que la valeur théorique pour 2 degrés de liberté est égale à χ_2 (théorique) = 5.9915 > 2.72 et la suite s passe de nouveau ce test.

3.4.3 Test de poker

Soit m un entier positif tel que [n/m] $\geq$ 5. 2^m, et soit k = [n/m]. On divise la séquence s en k parties disjointes de longueur m chacune ; soit n_i le nombre d'occurrences de la séquence de type i de longueur m, $1 \leq i \leq 2^m$. Le test de poker teste si les séquences de longueur m apparaissent un nombre identique de fois dans s, comme on pourrait l'attendre d'une séquence vraiment aléatoire, le critère du test est :

$$\chi_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

qui suit approximativement une loi de khi-deux à $2^m - 1$ degrés de liberté. Notons que le test de poker est une généralisation du test de fréquence (en posant m =1).

Exemple :

On considère la même séquence de l'exemple précédent :

On a m =2, k = 50 ; les blocs 00, 01, 10, 11 apparaissent 17, 8, 16, 9 respectivement, et la valeur calculée du critère $\chi_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k = \frac{2^2}{50}(17^2+8^2+16^2+9^2) - 50 = 5.2$, alors que la valeur théorique est χ_3 (théorique) = 14.0671> 5.2. On constate que la suite passe de nouveau ce test.

3.4.4 Test de séries (run test)

Le but ici est de déterminer si le nombre de séries (runs) (de 0 ou de 1) de différentes longueurs dans la séquence s est égal au nombre de séries attendu dans une séquence vraiment aléatoire. Le nombre de gaps (ou de blocs) de longueur i dans une séquence aléatoire de longueur n est égal à $e_i = (n - i + 3) / 2^{i+2}$. Soit k le plus grand entier i pour lequel $e_i \geq 5$. Soient d'autre part B_i et G_i les nombres de gaps et de blocs de longueur i dans s pour chaque i,

$1 \leq i \leq k$. Le critère du test est :

$$\chi_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$$

qui suit approximativement une loi de khi-deux à 2k-2 degrés de liberté.

Exemple :

On considère la même séquence de l'exemple précédent :

On a $e_1 = 12.75$, $e_2 = 6.3125$, k = 2 ; il y a 9 gaps de longueur 1 et 5 gaps de longueur 2 et il y a 15 blocs de longueur 1 et 9 blocs de longueur 2.

La valeur calculée du critère est

$$\chi_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i} = \frac{(9-12.75)^2}{12.75} + \frac{(5-6.31)^2}{6.31} + \frac{(15-12.75)^2}{12.75} + \frac{(9-6.31)^2}{6.31} =$$

2.9 et celle du khi-deux théorique $\chi_4 = 9.4877 > 2.9$. La séquence passe le test.

3.4.5 Test d'autocorrélation

Il permet de tester la corrélation entre2la séquence s et ses versions translatées (non cycliques). Soit d un entier fixé, $1 \leq d \leq [n/2]$. Le nombre de bits dans s qui ne sont pas égaux à leurs d-translatés est A (d) = $\sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$ s_i où $\oplus$ dénote l'opérateur XOR. Le test est basé sur le critère :

$$\chi_5 = 2\left[A(d) - \frac{n-d}{2} \right] / \sqrt{n-d}$$

qui suit approximativement une loi normale standard N(0 , 1) pour $n - d \geq 10$.

Le test rejette la séquence pour les grandes valeurs de la statistique du test.

Exemple :

On considère la même séquence de l'exemple précèdent :

Pour d = 8, on a A (d) = 42. La valeur calculée du critère $= 2\left[A(d) - \frac{n-d}{2} \right] / \sqrt{n-d}$

$= 2\left[42 - \frac{100-8}{2} \right] / \sqrt{100-8} =$ -0.8342, alors que la valeur théorique lue dans la table de la loi normale standard est $\chi_5 = -1.96$. Par conséquent, la séquence passe le test car -0.8342 > -1.96.

Conclusion :la séquence **11001 00100 00111 11101 10101 01000 10001 00001 01101 00011 00001 00011 01001 10001 00110 00110 01100 01010 00101 11000** est aléatoire.

3.5 Norme FIPS 140-1

Cette norme FIPS spécifie les exigences de sécurité qui doivent être satisfaites par un module cryptographique.

Elle spécifie 4 tests statistiques de l'uniformité (décrits précédemment dans la section 3.4). L'utilisateur ne sélectionne pas un niveau de signification α. Par contre, elle fournit des bornes que la valeur calculée de la statistique doit satisfaire. La sortie du générateur représentée par une séquence s de longueur 20 000 bits doit passer tous les tests suivants, faute de quoi elle est rejetée.

Test de fréquence (mono bit) :

Ici, le nombre n_1 de 1 doit satisfaire : $9654 < n_1 < 10346$.

Test de poker :

Ici la statistique χ_3 est calculée pour m = 4. Le test est positif si $1.03 < \chi_3 < 57.4$.

Test de séries (run test) :

Ici les nombres B_i de blocs et G_i de longueur i dans s sont dénombrés pour chaque i

($1 \leq i \leq 6$). Pour les besoins de ce test, les séries de longueur > 6 sont considérées égales à 6. Le test est positif si les 12 valeurs de B_i, G_i ($1 \leq i \leq 6$) se trouvent dans les limites spécifiées par les intervalles de la table ci-après :

Longueur de la serie	Intervalle requis
1	2267-2733
2	1079-1421
3	502-748
4	223-402
5	90-223
6	90-223

Test des longues séries (long run test) :

La séquence s passe ce test s'il n'y a pas de séries de longueur 34 ou plus.

Remarque : pour les applications à haute sécurité, la norme FIPS 140-1 demande à ce que les quatre tests soient réalisés chaque fois que le générateur de bits aléatoires est amélioré.

3.6 Tests statistiques du NIST

La suite des tests statistiques du NIST (cet organisme qui a pour mission de développer et promouvoir la mesure, les standards et la technologie) est un paquet de 16 tests qui ont été développés pour évaluer le caractère aléatoire des séquences binaires produites par des générateurs aléatoires ou pseudo-aléatoires.

Les tests se concentrent sur une variété de types différents de non incohérence qui pourrait exister dans une séquence. Chaque test est basé sur une valeur calculée du test statistique, qui est une fonction de la séquence évaluée.

Le test statistique est employé pour calculer une p-valeur qui récapitule la force de la preuve pour le caractère aléatoire. Chaque p-valeur peut être interprétée comme la probabilité qu'un générateur de nombres parfaitement aléatoire produise une séquence moins aléatoire que la séquence qui a été évaluée. L'utilisation de la p-valeur doit permettre à un individu d'interpréter facilement et objectivement les résultats du test et d'évaluer la qualité du générateur.

Nous récapitulons ici sous une forme succincte les différentes procédures de tests. On prend $\alpha = 0.01$.

3.6.1 Test de fréquence

Entrée

– Une séquence binaire s de longueur (n).

Procédure

– Calculer la fréquence des 1 (ou des zéros).

– Calculer la p-valeur = $\boldsymbol{erfc}\left(\frac{S_{obs}}{\sqrt{2}}\right)$ ($S_{obs} = \frac{|S_n|}{\sqrt{n}}$ et $S_n = X_1 + \ldots + X_n$ où $X_i = 2s_i - 1$).

Exemple

Soit s = 1011010101, n = 10, S_n = 1+ (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + (-1) + 1 = 2, $S_{obs} = \frac{|2|}{\sqrt{10}} = 0.63245$, p-valeur = $\boldsymbol{erfc}\left(\frac{0.63245}{\sqrt{2}}\right) = 0.52708 \geq 0.01$, la séquence s est aléatoire.

Défaut détecté

– La déviation attendue de la distribution théorique des zéros ou des 1 est plus grande, i.e., il y'a trop de zéros ou bien trop de 1.

Remarques

– Le test évalue la proximité de la proportion de 1 à ½, c'est-à-dire le nombre des 1 et de 0 dans une séquence doit être le même.

– Tous les tests suivants dépendent du passage de ce test.

– On recommande que chaque séquence évaluée doit être constituée au minimum de 100 bits (c'est-à-dire, $n \geq 100$).

3.6.2 Test de fréquence dans un bloc

Entrée

– Une séquence binaire s de longueur (n), un bloc de bits de longueur M.

Procédure

– Calculer la fréquence des 1 (ou des zéros) dans le bloc.

