Analyse des générateurs de nombres aléatoires dans des conditions anormales d'utilisation

Mathilde Soucarros

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques**

Arrêté ministériel : 7 août 2006

Présentée par

Mathilde SOUCARROS

Thèse dirigée par **Philippe ELBAZ-VINCENT**

préparée au sein de l' **Institut Fourier**
dans l'**École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**
et du **Centre d'Évaluation de la Sécurité des Technologies de l'Information**

Analyse des générateurs de nombres aléatoires dans des conditions anormales d'utilisation

Thèse soutenue publiquement le **15 octobre 2012**,
devant le jury composé de :

M. Louis GOUBIN
Professeur à l'Université de Versailles Saint-Quentin-en-Yvelines, Président
M. Jean-Claude BAJARD
Professeur à l'Université Pierre et Marie Curie, Rapporteur
M. Lionel TORRES
Professeur à l'Université de Montpellier 2, Rapporteur
M. Jessy CLÉDIÈRE
Ingénieur au CEA-Leti, Examinateur
Mme Cécile DUMAS
Ingénieur au CEA-Leti, Examinatrice
M. Antoine JOUX
Ingénieur à la DGA, Examinateur
M. Yannick TEGLIA
Ingénieur à STMicroelectronics, Examinateur
M. Philippe ELBAZ-VINCENT
Professeur à l'Université Joseph Fourier, Directeur de thèse

Remerciements

Je voudrais tout d'abord remercier ceux sans qui cette thèse n'aurait jamais existé, Frédéric Valette et Denis Réal de la DGA. Merci aussi à Denis pour son encadrement même s'il a tourné un peu court !

Cette thèse a été réalisée au sein du CESTI au CEA-Leti et je tiens à remercier tous les membres de ce laboratoire pour leur chaleureux accueil ainsi que leur aide pour la réalisation de mes travaux. En particulier, un grand merci à Cécile Dumas et Jessy Clédière, mes encadrants, pour m'avoir guidée durant ces trois années.

J'aimerais aussi remercier les personnes avec qui j'ai été en contact dans mon laboratoire universitaire, l'Institut Fourier, pour leur précieuse assistance. Tout particulièrement, merci à Philippe Elbaz-Vincent pour avoir dirigé cette thèse et pour ses conseils avisés.

Je tiens bien sûr à remercier Louis Goubin pour avoir présidé mon jury de thèse, ainsi que Jean-Claude Bajard et Lionel Torres pour avoir accepté de rapporter mon manuscrit. Merci aussi à Antoine Joux et Yannick Teglia pour leur participation comme examinateurs durant ma soutenance.

Je voudrais aussi remercier mes proches pour leur soutien attentionné. Enfin, j'ai une petite pensée pour tous mes camarades thésards : courage, la fin est plus proche que vous ne le pensez !

Table des matières

Table des figures

Liste des tableaux

Abréviations

AES	Advanced Encryption Standard
ANSI	American National Standard Institute
ASIC	Application-Specific Integrated Circuit
BBS	Blum-Blum-Shub
BSI	Bundesamt für Sicherheit in der Informationstechnik
CBC	Cipher Bloc Chaining
CESTI	Centre d'Évaluation de la Sécurité des Technologies de l'Information
CSPRNG	Cryptographically Secure Pseudo-Random Nunmber Generator
DAS	Digitalised Analog Signal
DEMA	Differential ElectroMagnetic Analysis
DES	Data Encryption Standard
DPA	Differential Power Analysis
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECDSA	Elliptic Curve Digital Signature Algorithm
EM	ÉlectroMagnétique
FIPS	Federal Information Processing Standards
FPGA	Field-Programmable Gate Array
IV	Initialisation Vector
LED	Light-Emitting Diode
LFSR	Linear Feedback Shift Register
MM	Multiplication de Montgomery
NIST	National Institute of Standards and Technology
PLL	Phase-Locked Loop
PRNG	Pseudo-Random Number Generator
QKD	Quantum Key Distribution

QRNG	Quantum Random Number Generator
RAM	Read Access Memory
RNG	Random Number Generator
RSA	Rivest-Shamir-Adleman
RTD	Resistance Thermometer Detector
SEMA	Simple ElectroMagnetic Analysis
SHA	Secure Hash Algorithm
SPA	Simple Power Analysis
TDEA	Triple Data Encryption Algorithm
TRNG	True Random Number Generator
VCO	Voltage Controlled Oscillator
XOR	eXclusive OR

Chapitre 1

Introduction

Les générateurs de nombres aléatoires ont de nombreuses applications dans des domaines très différents. Une utilisation populaire de ces nombres intervient dans les jeux de hasard : tirages de dés, de cartes, roulette, loterie, etc. Ceci a amené à appeler les méthodes utilisant des nombres aléatoires les méthodes de Monte-Carlo, en référence à ses casinos.

Avec l'avènement des ordinateurs nous avons vu arriver de nouvelles techniques pour simuler et prédire le comportement de phénomènes naturels comme des séismes (trajectoire d'un tsunami), la météo, le comportement d'une foule de personnes, etc. ou reproduire le fonctionnement de systèmes plus industriels par exemple dans le nucléaire, l'automobile ou l'aérospatial. Le système étudié est modélisé et les variables décrivant son état et son évolution sont réunies. Ces variables sont régies par des équations représentant les lois de la physique auxquelles est soumis le système. La simulation numérique du système consiste à déterminer les valeurs de ses paramètres en un certain nombre de points/moments. L'évolution du système peut être caractérisée par l'apparition de certains évènements ayant différentes probabilités d'apparition suivant une loi de distribution. Dans ce contexte, durant la simulation les évènements sont décidés par le tirage d'un nombre aléatoire.

Les nombres aléatoires peuvent aussi aider à tester des programmes informatiques. Ils sont générés dans des tests unitaires afin de vérifier le bon fonctionnement d'un programme en variant ses paramètres. De la même manière ils permettent d'obtenir une caractérisation du comportement d'un programme.

Au contraire, il est quelquefois avantageux (en temps et/ou en ressources) de ne tester qu'une partie des cas possibles et cela implique donc de choisir un échantillon approprié, de manière aléatoire dans l'ensemble de tous les cas possibles.

Les méthodes de Monte-Carlo et les manières de générer des nombres aléatoires qui leur sont adéquates sont expliquées en détail dans [Gen04]. Elles sont généralement utilisées afin de résoudre un problème difficile comme par exemple un système d'équation. Une simulation de Monte-Carlo évalue une expression mathématique à partir d'expériences avec des nombres aléatoires. Très souvent elle permet d'approcher la valeur d'une intégrale de dimension supérieure à 1. Par exemple nous considérons l'intégrale suivante :

$$\int_a^b f(x)dx \tag{1.1}$$

Cette intégrale peut être estimée en prenant m variables aléatoires y_i de distribution uniforme dans $[a,b]$ et en calculant :

$$(b-a)\sum_{i=1}^{m}\frac{f(y_i)}{m} \tag{1.2}$$

Dans la même famille nous allons trouver les algorithmes de Las Vegas. Contrairement aux mé-

thodes de Monte-Carlo pour lesquelles il existe une inexactitude sur le résultat obtenu, ces algorithmes donnent toujours un résultat correct. Une autre différence est le temps de calcul qui est déterministe avec les méthodes de Monte-Carlo et aléatoire avec les algorithmes de Las Vegas. Leur principe est l'utilisation de paramètres choisis aléatoirement. Par exemple l'algorithme de tri rapide (quicksort) est un algorithme de Las Vegas. En effet il choisit des pivots de manière aléatoire pour trier les données d'un tableau.

Un important domaine dans lequel les nombres aléatoires jouent un rôle vital est la cryptographie. Elle évoque en effet des notions de secret ou de hasard qui entrainent l'utilisation d'aléa. Elle est en particulier exploitée pour sécuriser des transactions bancaires (carte de paiement) ou des informations (carte Vitale, passeport, etc.) par exemple. La cryptographie impose des conditions plus restrictives sur la qualité de l'aléa utilisé et c'est dans ce contexte que se place cette thèse.

La cryptologie, la science du secret, est composée de la cryptographie qui comprend l'écriture des secrets et de la cryptanalyse qui les étudie. La cryptographie a une longue histoire, avec des exemples célèbres comme le chiffre de César ou la machine Enigma. Depuis l'explosion des télécommunications elle est maintenant très présente dans la vie courante, notamment avec le système bancaire et les cartes à puce.

Le principe d'origine de la cryptographie était de transmettre un message en toute confidentialité, en le chiffrant puis en le déchiffrant comme indiqué sur la figure 1.1. De nos jours elle doit souvent permettre :

- l'authentification (assurer l'identité de la personne envoyant un message),
- l'intégrité (assurer que le message n'a pas été modifié),
- la non-répudiation (l'auteur d'un message ne doit pas pouvoir nier qu'il l'a envoyé).

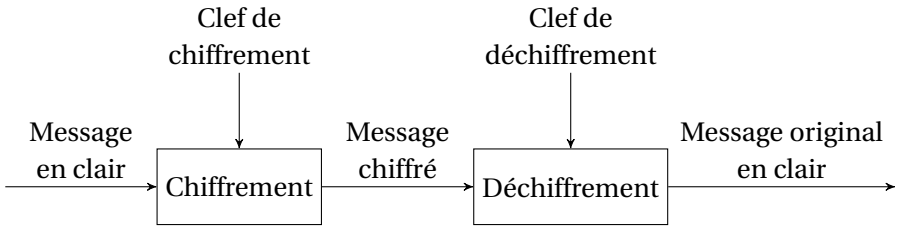


FIGURE 1.1 – Processus de chiffrement et déchiffrement d'un message.

En général la cryptographie est mise en œuvre par des algorithmes publics utilisant des clefs sur lesquelles repose la sécurité du processus. Les opérations de chiffrement et de déchiffrement se font avec deux clefs k_1 et k_2 telles que :

$$\begin{aligned} C_{k_1}(\mathscr{M}) &= \mathscr{C} \\ D_{k_2}(\mathscr{C}) &= \mathscr{M} \end{aligned} \tag{1.3}$$

avec $\mathscr{M}$ le message en clair et $\mathscr{C}$ le message chiffré.

Il existe deux types d'algorithmes. Les algorithmes à clef secrète ont leur clef de chiffrement qui peut être calculée à partir de la clef de déchiffrement en un temps polynomial et vice-versa. Dans ce cas k_1 et k_2 peuvent être identiques. Les algorithmes à clef publique ont leur clef de déchiffrement (clef privée) qui ne peut pas être calculée à partir de la clef de chiffrement (clef publique) en un temps raisonnable.

Ces algorithmes nécessitent des clefs secrètes qui ne puissent pas être retrouvées par une recherche exhaustive. C'est-à-dire que toutes les clefs possibles doivent avoir la même probabilité d'apparition. Les générateurs de nombres aléatoires sont donc utilisés pour produire de telles clefs.

Une autre application classique des nombres aléatoires est la signature numérique. En effet, si une signature «papier» permet de s'assurer de l'authenticité d'un document, il est tout aussi nécessaire de

faire une telle vérification pour un document numérique. Une signature numérique graphique étant facile à falsifier ou simplement copier, un autre mécanisme d'authentification est mis en œuvre. Le processus de signature et de vérification d'un message est décrit ci-dessous :

$$\begin{aligned} &S_{k_1}(\mathscr{M}) = \mathscr{C} \\ &V_{k_2}(\mathscr{C}, \mathscr{M}) = \text{vrai} \end{aligned} \tag{1.4}$$

avec $\mathscr{M}$ le message, $\mathscr{C}$ la signature, k_1 une clef privée et k_2 une clef publique. Les clefs doivent être produites par des générateurs de nombres aléatoires, de manière similaire au cas précédent.

Dans le cryptosystème du masque jetable (one-time pad), une suite de lettres aléatoires de même taille que le message à chiffrer est utilisée. Chaque lettre du masque chiffre une lettre correspondante du message. Le masque ne peut être utilisé qu'une seule fois et doit être produit par un générateur de nombres aléatoires.

Les algorithmes à clef secrète regroupent deux familles : les algorithmes de chiffrement par blocs et les algorithmes de chiffrement par flots. Les algorithmes de chiffrement par blocs, comme l'indique leur nom, réalisent les opérations de chiffrement et de déchiffrement sur des blocs de texte du message. Ceci implique qu'un chiffrement effectué sur des mêmes blocs avec des clefs identiques donne des chiffrés identiques. Afin de remédier à ce problème il est recommandé de chiffrer au préalable les blocs par des vecteurs d'initialisation qui «cassent» cette similarité. Ainsi des messages identiques auront des chiffrés différents, à condition que les vecteurs d'initialisation soient eux-mêmes différents. Un générateur de nombres aléatoires peut être utilisé pour créer ces vecteurs d'initialisation.

Un exemple célèbre de dépendance de la sécurité d'un algorithme à l'aléa des nombres utilisés est le DSA (Digital Signature Algorithm). La génération d'une signature numérique (r,s) avec cet algorithme se fait de la manière suivante :

$$\begin{aligned} r &= (\alpha^k \bmod p) \bmod q \\ s &= (k^{-1}(H(m) + a * r)) \bmod q \end{aligned} \tag{1.5}$$

avec p et q des nombres premiers, $\alpha = g^{\frac{p-1}{q}} \bmod p$ où $g \in \mathbb{Z}^*$, $1 < k < \alpha$ un nombre aléatoire, H une fonction de hachage, m un message et a la clef privée. La vérification de la signature se fait de la manière suivante :

$$\begin{aligned} w &= s^{-1} \bmod q \\ u_1 &= H(m)\, w \bmod q \\ u_2 &= r\, w \bmod q \\ v &= ((g^{u_1} y^{u_2}) \bmod p) \bmod q \end{aligned} \tag{1.6}$$

La signature est rejetée d'emblée si $0 < r < q$ ou $0 < s < q$ et elle est considérée valide si $v = r$. À partir de la connaissance du nombre aléatoire k, un attaquant peut retrouver la clef privée en calculant simplement :

$$a = (sk - H(m))r^{-1} \bmod q \tag{1.7}$$

Il lui est aussi possible de retrouver la clef privée s'il connait deux signatures différentes générées à partir d'un même k et deux messages. Pour cela il calcule tout d'abord k comme indiqué ci-dessous

puis il reprend le calcul précédent :

$$\begin{aligned} s_1 &= (k^{-1}(H(m_1) + ar)) \bmod q \\ s_2 &= (k^{-1}(H(m_2) + ar)) \bmod q \\ s_1 - s_2 &= k^{-1}(H(m_1) + ar) - k^{-1}(H(m_2) + ar) \\ &= k^{-1}(H(m_1) + xr - H(m_2) - xr) \\ &= k^{-1}(H(m_1) - H(m_2)) \\ k &= \frac{H(m_1) - H(m_2)}{s_1 - s_2} \end{aligned} \tag{1.8}$$

Enfin si un attaquant arrive à influencer ou à lire la sortie d'un RNG et connait ainsi quelques bits du nombre aléatoire k alors il peut appliquer une méthode qui consiste à retrouver la clef privée à partir d'un ensemble de signatures [NS00]. Ces travaux ont été transposés au cas de ECDSA, la variante de DSA sur courbes elliptiques [NS03].