– Calculer la p-valeur = ***igamc*** (N/2, $\chi^2(obs)$/2) où $N = \left\lfloor \frac{n}{M} \right\rfloor$ qui est le nombre de blocs,

$\chi^2(obs) = 4\,M\sum_{i=1}^{N}(\pi_i - \frac{1}{2})^2$ et $\pi_i = \frac{\sum_{j=1}^{M} s_{(i-1)M+j}}{M}$ pour $1 \leq i \leq N$.

Exemple

Soit s = 0110011010 et M = 3, n = 10, N = 3, π_1 = 2/3, π_2 = 1/3, π_3 = 2/3, $\chi^2(obs)$ = $4x3x\left(\left(\frac{2}{3}-\frac{1}{2}\right)^2+\left(\frac{1}{3}-\frac{1}{2}\right)^2+\left(\frac{2}{3}-\frac{1}{2}\right)^2\right)$= 1. La p-valeur= ***igamc***$\left(\frac{3}{2},\ \frac{1}{2}\right)$= 0.80125 ≥ 0.01, la séquence s est aléatoire.

Défaut détecté

– La déviation attendue de la distribution théorique des zéros ou des 1 dans le bloc est plus grande, i.e., il y'a trop de zéros ou bien trop de 1.

Remarques

– Le but de ce test est de déterminer si la fréquence des 1 dans un bloc à M bits est approximativement M/2, comme on l'aurait attendu sous une supposition d'incohérence.

– Pour la taille du bloc M =1, cet essai dégénère au test 1, test de fréquence (section 3.6.1).

– On recommande que chaque séquence à évaluer doit être constituée en un minimum de 100 bits (c'est-à-dire, $n \geq 100$).

3.6.3 Test de séries (runs test)

Entrée

– Une séquence binaire s de longueur (n).

Procédure

– Calculer le nombre total des séries de zéros et de 1 dans la séquence.

– Calculer la p-valeur = ***erfc*** $\left[\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n}\pi(1-\pi)}\right]$ où

$V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$, ($r(k) = 0$ si $s_k = s_{k+1}$, et $r(k) = 1$ autrement), et $\pi = \frac{\sum_j s_j}{n}$ (la proportion des 1 dans la séquence).

Exemple

Soit s = 1001101011, n = 10, π = 6/10 = 3/5, $V_n(obs)$ = (1 + 0 + 1 + 0 + 1 + 1 + 1 + 0) + 1 = 7, p-valeur = ***erfc*** $\left(\frac{\left|7 - \left(2.10.\frac{3}{5}\left(1-\frac{3}{5}\right)\right)\right|}{2.\sqrt{2.10}.\frac{3}{5}.\left(1-\frac{3}{5}\right)}\right)$ = 0.14723 ≥ 0.01, la séquence s est aléatoire.

Défaut détecté

– Une grande (ou petite) valeur du test statistique indique que l'oscillation dans la séquence est plus rapide (ou plus lente).

Remarques

– Une série de longueur k est constituée exactement de k bits identiques et elle est limitée auparavant et après avec le bit de la valeur opposée.

– Une oscillation est considérée pour être un changement de 1 à un zéro ou vice versa. Une oscillation rapide arrive quand il y a beaucoup de changements, par exemple, 010101010 oscille avec chaque bit. Une séquence avec une lente oscillation a moins de séries qu'on l'attendrait d'une séquence aléatoire.

– On recommande que chaque séquence à évaluer doit être constituée au minimum de 100 bits (c'est-à-dire, $n \geq 100$).

3.6.4 Test de longues séries de 1 dans un bloc

Le centre du test est la plus longue série de 1 dans un bloc à M bits. Le but de ce test est de déterminer si la longueur de la plus longue série de 1 dans la séquence évaluée est cohérente avec la longueur de la plus longue série de 1 de ce qu'on attendrait d'une séquence aléatoire.

Remarque

– On note qu'une irrégularité dans la longueur attendue de la plus longue série de 1 implique qu'il y a aussi une irrégularité dans la longueur attendue de la plus longue série de zéros. Donc, seulement un test sur les 1 est nécessaire.

3.6.5 Test de la matrice binaire *(de Marsaglia DIEHARD)*

Entrée

– Une séquence binaire s de longueur (n), un bloc de longueur (M).

Procédure

– Construire N matrices (M x M) de la séquence binaire.

– Pour chaque matrice, déterminer son rang.

– Calculer la statistique khi-deux basée sur la distribution théorique du rang.

– Calculer la p-valeur = $e^{-\chi^2(obs)/2}$ où

$$\chi^2(obs) = \frac{(F_M - 0.2888N)^2}{0.2888N} + \frac{(F_{M-1} - 0.5776N)^2}{0.5776N} + \frac{(N - F_M - F_{M-1} - 0.1336N)^2}{0.1336N}$$

F_M , $F_{M\text{-}1}$ sont les nombres de matrices de rang M et M-1 respectivement.

Exemple

Soit s = 01011001001010101101, n = 20, M = 3, $N = \left\lfloor \frac{n}{M.M} \right\rfloor$ = 2 matrices qui sont

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{vmatrix} \text{ et } \begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}$$

(la première ligne de la première matrice est le premier bloc de la séquence, la deuxième ligne est le deuxième bloc de la séquence et on poursuit la construction de la matrice ainsi que la deuxième matrice de la même manière).

$F_M = F_3 = 1$, $F_2 = 1$, $\chi^2(obs) = 0.59695$. La p-valeur = 0.74194 ≥ 0.01, la séquence s est aléatoire.

Défaut détecté

– Déviation de la distribution du rang correspondant à la séquence aléatoire.

Remarque

– Le test utilise des sous matrices disjointes formées à partir de la séquence entière pour vérifier la dépendance linéaire entre les sous séries de longueur fixée de la séquence testée.

3.6.6 Test (spectral) de la transformée de Fourier

Entrée

– Séquence binaire de longueur (n).

Procédure

– Appliquer la transformée discrète de Fourier à la séquence et déterminer son module.

– Choisir les $n/2$ premiers éléments (des sommets) et compter le nombre de sommets moins que $\sqrt{3n}$.

– Calculer la p-valeur = $\boldsymbol{erfc}\left(\frac{|d|}{\sqrt{2}}\right)$ où $d = \frac{(N_1 - N_0)}{\sqrt{10(0.95)(0.05)/2}}$, $N_0 = 0.95n/2$ (le nombre de sommets moins que $\sqrt{3n}$) et N_1 est le nombre de sommets moins que $\sqrt{3n}$ observé dans le module.

Exemple

Soit s = 1001010011, n =10, N_0 = 4.75, N_1 = 4, d = -1.53896. La p-valeur = $\boldsymbol{erfc}\left(\frac{1.53896}{\sqrt{2}}\right) = 0.12381 \geq 0.01$, la séquence s est aléatoire.

Défaut détecté

– Détecte des particularités périodiques dans la séquence de bits.

Remarques

– Le but de ce test est de détecter des particularités périodiques (c'est-à-dire, les modèles répétitifs qui sont près l'un de l'autre) dans la séquence évaluée qui indiquerait une déviation de la supposition d'incohérence.

– On recommande que chaque séquence à évaluer doit être constituée au minimum de 1000 bits (c'est-à-dire, $n \geq 1000$).

3.6.7 Non overlapping template matching test[4]

Entrée

– Séquence *s* de longueur (*n*), *M* la longueur de chaque sous séquence testée, calibre (*B*) de longueur (*m*).

Procédure

– Calculer le nombre de fois où *B* apparaît dans le flot.

– Calculer la p-valeur = ***igamc*** $\left(\frac{N}{2}, \frac{\chi^2(obs)}{2}\right)$, *N* est le nombre de blocs, $\chi^2(obs)$ $= \sum_{j=1}^{N} \frac{(W_j - \mu)^2}{\sigma^2}$, W_j est le nombre de fois où *B* apparaît dans le bloc *j*, $\mu = (M\text{-}m+1) / 2^m$ et $\sigma^2 = M\left(\frac{1}{2^m} - \frac{2m-1}{2^{2m}}\right)$.

Exemple

Soit *s* = 10100100101110010110, *n* = 20. Si *N* = 2 et *M* = 10 alors les 2 blocs sont 1010010010 et 1110010110.

On crée une fenêtre à *m* bits et on compare les bits de cette fenêtre au calibre. Si le calibre n'est pas trouvé, la fenêtre fait glisser une position de bit. Par exemple si *m* = 3 et la fenêtre contient les bits 3 à 5, la fenêtre suivante contiendra les bits 4 à 6. Par contre si le calibre est trouvé la fenêtre fait glisser *m* positions, si la fenêtre (couronnée de succès) contient des bits 3 à 5, donc la fenêtre suivante contiendra des bits 6 à 8.

Si *m* = 3 et le calibre *B* = 001, l'évaluation sera comme suit :

Positions des bits	**Bloc 1**		**Bloc 2**	
	Bits	W_1	**Bits**	W_2
1-3	101	0	111	0
2-4	010	0	110	0
3-5	100	0	100	0
4-6	001(succès)	1	001(succès)	1
5-7	Non examiné		Non examiné	
6-8	Non examiné		Non examiné	
7-9	001	2	011	1
8-10	010	2	110	1

[4] Non-overlappings template matchings test est le test de non-chevauchement de calibres (formes).

$W_1 = 2,\ W_2 = 1,\ \mu = 1$ et $\sigma^2 = 0.46875$. $\chi^2(obs) = \dfrac{(2-1)^2 + (1-1)^2}{0.46875} = 2.13333$. La p-valeur = ***igamc*** $\left(\dfrac{2}{2}, \dfrac{2.13333}{2}\right) = 0.34415 \geq 0.01$, la séquence *s* est aléatoire.

Défauts détectés

– Trop d'occurrences des calibres non- périodiques.

Remarques

– Pour ce test une fenêtre à *m* bits est employée pour chercher un modèle spécifique à *m* bits.

– Si le modèle n'est pas trouvé, la fenêtre fait glisser une position de bit.

– Si le modèle est trouvé, la fenêtre est remise au bit après le modèle trouvé.

– On recommande $m = 9$ ou $m = 10$ pour obtenir des résultats significatifs.

– Le test de non chevauchement de calibres détermine la fréquence des calibres non périodiques.

– Le test rejette les séquences présentant trop ou trop peu d'occurrences d'un modèle donné apériodique.

3.6.8 Overlapping template matching test[5]

– Ce test détermine aussi s'il y a trop d'occurrences de formes prédéfinies. La différence par rapport au test précèdent (3.6.7), s'il y a chevauchement la fenêtre fait glisser seulement un bit avant la reprise de la recherche. Le test de chevauchement de calibres détermine la fréquence des séries de *m*-bits de 1.

– Le test rejette les séquences présentant trop ou trop peu d'occurrences de *m*-series de 1, mais peut facilement être modifié pour détecter les occurrences irrégulières de n'importe quel modèle périodique *B*.

3.6.9 Test statistique universel *(d'Ueli Maurer)*

Entrée

– Séquence *s* de longueur (*n*), bloc de longueur (*L*), segment d'initialisation (*Q*), segment du test (*K*).

[5] Overlapping template matching test est le test de chevauchement de calibres (formes).

Procédure

– Compter le nombre de différents calibres dans la séquence.

– Calculer la p-valeur (voir la section 4.3.5).

Défauts détectés

– Détecte les générateurs pseudo-aléatoires qui peuvent être modélisés par une source stationnaire ergodique à mémoire finie.

– Critère de compressibilité (régularité).

Remarque

– Une séquence excessivement compressible est considérée pour être non aléatoire.

3.6.10 Test de compression de Lempel-Ziv

Entrée

–Séquence de bits.

Procédure

– Soit W_{obs} : le nombre de mots disjoints et cumulativement distincts dans la séquence.

– Faire l'analyse syntaxique des mots consécutifs disjoints et distincts qui forment le "dictionnaire" de mots dans la séquence.

– Créer des sous séries de bits consécutifs de la séquence jusqu'à trouver une sous série qui n'a pas été trouvée précédemment. La sous-série résultante est un nouveau mot du dictionnaire.

Exemple

Soit la séquence s = 010110010

Position du bit	Bit	Mot nouveau ?	Le mot est :
1	0	Oui	0 (bit 1)
2	1	Oui	1 (bit 2)
3	0	Non	
4	1	Oui	01 (bits 3-4)
5	1	Non	
6	0	Oui	10 (bits 5-6)
7	0	Non	
8	1	Non	
9	0	Oui	010(bits7-9)

On a trouvé cinq mots dans le "dictionnaire" : 0, 1, 01, 10, 010. Donc, $W_{obs} = 5$.

– Calculer la p-valeur = ½ ***erfc***$\left(\frac{\mu - W_{obs}}{\sqrt{2\sigma^2}}\right)$ où μ = 69586.25 et $\sigma = \sqrt{70.448718}$ pour $n = 10^6$.

Remarques

– Il examine le nombre de formes cumulativement distinctes dans la séquence du test de manière à déterminer de combien la séquence peut être compressée.

– Une séquence vraiment aléatoire possède un nombre caractéristique de formes distinctes, elle est considérée pour être non aléatoire s'elle peut être significativement compressée.

3.6.11 Test de complexité linéaire

Entrée

– Une séquence de bits.

Procédure

– Partitionner la séquence de n bits en N blocs de M bits où $n = MN$.

– Déterminer la complexité linéaire L_i pour chacun des N blocs (i =1,…,N). L_i est la longueur du plus court registre à décalage linéaire.

– Calculer la p-valeur = ***igamc*** $\left(\frac{K}{2}, \frac{\chi^2(obs)}{2}\right)$, K est le nombre de degrés de liberté et $\chi^2(obs)$ est décrite dans la section 2.11.4 de [**1**].

Défaut détecté

– La séquence n'est pas assez complexe pour être considérée aléatoire.

Remarques

– Le centre du test est la longueur d'un registre linéaire à décalage.

– Le but de ce test est de déterminer si vraiment la séquence est assez complexe pour être considérée aléatoire.

– La séquence aléatoire est caractérisée par le plus long registre linéaire à décalage. Un court registre implique que la suite n'est pas aléatoire.

3.6.12 Test de séries (serial test)

Le test vérifie l'uniformité de(s) distribution(s) de formes à m-bits pour différentes longueurs des formes m. Une séquence aléatoire exhibe une uniformité : chaque forme à m-bit doit apparaître avec la même fréquence que toute autre forme à m-bit, en moyenne. Pour m = 1, cet essai dégénère au test 1, test de fréquence (section 3.6.1).

3.6.13 Test de l'entropie approximative (*de Pincus & Singer (1997))*

Entrée

– Séquence de bits, bloc de longueur (*m*).

Procédure

– Augmenter la longueur de la séquence en ajoutant *m* -1 bits.

– Calculer la fréquence de blocs de *m*-bits possibles.

– Calculer C_i^m où *i* est la fréquence de blocs de *m*-bits, et $C_i^m = \#i / n$ pour chaque valeur de *i*.

– Calculer $\varphi^{(m)} = \sum_{i=0}^{2^m-1} \pi_i \log \pi_i \;\; ou \;\; \pi_i = C_j^3 \; et \; j = \log_2 i$.

– Répéter la procédure en remplaçant *m* par *m*+1.

– Calculer la statistique $\chi^2 = 2n\left[\log 2 - ApEn(m)\right] où \; ApEn(m) = \varphi^{(m)} - \varphi^{(m-1)}$.

– Calculer la p-valeur= ***igamc*** $\left(2^{m-1}, \frac{\chi^2}{2}\right)$

Exemple

– Soit *s* = 0100110101 et *m* = 3, *n* = 10 ; on ajoute deux (*m* -1) bits 0 et 1 à la séquence pour tester la nouvelle séquence 010011010101.

– On calcule la fréquence de blocs de 3-bits possibles :

#000 = 0, #001 = 1, #010 = 3, # 011 = 1, #100 = 1, #101 = 3, #110 = 1, #111= 0.

– On calcule C_i^m où *i* est la fréquence de blocs de *m*-bits, et $C_i^m = \#i / n$ pour chaque valeur de *i*.

$C_{000}^3 = 0,\; C_{001}^3 = 0.1,\; C_{010}^3 = 0.3,\; C_{011}^3 = 0.1,\; C_{100}^3 = 0.1,\; C_{101}^3 = 0.3,\; C_{110}^3 = 0.1,\; C_{111}^3 = 0.$

– On calcule $\varphi^{(m)} = \sum_{i=0}^{2^m-1} \pi_i \log \pi_i \;\; où \;\; \pi_i = C_j^3 \; et \; j = \log_2 i$. Pour notre exemple $\varphi^{(3)}$ = -1.643417.

On répète les étapes précédentes en remplaçant *m* par *m* + 1 (on remplace 3 par 4). Après calculs, $\varphi^{(4)}$ = -1.83437.

χ^2 = 0.50219. La p-valeur = ***igamc*** $\left(2^2, \frac{0.50219}{2}\right)$ = 0.26196 ≥ 0.01, la séquence *s* est aléatoire.