En 2010, un groupe de personnes a mis en évidence une anomalie dans la console de jeu PS3 [Fai10]. Celle-ci vérifie la légitimité des jeux utilisés grâce à l'algorithme de signature ECDSA. Elle intègre donc un générateur de nombres aléatoires. L'appel à ce générateur était dans ce cas mal implanté et produisait toujours le même nombre «aléatoire». À partir de la connaissance de ce nombre et de deux signatures, il est possible de retrouver la clef privée. Une fois cette clef connue, il est possible de signer n'importe quel jeu (par exemple piraté) correctement.

Plus récemment, des études se sont penchées sur la redondance des clefs circulant sur internet [LHA+12, Hen12]. Un grand nombre de clefs publiques, notamment du cryptosystème RSA (Rivest-Shamir-Adleman), ont ainsi été récupérées. Cela a permis de trouver la proportion de clefs se recouvrant, ainsi qu'un certain nombre de clefs faibles (facilement calculable). Dans RSA, la clef publique consiste en $N = p * q$ le produit de deux nombres premiers et d'un exposant e. La sécurité de ce cryptosystème repose sur l'impossibilité de factoriser N en un temps raisonnable. En supposant qu'un certain nombre de modules RSA partagent un facteur p identique, il est possible de les factoriser :

$$\begin{aligned} p &= \gcd(N_1, N_2) \\ q_1 &= \frac{N_1}{p} \\ q_2 &= \frac{N_2}{p} \end{aligned} \tag{1.9}$$

Ces travaux ont montré qu'il était possible de réaliser une telle factorisation, grâce à la redondance des facteurs premiers. Cela signifie en pratique que les systèmes générant les nombres premiers (à partir de nombres aléatoires) n'ont pas une entropie suffisante.

Ces exemples illustrent, entre autres, les raisons d'utiliser des générateurs de nombres aléatoires en cryptographie. Un point critique est la qualité des nombres aléatoires générés. En effet dans certains cas il est essentiel pour la sécurité des fonctions cryptographiques mises en œuvre que ces nombres soient parfaitement aléatoires.

Le travail que nous avons effectué et qui est décrit dans cette thèse est orienté d'un point de vue attaquant, sur les possibilités de perturbation des générateurs de nombres aléatoires et leurs conséquences.

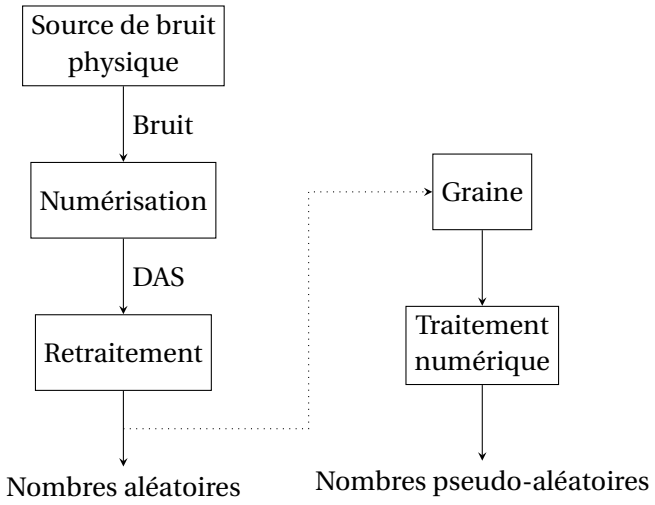
Chapitre 2

Les générateurs de nombres aléatoires

Ce chapitre introduit l'état de l'art des générateurs de nombres aléatoires. Il explique leur fonctionnement général et les différentes conceptions existantes.

2.1 Principe de fonctionnement

Les générateurs de nombres aléatoires (RNG : Random Number Generator) peuvent être réalisés de différentes façons. Ils peuvent être classés dans deux grandes familles : ceux qui extraient de l'aléa à partir de phénomènes physiques (TRNG : True RNG) et ceux qui le créent grâce à des algorithmes déterministes (PRNG : Pseudo-RNG). Il existe aussi des systèmes hybrides qui combinent des techniques de ces deux familles. La figure 2.1 décrit le fonctionnement de ces différents RNG. Les TRNGs produisent de l'aléa à partir d'une source physique d'entropie en transformant des signaux analogiques en numériques (DAS : Digitalised Analog Signal) puis en leur appliquant un retraitement permettant d'améliorer leur qualité. Les PRNGs sont initialisés par une graine «vraiment» aléatoire.



(a) TRNG (b) PRNG

FIGURE 2.1 – Principe de fonctionnement général des RNGs.

2.2 Les générateurs de nombres vraiment aléatoires

Il existe de nombreuses possibilités de construire des générateurs de nombres aléatoires à partir de phénomènes physiques. La plupart de ces générateurs exploitent la gigue de phase d'un ou plusieurs oscillateurs en anneaux. Un autre phénomène aléatoire populaire est l'observation du bruit thermique dans un semiconducteur. Le temps entre des émissions de particules durant une désintégration radioactive est exploité par Hotbits de Walker [Wal96], et il est possible de récupérer des fichiers de nombres aléatoires via son site internet. Disponible aussi sur internet, random.org [Haa98] propose des séquences de nombres aléatoires générées à partir de mesures des variations d'amplitude du bruit atmosphérique. Lavarand était lui un générateur créant des nombres aléatoires à partir de photographies de l'activité de lampes à bulles. Le projet a évolué [NCP01] et le principe est maintenant de faire des photographies dans un espace complètement sans lumière pour ne garder que du bruit. D'autres générateurs sont basés sur les processus relatifs à un ordinateur : turbulence de l'air dans un lecteur de disque [DIF94], variation de son temps de réponse [JSHJ98], mouvements de souris [HLWZ09], etc.

2.2.1 TRNGs exploitant le bruit de semiconducteurs

Nous considérons ici le bruit thermique et le bruit de Schottky (shot noise) dans des semiconducteurs.

Le bruit thermique aux bornes d'un conducteur, dû au mouvement des électrons, a été mesuré pour la première fois par Johnson [Joh28] et expliqué théoriquement par Nyquist en 1928 [Nyq28]. Dans leurs papiers ils montrent que la variation de tension aux bornes d'un conducteur électrique peut être exprimée de la façon suivante :

$$\overline{v^2} = 4k_B TR\Delta f \tag{2.1}$$

où $k_B \simeq 1.3806 \times 10^{-23} J.K^{-1}$ est la constante de Boltzmann, T est la température du conducteur en Kelvin, R est la résistance du conducteur en Ohms, et Δf est la bande passante considérée.

Le bruit de Schottky (shot noise) est la fluctuation du courant due à la variation de son nombre k de porteurs de charge durant un certain temps, qui suit une distribution de Poisson ($p(k) = e^{-\lambda}\frac{\lambda^k}{k!}$ avec λ le nombre moyen d'occurrences des porteurs de charge durant ce temps). Son amplitude est exprimée par :

$$\overline{i^2} = 2eI\Delta f \tag{2.2}$$

où $e = 1,602176565 \times 10^{-19} C$ est la valeur de la charge e^-, I est la valeur du courant en Ampères et Δf est la bande passante considérée.

Les semiconducteurs utilisés sont principalement des résistances et des diodes dont le bruit thermique ou de Schottky est soit amplifié directement soit amplifié de manière différentielle.

2.2.1.1 Amplification directe

La figure 2.2 montre le schéma d'amplification directe d'un bruit thermique d'une résistance. Un faible signal est produit aux bornes de la résistance R, dû à son bruit thermique. L'amplificateur permet d'obtenir un signal de plus grande amplitude. Le comparateur effectue la numérisation du signal de sortie : si le signal amplifié est supérieur à celui de référence alors le bit en sortie est un 1 sinon c'est un 0. Dans le même temps il échantillonne le signal de sortie à chaque coup d'horloge. L'amplification se fait de manière à ce que la composante continue du bruit thermique amplifié soit similaire au signal de référence V_{ref}. En effet ceci permet d'avoir en sortie du comparateur un signal non biaisé. De

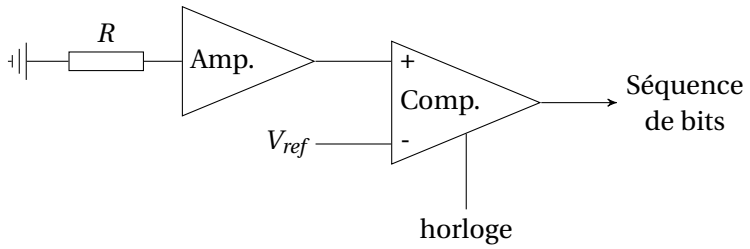


FIGURE 2.2 – Amplification directe du bruit thermique.

tels générateurs sont décrits dans [HCD97, PC00]. Ce circuit doit être utilisé prudemment car le fait que la séquence de bits soit non biaisée repose sur le bon réglage de V_{ref} et les paramètres de l'amplificateur. Une perturbation dans l'environnement du circuit peut déstabiliser cet équilibre. En effet le bruit dans un semiconducteur est très faible et est donc potentiellement négligeable par rapport à d'autres effets comme une variation de l'alimentation par exemple.

2.2.1.2 Amplification différentielle

Afin de remédier à ce problème de déséquilibre, le montage peut être modifié comme indiqué sur la figure 2.3. Au lieu d'utiliser une résistance le circuit en intègre deux (R_1 et R_2). Les bruits thermiques issus de chacune de ces résistances sont soustraits l'un à l'autre puis le résultat est amplifié. Ceci permet de supprimer la composante continue sur le signal en entrée du comparateur. En effet la composante continue du bruit thermique est due à l'environnement autour des résistances qui est ici le même pour R_1 et R_2 et est donc annulée. Ce type de circuit est utilisé dans [TBDSL01].

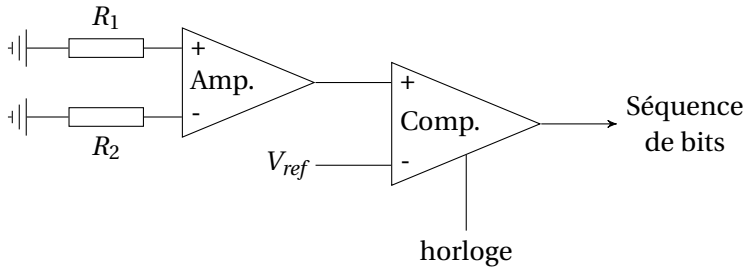


FIGURE 2.3 – Amplification différentielle du bruit thermique.

La branche (−) de l'amplificateur peut aussi provenir d'une boucle à rétroaction provenant de la sortie du comparateur afin de régler automatiquement la valeur de la composante continue à soustraire [ZH01]. Des variations sur ce thème sont possibles avec plusieurs boucles à rétroaction (branche (−) de l'amplificateur, V_{ref}) [HCD97, BGL+03a, WZSL05].

Alternativement, dans [KS08] le circuit d'amplification différentielle est transposé à des diodes Zener au lieu de résistances mais le principe reste le même.

2.2.2 TRNGs basés sur des oscillateurs

Les TRNGs composés d'oscillateurs exploitent le phénomène de gigue de phase (jitter). En effet, dû à l'implantation des oscillateurs dans des FPGA ou ASIC, ceux-ci ont des temps de propagation instables dans les portes logiques, ce qui crée de l'incertitude. Ces différences dans le temps de propagation peuvent être produites par le bruit thermique dans les composants des oscillateurs, par leur

fabrication, etc.

Les circuits à oscillateurs sont sans doute les plus utilisés des TRNG et il existe deux grandes familles qui sont l'échantillonnage d'un oscillateur et les oscillateurs XORés.

2.2.2.1 Échantillonnage d'un oscillateur

Le principe de ce circuit est d'échantillonner un oscillateur de fréquence rapide par une horloge de fréquence lente comme indiqué sur la figure 2.4. En pratique le signal d'horloge peut être réalisé par un autre oscillateur. La fréquence lente doit être réglée de manière à permettre une accumulation suffisante de jitter dans l'oscillateur à fréquence rapide pour garantir assez d'incertitude sur son état au moment où il est échantillonné.

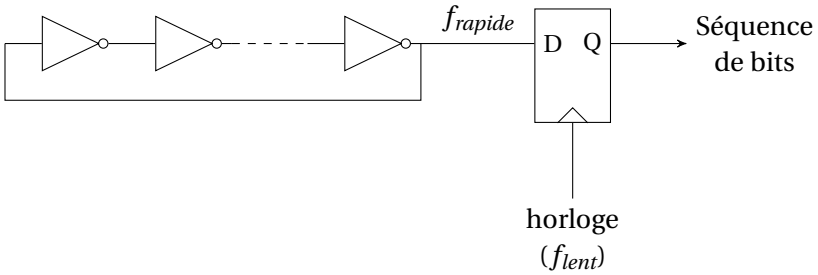


FIGURE 2.4 – Échantillonnage d'un oscillateur à grande fréquence par une horloge de faible fréquence.

Ce type de génération de nombres aléatoires est notamment utilisé par Intel [JK99]. Dans ce générateur le signal d'horloge est généré par un VCO (Voltage Controlled Oscillator), dont la fréquence est contrôlée par un signal produit par amplification directe du bruit thermique dans une résistance, il s'agit donc d'un système mixte bruit/oscillateur. Dans [BGL+03b], une amplification différentielle du bruit thermique dans des résistances est utilisée pour créer un signal oscillant qui sert d'horloge à un oscillateur. Le montage d'amplification du bruit thermique intègre une boucle de rétroaction afin d'augmenter le jitter. Une autre manière d'augmenter le jitter est décrite dans [KG04] où deux oscillateurs de fréquences similaires sont utilisés avec des montages composés de deux bascules D, d'un inverseur et d'un buffer. D'autres montages essaient de contrôler les paramètres du générateur, [LM05] utilise deux oscillateurs et des boucles à rétroaction pour corriger la fréquence rapide d'un oscillateur et la phase de l'autre. Le générateur d'un processeur Via [Cry03] utilise une combinaison de trois oscillateurs pour générer l'horloge. D'autres modifications sont décrites dans le générateur d'Infineon [DJ00] où l'échantillonnage s'effectue après le passage du signal dans la bascule D ce qui permet de réduire le biais. Enfin [TLL03] génère son oscillateur non pas avec des inverseurs mais avec une série de bascules D XORées entre elles. Dans [SDFF06], les oscillateurs sont composés de deux PLLs, dont une a son signal passant dans deux bascules D qui est ensuite échantillonné et l'autre est le signal d'horloge de ces composants. La description de ce TRNG est de plus accompagnée de sa modélisation.

2.2.2.2 Oscillateurs XORés

Une autre configuration classique intégrant des oscillateurs est décrite dans la figure 2.5. Plusieurs oscillateurs tournent en parallèle de façon indépendante, leurs signaux sont XORés et le résultat est échantillonné. L'aléa est créé par le mélange de l'accumulation du jitter de chaque oscillateur. Ce

montage a été longuement étudié dans [SMS07], avec la construction d'un modèle mathématique et la recherche d'une configuration maximisant l'entropie en sortie du TRNG.