Défaut détecté

– Les petites valeurs d'*ApEn* (*m*) impliquent la régularité forte.

Remarque

– Le test compare les fréquences de blocs de longueurs (m) et (m+1) avec celles d'une suite vraiment aléatoire.

3.6.14 Test des sommes cumulées

Entrée

– Séquence *s* de longueur (*n*).

Procédure

– Normaliser la séquence en convertissant les zéros et les 1 en valeurs (-1) et (+1).

– Calculer le maximum des sommes partielles S_i des sous séquences successives.

– Calculer le test statistique $z = \max_{1 \le k \le n} |S_k|$ où $max\,|S_k|$ est la plus grande valeur absolue des sommes partielles S_k.

– Calculer la p-valeur =

$$1 - \sum_{k=\left(\frac{-n}{z}+1\right)/4}^{\left(\frac{n}{z}-1\right)/4} \left(\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right) + \sum_{k=\left(\frac{-n}{z}-3\right)/4}^{\left(\frac{n}{z}-1\right)/4} \left(\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)\right)$$

où Φ est la fonction de distribution de la loi normale.

Exemple

Soit *s* = 1011010111, on convertit les 0 et les 1 en (-1) et (+1) en utilisant $X_i = 2s_i - 1$, alors

X = 1, (-1), 1, 1, (-1), 1, (-1), 1, 1, 1.

On calcule les sommes successives,

$S_1 = 1$,

$S_2 = 1 + (-1) = 0$,

$S_3 = 1 + (-1) + 1 = 1$,

$S_4 = 1 + (-1) + 1 + 1 = 2$,

$S_5 = 1 + (-1) + 1 + 1 + (-1) = 1$,

$S_6 = 1 + (-1) + 1 + 1 + (-1) + 1 = 2$,

$S_7 = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) = 1$,

$S_8 = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 = 2$,

$S_9 = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + 1 = 3$,

$S_{10} = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + 1 + 1 = 4$.

La plus grande valeur de S_k est 4, donc *z* = 10. La p-valeur = 0.411658 ≥ 0.01, la séquence *s* est aléatoire.

Défauts détectés

– Ou bien trop de 1 ou bien trop de zéros aux premières étapes.

– Détermine si la valeur cumulée d'une somme de séquences partielles occurant dans la séquence testée est trop grande ou trop petite par rapport à l'évolution espérée de telles sommes dans les séquences vraiment aléatoires.

Remarques

– La somme cumulée est assimilée à une marche aléatoire.
– Pour une séquence aléatoire, la marche aléatoire oscille autour de 0.
– De grandes valeurs de la marche indiquent trop de 0 ou de 1 au voisinage du départ de la séquence.
– De petites valeurs indiquent que les 0 et les 1 sont mélangés fréquemment.
– Le but du test est de déterminer si la somme cumulative des séquences partielles arrivant dans la séquence testée est trop grande ou trop petite quant au comportement attendu de cette somme cumulative pour des séquences aléatoires.

3.6.15 Test des excursions aléatoires

Entrée

– Une séquence de bits de longueur n.

Procédure

– Convertir les 0 et 1 en (-1) et (+1) et calculer les sommes partielles S_i (i =1,…,n).
– Construire une nouvelle séquence $S^{'}$ en ajoutant 0 au début et à la fin de la séquence S. $S^{'}$=0, S_1, S_2,…, S_n, 0.
– Soit J = le nombre total de croisement zéro dans la séquence S$^{'}$, où le croisement zéro est la valeur de zéro dans $S^{'}$ qui arrive après le départ zéro. J est aussi le nombre de cycles dans $S^{'}$, où le cycle est la sous séquence de $S^{'}$ qui consiste en la présence de zéro, suivi par des valeurs non nulles et finissant avec un autre zéro. Le zéro de la fin d'un cycle est le zéro du début d'un autre cycle. Le nombre de cycles dans $S^{'}$ est le nombre de croisements zéro. Si J < 500, arrêter le test.
– Pour chaque cycle et chaque valeur d'état x (x dans [-4,-1] et [1,4]), calculer la fréquence de chaque valeur x dans chaque cycle.
– Calculer $\mu_k(x)$= le nombre total de cycles dans lequel l'état x arrive exactement k fois parmi tous les cycles.
– Pour chaque état x, on calcule sa p-valeur= ***igamc*** (5/2, $\chi^2(obs)/2$) où $\chi^2(obs) = \sum_{k=0}^{5} \frac{(\mu_k(x) - J\pi_k(x))^2}{J\pi_k(x)}$ et $\pi_k(x)$ est la probabilité que l'état x soit présent k (k = 0, 1,…, 5, toutes les fréquences ≥ 5 sont stockées dans $\mu_5(x)$ et $\sum_{k=0}^{5} \mu_k(x) = J$) fois dans la distribution aléatoire (pour la table de valeurs de π_k, voir la section 3.15 de **[1]**).

Exemple

Soit s = 0110110101, on convertit les 0 et les 1 en (-1) et (+1) en utilisant $X_i = 2\ s_i - 1$.

X = -1, 1, 1, -1, 1, 1, -1, 1, -1, 1. On calcule les sommes partielles successives :

$S_1 = -1$, $S_2 = 0$, $S_3 = 1$, $S_4 = 0$, $S_5 = 1$, $S_6 = 2$, $S_7 = 1$, $S_8 = 2$, $S_9 = 1$, $S_{10} = 2$.

S = {-1, 0, 1, 0, 1, 2, 1, 2, 1, 2} et on forme une nouvelle séquence S' en ajoutant 0 au début et à la fin de la séquence S, S' = {0, -1, 0, 1, 0, 1, 2, 1, 2, 1, 2, 0}.

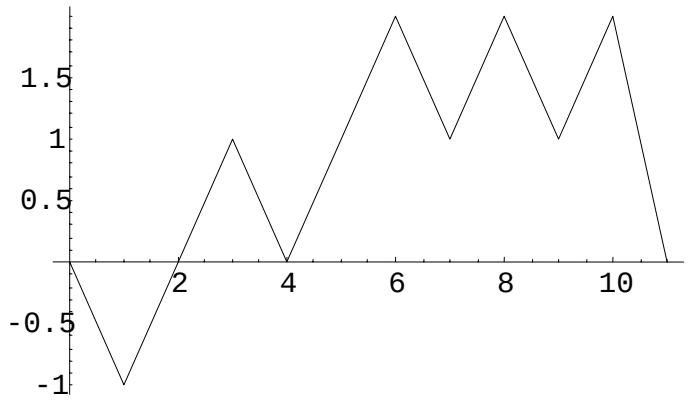


Figure 3 : exemple d'excursion aléatoire (S')

J = 3 (il y a 3 zéros dans les positions 3, 5 et 12 de S'), par conséquent il y a 3 cycles qui sont {0, -1, 0}, {0, 1, 0} et {0, 1, 2, 1, 2, 1, 2, 0}.

Pour chaque cycle et chaque valeur d'état x (x dans [-4,-1] et [1,4]), on calcule la fréquence de chaque valeur x dans chaque cycle.

On constate que (-1) est présent une seule fois au 1er cycle, 1 est présent une seule fois au 2eme cycle, 1 et 2 sont présents trois fois chacun au 3eme cycle. Cela peut être visualisé dans le tableau qui suit :

Etat *x*	**Cycles**		
	Cycle 1 {0, -1, 0}	**Cycle 2 {0, 1, 0}**	**Cycle 3 {0, 1, 2, 1, 2, 1, 2,0}**
-4	0	0	0
-3	0	0	0
-2	0	0	0
-1	1	0	0
1	0	1	3
2	0	0	3
3	0	0	0
4	0	0	0

On calcule $\mu_k(x)$= le nombre total de cycles dans lequel l'état x arrive exactement k fois parmi tous les cycles, qu'on résume dans le tableau qui suit :

Etat x	Nombre de cycles (k)					
	0	**1**	**2**	**3**	**4**	**5**
-4	3	0	0	0	0	0
-3	3	0	0	0	0	0
-2	3	0	0	0	0	0
-1	2	1	0	0	0	0
1	1	1	0	1	0	0
2	2	0	0	1	0	0
3	3	0	0	0	0	0
4	3	0	0	0	0	0

Par exemple l'état (-4) est présent 0 fois dans tous les cycles donc μ_0 (-4) = 3, l'état 1 est présent une fois dans un cycle donc μ_1 (1) =1, l'état 2 est présent 3 fois dans un cycle donc μ_3 (2) = 1.