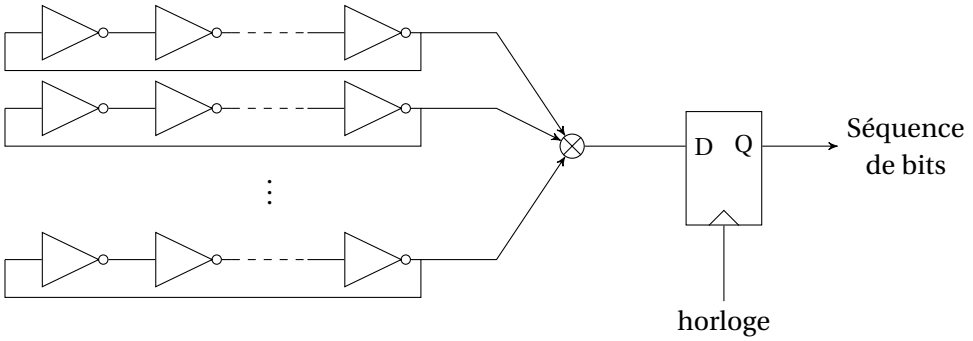


FIGURE 2.5 – Échantillonnage du XOR d'oscillateurs en anneaux tournant en parallèle.

Dans [WT08] le montage est amélioré pour assurer une meilleure prise en charge des transitions dans le XOR et la bascule D. Pour cela des bascules D sont ajoutées entre les oscillateurs et le XOR. Les oscillateurs simples peuvent être remplacés par des oscillateurs de Fibonacci (figure 2.6) pour lesquels la rétroaction est une fonction des bits en sortie des inverseurs, ou des oscillateurs de Galois [Gol06] (figure 2.7) pour lesquels la rétroaction combine les bits à la sortie des inverseurs avec le bit de sortie.

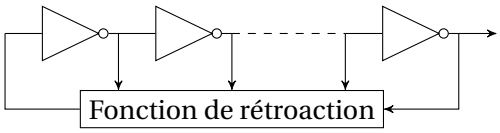


FIGURE 2.6 – Oscillateur de Fibonacci.

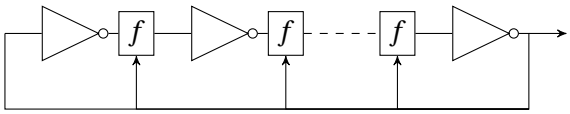


FIGURE 2.7 – Oscillateur de Galois avec une fonction de rétroaction f.

Dans [FD03] un signal provenant d'une PLL (Phase-Locked Loop) est répliqué avec des retards différents et les signaux résultants sont XORés.

2.2.3 TRNGs basés sur des phénomènes quantiques

Un phénomène quantique est considéré comme une expérience pour laquelle nous ne pouvons pas connaître ou même deviner le résultat. La construction de générateurs de nombres aléatoires à partir de phénomènes quantiques (QRNG : Quantic RNG) est un domaine encore peu exploité, sans doute dû aux problèmes de miniaturisation de tels systèmes. Par exemple la désintégration radioactive est un phénomène quantique. Nous présentons ici deux exemples courants de TRNGs se basant sur des expériences d'optique quantique. Le générateur Quantis [ID 10b] commercialisé par ID Quantique exploite l'aléa mis en jeu dans de tels phénomènes.

2.2.3.1 Le miroir semi-réfléchissant

Dans cette expérience, un miroir semi-réfléchissant reçoit des photons envoyés un à un par une source laser. Les photons sont soit réfléchis soit transmis avec des probabilités égales comme indiqué sur la figure 2.8. Les photons sont détectés et un bit est généré, à 0 ou à 1 selon la trajectoire empruntée. Le passage d'un photon précédant n'influence pas le résultat courant. Si le miroir est semi-réfléchissant à une certaine erreur près alors la séquence de sortie a un biais sur la distribution des bits (plus de 0 ou plus de 1). Le faisceau laser est filtré pour ne laisser passer qu'une partie des photons afin de limiter des problèmes de détection. Un circuit électronique exploitant ce montage est décrit dans [JAW+99]. Dans [SGG+00] le photon est dirigé dans une fibre ayant deux branches. Selon la branche empruntée le photon arrive plus ou moins rapidement et cela génère un bit à 0 ou 1.

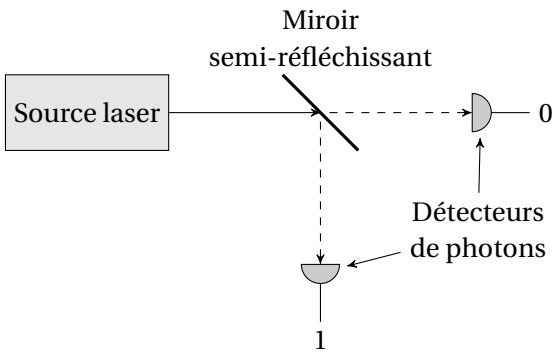


FIGURE 2.8 – Trajectoire d'un photon quand il atteint un miroir semi-réfléchissant.

Une alternative est d'utiliser un polariseur par séparation de faisceau au lieu d'un miroir semi-réfléchissant. Les photons sont polarisés de 45° par rapport à l'axe du polariseur avant de l'atteindre. Dans ce cas les photons ont autant de chance d'être transmis dans une polarisation horizontale (détecteur «0») ou verticale (détecteur »1»). Ce type d'expériences est exploité dans [JAW+99, FSS+06].

Ce montage peut être utilisé un peu différemment en utilisant des paires de photons intriqués. En effet quand ils arrivent sur le polariseur, un des photons est transmis et l'autre est réfléchi. Dans [HQSMD+04], le photon réfléchi arrive sur un miroir semi-transparent et déclenche un détecteur. Le photon transmis annonce l'arrivée du photon réfléchi ce qui permet de s'affranchir des problèmes de détection du montage original et d'utiliser un faisceau laser non atténué. Chaque photon de la paire peut aussi être envoyé sur un polariseur via un miroir semi-réfléchissant comme expliqué dans [FSS+06]. Le bit généré dépend dans ce cas du résultat des quatre détecteurs en place.

2.2.3.2 Le compteur de photons

Dans cette configuration le principe est qu'un détecteur de photons compte les photons émis par une source laser comme représenté sur la figure 2.9. Ceci est lié à la distribution des photons qui est Poissonienne.

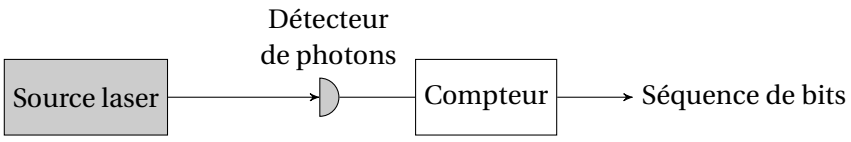


FIGURE 2.9 – Comptage des photons émis par une source laser.

Le comptage des photons se fait dans un intervalle de temps précis et selon que ce nombre est pair ou impair, le bit généré est 1 ou 0 [FWN+10]. Dans [WG09] deux compteurs successifs de bits n_1 et n_2 sont comparés. Comme ils suivent la même loi alors les probabilités que l'un des compteurs soit à zéro tandis que l'autre est supérieur à zéro sont égales. Quand ces conditions sont remplies alors un bit est généré. Une autre exploitation possible de ce montage est de récupérer le temps d'attente entre deux photons qui peut être directement numérisé, générant ainsi plusieurs bits à la fois [WK10].

2.3 Les générateurs de nombres pseudo-aléatoires

Les générateurs de nombres pseudo-aléatoires produisent des nombres qui semblent aléatoires mais qui en réalité ne le sont pas. En effet il s'agit d'algorithmes déterministes qui, à partir d'une graine, génèrent des suites de nombres dont les propriétés statistiques respectent ce qui est attendu de vrais nombres aléatoires. Une même graine produit les même nombres.

Les PRNG ont souvent de meilleures propriétés aléatoires que les TRNG, qui doivent être retraités la plupart du temps. De plus, ils sont en général plus rapides ce qui est important par exemple pour une application de simulation. Par contre, du fait de leur implantation matérielle, le nombre d'états possibles de ces générateurs est limité (par la taille des registres par exemple). En conséquence, en observant leur fonctionnement indéfiniment, il arrive un moment ou les générateurs passent par un état similaire à un état précédent, ils sont donc périodiques.

Dans [BK12], le NIST (National Institute of Standards and Technology) présente ses recommandations pour l'utilisation d'un PRNG. Trois familles d'algorithmes sont considérées appropriées : des fonctions de hachages, des algorithmes de chiffrements par blocs ainsi que des algorithmes basés sur des problèmes de la théorie des nombres.

Nous présentons ici deux méthodes de génération de nombres pseudo-aléatoires très utilisées mais non convenables pour une utilisation en cryptographie, ainsi que des générateurs cryptographiquement sûrs.

2.3.1 PRNGs basés sur les méthodes de congruence linéaire

Les générateurs linéaires congruentiels ont été introduits par Lehmer [Leh51] et sont aujourd'hui encore très populaires. En effet il s'agit de générateurs nécessitant peu d'opérations et la production des nombres est rapide. Une séquence de nombres aléatoires X_n est créée de la manière suivante :

$$X_{n+1} = (aX_n + c) \bmod m \tag{2.3}$$

avec m le module, $0 \leqslant a < m$ le multiplicateur, $0 \leqslant c < m$ l'incrément et X_0 la graine.

La période de la séquence aléatoire est inférieure à m. Un choix judicieux de a, b et m permet de maximiser cette période, notamment en prenant b et m premiers entre eux. Il existe des tables recensant les valeurs de ces paramètres qui maximisent la période. Ces générateurs ne sont toutefois pas utilisables en cryptographie, des études ayant montré qu'ils peuvent être cassés [Boy89]. Il est possible d'utiliser des combinaisons de ces générateurs linéaires congruentiels pour obtenir des séquences de bits avec de meilleures propriétés aléatoires.

2.3.2 PRNGs basés sur les registres à décalage à rétroaction linéaire

La théorie des séquences à registres à décalage est décrite dans [Sel66]. Ces PRNG, aussi appelés LFSR (Linear Feedback Shift Register) sont constitués d'un registre de n bits et d'une boucle à rétroaction intégrant une fonction linéaire de ces bits comme indiqué sur la figure 2.10. À chaque décalage

du registre (un cran vers la droite) le bit b_1 est ajouté à la séquence générée et un nouveau bit est créé dans b_n en combinant tous les bits dans le registre par la fonction de rétroaction linéaire, selon une configuration de Fibonacci.

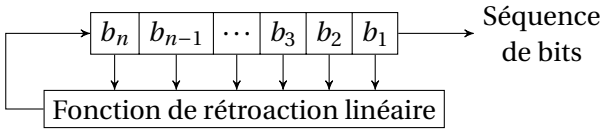


FIGURE 2.10 – Principe de fonctionnement d'un registre à décalage à rétroaction linéaire selon une configuration de Fibonacci.

Certaines combinaisons de ces bits permettent d'obtenir une période maximale de la séquence de bits, c'est-à-dire $2^n - 1$ bits (l'état 0 est exclu car il ne permet pas d'évolution). Pour cela le polynôme formé par les bits de la combinaison en leur ajoutant la valeur 1 doit être un polynôme primitif (par exemple si les bits $b1$, $b3$ et $b4$ sont les entrées de la fonction de rétroaction linéaire alors le polynôme correspondant est $x^4 + x^3 + x^1 + 1$). Le degré du polynôme est la longueur du registre. Un polynôme est primitif de degré n s'il est irréductible et divise $x^{2^n-1}+1$ mais pas x^d+1 pour tout d divisant 2^n-1.

La plus simple fonction de rétroaction linéaire est un XOR de certains bits du registre. Des nombres générés par un tel algorithme ont le défaut d'être très corrélés. De plus les suites de bits sont linéaires et il est possible de retrouver la fonction de rétroaction linéaire à partir de $2n$ bits consécutifs avec l'algorithme de Berkelamp-Massey [Mas69].

Il existe une configuration appelée de Galois où la boucle de rétroaction prend sa source à la sortie de la séquence de bits. Dans cette configuration la fonction de rétroaction linéaire met à jour les bits du registre en prenant comme paramètres d'entrée ces mêmes bits et le bit généré dans la séquence en sortie comme indiqué sur la figure 2.11. Les deux configurations (Fibonacci et Galois) sont équivalentes, à part leur implantation qui est plus ou moins efficace selon les plateformes.

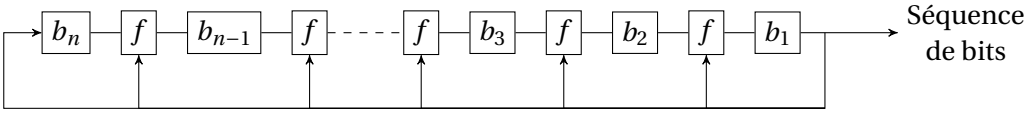


FIGURE 2.11 – Principe de fonctionnement d'un registre à décalage à rétroaction linéaire f selon une configuration de Galois.

De nombreuses combinaisons possibles de ces générateurs existent, en les utilisant par exemple en parallèle et/ou en série et en les joignant par différentes fonctions. La sécurité de ces générateurs dépend de la complexité de la fonction les combinant, et même ainsi elle n'est pas suffisante. Il existe en effet des attaques permettant de retrouver les paramètres de ces générateurs, par exemple l'algorithme de Berkelamp-Massey, des attaques par corrélation (le compromis entre la complexité linéaire et l'immunité à cette attaque est décrite dans [Sie84]) ou des attaques algébriques [CM03, CAAC08].

2.3.3 PRNGs cryptographiquement sûrs

Les CSPRNGs (Cryptographically Secure PRNG) sont des générateurs qui peuvent être utilisés en cryptographie contrairement à ceux décrits précédemment. Un PRNG est dit cryptographiquement sûr s'il passe le test du bit suivant, c'est-à-dire qu'il n'existe pas d'algorithme de temps polynomial tel que connaissant les l premiers bits d'une séquence s on peux en déduire le $(l+1)^e$ bit de s avec une probabilité supérieure à 0.5. Il existe plusieurs techniques pour la construction de tels générateurs.

Le principe peut être d'accumuler de l'entropie via diverses sources et de passer le résultat dans une fonction de hachage. Une autre technique assure la sécurité des générateurs grâce à l'impossibilité de résoudre un certain problème en temps polynomial.

2.3.3.1 Fortuna

Fortuna a été inventé par Bruce Shneier et Niels Ferguson [FS03]. Le générateur est composé de plusieurs parties. Il utilise un algorithme de chiffrement par bloc utilisé en mode compteur c'est-à-dire le chiffrement successif d'un compteur s'incrémentant. La clef de chiffrement est régulièrement modifiée afin d'éviter la périodicité due au nombre d'états limité que peut prendre ce compteur. Un accumulateur d'entropie est constitué de plusieurs sources d'entropie remplissant différents ensembles de nombres aléatoires de manière uniforme. A chaque tirage d'un nouvelle clef pour l'algorithme de chiffrement, un ensemble différent est utilisé. Pour la n-ième clef l'ensemble k est utilisé si 2^k divise n. La clef est générée via deux hachages (SHA-256) de l'ensemble considéré.

2.3.3.2 Blum-Blum-Shub

Le générateur de Blum-Blum-Shub (BBS) [BBS86] est construit de la manière suivante. Soient p et q deux grands nombres premiers congruant à 3 modulo 4, $n = p * q$ est un entier de Blum. Soit x_0 un entier aléatoire premier par rapport à n alors :

$$\begin{aligned} x_i &= x_{i-1}^2 \bmod n \\ &= x_0^{2^i \bmod (p-1)(q-1)} \end{aligned} \tag{2.4}$$

La séquence de sortie du générateur est composée des bits les moins significatifs de chaque x_i. Il a été montré qu'au plus $\log_2(\log_2(n))$ bits peuvent être extraits à chaque itération [VV84]. La sécurité du générateur repose sur la difficulté à factoriser n. Il est dit imprévisible à gauche et à droite car connaissant la suite engendrée par l'algorithme il est impossible de prédire le bit suivant ou précédant de la suite.

2.3.3.3 NIST SP 800-90A

Dans [BK12], le NIST explique des mécanismes à utiliser pour la construction de PRNGs convenables à une utilisation en cryptographie. Il est par exemple possible de baser la conception d'un PRNG sur une fonction de hachage non inversible. Deux mécanismes sont ainsi approuvés par le NIST : Hash_DRBG simple fonction de hachage) et HMAC_DRBG (plusieurs occurrences d'une fonction de hachage combinée avec une clef secrète). Un autre mécanisme met en œuvre des algorithmes de chiffrement par bloc : CTR_DRBG (chiffrement en mode compteur). Enfin la dernière classe de PRNGs citée dans la publication est basée sur des problèmes de la théorie des nombres. Un exemple de ce mécanisme est le Dual_EC_DRBG dont la sécurité est assurée par le problème du logarithme discret sur des courbes elliptiques (trouver a tel que $Q = aP$ avec P et Q des points d'une courbe elliptique d'ordre n).

2.3.3.4 ANSI X9.31 Appendix A.2.4

Ce PRNG a été validé par l'ANSI (American National Standard Institute)[ANS98] afin de générer des clefs pour des algorithmes de signature dans le domaine de la finance. Son principe est le suivant :

$$\begin{aligned} I &= \text{TDEA}_k(\text{DT}) \\ R &= \text{TDEA}_k(I \oplus V) \\ V &= \text{TDEA}_k(R \oplus I) \end{aligned} \tag{2.5}$$

où TDEA est un triple DES (Triple Data Encryption Algorithm), DT est la date, k est la clef et V est une graine aléatoire initialement. Le NIST a approuvé son utilisation en spécifiant l'utilisation d'un AES ou d'un triple DES au cœur du dispositif[Kel05].

2.3.3.5 FIPS PUB 186.2

Le standard DSS (Digital Signature Standard)[FIP00] publié par le Federal Information Processing Standards (FIPS), définit l'algorithme de signature DSA. Il apporte aussi deux solutions pour la génération de nombres aléatoires. Ces algorithmes utilisent des fonctions à sens unique et peuvent être générés par une fonction de hachage SHA-1 ou un DES.

2.4 Les retraitements algorithmiques

Les séquences de nombres aléatoires générées par des TRNG ou des PRNG ne sont pas toujours parfaites. En effet, et plus particulièrement pour les TRNG, ces séquences peuvent être biaisées, corrélées, etc. Ceci ne remet pas en cause l'utilisation de ces générateurs comme sources d'aléa mais met en évidence la nécessité de leur adjoindre un retraitement algorithmique matériel. Il existe de nombreuses manières de construire de tels retraitements, qu'il s'agisse de la correction d'un simple biais ou d'un mixage de la séquence de bits. Quelques retraitements parmi les plus utilisés sont détaillés ci-dessous.

2.4.1 Le correcteur de von Neumann et le ou exclusif

Ces deux algorithmes sont basiques donc faciles à mettre en œuvre mais leurs facultés de correction sont limitées. En effet ces retraitements corrigent seulement un biais dans la distribution des bits, la séquence doit donc avoir ses bits indépendants. L'opération ou exclusif (XOR) traite les bits de la séquence deux à deux et génère un bit pour chaque opération comme illustré dans le tableau 2.1a. Une étude de cette opération pour des séquences de bits (corrélés ou non) biaisées a été faite dans [Dav02]. Le correcteur de von Neumann [vN63] prend lui aussi deux bits en entrée mais ne génère de bit que s'ils sont différents comme indiqué dans le tableau 2.1b.

La théorie de ces correcteurs est simple. On suppose qu'on a une séquence biaisée de bits indépendants avec un biais stable au cours du temps ou identique pour chaque bit. La probabilité d'avoir un bit à 1 dans une telle séquence est $p = \frac{1}{2} + \epsilon$ et la probabilité d'avoir un 0 est $q = \frac{1}{2} - \epsilon$ avec ϵ le biais. Pour l'opération XOR, les probabilités d'obtenir un 0 (Q_{XOR}) et un 1 (P_{XOR}) en sortie sont :

$$P_{XOR} = 2 * p * q = \frac{1}{2} - 2\epsilon^2 \tag{2.6}$$

$$Q_{XOR} = p * p + q * q = \frac{1}{2} + 2\epsilon^2 \tag{2.7}$$

Bit n° 1	Bit n° 2	Bit généré
0	0	0
0	1	1
1	0	1
1	1	0

(a) Opération XOR.

Bit n° 1	Bit n° 2	Bit généré
0	0	-
0	1	0
1	0	1
1	1	-

(b) Correcteur de von Neumann.

TABLE 2.1 – Génération d'un bit à partir d'une paire de bits par des retraitements simples.

P_{XOR} et Q_{XOR} ont toujours un biais en sortie du XOR mais ce biais a été réduit fortement. En effet pour un $\epsilon \leqslant \frac{1}{2}$ petit alors ϵ^2 est encore plus faible. L'opérateur XOR réduit la taille de la séquence de moitié.

Pour le correcteur de von Neumann, les probabilités d'obtenir un 0 (Q_{vN}) et un 1 (P_{vN}) en sortie sont :

$$P_{vN} = p * q = \frac{1}{4} - \epsilon^2 \tag{2.8}$$

$$Q_{vN} = p * q = \frac{1}{4} - \epsilon^2 \tag{2.9}$$

P_{vN} et Q_{vN} sont identiques en sortie du correcteur de von Neumann, le biais est donc complètement éliminé. Le débit en sortie est ici réduit d'au moins un quart car les paires 00 et 11 sont ignorées.

Dans [Dic07] plusieurs XOR sont combinés pour augmenter l'entropie de la source quand elle est biaisée. L'algorithme agit sur des suites de bits en les combinant (XOR) par elles-mêmes en série.

2.4.2 Les fonctions résilientes

Les fonctions résilientes sont dérivées des fonctions booléennes. Elles ont été inventées initialement pour des algorithmes de chiffrement symétrique, notamment pour le chiffrement par bloc et les combinaisons dans les S-box. Leur intérêt réside dans le fait que la connaissance de m bits en entrée d'une telle fonction ne permet pas de faire d'hypothèse sur son bit de sortie. Une fonction booléenne à n variables (fonction de $\{0,1\}^n$ dans $\{0,1\}$) est dite équilibrée si son poids de Hamming est de 2^{n-1}, c'est-à-dire qu'elle prend autant de fois la valeur 0 que 1. Elle est dite sans corrélation d'ordre t si sa distribution de valeurs ne change pas lorsque l'on fixe au plus t entrées. Une fonction équilibrée sans corrélation d'ordre t est dite résiliente d'ordre t. Ces fonctions font l'objet de beaucoup d'études, notamment [CH10] est consacré aux fonctions booléennes. Un gros désavantage est le débit réduit en sortie de ces fonctions.

Un exemple de fonction résiliente peut être trouvé dans [SMS07] où un code BCH (256,16,113) [Hoc59, BRC60] est utilisé. Les bits sont regroupés par blocs de 256 et la fonction résiliente en rend 16. L'ordre de la fonction est 113 ce qui signifie que la fonction corrige au plus 112 bits corrélés en entrée.

2.4.3 Les fonctions cryptographiques

Ces fonctions peuvent aussi être utilisées comme PRNG. Dans le cadre d'un retraitement ce sont leurs propriétés de diffusion (les redondances locales se dissipent à plus grande échelle) et confusion (chaque bit de sortie provient d'une transformation complexe d'un grand nombre de bits d'entrée) qui sont exploitées. En effet elles n'éliminent pas spécialement des biais ou des corrélations mais leur

principe est de mélanger les bits ce qui rend de tels défauts «invisibles». En particulier les fonctions de hachage sont idéales pour une telle tâche.

Une fonction de hachage H prend un message en entrée $\mathscr{M}$ et calcule $h = H(\mathscr{M})$. Si la fonction de hachage est à sens unique alors sachant $\mathscr{M}$ il est facile de calculer h, sachant h il est difficile de calculer $\mathscr{M}$ et sachant $\mathscr{M}$ il est difficile de trouver $\mathscr{M}'$ tel que leurs fonctions de hachage soient identiques. Il faut aussi que la fonction soit résistante aux collisions, c'est-à-dire qu'il est difficile de trouver un $\mathscr{M}$ et un $\mathscr{M}'$ tels quel leurs fonctions de hachage soient identiques.

Chapitre 3

Évaluation de l'aléa

Déterminer si un générateur est aléatoire ou non est un problème délicat. En effet il n'existe pas de test universel pouvant affirmer avec certitude qu'un générateur est aléatoire. Le principe est donc de montrer qu'il n'est pas biaisé en étudiant les propriétés des nombres qu'il génère.

En pratique un générateur aléatoire produit une séquence de nombres ayant des propriétés d'imprédictibilité et d'indépendance, et elle suit une certaine distribution (uniforme en cryptographie, gaussienne en télécommunications, etc.). L'évaluation de la qualité aléatoire d'un générateur passe donc par le contrôle des propriétés de la séquence qu'il génère. Ceci est réalisé grâce à des tests statistiques qui permettent de comparer les performances du générateur étudié par rapport à celles, théoriques, qu'aurait une séquence de bits réellement aléatoires. Les tests se font sur de grandes séquences de bits pour une meilleure représentation du RNG.

Une base des tests statistiques utilisés aujourd'hui a été écrite par Donald Knuth dans «The Art of Computer Programming» [Knu97] et dans [MVO96] quelques tests très simples sont expliqués. Ils ont ensuite été repris et d'autres tests ont été ajoutés dans diverses batteries. Les tests peuvent être choisis suivant la qualité des nombres aléatoires désirée et la taille de la séquence à tester. Pour une application en cryptographie la qualité doit être la meilleure possible mais pour d'autres applications la séquence peut être «imparfaite».

3.1 Principe

Comme expliqué dans [Knu97], les tests statistiques sont formulés de telle manière à tester une hypothèse $\mathscr{H}_0$ qui est que la séquence testée est aléatoire. Le résultat de chaque test permet de prendre une décision sur l'acceptation ou le rejet de $\mathscr{H}_0$.

Une statistique permettant de prendre une telle décision est choisie pour chaque test et sa distribution théorique est calculée. Une valeur critique est déterminée à partir de la distribution, de manière générale à ses extrémités où les valeurs expérimentales suivent moins bien la théorie. La statistique est calculée à partir de la séquence sous test et si elle dépasse la valeur critique $\mathscr{H}_0$ elle est rejetée sinon elle est acceptée. Ceci repose sur le fait que pour une séquence vraiment aléatoire la statistique a peu de chances de s'éloigner de sa distribution théorique et donc de dépasser la valeur critique.

En réalité la statistique dépassera sa valeur critique un certain nombre de fois et $\mathscr{H}_0$ sera rejetée alors que la séquence est aléatoire. La probabilité de cet évènement est appelée α ou erreur de Type I. Au contraire la probabilité d'accepter $\mathscr{H}_0$ alors que la séquence est non aléatoire est appelée β ou erreur de Type II et n'a pas de valeur fixe. En effet, la détection d'un tel phénomène ne dépend que des tests (non exhaustifs) existants. Comme nous n'avons pas garantie qu'ils détectent tous les défauts

possibles, β peut donc prendre une infinité de valeurs. La valeur α peut être choisie en fonction de l'usage qui est fait des séquences aléatoires, en général elle est comprise dans [0,001;0,05]. En effet si α est importante alors plus de séquences seront rejetées et donc plus les tests sont durs à passer (pour $\alpha = 0{,}001$ une séquence sur mille est rejetée, pour $\alpha = 0{,}05$ il s'agit de cinq séquences sur cent).

3.2 Tests statistiques existants

Nous présentons ici des tests communément utilisés afin d'évaluer les performances des RNG. Ils sont organisés dans différentes batteries.

3.2.1 Le programme ENT

Ce programme a été créé par Walker en 1985 [Wal08]. Il permet d'avoir un aperçu rapide de certaines propriétés de la séquence testée grâce à six tests très simples. Ceux-ci calculent des statistiques sur les séquences entières et permettent donc d'avoir un aperçu global de quelques unes de leurs propriétés.

3.2.1.1 L'entropie

Dans le programme, l'entropie calcule la quantité d'information contenue dans un fichier. Le fichier est considéré comme une suite de mots de 1 ou 8 bits. L'entropie est calculée comme indiqué ci-dessous :

$$H(X) = -\sum_{i=0}^{2^n-1} P_i(X)\log_2(P_i(X)) \tag{3.1}$$

où X est la source étudiée, P_i est la probabilité d'apparition du mot i de n bits.

Le calcul de l'entropie rend le nombre minimum de bits par mot contenant toute l'information. Par exemple si l'entropie est de 6 bits/mot pour des mots de 8 bits alors 2 bits portent de l'information redondante et le fichier pourrait être compressé pour atteindre trois quarts de sa taille originale.

3.2.1.2 Le test du khi deux

Aussi appelé test d'adéquation («goodness-of-fit»), ce test exprime la capacité d'une variable aléatoire à suivre son comportement théorique. Il est très sensible aux déviations qui peuvent apparaître. La formule utilisée est la suivante :

$$\chi^2 = -\sum_{i=0}^{2^n-1} \frac{N_i - Nt_i}{Nt_i} \tag{3.2}$$

où N_i est le nombre d'occurrences du mot i de n bits et Nt_i est sa fréquence d'apparition théorique.

Le programme calcule aussi un pourcentage indiquant la fréquence à laquelle une séquence de nombres vraiment aléatoires excéderait cette valeur. Ceci représente un degré de suspicion sur le fait que la séquence testée est aléatoire. Si le pourcentage est supérieur à 99% ou inférieur à 1% alors la séquence n'est pas considérée aléatoire, entre 95% et 99% ou entre 1% et 5% il y a une suspicion, entre 90% et 95% ou entre 5% et 10% la suspicion est faible.