On calcule $\chi^2(obs) = \sum_{k=0}^{5} \frac{(\mu_k(x) - J\pi_k(x))^2}{J\pi_k(x)}$ où $\pi_k(x)$ est la probabilité que l'état x soit présent k fois dans la distribution aléatoire (pour la table de valeurs de π_k, voir la section 3.15 de **[1]**). La p-valeur = ***igamc*** $\left(\frac{5}{2}, \frac{4.33303}{2}\right)$ = 0.502529, pour l'état x =1.

N.B : on calcule une p-valeur pour chaque état.

Défaut détecté

– Détecte la déviation de la distribution du nombre de visites de l'excursion aléatoire à un certain état.

Remarque

– Le but du test est de déterminer si le nombre de visites à un état particulier dans un cycle dévie de ce que l'on attendrait pour une séquence aléatoire.

– Ce test est en réalité une série de huit tests (et des conclusions), un test et une conclusion pour chacun des états :-4,-3,-2,-1 et +1, +2, +3, +4.

3.6.16 Variante du test des excursions aléatoires

– Ce test est en réalité une `extension du précédent test, c'est une série` de dix-huit tests (et des conclusions), un essai et conclusion pour chacun des états :-9,-8, …,-1 et +1, …, +9.

– Le centre du test est le nombre total de fois qu'un état particulier est visité dans une somme cumulative d'excursion aléatoire.

– Le but du test est de détecter des déviations du nombre attendu de visites aux états divers dans l'excursion aléatoire.

Chapitre 4

TEST STATISTIQUE UNIVERSEL DE MAURER

4. TEST STATISTIQUE UNIVERSEL DE MAURER

Dans la suite, nous nous limiterons à l'étude du test de Maurer et ses applications.

4.1 Modèles statistiques pour les générateurs de bits

Le modèle probabiliste le plus simple d'un générateur de bit est une *source binaire à mémoire finie* (Binary Memoryless Source, *BMS*) qui produit des variables binaires aléatoires statistiquement indépendantes et identiquement distribuées. Il est caractérisé par un paramètre simple, la probabilité p d'émettre 1. Ce modèle sera noté par BMS_p. Notons que le $BMS_{1/2}$ est équivalent à une source binaire symétrique (Binary Symmetric Source, *BSS*). Un autre modèle simple, noté par ST_p, émet 0 et 1 avec la même probabilité p, mais ses probabilités de transition sont biaisées : un chiffre binaire est suivi par son complément avec la probabilité p et par le même chiffre avec la probabilité 1-p. C'est un exemple d'une source binaire stationnaire avec un bit de mémoire. En général, la distribution de probabilité du *ieme* bit peut dépendre des M bits précédents où M est la taille de la mémoire de la source. Dans beaucoup d'applications, il est raisonnable d'assumer qu'un générateur de bits aléatoires défectueux ou mal conçu peut être modélisé par une telle source avec une mémoire relativement petite.

Considérons une source S qui émet une séquence U_1, U_2.... de variables binaires aléatoires. Supposons qu'il existe un entier positif M tel que pour tout $n > M$, la distribution de probabilité conditionnelle de U_n, étant donné U_1..., U_{n-1}, dépend seulement des M bits les plus récents , c'est-à-dire, tel que :

$$P_{U_n|U_1\ldots U_{n-1}}(u_n \mid u_1 \ldots u_{n-1}) = P_{U_n|U_{n-1}\ldots U_{n-M}}(u_n \mid u_{n-1} \ldots u_{n-M}) \qquad \textbf{(1)}$$

Définition : Pour $n > M$ et pour chaque séquence binaire (u_1...,u_n), alors le plus petit M vérifiant (1) est appelé *mémoire* de la source S et $\sum_n = [U_{n-1},\ldots,U_{n-M}]$ dénote son état au temps n.

Soit $\sum_1 = [U_0,\ldots,U_{1-M}]$ l'état initial où U_{1-M} ,..., U_0 sont les variables simulées aléatoires. Si en plus de la condition (1) la source satisfait :

$$P_{U_n|\Sigma_n}(u|\sigma) = P_{U_1|\Sigma_1}(u|\sigma)$$

pour tout $n > M$ et pour tout $u \in B$ et $\sigma \in B^M$, (B dénote la suite {0,1}), alors elle est dite *stationnaire*. Une source stationnaire de mémoire M est ainsi complètement spécifiée par la

distribution de probabilité de l'état initial P_{Σ_1} , et la distribution de probabilité d'un état à un autre $P_{\Sigma_2|\Sigma_1}$.

La séquence d'état forme ainsi une chaîne de Markov avec la propriété particulière que chacun des 2^M états possèdent au moins 2 états successeurs avec une probabilité non nulle.

Nous noterons les 2^M états possibles de la source (ou la chaîne de Markov) par les entiers de l'intervalle [0, 2^M-1].

($\sum_n = j$ signifie que $U_{n-1}\ldots U_{n-M}$ est la représentation binaire de j).

Pour la classe des chaînes de Markov ergodiques, qui incluent beaucoup de cas ayant un intérêt pratique, il existe une distribution de probabilité des états invariante p_0 ,..., p_{2^M-1} telle que :

$$\lim_{n\to\infty} P_{\Sigma_n}(j) = p_j \qquad \text{pour } 0 \le j \le 2^M\text{-}1$$

De plus, les probabilités p_j sont solution du système de 2^M équations linéaires suivant

$$\sum_{j=0}^{2^M-1} p_j = 1 \qquad \textbf{(2)}$$

$$p_j = \sum_{k=0}^{2^M-1} P_{\Sigma_2|\Sigma_1}(j|k)\, p_k \qquad \text{pour } 0 \le j \le 2^M\text{-}2 \qquad \textbf{(3)}$$

4.2 Taille efficace de la clef de système de chiffrement

Un système de chiffrement est bon s'il est conçu de telle manière qu'on ne connaisse pas de meilleure attaque que la recherche exhaustive. La dimension de l'espace des clefs est choisie de telle manière que pour réussir une telle recherche exhaustive, même avec une très petite probabilité de succès, requiert un effort de recherche impossible (ou important). Si toutes les valeurs possibles de la clef secrète ne sont pas équiprobables à priori, alors la stratégie optimale de l'ennemi dans une recherche complète de la clef est de commencer par la clef la plus probable et continuer à évaluer des clefs dans l'ordre décroissant des probabilités. On note par *Z*, la clef secrète, *n* sa longueur de bits et $z_1 ..., z_{2^n}$ une liste de valeurs de clefs satisfaisant :

$$P_Z(z_1) \ge \ldots \ge P_Z(z_{2^n})$$

Pour une source donnée S et pour δ qui satisfait $0 \leq \delta \leq 1$, soit $\mu_S(n, \delta)$ le nombre minimum de clefs estimées que l'ennemi doit évaluer (en utilisant la stratégie optimale de recherche de clef) pour trouver la clef correcte avec une probabilité au moins égale δ quand S est employée pour produire la clef de n-bit Z,i.e.,

$$\mu_S(n, \delta) = min\{k : \sum_{i=1}^{k} P_Z(z_i) \geq \delta\} \quad \textbf{(4)}$$

On définit la taille efficace de la clef d'un système de chiffrement avec source clef S comme étant $log_2\ \mu_S(n, ½)$, c'est-à-dire, le logarithme du nombre minimum de clefs qu'un ennemi doit essayer pour trouver la clef correcte avec une probabilité d'au moins 50 %. Le choix $\delta = ½$ dans cette définition est quelque peu arbitraire, mais en général, pour n assez grand, $log_2\ \mu_S(n, \delta)/n$ est presque indépendant de δ quand δ n'est pas trop proche de 0 ou 1. On note que quand la clef est vraiment aléatoire, c'est-à-dire, quand S est une source binaire symétrique, alors $log_2\ \mu_S(n, ½) = n - 1$.