La fonction de densité de probabilité d'une variable aléatoire ayant une distribution de khi deux

avec v degrés de liberté est définie par :

$$f(x) = \begin{cases} \frac{1}{\Gamma(\frac{v}{2})2^{\frac{v}{2}}} x^{\frac{v}{2}-1} e^{-\frac{x}{2}}, & 0 \leq x < \infty \\ 0, & x < 0 \end{cases} \tag{3.3}$$

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx, \quad t > 0$$

où v est égal au nombre de mots possibles de la variable aléatoire moins 1.

Le pourcentage de séquences vraiment aléatoires ayant une valeur de χ^2 supérieure à celle calculée est donc obtenu à partir de cette fonction :

$$P = 100 * \frac{1}{\Gamma(\frac{v}{2})2^{\frac{v}{2}}} \int_{\chi^2}^\infty x^{\frac{v}{2}-1} e^{-\frac{x}{2}} dx \tag{3.4}$$

3.2.1.3 La moyenne

Elle est calculée en sommant tous les mots de la séquence et en divisant par le nombre de mots total. Cela permet de détecter un biais sur la distribution des bits et des octets si elle s'éloigne de 0.5 et 127.5 respectivement.

3.2.1.4 La valeur de Pi par la méthode de Monte-Carlo

Des couples successifs (x,y) sont générés à partir de six octets. Chaque (x,y) représente la coordonnée d'un point dans un carré avec l'origine localisée sur un de ses coins. Si ($x^2 + y^2 <$ côté du carré), alors le point appartient au cercle centré sur l'origine et de rayon le côté du carré. Le pourcentage de points appartenant à ce cercle est égal au rapport de son aire (un quart d'un cercle) et de celle du carré, ce qui donne $\frac{\pi}{4}$. Pi est donc égal à quatre fois la proportion de points à l'intérieur du cercle. Une déviation par rapport à la valeur théorique montre un déséquilibre dans la répartition des points dans le carré et donc des bits dans les coordonnées.

3.2.1.5 L'autocorrélation

L'autocorrélation calcule l'intensité de la relation entre un mot et son successeur. Elle se calcule de la manière suivante :

$$A = \sum_{i=0}^{N-2} \frac{(X_i - \mu)(X_{i+1} - \mu)}{\sigma^2} \tag{3.5}$$

où N est le nombre de mots dans la séquence, X_i est le i^e mot, μ est la moyenne de la séquence et σ sa variance.

Dans le programme, s'il y a une dépendance forte entre les mots successifs alors le résultat est 1. Au contraire, s'ils sont complètement différents alors le calcul rend 0.

3.2.2 Le standard FIPS PUB 140-2

La publication 140-2 du FIPS de Mai 2001[NIS01] décrit un standard pour l'accréditation de modules cryptographiques. Présentée par le NIST, elle contient des tests assez simples. Ceux-ci permettent de dégager rapidement des défauts «évidents» à partir de la séquence testée.

La séquence est considérée comme des suites adjacentes de 20000 bits. Les tests sont appliqués à toutes les suites. Pour chaque test, une statistique est calculée sur les 20000 bits. Cette statistique a une valeur théorique et en pratique la valeur calculée va plus ou moins bien s'en approcher. Les

occurrences des valeurs calculées peuvent être représentées par une gaussienne centrée sur la valeur théorique. Un intervalle est donc fixé avec en son centre la valeur théorique attendue. Si la valeur calculée en est trop éloignée elle ne rentrera pas dans l'intervalle et le test est considéré raté. Au contraire, si elle rentre dans l'intervalle il est réussi. Le pourcentage de réussite d'un test est ainsi calculé pour la séquence entière. Les statistiques calculées étant dispersées autour de la valeur théorique, nous nous attendons à en avoir un petit nombre en dehors de l'intervalle. Un faible pourcentage d'échec à un test ne signifiera donc pas que la séquence est non aléatoire.

3.2.2.1 Monobit

Le nombre de bits à 1 est calculé. Le test est réussi si cette valeur est comprise entre 9725 et 10275. Ceci permet de détecter des déviations dans la distribution des bits.

3.2.2.2 Poker

La suite de 20000 bits est divisée en mots de 4 bits. Les occurrences $f(i)$ de chaque mot i sont comptées et la statistique suivante est calculée :

$$X = \frac{16}{5000} \sum_{i=0}^{15} f(i)^2 \tag{3.6}$$

Le test est réussi si $2{,}16 < X < 46{,}17$. Il permet de détecter des déviations dans la distribution des mots de 4 bits.

3.2.2.3 Runs

Un run est une séquence de bits consécutifs tous à 0 ou à 1. Les occurrences des runs de différentes tailles sont comptées et doivent toutes appartenir aux intervalles donnés dans le tableau 3.1 pour que le test soit réussi. Dans une suite de 20000 bits le nombre de runs de longueur i r_i suit la distribution suivante :

$$r_i = \frac{20000 - i + 3}{2^{i+2}} \tag{3.7}$$

Taille des runs	Occurrences théoriques	Occurrences acceptées
1	2500	2343 - 2657
2	1250	1135 - 1365
3	625	542 - 708
4	312	251 - 373
5	156	111 - 201
$\geqslant$6	156	111 - 201

TABLE 3.1 – Occurrences acceptées des runs de différentes tailles.

Ce test détecte des anomalies dans la distribution des bits.

3.2.2.4 LongRuns

Un long run est un run de 26 bits ou plus. Le test est réussi si la séquence ne contient aucun long run.

3.2.3 La méthode AIS31

L'agence allemande de la sécurité des technologies de l'information (BSI : Bundesamt für Sicherheit in der Informationstechnik) a présenté une méthodologie pour l'évaluation des générateurs de nombres aléatoires [SK03]. Cette évaluation passe par l'exécution de différentes procédures de test selon le niveau de sécurité demandé. Une partie des tests utilisés est composée des tests du FIPS 140-2.

3.2.3.1 Disjointness (T0)

Soient 2^{16} mots consécutifs de 48 bits. Le test est réussi si les mots sont tous différents. La probabilité d'échec pour un RNG idéal est de 2^{-17}. Ce test détecte des occurrences suspicieuses de mots.

3.2.3.2 Tests du FIPS (T1 à T4)

Ce sont les test du FIPS avec des probabilités d'échec de 10^{-6} pour le test du monobit, de $1{,}014 \times 10^{-6}$ pour le test du poker, de 10^{-6} pour le test du run et idem pour le long run. Pour ce dernier test un run est considéré long à partir de 34 bits.

3.2.3.3 Autocorrelation (T5)

Ce test se fait sur une séquence de 10 000 bits. L'autocorrélation est calculée avec des décalages τ de 1 à 5000 bits. La formule suivante est utilisée :

$$T_5(\tau) = \sum_{i=1}^{5000} (b_i \otimes b_{i+\tau}) \tag{3.8}$$

Le test est réussi si $2326 < T_5(\tau) < 2674$. La probabilité d'échec pour un RNG idéal est de 2^{-6}.

3.2.3.4 Uniform distribution (T6)

La séquence est divisée en mots de n bits. La statistique suivante est calculée :

$$T_6(i) = \sum_{i=0}^{2^n-1} \frac{N_i}{N} \tag{3.9}$$

où N_i est le nombre d'occurrences du mot i sur la séquence de N mots.

Le test est réussi si $T_6(i) \in [2^{-n} - a; 2^{-n} + a]$ pour tous les mots i et avec n, N et a en paramètres. Il vérifie la distribution uniforme des mots de différentes tailles.

3.2.3.5 Homogeneity (T7)

La séquence est divisée en S sous-séquences composées de suites de mots. A chaque position h dans toutes les sous-séquences, les occurrences $f_h(i)$ des mots i sont calculées. La statistiques $T_7(H, S)$ est alors calculée comme suit :

$$T_7(H, M) = \sum_{h=1}^{H} \sum_{i=0}^{M-1} \frac{(f_h(i) - P_i}{P_i} \tag{3.10}$$

$$P_i = \frac{1}{H} \sum_{h=1}^{H} f_h(i)$$

où H est le nombre de positions des mots dans les sous-séquences et M est le nombre de mots possibles.

Le test échoue si $T_7(H,M) > \chi^2(\alpha,(H-1)(M-1))$ qui est la limite de rejection pour un test de khi deux avec $(H-1)(M-1)$ degrés de liberté et α le seuil de signification.

3.2.3.6 Entropy estimation (T8)

La séquence étudiée est une suite de mots. Pour tous les mots i de la séquence, la distance D_i entre le mot considéré et son prédécesseur le plus proche ayant une même valeur est déterminée. La valeur suivante est calculée :

$$f_c = \frac{1}{K}\sum_{i=Q+1}^{Q+K} g(A_i) \tag{3.11}$$

$$g(n) = \frac{1}{\log(2)}\sum_{k=1}^{n-1}\frac{1}{k}$$

où $Q+K$ est la taille de la séquence.

Le test est réussi si f_c est supérieure à une certaine valeur, dépendante des paramètres Q, P, et de la taille des mots.

3.2.4 La batterie Diehard

La batterie de tests Diehard a été développée par George Marsaglia et publiée en 1995 [Mar95]. Les séquences testées doivent contenir 3 millions d'entiers (de 32 bits). La plupart des tests renvoient une p-valeur donnée par :

$$p = 1 - F(X) \tag{3.12}$$

où X est la variable aléatoire utilisée dans le test et F est sa distribution (le plus souvent normale).

Pour une séquence vraiment aléatoire les p-valeurs ont une distribution uniforme dans $[0\,;1[$. F est une approximation de la distribution théorique attendue et montrera donc quelques divergences par rapport à celle-ci. Ceci est d'autant plus vrai aux extrémités de la distribution avec des p-valeurs environ égales à 0 ou 1. De telles valeurs peuvent tout de même arriver, la séquence est donc considérée non aléatoire si plus de 5 p-valeurs sont égales à 0 ou 1 plus ou moins un certain intervalle. Par exemple pour un intervalle de 0.01 la p-valeur doit appartenir à [0,01;0,099].

3.2.4.1 Birthday spacings

Ce test fait référence au paradoxe des anniversaires qui est le suivant : dans un groupe de 23 personnes il y a 50% de chance que deux personnes aient la même date d'anniversaire, et à partir de 57 personnes la probabilité est supérieure à 99%.

Dans le test du birthday spacings il y a m anniversaires dans une année de n jours. Les distances entre les anniversaires sont comptabilisées. Soit j le nombre de distances dont les occurrences sont supérieures à un, j suit une loi de Poisson :

$$P(j=k) = e^{-\lambda}\frac{\lambda^k}{k!} \tag{3.13}$$

où $\lambda = \frac{m^3}{4n}$ est la moyenne.

Un échantillon comprenant 200 valeurs de j est étudié et un test de khi deux rend une p-valeur. 9 p-valeurs sont calculées ainsi avec les bits 1 à 24 des entiers de la séquence testée, puis les bits 2 à

25, etc. jusqu'aux bits 9 à 32. Les p-valeurs font ensuite l'objet d'un test de Kolmogorov-Smirnov qui rend la p-valeur finale.

Le test de Kolmogorov-Smirnov, de manière similaire au test du khi deux, est un test d'adéquation. Il compare la fonction de répartition théorique d'une variable aléatoire avec sa répartition empirique. En pratique il calcule la différence entre ces deux répartitions et compare la valeur maximale obtenue à un seuil donné par une table. Si la valeur est supérieure à ce seuil alors la séquence testée est considérée mauvaise.

3.2.4.2 Overlapping 5-permutation

Dans une séquence d'un million d'entiers chaque groupe de 5 entiers consécutifs peut prendre 120 états différents possibles (5! = 120). Toutes les occurrences de ces états sont comptées et elles doivent suivre une distribution normale. Un test du rapport de vraisemblance calcule la p-valeur. Il compare pour cela la probabilité que les occurrences suivent leur distribution théorique à la probabilité qu'elles ne la suivent pas.

3.2.4.3 Binary rank

Ce test se fait sur des matrices de 6×8 bits, 31×31 bits ou 32×32 bits. Par exemple la matrice de 6×8 bits est formée par les 8 premiers bits de 6 entiers consécutifs. Plusieurs centaines de matrices sont ainsi formées et leur rang est calculé. Les occurrences de ces rangs sont comptées, les rangs les plus faibles apparaissant rarement ils sont regroupés à partir d'un seuil. Un test de khi deux permet de vérifier que les occurrences ne diffèrent pas trop de leurs valeurs théoriques et rend une p-valeur.

3.2.4.4 Bitstream

La suite considérée est composée de 2^{21} mots de 20 bits. Les mots sont formés en se décalant d'un bit à la fois dans la séquence initiale ($2^{21}+19$ bits au total). Les mots peuvent prendre 2^{20} valeurs différentes et le nombre j de valeurs n'apparaissant pas est compté. Pour une suite aléatoire, j suit une distribution normale de moyenne $\mu = 141{,}909$ et variance $\sigma = 428$. La variable $(j - 141\,909)/428$ permet ainsi la génération d'une p-valeur uniformément distribuée. Le test est répété 20 fois.

3.2.4.5 Monkey

Ce test fait référence au paradoxe du singe savant, un théorème disant qu'un singe tapant au hasard sur le clavier d'une machine à écrire pourra presque sûrement écrire tous les livres de la Bibliothèque Nationale de France.

Ce test est comparable au précédent et a trois déclinaisons : OPSO, OQSO et DNA. OPSO (Overlapping Pairs Sparse Occupancy) considère des mots composés de 2 lettres de 10 bits dans un alphabet de 210 lettres. Il génère 2^{21} mots et compte le nombre j de mots n'apparaissant pas, j suivant une distribution normale ($\mu = 141{,}909$), $\sigma = 290$).

OQSO (Overlapping Quadruples Sparse Occupancy) considère des mots composés de 4 lettres de 5 bits dans un alphabet de 2^5 lettres. Il génère 2^{21} mots et compte le nombre j de mots n'apparaissant pas, j suivant une distribution normale ($\mu = 141{,}909$, $\sigma = 295$).

DNA considère des mots composés de de 10 lettres de 2 bits dans un alphabet de 2^2 lettres (en référence aux composants de l'ADN : A, T, G et C). Il génère 2^{21} mots et compte le nombre j de mots n'apparaissant pas, j suivant une distribution normale ($\mu = 141{,}909$, $\sigma = 339$).

3.2.4.6 Count the 1's

Il existe deux variations à ce test : il peut être utilisé sur la suite des entiers de la séquence dont on ne garde qu'un octet spécifique ou la suite de tous les octets successifs disponibles. Les octets sont formés en se décalant d'un bit à la fois dans la séquence initiale. Des mots de 5 octets sont considérés, chaque octet pouvant prendre les valeurs suivantes selon le nombre de bits à 1 qu'il contient : A (0 à 2 bits), B (3 bits), C (4 bits), D (5 bits), E (6 à 8 bits). Sur 256000 mots examinés, chaque occurrence est comptée et un test de khi deux est exécuté.