Déterminons maintenant la taille efficace de la clef d'un système de chiffrement dont la source clef est BMS_P. Sans perte de généralité on suppose que $0 < p < ½$. On note que la source ST_p décrite précédemment peut être modélisée par la source BMS_p avec une sommation à la sortie (addition modulo 2 des bits de sortie du BMS_P). Donc la suite de probabilités des clefs et donc aussi la taille efficace de la clef est identique pour les deux sources. La distribution de probabilité de Z est donnée par :

$$P_Z(z) = p^{w(z)}\ (1-p)^{n-w(z)}$$

où $w(z)$ dénote le poids de Hamming de z. Pour réussir avec une probabilité approximative de ½, l'ennemi doit examiner toutes les clefs z de poids de Hamming $w(z) < pn$. La taille efficace de la clef peut être approchée par :

$$log_2\ \mu_{\mathrm{BMS}p}(n, 1/2) \approx log_2 \sum_{i=0}^{pn} \binom{n}{i} \quad \textbf{(5)}$$

En utilisant l'estimation suivante des coefficients binomiaux (équation A.21 **[7]**) :

$$\frac{1}{\sqrt{8t(n-t)/n}}\,2^{nH(t/n)} \le \binom{n}{t} \le \sum_{i=0}^{t}\binom{n}{i} \le 2^{nH(t/n)} \qquad \textbf{(6)}$$

où $H(x)$ est la fonction d'entropie binaire définie par :

$$H(x) = -x\,log_2 x - (1-x)\,log_2(1-x) \qquad \text{pour } 0 < x < 1 \qquad \textbf{(7)}$$
$$\text{et } H(0) = H(1) = 0.$$

On note que $H(x) = H(1-x)$ pour $0 \le x \le 1$.
Les inégalités (6) suggèrent l'approximation suivante :

$$log_2 \sum_{i=0}^{t}\binom{n}{i} \approx n\,H(t/n)$$

qui avec (5) donne :

$$log_2\,\mu_{\mathrm{BMS}p}(n, 1/2) \approx n\,H(p)$$

En utilisant (6) on peut prouver que cette approximation est asymptotiquement exacte, c'est-à-dire, que

$$\lim_{n\to\infty}\frac{log_2\,\mu_{BMS_P}(n,\delta)}{n} = H(p) \qquad \text{pour } 0 < \delta < 1.$$

Notons que l'entropie par bit de sortie de la source BMS_p, $H(p)$ est donc égale au facteur par lequel la taille efficace de clef est réduite. Shannon (voir **[3]**, théorème 4) a prouvé que pour une source générale ergodique stationnaire,

$$\lim_{n\to\infty} log_2\,\mu_S(n, \delta)/n = H_S \qquad \text{pour } 0 < \delta < 1\ ,$$

où H_S est l'entropie par bit de S définie comme suit :

$$H_S = -\sum_{j=0}^{2^M-1} p_j \sum_{k=0}^{2^M-1} P_{\Sigma_2|\Sigma_1}(k|j)\,log_2\,P_{\Sigma_2|\Sigma_1}(k|j) \qquad \textbf{(8)}$$

Et où les probabilités stationnaires d'état p_j sont pour $0 \leq j \leq 2^M$-1 définies par (3). Autrement dit, pour la classe générale de sources ergodiques stationnaires, l'entropie par-bit H_S est la mesure de leur qualité cryptographique quand ils sont employés comme source secrète de clef d'un système de chiffrement. Inversement, la redondance par-bit, 1-H_S, est la mesure du mauvais état cryptographique d'une source de clef. Puisque chaque état j peut avoir au plus deux successeurs avec une probabilité non nulle, i.e.,

$j^* = (2j) \bmod 2^M$ et $j^{**} = (2j+1) \bmod 2^M$, l'expression (8) peut être simplifiée

$$H_S = \sum_{j=0}^{2^M-1} p_j \, H(P_{\Sigma_2|\Sigma_1}(j^*|j)) \qquad \textbf{(9)}$$

Exemple: Considérons une source qui émet des bits indépendants et symétriquement distribués sauf quand deux bits consécutifs sont identiques, dans ce cas le bit suivant est différent avec la probabilité 0.8. Par exemple, quand deux 0 arrivent, le bit suivant est 1 avec la probabilité 0.8 et 0 avec la probabilité 0.2, mais quand la paire 01 arrive, le bit suivant est 0 ou 1 , tous les deux avec la probabilité 0.5.

Cette source est une *source ergodique stationnaire* avec la mémoire $M = 2$ et il est facile de vérifier que l'on donne les probabilités de transition d'état $P_{\Sigma_2|\Sigma_1}(0|0) = 0.2$, $P_{\Sigma_2|\Sigma_1}(1|0) = 0.8$, $P_{\Sigma_2|\Sigma_1}(2|1) = 0.5$, $P_{\Sigma_2|\Sigma_1}(3|1) = 0.5$, $P_{\Sigma_2|\Sigma_1}(1|2) = 0.5$, $P_{\Sigma_2|\Sigma_1}(3|2) = 0.5$, $P_{\Sigma_2|\Sigma_1}(1|3) = 0.8$, $P_{\Sigma_2|\Sigma_1}(3|3) = 0.2$.

Les probabilités stationnaires d'état peuvent être obtenues comme une solution du système (2) et (3) : $p_0 = p_3 = 5/26$ et $p_1 = p_2 = 4/13$

L'entropie par-bit est selon (9) égale à 2 (5/26) H (0.2) + 2 (4/13) H (0.5) = (5/13).0.7219 + (8/13).1= 0.893.

La sortie de cette source a ainsi une redondance de 10.7%.

4.3 Description du test statistique universel

Ce test a été conçu en 1992 par Ueli Maurer au département d'informatique de l'université Princeton, et est étroitement liée à l'entropie " la mesure de la qualité d'une source de la clef secrète dans une application cryptographique ".

Ueli Maurer a montré que les tests statistiques sur les suites aléatoires pouvaient se déduire de tentatives de compresser la suite. Une suite aléatoire est bien sûr non compressible car son entropie est maximale, une compression cherchant à éliminer les redondances du message.

Le test mesure la quantité réelle par laquelle la sécurité d'un système de chiffre serait réduite si le générateur évalué *G* avait été employé comme source de clef, i.e., il mesure la taille efficace de la clef μ_G (n, ½) d'un système de chiffrement avec source de clef *G*. Donc, les défauts statistiques sont pondérés selon les dégâts potentiels qu'ils pourraient causer dans une application cryptographique.

Le test se propose de mesurer la signification réelle cryptographique d'un défaut. Le test n'est pas conçu pour détecter un type de défaut statistique, par contre, il doit être capable de détecter des classes générales de défauts statistiques pouvant être modélisés par une source ergodique stationnaire avec mémoire finie. C'est pourquoi Maurer affirme que le test englobe les quelques tests standards statistiques, c'est en ce sens qu'il est qualifié d'universel.

Le test est de type compression basé sur l'idée de Ziv. Le générateur doit passer le test si et seulement si la séquence de sortie ne peut pas être compressée significativement.

4.3.1 Principe du test

L'idée de base est qu'Il n'est pas possible de compresser significativement (sans perte d'information) la séquence de sortie d'un générateur binaire aléatoire. Donc, si une séquence produite par un tel générateur peut l'être, ce générateur doit être rejeté. Au lieu de compresser la séquence actuelle, le test universel calcule plutôt une quantité relative à la longueur de la séquence compressée.

Le test est universel au sens où il est capable de détecter tout défaut d'un générateur binaire aléatoire parmi une large classe de défauts.

Le test exige une séquence de bits suffisamment longue (de l'ordre de $10.2^L + 1000.2^L$ pour $6 \leq L \leq 16$) qui est divisée en deux blocs adjacents de L-bits ($6 \leq L \leq 16$), Q blocs d'initialisation ($Q \geq 10.2^L$) et K blocs de test ($K \approx 1000.2^L$). Nous prenons $K = [n/L] - Q$ pour maximiser sa valeur. L'ordre de grandeur de Q doit être choisi de manière à assurer que tous les modèles de fichiers binaires de L-bits possibles puissent être contenus dans le

bloc d'initialisation. Le test n'est pas convenu pour les très grandes valeurs de L parce que l'initialisation prend un temps exponentiel.

4.3.2 Algorithme du test

Cet algorithme calcule une statistique f_n, pour une séquence d'échantillonnage $s = s_0, s_1,\ldots, s_{n-1}$ à utiliser dans le test universel. La séquence est alors partitionnée en blocs de L-bits en éliminant les bits qui ne forment pas un bloc complet de L-bits. Le nombre total des blocs est Q+K, où Q et K sont définis ci-dessus.

Pour chaque i, $1 \leq i \leq Q+K$, soit b_i l'entier dont la représentation binaire est le ieme bloc. Les blocs sont balayés dans l'ordre. On conserve une table T telle qu'à chaque étape, T_j est la position de la dernière occurrence du bloc correspondant à l'entier j, $0 \leq j \leq 2^L-1$. Les Q premiers blocs de s sont utilisés pour initialiser la table T. Q doit être choisi au moins égal à 10.2^L de manière à avoir une grande vraisemblance que chacun des 2^L blocs de L bits apparaît au moins une fois dans les Q premiers blocs. Les K blocs restants sont utilisés pour définir la statistique du test. Pour chaque i, $Q+1 \leq i \leq Q+K$, soit $A_i = i - T[b_i]$, c'est le nombre de positions depuis la dernière occurrence du bloc b_i, alors $f_n = \frac{1}{K}\sum_{i=Q+1}^{Q+K} log_2 A_i$.

Algorithme :

Entrée : une séquence binaire s de longueur n et les paramètres L, Q, K.

Sortie : la valeur de la statistique f_n pour la séquence s.

1. Initialisation de la table. Pour j = 0 à 2^L-1 faire $T[j] \leftarrow 0$
2. Initialiser la table T. Pour i =1 à Q faire $T[b_i] \leftarrow i$
3. Initialiser la variable somme : somme $\leftarrow 0$
4. Pour i = Q+1 à Q+K faire
   4.1 somme $\leftarrow$ somme + $\log_2 (i-T[b_i])$
   4.2 $T[b_i] \leftarrow i$
5. $f_n \leftarrow$ somme / K.
6. Retour à f_n.

4.3.3 But du test

Le but du test est de détecter si la séquence peut être significativement compressée sans perte d'information. Une séquence significativement compressible est considérée pour être non aléatoire.

4.3.4 Appel de fonction

L La longueur de chaque bloc.

*Note : l'utilisation de L comme la taille de bloc n'est pas compatible avec la notation de taille de bloc (M) employée pour les autres tests. Cependant, l'utilisation de L comme la taille de bloc a été spécifiée dans la source originale du test de Maurer.

Q Le nombre de blocs de la séquence d'initialisation.

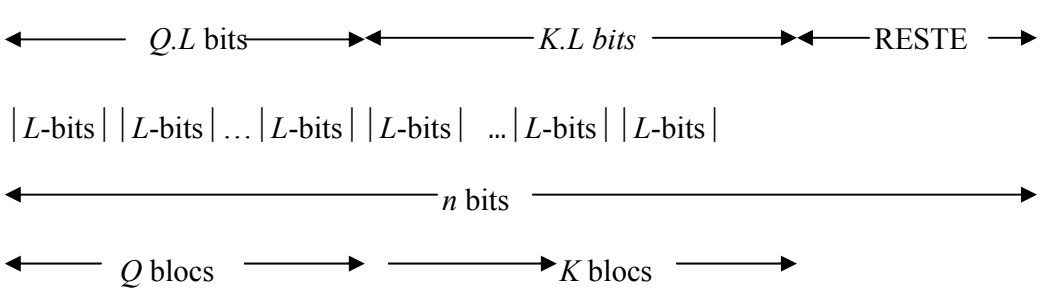
n La longueur de la série de bits.

La distribution de référence pour le test statistique est la distribution semi normale (variante de la distribution normale unilatérale).

4.3.5 Description du test

(1) La séquence de n-bits (s) est divisée en deux segments, le segment d'initialisation qui est constitué des Q (L-bits) et le segment du test qui est constitué des K (L-bits). Les bits restants à la fin qui ne forment pas un bloc de L-bits complet sont éliminés.

Segment d'initialisation **Segment du test**



Les premiers blocs de *Q* sont utilisés pour initialiser le test. Les *K* blocs restants sont les blocs du test ($K = [n/L] - Q$).

Par exemple si $s = 01011001001010101101$, donc $n = 20$.

Si $L = 2$ *et* $Q = 4$ donc $K = [n/L] - Q = [20/2] - 4 = 6$. Le segment d'initialisation est 01011001; le segment du test est 001010101101.

Les blocs de *L*-bits sont représentés dans le tableau suivant :

Bloc	Type	Contenu
1	Segment d'initialisation	01
2		01
3		10
4		01
5	Segment du test	00
6		10
7		10
8		10
9		11
10		01

(2) En utilisant le segment d'initialisation, une table est créé pour chaque valeur de *L*-bits possible (la valeur de *L*-bit est utilisée comme un index dans la table). Le numéro du bloc de la dernière occurrence pour chaque bloc de *L*-bits est noté dans la table (i.e., pour *i* de 1 à *Q*, $T_j = i$, où *j* est la représentation décimale du contenu du *iéme* bloc de *L*-bits).

Pour l'exemple ci-dessus, la table suivante est créée en utilisant les 4 blocs d'initialisation.

	Valeur de *L*-bit possible			
	00 (enregistré dans T_0)	01 (enregistré dans T_1)	10 (enregistré dans T_2)	11 (enregistré dans T_3)
Initialisation	0	4	3	0

(3) On examine chacun des *K* blocs du segment du test et on détermine le numéro du bloc depuis la dernière occurrence du même bloc de *L*-bits (i.e., $i - T_j$). On replace la valeur dans la table avec l'emplacement du bloc actuel (i.e., $T_j = i$). On ajoute le log_2 de la distance calculée entre les re-occurrences du même bloc de *L*-bits à la somme de toutes les différences détectées dans les *K* blocs (i.e., $somme = somme + log_2 (i - T_j)$).

Pour notre exemple la table et la somme cumulative sont développées comme suit :

Pour le bloc 5 (le 1er bloc test) : 5 est placé dans la rangée ″00″ de la table (i.e.,T_0), et somme = $\log_2 (5-0) = 2.3219280948$.

Pour le bloc 6 : 6 est placé dans la rangée ″10″ de la table (i.e.,T_2), et somme = $2.3219280948 + log_2 (6-3) = 2.3219280948 + 1.5849625007 = 3.9068905955$.

Pour le bloc 7 : 7 est placé dans la rangée ″10″ de la table (i.e.,T_2), et *somme* = $3.9068905955 + log_2 (7-6) = 3.9068905955 + 0 = 3.9068905955$.

Pour le bloc 8 : 8 est placé dans la rangée ″10″ de la table (i.e.,T_2), et *somme* = *3.9068905955 + log_2 (8-7) = 3.9068905955 + 0 = 3.9068905955.*

Pour le bloc 9 : 9 est placé dans la rangée ″11″ de la table (i.e.,T_3), et *somme* = *3.9068905955 + log_2(9-0) = 3.9068905955 + 3.1699250014 = 7.0768155969.*

Pour le bloc 10 : 10 est placé dans la rangée ″01″ de la table (i.e.,T_1), et *somme* = *7.0768155969+ log_2 (10-4) = 7.0768155969 + 2.5849625007 = 9.6617780976.*

(4) On calcule le test statistique $f_n = \frac{1}{K} \sum_{i=Q+1}^{Q+K} log_2(i - T_j)$ où T_j est l'entrée de la table correspondant à la représentation décimale du contenu du bloc du ieme bit des *L*-bits. Pour notre exemple $f_n = \frac{9.6617780976}{6} =$ ***1.6102963496***.

(5) On calcule la p-valeur = ***erfc***$\left|\frac{f_n - esperance(L)}{\sigma\sqrt{2}}\right|$ *où* ***erfc*** *est définie par :* *erfc(z)* = $\frac{2}{\sqrt{\pi}} \int_z^\infty e^{-u^2} du$, et l'*espérance* (*L*) et σ sont pris d'une table de valeurs (voir la table qui suit) .

La déviation théorique standard est donnée par $\sigma = c\sqrt{\frac{variance(L)}{K}}$ où

$c = 0.7 - \frac{0.8}{L} + \left[4 + \frac{32}{L}\right]\frac{K^{-3/L}}{15}$.

L	*esperance*	*variance*	*L*	*esperance*	*variance*
1	0.7326495	0.690	9	8.1764248	3.311
2	1.5374383	1.338	10	9.1723243	3.356
3	2.4016068	1.901	11	10.170032	3.384
4	3.3112247	2.358	12	11.168765	3.401
5	4.2534266	2.705	13	12.168070	3.410
6	5.2177052	2.954	14	13.167693	3.416
7	6.1962507	3.125	15	14.167488	3.419
8	7.1836656	3.238	16	15.167379	3.421

Pour l'exemple ci-dessus, p-valeur = ***erfc***$\left|\frac{1.6102963496 - 1.5374383}{\sqrt{2}\sqrt{1.338}}\right|$ = 0.949846.

Remarque 1: on a pris dans cet exemple $L = 2$ pour illustration, car cette valeur n'est pas recommandée pour tester.