3.2.4.7 Parking lot

Des voitures (cercles de rayon 1) essayent de se garer dans un carré de côté 100. La position de ces voitures est donnée par deux entiers de 32 bits pour l'ordonnée et l'abscisse. Si la place de parking est déjà prise la voiture fait un nouvel essai jusqu'à en trouver une de libre. Le nombre de voitures garées après 12000 essais est k, qui suit une distribution normale avec $\mu = 3523$ et $\sigma = 21{,}9$. Ceci est effectué 10 fois puis les k normalisés sont passés à un test de Kolmogorov-Smirnov qui rend la p-valeur.

3.2.4.8 Minimum distance

Dans un carré de côté 10000, 8000 points sont placés, leurs coordonnées étant deux entiers de la séquence testée. La distance minimale d entre deux points est calculée. d^2 suit une distribution exponentielle de moyenne $\mu = 0{,}995$ donc $1 - e^{\frac{-d^2}{0{,}995}}$ est uniforme sur $[0,1[$. Ceci est répété 100 fois et un test de Kolmogorov-Smirnov rend une p-valeur.

3.2.4.9 3D spheres

Dans un carré de côté 1000, 4000 points sont placés, leurs coordonnées étant deux entiers de la séquence testée. Une sphère est placée à chaque point de telle manière qu'elle touche son point le plus proche. Le volume de la plus petite sphère suit une distribution exponentielle de moyenne $\mu = \frac{120\pi}{3}$. Ceci est effectué 20 fois, chaque rayon minimum donne une variable uniforme $1 - e^{\frac{-d^3}{30}}$ et un test de Kolmogorov-Smirnov est effectué.

3.2.4.10 Squeeze

Les entiers U successifs de la séquence testée sont transformés en réels dans $[0;1[$. Le calcul suivant est effectué : $k = \lceil kU \rceil$ avec k initialisé à 2^{31} et jusqu'à ce qu'il atteigne 1. j est le nombre d'itération nécessaires pour compléter le calcul. Ceci est effectué 100000 fois et le nombre de fois pour lesquelles j est inférieur à $6, 7, 8, \cdots, 47$ et supérieur à 48 est compté puis un test de khi deux est effectué.

3.2.4.11 Overlapping sums

Les entiers U successifs de la séquence testée sont transformés en réels dans $[0;1[$. Les sommes de 100 nombres successifs suivent une loi normale. En les transformant en variables uniformes, elles sont ensuite utilisées dans un test de Kolmogorov-Smirnov.

3.2.4.12 Runs

Dix fois 10000 entiers successifs de la séquence testée sont transformés en réels dans $[0;1[$. Un up-run est une succession de nombres croissants et un down-run de nombres décroissants. Les nombres

de runs de différentes longueurs sont calculés et leurs valeurs théoriques sont connues. Un test de khi deux est effectué sur chacun des runs.

3.2.4.13 Craps

Dans ce test, 200 000 parties de craps sont jouées avec la valeur du jet de dé donnée par un entier de la séquence. Les nombres de jeux gagnés et de jets nécessaires pour finir chaque partie sont comptés. Le nombre de jeux gagnés suit une loi normale avec $\mu = 200\,000p$ et $\sigma = 200\,000p(1-p)$ avec $p = \frac{244}{495}$. Les occurrences des nombres de jets sont comptées et un test de khi deux est effectué.

3.2.5 La batterie du NIST

Cette batterie de test a été développée par le NIST pour évaluer des RNGs utilisés dans des applications cryptographiques [RSN+01]. Chaque test statistique de la batterie rend une p-valeur p qui est la probabilité qu'un générateur de nombres vraiment aléatoires produise une séquence moins aléatoire que celle testée selon une certaine propriété. Une p-valeur de 1 signifie que la séquence est parfaitement aléatoire et de 0 qu'elle ne l'est pas du tout. $\mathscr{H}_0$ est acceptée si $p \geqslant \alpha$ pour $\alpha \in [0{,}001; 0{,}01]$. Par exemple un α de 0,01 indique qu'une séquence sur 100 devrait être rejetée et une p-valeur $p \geqslant 0{,}01$ signifie que la séquence est aléatoire à 99,99% de chance.

La séquence testée est divisée en suites d'au moins 1000 bits et le test de toutes les suites génère un certain nombre de p-valeurs. La proportion de p-valeurs supérieures à α est calculée et doit appartenir à l'intervalle donné ci-dessous pour que la séquence soit considérée aléatoire :

$$\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}} \tag{3.14}$$

où $\hat{p} = 1 - \alpha$ et m est le nombre de suites.

La distribution des p-valeurs est aussi examinée pour assurer leur uniformité. Pour cela un test de χ^2 est exécuté avec :

$$\chi^2 = \sum_{i=1}^{10} \frac{F_i - s/10}{s/10} \tag{3.15}$$

où F_i est le nombre de p-valeurs dans l'intervalle i et s est la taille de la suite. Une nouvelle p-valeur est calculée avec :

$$\begin{aligned} p - \text{valeur}_T &= \text{igamc}(9/2, \chi^2/2) \\ \text{igamc}(a,x) &= \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} dt \\ \Gamma(z) &= \int_0^\infty e^{-t} t^{z-1} dt \end{aligned} \tag{3.16}$$

où igamc est la fonction gamma incomplète et Γ la fonction gamma.

Si $p - \text{valeur}_T \geqslant 0{,}0001$ alors les suites sont considérées uniformément distribuées.

Cette batterie de tests assume que les séquences testées ont les propriétés suivantes :

- uniformité (les probabilités d'apparition des bits 0 et 1 sont égales à 0,5),
- scalabilité (si une séquence est aléatoire alors ses sous-séquences doivent aussi réussir les tests),
- consistance (un générateur doit avoir le même comportement quel que soit sa graine et ne doit donc pas être testé qu'avec une seule graine).

3.2.5.1 Frequency

Ce test détermine si les occurrences de bits à 0 et à 1 dans la séquence examinée sont égales. Une séquence vraiment aléatoire aura ces bits en proportions égales (uniformité), la statistique suit donc une distribution normale. Il détecte aussi l'importance du biais dans la distribution des bits selon que leur proportion d'apparitions se rapprochent de 0,5 ou pas.

3.2.5.2 Frequency within a block

Ce test est identique au précédent mais il calcule les occurrences des bits sur des sous-séquences de taille M. Le nombre de bits à 1 est donc en principe égal à $\frac{M}{2}$. Ce test mesure à quel point cela est vrai grâce à un test de khi deux.

3.2.5.3 Runs

Ce test détermine si les oscillations entre des runs de bits à 0 ou à 1 sont trop lentes (pour des transitions peu rapprochées il y a plus de runs) ou trop rapides (moins de runs). Il calcule le nombre total de runs et le compare à sa valeur attendue par un test de khi deux.

3.2.5.4 Longest run of ones in a block

Ce test examine la longueur maximale des runs de bits à 1 dans des sous-séquences de M bits. Il compare la valeur obtenue à celle attendue pour une séquence aléatoire par un test de khi deux. Une déviation observée sur cette longueur implique la réciproque sur des runs de bits à 0, ceux-ci ne sont donc pas examinés.

3.2.5.5 Binary matrix rank

Ce test détermine si des dépendances linéaires existent dans la séquence. Des matrices sont formées par les suites de bits dans la séquence testée et leur rang est calculé. Les occurrences des différents rangs possibles sont comparés à celles théoriques pour une séquence aléatoire par un test de khi deux.

3.2.5.6 Discrete Fourier transform

Le but de ce test est de déterminer si la séquence détient des caractéristiques de périodiques (motifs se répétant) ce qui se traduirait par des valeurs de pics non homogènes à certaines fréquences. Il calcule la DFT (Discrete Fourier Transform) de la séquence et compte le nombre de pics de la DFT dépassant un seuil qu'en théorie 95% des pics ne devraient pas dépasser. Ce nombre est comparé aux 5% de pics dépassant le seuil en temps normal. La statistique suit donc une distribution normale.

3.2.5.7 Non-overlapping template matching

Le but de ce test est de détecter si la séquence contient un ou plusieurs motifs se répétant trop ou pas assez souvent. Ce test compte les occurrences de toutes les combinaisons possibles d'un motif ayant un certain nombre m de bits et les compare à des valeurs théoriques par un test de khi deux. Si le motif n'est pas trouvé la séquence est examinée en la décalant d'un bit sinon elle est décalée de m bits.

3.2.5.8 Overlapping template matching

Ce test est similaire au précédent, à la différence que lorsque un motif est trouvé la séquence n'est ensuite décalée que d'un bit. Il y a donc un recouvrement des données.

3.2.5.9 Maurer's «Universal Statistical» test

Ce test estime la compressibilité sans perte d'information d'une séquence. Pour cela il compte le nombre de bits entre deux motifs identiques, une mesure liée à la longueur de la séquence compressée. Il somme le $\log_2$ de ces distances et la statistique suit une distribution semi-normale (seulement un côté de la distribution).

3.2.5.10 Linear complexity

Ce test détermine la complexité de la séquence en la décrivant par des LFSRs. Plus une séquence est aléatoire plus elle est caractérisée par un LFSR long. En pratique les occurrences des LFSRs de différentes tailles sont comptées et comparées aux valeurs attendues pour une séquence aléatoire par un test de khi deux.

3.2.5.11 Serial

Ce test estime l'uniformité de la séquence. Des motifs de m bits sont pris dans la séquence en se décalant d'un bit à chaque fois. Les occurrences de chaque motif sont comptées puis comparées à leurs valeurs théoriques (motifs équiprobables) par un test de khi deux.

3.2.5.12 Approximate entropy

Ce test estime la régularité de la séquence. Des motifs de m bits et de $m+1$ bits sont pris dans la séquence en décalant d'un bit à chaque fois. Leurs occurences sont comptées et ensuite comparées aux valeurs théoriques attendues par un test de khi deux.

3.2.5.13 Cumulative sum

Ce test montre si la séquence contient des suites trop importantes de bits à 0 ou à 1 ou au contraire si les bits sont trop bien mélangés. Les bits sont additionnés (un bit à 0 donne la valeur -1 et un bit à 1 donne 1) à mesure qu'ils sont parcourus (du début de la séquence à la fin et vice-versa). La valeur absolue maximale des sommes partielles est retenue, elle doit suivre une distribution normale.

3.2.5.14 Random excursions

Ce test considère la suite des sommes partielles de la séquence (constituée de bits à 0 et à 1). Quand un bit à 1 est rencontré alors la somme est incrémentée de 1 et s'il s'agit d'un bit à 0 elle est décrémentée de 1. Un cycle est une partie de cette suite débutant et finissant par une somme à 0. Pour chaque cycle les occurrences des différentes valeurs des sommes (états) sont comptées puis comparées aux valeurs attendues par un test de khi deux.

3.2.5.15 Random excursions variants

Ce test considère la suite des sommes partielles de la séquence comme précédemment. Les occurrences de chaque état sont comptées sur toute la séquence et doivent suivre une distribution normale.

3.2.6 La batterie TestU01

TestU01 est une librairie créée par Pierre L'Ecuyer et Richard Simard [LS07] contenant plusieurs batteries de tests statistiques. Dans TestU01 la séquence de bits peut aussi être considérée comme une suite de réels dans [0;1]. L'hypothèse $\mathscr{H}_0$ est alors que la séquence est une suite de réels indépendants de distribution uniforme sur [0;1] ($\mathscr{H}_0^A$) ou qu'elle est une suite de bits indépendants entre eux et prenant la valeur 0 ou 1 avec la même probabilité ($\mathscr{H}_0^B$). En conséquence les tests sont écrits de manière à vérifier soit $\mathscr{H}_0^A$ soit $\mathscr{H}_0^B$.

3.2.6.1 MultinomialBitsOver

Ce test mesure l'uniformité de points aléatoires dans un hypercube unité de dimension t. Cet hypercube est divisé en k cellules cubiques de même taille. En ensemble de n points est formé en générant leurs coordonnées à partir de L bits successifs pris dans la séquence testée, chaque groupe de L bits étant décalé d'un bit par rapport au précédent. Le nombre de points appartenant à chaque cellule est comparé aux valeurs attendues.

3.2.6.2 ClosePairsBitMatch

Ce test reprend le même hypercube que précédemment avec des points générés de manière identique. Il est basé sur la distance minimale entre toutes les paires de points. Cette distance est mesurée en comparant les bits de poids fort entre les points et en comptant les bits identiques.

3.2.6.3 AppearanceSpacings

Il s'agit du «Universal Test» de Maurer qui mesure l'entropie d'une séquence de bits. Le test forme une nouvelle séquence à partir de la concaténation des s bits de poids fort parmi la suite de nombres composant la séquence initiale testée. Il parcourt ensuite les mots de taille L bits et crée la suite des distances entre le mot examiné et sa précédente occurrence dans la séquence. Il calcule enfin la moyenne de ces distances qui doit suivre une distribution normale.

3.2.6.4 LinearComp

Deux tests sont appliqués ici pour évaluer l'évolution de la complexité linéaire de la séquence quand elle est parcourue. Cela signifie que l'on cherche la taille minimale d'un LFSR qui génèrerait la séquence testée. Des séquences aléatoires sont caractérisées par des LFSRs longs.

Le test du saut de complexité compte le nombre de sauts apparaissant dans la complexité linéaire de la séquence. Un saut apparaît quand l'addition d'un bit à la séquence parcourue augmente sa complexité linéaire. Le nombre de sauts est comparé à sa distribution normale théorique.

Le test de la taille du saut mesure la quantité de complexité ajoutée avec chaque bit. Il compare ensuite les occurrences de chaque taille de saut avec leurs valeurs théoriques en utilisant un test de khi deux.

3.2.6.5 LempelZiv

Ce test mesure la compressibilité de la séquence testée selon l'algorithme de Lempel-Ziv. Il compte pour cela le nombre de mots différents apparaissant dans la séquence, cette valeur doit être approximativement distribuée suivant une loi normale.

3.2.6.6 Fourier1

La séquence est divisée en N suites de n bits. Les coefficients de la transformée de Fourier de chaque suite sont calculés et le nombre de coefficients ayant une valeur absolue inférieure à un certain seuil est compté. Le test vérifie que ce nombre suit bien une distribution normale de moyenne $\mu = \frac{0,95n}{2}$ et de variance $\sigma^2 = 0,05\mu$.

3.2.6.7 Fourier3

Ce test est basé sur le précédent. Les coefficients de la transformée de Fourier de chaque suite sont mis au carré et divisés par le nombre de bits. Les coefficients de même fréquence pour toutes les suites sont additionnés et le test vérifie qu'ils suivent une distribution normale.

3.2.6.8 LongestHeadRun

Ce test génère une séquence de n mots de L bits à partir des s bits extraient dans chacun des $n\lfloor\frac{L}{s}\rfloor$ nombres successifs de la séquence initiale. Dans chaque mot, le plus long run de bits à 1 est trouvé. Les occurrences des runs de différentes longueurs sont comparées à leurs valeurs théoriques par un test de khi deux.