Remarque 2: Une version standard de la statistique - la normalisation étant prescrite par le test - est comparée à une gamme acceptable basée sur une densité normale standard (Gaussienne), utilisant la moyenne de la statistique du test qui est donnée par la formule (16) de Maurer (1992) **[10]** :

$$E(f_n) = 2^{-L} \sum_{i=1}^{\infty} (1 - 2^{-L})^{i-1} \log_2 i$$

L'espérance de f_n du test statistique est celle de la variable aléatoire $\log_2 G$ où $G = G_L$ est une variable géométrique aléatoire de paramètre $1 - 2^{-L}$.
Il y a plusieurs versions de formules approximatives empiriques pour la variance de la forme

$$\text{Var}(f_n) = c(L, K)\ \text{Var}(\log_2 G) / K$$

Ici, $c(L, K)$ représente le facteur qui tient compte de la nature de la dépendance des occurrences de calibres. L'une des dernières approximations **[8]** a la forme

$$c(L, K) = 0.7 - \frac{0.8}{L} + \left(1.6 + \frac{12.8}{L}\right) K^{-4/L}$$

Règle de décision (au niveau de 1 %)
Si la p-valeur calculée est < 0.01, on peut conclure que la séquence est non aléatoire. Autrement, conclure que la séquence est aléatoire.

Conclusion et interprétation des résultats du test
Puisque la p-valeur obtenue est $\geq$ 0.01 (la p-valeur = 0.94984), La conclusion est que la séquence est aléatoire.

Remarque : Si f_n diffère significativement de l'espérance (L), alors la séquence est significativement compressible.

Recommandations pour le choix de la taille de la séquence

Les valeurs de *L, Q* et *n* doivent être choisies comme suit **[1]:**

n	*L*	$Q=10.2^L$
$\geq$ 387 840	6	640
$\geq$ 904 960	7	1280
$\geq$ 2 068 480	8	2560
$\geq$ 4 654 080	9	5120
$\geq$ 1 342 400	10	10240
$\geq$ 22 753 280	11	20480
$\geq$ 49 643 520	12	40960
$\geq$ 107 560 960	13	81920
$\geq$ 231 669 760	14	163840
$\geq$ 496 435 200	15	327680
$\geq$ 1 059 061 760	16	655360

Chapitre 5

APPLICATION

5. APPLICATION

5.1 Programme en langage C du test de MAURER pour L = 8 bits.

```
#include <stdio.h>
#include <math.h>
void main()
{
float esperance=7.1836656;
int l=8;
int v=256;
int q=2000;
int k;
int i,n;
int tab[256];
float somme=0,ftu;
unsigned char serie1[22000];
int nb_oct=0 ;

FILE *fout;
fout=fopen,"fichier","r") ;
for(i=0;i<22000;i++)serie1[i]=0;

i=0;
while(!feof(fout)&&i<22000){
      int c=fgetc(fout));
      serie1[nb_oct]=c;
      nb_octi++;
      }
fclose(fout);
if(i<22000)
printf("  Suite trop petite !!! \n La longueur de la suite doit être supérieure à 22000 octets
\n");

else{
         k=22000-q;
         for(i=0;i<256;i++)     tab[i]=0;
         for(i=1;i<=q;i++)
               tab[serie1[i-1]]=i;
         for(i=q+1;i<=q+k;i++) {
                  somme=somme+log(i-tab[serie1[i-1]]);
                  tab[serie1[i-1]]=i;
         }

ftu=(somme/k)/log(2);
printf("ftu = %f",ftu);

}
}
```

5.2 Application du programme

On teste une séquence *s* générée à l'aide de la fonction [Random], de longueur n =176000 bits, en utilisant les programmes de 4 tests décrits précédemment.

1. **Test de fréquence (test mono bit)**

n =176000 bits

S_n=14 (en convertissant les zéros et les 1 de la séquence *s* en (-1) et (+1), $S_n = X_1+X_2+...+X_n$ où $X_i=2s_i-1$).

$S_{obs} = \frac{|S_n|}{\sqrt{n}} = 0.094387980744853$

p-valeur = ***erfc*** $\left(\frac{S_{obs}}{\sqrt{2}}\right) = 0.924801$. La p-valeur ≥ 0.01, donc la séquence *s* est aléatoire.

2. Test de fréquence dans un bloc

Nombre de blocs $N = 8$, nombre de bits par bloc $M = 22000$, nombre de bits n =176000.

On détermine la proportion π_i des 1 dans chaque bloc en utilisant l'équation :

$$\pi_i = \frac{\sum_{j=1}^{M} s_{(i-1)M+j}}{M} \text{ pour } 1 \leq i \leq N$$

On calcule la statistique χ^2 : $\chi^2(obs) = 4M\sum_{i=1}^{N}\left(\pi_i - \frac{1}{2}\right)^2 = 5.33634220043197$.

p-valeur = ***igamc*** $(N/2, \chi^2(obs)/2) = 0.717799$. La p-valeur ≥ 0.01, donc la séquence *s* est aléatoire.

3. Test de series (run test)

n = 176000, on calcule la proportion des 1 dans la séquence: $\pi = \frac{\sum_j s_j}{n} = 0.50071346759796$

$V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1 = 88043$ ($r(k) = 0$ si $s_k=s_{k+1}$, et $r(k) = 1$ autrement).

p-valeur = ***erfc*** $\left[\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n}\pi(1-\pi)}\right] = 0.836909$. La p-valeur ≥ 0.01, donc la séquence *s* est aléatoire.

4. Test de Maurer

n =176000, f_n= 7.18794250488281, espérance (L) = 7.1836656 et σ=3.238 (pour L = 8).

p-valeur = ***erfc*** $\left(\left| \frac{f_n - espérance(L)}{\sqrt{2}\sigma} \right| \right)$ = 0.998104. La p-valeur ≥ 0.01, donc la séquence s est aléatoire.

On peut résumer ces résultats comme suit :

Test	**p-valeur**
Test de fréquence	0.924801
Test de fréquence dans un bloc	0.717799
Test de series (run test)	0.836909
Test de Maurer	0.998104

On remarque que la séquence s passe tous les tests, par conséquent elle est aléatoire. La p-valeur du test de Maurer est la plus proche de 1.

CONCLUSION

Le test statistique universel d'UELI MAURER est un test particulier par rapport aux autres tests, il est basé sur un modèle statistique plus général que ceux précédemment considérés dans le contexte de tests statistiques, à savoir une source ergodique stationnaire avec une mémoire $M \leq L$, où L est un paramètre du test. Le test mesure l'entropie qui est un élément fondamental dans la théorie de l'information auquel d'autres tests statistiques font seulement allusion. Il est capable de détecter tout défaut d'un générateur binaire aléatoire parmi une large classe de défauts (inclus les 5 défauts détectables par les tests de base). Le test donne une mesure correcte de la qualité cryptographique d'une source clef, comme il mesure la taille efficace d'un système de chiffrement. Parmi les inconvénients de ce test, il n'est pas convenu pour les grandes valeurs de L parce que l'initialisation prend un temps exponentiel, et il nécessite une séquence d'échantillonnage plus longue pour être efficace ($n \geq 387\,840$ pour $L = 6$).

Références

[1] **A**ndrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, San Vo, A statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST Special Publication 800-22 (avec révision datée du 15 mai(2001).

[2] **A**.Menezes, P.van Oorschort, et S.Vanstone, Handbook of Applied Cryptography, CRC Press, 1996. Disponible gratuitement sur http://cacr.math.uwaterloo.ca/hac/.

[3] **C**.Shannon, A mathematical theory of communication, The Bell system technical journal, vol.27, pp.379-423, 623-656, Juillet-octobre 1948.

[4] **F**IPS 140-1, Security requirements for cryptographic modules, Federal Information Processing Standards Publications 140-1, U.S Departement of commerce / N.I.S.T, National Technical Information Service, Springfield, Virginia, 1994.

[5] **G**.Florin, S.Natkin, les techniques de la cryptographie, cryptographie cours CNAM, Janvier 2001.

[6] **J**ames L. Massey, Some Applications of Source Coding in Cryptography,
Paper from European Trans. on Telecomm., Vol. 5, pp. 421-429, Juillet-Aout 1994.

[7] **J**.M.Wozncraft et B.Reiffen, sequential decoding, Cambridge, MA:Techn. Press of the M.I.T., 1960.

[8] **J**.S.Coron et D.Naccache, "An Accurate Evaluation of Maurer's Universal Test" proceedings of SAC'98, Springer-Verlag, 1998. Disponible sur http://www.eleves.ens.fr:8080/home/coron/index.html.

[9] **P**ierre L'Ecuyer, "uniform random number generation", apparu dans "annals of operations research", 1994.

[10] **U**eli Maurer, A universal statistical test for random bit generators, journal of cryptology, vol.5, no.2, pp.89-105, 1992.

[11] http://www.nist.gov/

[12] http://www.stat.ucl.ac.be/