Le test cherche aussi le plus long run de bits à 1 dans la séquence entière et en répétant ceci N fois il compare la statistique à sa distribution théorique.

3.2.6.9 PeriodsInString

Ce test est basé sur la distribution des autocorrélations des séquences de s bits. L'autocorrélation d'une suite est donnée par un vecteur de bits. Le bit d'index p de ce vecteur est à 1 si p est une période de la séquence, i.e. la séquence testée est constituée d'une suite de p bits se répétant. n séquences sont générées et leurs autocorrélations sont calculées. Le nombre d'occurrences de chaque autocorrélation est comparé aux valeurs théoriques par un test de khi deux.

3.2.6.10 HammingWeight

Ce test examine le nombre de bits à 1 pour chaque mot de L bits. Les occurrences possibles de bits à 1 dans un mot suivent une loi binomiale de moyenne $\frac{L}{2}$ et de variance $\frac{L}{4}$. Elles sont comparées aux occurrences obtenues dans la séquence par un test de khi deux.

3.2.6.11 HammingCorr

Il teste l'indépendance des poids de Hamming des mots successifs de la séquence. Il génère pour cela une séquence composée des bits de poids les plus forts à partir des nombres composant la séquence originale. Il calcule le poids de Hamming des mots successifs de L bits de la séquence et leur corrélation par la formule suivante :

$$\hat{\rho} = \frac{4}{(n-1)L} \sum_{i=1}^{n-1} \left(HW_i - \frac{L}{2}\right)\left(HW_{i+1} - \frac{L}{2}\right) \tag{3.17}$$

où HW_i est le poids de Hamming du mot i et n est le nombre de mots dans la séquence.

Le test vérifie que $\hat{\rho}\sqrt{n-1}$ a une distribution normale.

3.2.6.12 HammingIndep

Ce test est une extension du précédent. Les occurrences de chaque paire possible des poids de Hamming sont comptées et comparées à leurs valeurs théoriques par un test de khi deux.

Un deuxième test est effectué : les occurrences calculées précédemment sont placées dans une matrice, ses lignes et colonnes centrales sont éliminées pour ne garder que les coins. Des opérations linéaires sont exécutées sur ces sous-matrices pour créer trois variables. Leurs valeurs sont comparées à la théorie par un test de khi deux.

3.2.6.13 Autocor

Ce test calcule l'autocorrélation au rang d des séquences de n bits comme suit :

$$A_d = \sum_{i=1}^{n-d} b_i \otimes b_{i+d} \tag{3.18}$$

Elle doit suivre une distribution binomiale de paramètres $(n-d, \frac{1}{2})$ et la statistique suivante est vérifiée comme ayant une distribution normale :

$$\frac{2A_d - (n-d)}{\sqrt{n-d}} \tag{3.19}$$

3.2.6.14 Run

Pour des séquences de nombres réels, un run est une suite de nombres ayant des valeurs successives. Si les valeurs augmentent alors il s'agit d'un run up, sinon c'est un run down. Les occurrences des différentes longueurs des runs sont comptées et comparées à leur distribution théorique.

Pour des séquences de bits, un run est une succession de bits à 1 et un gap de bits à 0. Les occurrences des différentes longueurs des runs et gaps sont comptées et comparées à leur distribution théorique. La statistique doit suivre une distribution de khi deux.

3.2.6.15 MatrixRank

Ce test est basé sur le rang d'une matrice de bits aléatoires. Les s bits les plus significatifs des nombres successifs de la séquence testée sont extraits et forment une nouvelle séquence. Une matrice $L * k$ est formée en la remplissant par les bits de la séquence ligne par ligne. Le rang de la matrice est calculé et donne le nombre de lignes indépendantes. Ceci est répété afin de comparer la statistique à sa distribution théorique par un test de khi deux.

3.2.6.16 RandomWalk1

Ce test compare les propriétés attendues d'une marche aléatoire avec celles de la séquence étudiée grâce à un test de khi deux. Les s bits les plus significatifs des nombres successifs de la séquence sont extraits et forment une nouvelle séquence. Celle-ci est parcourue et une somme est incrémentée selon la valeur du bit lu (un bit à 0 donne la valeur -1, un bit à 1 donne 1). n marche aléatoires sont ainsi produites. Les différentes statistiques étudiées sont :

- le nombre de bits à 1,
- la valeur maximale atteinte par la somme,
- la proportion du temps pour lequel la somme est positive,
- le temps minimal avant que la somme atteigne une certaine valeur,

– le nombre de fois où la somme est égale à 0,
– le nombre de changements de signe de la somme.

3.3 Limitations

Nous discutons ici des limitations des tests statistiques. En effet, ceux-ci permettent de récupérer de l'information sur le comportement d'un RNG mais ne sont pas suffisants pour s'assurer de son bon fonctionnement.

3.3.1 Informations données par les tests statistiques

Le principe de l'évaluation statistique d'un RNG fait que le résultat obtenu est soit directement une décision (séquence aléatoire ou non) soit un ensemble de p-valeurs. Ceci résulte dans le manque d'informations concrètes sur les propriétés de la séquence testée. Il peut être bénéfique d'observer directement ces propriétés afin d'avoir une meilleure idée du fonctionnement du RNG visé. Nous pouvons par exemple afficher les occurrences des mots de la séquence sur un graphe. Si ces mots ne sont pas uniformément distribués alors cela ressortira sur un test de khi deux et grâce à la représentation visuelle nous pouvons en plus identifier les motifs en cause.

3.3.2 Évolution dans le temps

Ces séquences produites par les RNGs sont, d'une certaine manière, le reflet de leur fonctionnement au cours d'un certain intervalle de temps. Les informations sur leur évolution durant ce temps disparaissent dans le passage des tests statistiques. En effet ces derniers vont avoir tendance à moyenner les résultats et si des différences de comportement existent dans la séquence, elles ne sont pas immédiatement distinguables en sortie des tests. Par exemple, imaginons qu'un biais existe au début d'une séquence avec plus de bits à 1 et qu'il se transforme à la fin par un biais avec plus de bits à 0. Dans ce cas le biais calculé par les tests sur la séquence entière est nul, ce qui ne nous informe pas correctement sur le comportement du RNG. Certains tests font des statistiques sur la séquence dans laquelle l'ordre des bits est changé ou sur des morceaux de séquences uniquement, ce qui là aussi biaise l'information sur l'évolution du système.

3.3.3 Attaque localisée

Les tests statistiques font état d'un comportement global d'un RNG. Ils se basent sur des répétitions de déviations par rapport à un comportement normal. Une perturbation très localisée ne sera donc pas détectée par les tests statistiques, ou en tout cas elle ne sera pas détectée comme étant anormale. Ainsi, la connaissance du genre de perturbation induite ainsi que le moment à laquelle elle a été réalisée peut être suffisante pour être exploitée par un attaquant. Par exemple, si un attaquant arrive à manipuler la valeur d'un octet à sa guise, il pourrait décider de le mettre à 0 lors de la génération d'un nombre premier, d'un vecteur d'initialisation, d'un nonce, etc. La connaissance de la valeur d'un octet dans ces nombres peut être suffisante pour révéler des informations normalement secrètes.

Chapitre 4

Les attaques par perturbation

Nous introduisons ici les attaques réalisables sur des composants cryptographiques. Le but de ces attaques est de dévier le comportement attendu des composants ou d'obtenir des informations sur les données qu'ils manipulent. La connaissance de ces informations induit des vulnérabilités sur les algorithmes ou données secrètes que contient ce composant. En appliquant des perturbations aux RNGs, il est possible d'influencer les séquences de bits générés en sortie. Pour certaines applications il est crucial que de telles perturbations ne se produisent pas. En effet, la sécurité de certains algorithmes cryptographiques dépend, entre autres, de la qualité des nombres aléatoires utilisés.

4.1 Les attaques physiques

Les attaques sur des composants cryptographiques peuvent être réalisées par deux types de méthodes : il s'agit soit d'attaques passives ou actives. Dans une attaque passive, le composant fonctionne normalement et la récupération d'informations s'effectue par l'observation de ses différentes propriétés physiques (consommation de courant, temps d'exécution, etc.). Dans une attaque active, une perturbation est appliquée afin que le composant ait un comportement anormal (température, tension d'alimentation, etc. en dehors de leur limites de fonctionnement). Les informations sont révélées via les anomalies qui apparaissent dans les opérations effectuées.

Les attaques invasives [KK99, Sko05] récupèrent les informations contenues dans un composant en étudiant au plus près son circuit électronique. Ces attaques permettent de retrouver les diverses informations contenues dans le composant, par exemple des implantations d'algorithmes, des données en mémoire, etc. Le but peut être d'obtenir une image du circuit et faire ainsi de la rétro-ingénierie, de poser des sondes sur les bus du circuit afin de lire les informations qui y transitent ou de modifier le circuit en coupant/ajoutant des pistes. Les méthodes de préparation des composants pour ces attaques peuvent être destructives et ne permettent alors pas leur utilisation ultérieure. Ces méthodes mettent en œuvre des procédés chimiques afin d'atteindre les couches intéressantes du composant ou utilisent des outils spécialisés comme un FIB (Focused Ion Beam).

Les attaques non invasives analysent les informations directement accessibles du composant étudié. Les attaques actives consistent à modifier le fonctionnement normal du composant en perturbant son environnement. Il s'agit par exemple de faire varier la température ou la tension d'alimentation hors des limites normales de fonctionnement préconisées, d'appliquer des impulsions électriques (glitches), ou de jouer sur les fronts d'horloge [BECN$^+$04]. Les attaques passives se basent sur la récupération de signaux à partir des canaux cachés du composant étudié et leur analyse statistique. Différents paramètres peuvent être étudiés comme le temps d'exécution des différentes instructions d'un algorithme [Koc96], la consommation électrique du composant liée aux opérations effectuées et

aux des données manipulées (SPA : Simple Power Analysis, DPA : Differential Power Analysis) [KJJ99].

Enfin les attaques semi-invasives [Sko05] nécessitent un accès rapproché mais sans contact à la partie du circuit ciblée. Cela signifie qu'il y a une phase de préparation du composant afin de faciliter cet accès. Un objectif de cette attaque peut être de lire le contenu de cellules mémoires, par exemple en utilisant un laser ou un champ magnétique [SSAQ02]. Nous retrouvons aussi dans cette catégorie les équivalents des attaques SPA et DPA en émissions électromagnétiques (SEMA : Simple ElectroMagnetic Analysis, DEMA : Differential Electromagnetic Power Analysis) [QS00, GMO01]. En effet, plus le composant a de couches, plus le champ électromagnétique devient faible et il est donc nécessaire de s'approcher au plus près du circuit électronique. De manière active, le principe de l'attaque est d'injecter des fautes dans le fonctionnement du composant grâce à des impulsions magnétiques ou lumineuses par exemple [SA03].

4.2 Perturbations des systèmes électroniques

Nous détaillons ici des attaques par perturbation utilisées en cryptographie [Sko05, BECN+04], ou pour des études de fiabilité des circuits électroniques. Nous distinguons deux types d'effets observables : soit la perturbation impacte globalement un composant (localisation non évidente des erreurs générées), soit des fautes sont générées de manière précise (par exemple un bit ou un registre). En cryptographie les attaques se focalisent généralement sur ce dernier point. En effet, en plus de la perturbation physique du circuit visé, l'attaque comprend l'analyse des algorithmes qu'il contient afin de déterminer les instructions/données à compromettre.

4.2.1 Variation de la température

Selon [BECN+04], une telle attaque peut produire deux types de fautes sur des cartes à puce. En dépassant les limites préconisées de leur température ambiante ces puces entrent dans un mode de fonctionnement dans lequel les opérations d'écriture sont possibles mais pas celles de lecture et vice-versa. Ceci est dû au fait que ces deux opérations n'ont pas les mêmes caractéristiques de fonctionnement.

Un autre effet est la modification aléatoire de l'état d'une cellule mémoire dans une RAM (Read Access Memory). Dans [GA03], des erreurs sont induites dans la mémoire d'un PC. L'expérimentation consiste à diriger une lampe vers cette mémoire, en variant sa distance à la mémoire jusqu'à atteindre une température d'environ 100 °C pour laquelle des fautes commencent à apparaître.

Dans [LWW+10b], un système de cryptographie quantique et plus particulièrement ses détecteurs de photons est attaqué. En envoyant de la lumière sur les détecteurs ils chauffent et peuvent entrer dans un mode de fonctionnement dans lequel ils sont aveuglés, c'est-à-dire qu'ils ne détectent plus les photons qui leur sont envoyés. Il est aussi possible pour un attaquant d'ajouter des coups de lasers qui simulent l'arrivée de photons. Ces attaques sont en fait similaires à celles décrites dans la section 4.2.4.3, où la lumière suffit à générer ces effets.

4.2.2 Variation de la tension d'alimentation

Cette attaque est du même type que la précédente et profite de failles de sécurité dues à une tension d'alimentation ayant une valeur sortant des limites de fonctionnement fixées. Plusieurs exemples d'applications sont cités dans [AK96] :

– en augmentant la tension d'alimentation d'un micro-contrôleur de 0,5 V il est possible de mettre un bit accédé en écriture à 0,

– en diminuant légèrement la tension d'alimentation un RNG produit quasiment uniquement des bits à 1.

4.2.3 Impulsions électriques

Ces attaques consistent à appliquer une variation brève de tension (glitch) par différents moyens. Il existe plusieurs chemins d'attaque : le signal d'alimentation, d'horloge, un champ électrique externe ou encore un champ électromagnétique. Le principe de l'attaque est de perturber le déroulement normal d'un programme en lui faisant sauter des instructions ou en corrompant les données manipulées [AK98, KK99, Sko05, ADN+10]. Ceci est possible en sortant les composants électroniques du circuit sous attaque hors de leurs limites normales de fonctionnement.

Par exemple les bascules mémorisent l'état de leur signal en entrée sur réception d'un coup d'horloge. Si un glitch est appliqué sur le signal d'horloge alors il va simuler ce coup. Ceci va entraîner la mémorisation d'un état qui ne sera pas forcément celui attendu. Une autre possibilité est que le glitch va déclencher le déroulement d'une instruction avant que la précédente ne soit terminée et que les données ne soient à jour.

Les glitches appliqués à un signal d'alimentation peuvent modifier le seuil des transistors. Le circuit attaqué va ainsi avoir un comportement différent pendant un court moment.

Des applications intéressantes de cette attaque sont de supprimer des instructions conditionnelles effectuant des vérifications de sécurité, rallonger les itérations de boucles afin de lire plus de mémoire ou les réduire afin de diminuer la sécurité d'opérations cryptographiques.

4.2.4 Impulsions lumineuses

4.2.4.1 Lumière blanche

Il a été montré que l'exposition à un flash de lumière peut provoquer une erreur sur les portes logiques illuminées. Raphaël Bauduin au CESTI (Centre d'Évaluation de la Sécurité des Technologies de l'Information) du CNET (Centre National d'Études des Télécommunications) à Caen a été le pionnier des attaques par application de lumière dans le milieu des années 1990. Dans [SA03], l'état d'une cellule mémoire est modifié grâce au flash d'un appareil photo. Ceci est dû aux courant induits par les photons de la lumière qui est appliquée.

4.2.4.2 Laser

De nos jours les attaques par illumination sont produites grâce à des bancs lasers pour des résultats améliorés par rapport à ceux par exposition à un flash de lumière. En effet avec un laser nous maîtrisons la focalisation du faisceau sur des localisations très précises des circuits électroniques ainsi que la durée pendant laquelle ils sont éclairés. L'idée de départ de l'utilisation des lasers [Hab65] était la reproduction et simulation des effets causés par des radiations ionisantes sur des semiconducteurs [Fou90, Pou00]. Ces effets ont été observés initialement par la présence de particules radioactives dans les matériaux utilisés en électronique [MW78] qui causaient des fautes dans les circuits. Il existe aussi une problématique issue du domaine spatial avec l'étude des effets des rayons cosmiques sur les semi-conducteurs [ZL79].

Les faisceaux laser sont aujourd'hui très utilisés pour opérer des attaques par fautes sur des circuits électroniques. Le principe de ces attaques est la formation de paires électrons-trous par un photons reçus dans le silicium au voisinage des zones polarisées. Ces paires électrons-trous créent des courants qui peuvent provoquer des court-circuits ou une mise à la masse par exemple. De plus, l'utilisation d'un laser permet de focaliser la lumière sur un point précis d'un circuit.

4.2.4.3 Applications aux systèmes quantiques

Une application particulière de l'attaque par injection de lumière est possible sur des systèmes de cryptographie quantique. En effet, dans le cas où ceux-ci se basent sur de la quantique optique alors des détecteurs de photons sont utilisés. Les attaques avec injection de lumière se focalisent donc sur ces détecteurs. Le système cryptographique considéré est l'échange de clef (QKD : Quantum Key Distribution).

Dans [Mak09], l'attaque décrite consiste à aveugler les deux détecteurs de photons (composés de photodiodes) du système. En illuminant les deux détecteurs, l'échange de clef est arrêté. En effet, l'envoi d'un grand nombre de photons sur un détecteur le fait entrer dans un régime de saturation et son taux de détections diminue jusqu'à s'annuler. De plus, en modifiant la polarité de la lumière injectée un attaquant peut choisir d'illuminer un seul détecteur et l'autre fonctionne alors normalement. Ainsi l'attaquant a la maîtrise temporelle de l'échange des clefs. L'attaque peut être améliorée pour permettre la maîtrise totale des informations envoyées [LWW+10a]. En effet, en aveuglant les détecteurs ils entrent dans un nouveau mode de fonctionnement. Il est alors possible de superposer des impulsions laser à la lumière, qui sont détectées comme les photons initialement.

Une autre application utilisant l'injection de lumière est la suivante [VMH01, GFK+06]. De la lumière est envoyée sur le système à la manière d'une sonde. Dans ce système, l'information est codée par la phase des photons. La lumière traverse le chemin emprunté par les photons et récupère de l'information sur leur phase donc sur les bits transmis.

4.2.5 Champ électromagnétique

Les perturbations électromagnétiques sont une problématique importante dans le domaine des circuits électroniques. En effet ceux-ci sont sensibles aux ondes électromagnétiques et doivent donc en être protégés (d'où les normes de compatibilité électromagnétiques). Cette tâche est compliquée par le fait que les circuits électroniques sont eux même générateurs d'ondes électromagnétiques. En effet ces ondes sont créées soit par la circulation d'un courant dans une boucle soit par une chute de tension dans un circuit. Les composants par lesquels ces ondes sont créées sont alors appelés des antennes.

Les perturbations induites sur les circuits sont dues au couplage de ces derniers avec le rayonnement émis. Le couplage est créé via les antennes du circuit visé ou directement sur ses câbles et différents effets peuvent être observés suivant les composants perturbés. [Dub09] présente les effets des ondes électromagnétiques sur des circuits électroniques. Il cite entre autres : des perturbations de signaux entraînant la modification d'un état logique, l'apparition de délais pouvant générer une désynchronisation de plusieurs signaux, la modification du point de fonctionnement de composants ou la variation en fréquence et gigue de phase de systèmes oscillants.

4.2.6 Rayonnements ionisants

Un rayon est dit ionisant si l'énergie qu'il transfère vers des électrons d'une matière est suffisante pour les arracher à leurs atomes. Dans ce processus des atomes vont perdre des électrons et être chargés positivement (cations) tandis que leur voisins vont récupérer ces électrons et être chargés négativement (anions). Des rayonnements à très forte énergie sont même susceptibles de modifier le réseau cristallin d'un matériau et d'en modifier ses propriétés électroniques [Bou73].

Les rayonnements ionisants sont composés [UNS00] :

– des rayonnements cosmiques produits par le flux des particules et des noyaux atomiques circulant dans le vide,

– des ondes électromagnétiques très énergétiques produites par des rayons X (produits par l'interaction de faisceaux d'électrons avec un métal) ou γ (émis par les noyaux radioactifs excités),
– des rayonnements α et β composés des particules émises lors d'une désintégration radioactive,
– des rayonnements produits par l'interaction des neutrons avec la matière.

De nombreuses études ont montré que les rayonnements ionisants peuvent causer des erreurs sur des composants électroniques. Les répercussions de tels effets sont donc importantes, notamment dans les domaines du spatial avec les rayons cosmiques, mais aussi et plus simplement pour des applications courantes en électronique. [Ter04] propose une bibliographie sur les études des effets engendrés par de telles radiations.

4.3 Perturbations particulières aux TRNGs

La sécurité des RNGs peut être étudiée sous un aspect mathématique [SMS07, FD03, BLMT10] en les modélisant et en étudiant l'influence de possibles perturbations sur leur fonctionnement.

Nous présentons ici différentes manières de perturber physiquement des TRNGs. De telles études ont été effectuées sur des TRNGs disponibles dans le commerce [DJ00, Cry03], il s'agit en effet d'une étape nécessaire pour la validation de la performance de tels composants. En l'occurrence ces TRNGs n'ont pas montré (ou très peu) d'influence aux perturbations qui leur ont été appliquées (variation de température, de fréquence d'horloge, de tension d'alimentation, application d'activités environnantes).

Nous expliquerons dans le chapitre 5 les expériences que nous avons nous-mêmes menées, et en particulier les bancs de test que nous avons mis en place pour les réaliser.

4.3.1 La variation de température

Sans doute l'expérience la plus facile à mettre en œuvre, la variation de température consiste à chauffer ou refroidir un composant intégrant un RNG.

Dans [ŠDF11] le TRNG perturbé est celui décrit dans [FD03] et implémenté dans un FPGA. Il est composé d'un signal d'horloge contrôlant une PLL, dont la sortie est retardée plusieurs fois, créant ainsi des signaux différents qui sont ensuite XORés. Deux implantations sont étudiées, une première où les paramètres du TRNG sont automatiquement fixés et une seconde où ils sont déterminés manuellement en réglant une bande passante faible dans la boucle de la PLL. Le RNG atteint 30 °C en fonctionnement et il a été refroidi jusqu'à −40 °C grâce à un spray givreur. Dans la deuxième implantation, le nombre de bits est plus important et plus équilibré ce qui rend le résultat plus fiable et stable. Le premier montage est lui sensible à la variation de température et les tests du FIPS révèlent un biais dans la distribution des bits.

Dans [YKBS10] le TRNG perturbé est celui décrit dans [SMS07] et implémenté dans un FPGA. Il est composé d'oscillateurs en anneaux tournant en parallèle dont les sorties sont XORées. L'étude est faite sur la plage de 0 °C à 50 °C, pour 16 à 256 oscillateurs en parallèle de longueurs de 3 à 27 inverseurs et dont le XOR est échantillonné à 100 MHz. La fréquence à laquelle les oscillateurs fonctionnent est mesurée. En effet les fréquences des oscillateurs doivent être différentes d'un multiple de la fréquence d'échantillonnage et ne pas avoir un PPCM (Plus Petit Multiplicateur Commun) petit pour éviter l'apparition de motifs dans les séquences due à la périodicité des bits générés. Cette expérience montre que plus le RNG chauffe plus la fréquence des oscillateurs diminue et vice-versa, et quelquefois prend une valeur proche d'un multiple de la fréquence d'échantillonnage.

Dans [SSR09] trois TRNG différents sont comparés sous perturbation. Les RNG utilisés sont décrits dans [SMS07], [VHKK08] et [DG07], ils sont implantés dans un FPGA et testés à 25 °C, 55 °C et

75 °C. Les tests du FIPS sont utilisés et montrent que le TRNG de [SMS07] n'est pas (ou infiniment peu) influencé par la température, le TRNG de [VHKK08] a des performances bien dégradées plus la température augmente et cette dernière a peu d'impact sur le TRNG de [DG07].

4.3.2 La variation de l'alimentation

Varier la tension d'alimentation des composants d'un TRNG est une autre méthode populaire pour perturber celui-ci.

Dans [YKBS10], cité précédemment avec la variation de température, une autre expérience est effectuée en modifiant la tension d'alimentation du FPGA dans lequel le TRNG est implanté. Les paramètres de tests sont identiques, la température ne change pas mais la tension d'alimentation prend des valeurs entre 1,4 V et 1,6 V. La fréquence des oscillateurs est mesurée et montre qu'elle augmente avec le voltage et vice-versa, et dans quelques cas prend une valeur proche d'un multiple de la fréquence d'échantillonnage.

4.3.3 L'activité avoisinante

Les TRNGs sont implantés dans des circuits et sont donc environnés par d'autres activités que la leur. Le but de cette expérience consiste à comparer les performances d'un TRNG avec et sans activités environnantes. En fait l'application d'une telle perturbation essaye de simuler un environnement réaliste pour le fonctionnement des TRNG.

Dans [SSR09] cité précédemment avec la variation de température, trois TRNGs sont étudiés. La même expérience est réalisée sans modifier la température mais en créant de l'activité dans le FPGA en implantant des LFSRs autour des TRNGs. Comme pour les tests avec variation de température, le TRNG de [SMS07] reste robuste, celui de [VHKK08] voit ses performances dégradées, enfin le TRNG de [DG07] est influencé et montre en particulier l'apparition d'un biais dans la distribution des bits.

4.3.4 L'injection de fréquence

Cette attaque peut être rapprochée de la précédente. En effet, en simulant une activité au voisinage d'un TRNG nous utilisons certaines parties du circuit électronique et créons ainsi des champs électromagnétiques.

Cette expérience est plus récente et a été en particulier introduite par [MM09]. Dans cet article une méthode est décrite pour réduire l'entropie d'un TRNG implanté dans une carte de paiement EMV (Europay Mastercard Visa). Elle se base sur un TRNG de type [SMS07] avec des oscillateurs à anneaux tournant en parallèle et dont les sorties sont XORées. Le principe de l'attaque est d'envoyer un signal sinusoïdal ayant une certaine fréquence à travers la masse de l'alimentation des inverseurs composant les oscillateurs. La fréquence de ce signal peut être déterminée en balayant une certaine gamme de fréquences et en testant les nombres générés, ou en analysant le spectre du signal électromagnétique émis par la carte et en choisissant une fréquence qui en ressort. Les tests du NIST et Diehard sont utilisés afin d'évaluer la qualité des nombres aléatoires générés et montrent une périodicité pour une certaine fréquence bien choisie. Ceci est basé sur le fait que le signal est injecté à une fréquence telle qu'elle permet de coupler les oscillateurs composant le TRNG d'où la perte d'aléa.

Une expérience différente est réalisée dans [PTL+11]. Le TRNG considéré est constitué d'un oscillateur de 101 inverseurs et est implanté dans un FPGA. Une sonde placée au dessus de ce FPGA permet d'injecter un signal électromagnétique, dans ce cas à une fréquence de 1 GHz. Une cartographie du FPGA est réalisée en déplaçant la sonde et en mesurant la fréquence à laquelle l'oscillateur tourne. Quand la sonde est placée au dessus des rails d'alimentation de l'oscillateur, la fréquence de ce dernier augmente et des motifs apparaissent dans la séquence de bits générée.

4.4 Exploitation des attaques cryptographiques

Dans cette partie nous détaillons les exploitations possibles de la perturbation d'un RNG. En effet certains algorithmes cryptographiques se basent sur le fait que les nombres aléatoires utilisés ont une entropie maximale pour assurer leur sécurité. La conséquence d'une attaque physique sur un RNG a pour effet de diminuer l'entropie des nombres qu'il génère et la sécurité des algorithmes est alors en doute.

4.4.1 Recherche exhaustive

Cette attaque est sans doute la plus intuitive et la plus simple. Son principe est d'essayer toutes les combinaisons possibles d'un nombre aléatoire jusqu'à trouver celui utilisé. Par exemple, dans un cryptosystème à clef privée en connaissant le message chiffré et en essayant toutes les clefs possibles nous pouvons retrouver le message en clair. Le choix de la taille de la clef permet d'éviter ces attaques, en rendant cette recherche impossible à effectuer dans un temps raisonnable.

Dans le cas où le RNG générant la clef est perturbé alors les probabilités d'occurrence des nombres aléatoires générés ne sont pas toutes égales. Les clefs peuvent alors être classées de la plus probable à la moins probable. La recherche exhaustive aura ainsi plus de chance d'aboutir plus rapidement.

4.4.2 Clefs apparentées

Les attaques par clefs apparentées exploitent le fait que des clefs utilisées dans un algorithme aient des similarités (bits identiques, etc.).

Un exemple bien connu de cette attaque a été réalisé sur l'algorithme RC4 et en particulier dans le cadre de son utilisation dans le protocole WEP (Wired Equivalent Privacy) [FMS01]. RC4 est un algorithme de chiffrement par flots, il combine les octets du message avec des octets issus d'un PRNG grâce à une opération XOR. Le PRNG est constitué d'un tableau de permutations rangées en mélangeant ces cases avec une clef. Un problème de cet algorithme concerne le fait qu'un grand nombre de clefs sont faibles : une petite partie de leurs bits détermine une grande partie de la permutation. Ceci se retrouve ensuite dans les premiers bits chiffrés. En conséquence, si un attaquant utilise des clefs ayant une partie similaire avec différents messages, il peut retrouver cette partie en étudiant le début des messages chiffrés.

Cette attaque a été décrite pour une application à des versions simplifiées de l'algorithme de chiffrement par blocs AES [BK09, BKN09, BDK+10]. Dans ce cas, le principe de l'attaque consiste à «traquer» la relation entre les sous-clefs des différents tours de l'AES durant son calcul. L'examen des chiffrés obtenus permet de retrouver les clefs utilisées.

4.4.3 Collisions

Ce type d'attaques exploite le fait que des nombres puissent être totalement ou partiellement identiques.

4.4.3.1 Nombres identiques

Dans ce cas de figure nous considérons la possibilité d'utiliser des paramètres identiques de manière involontaire. Ces paramètres peuvent être produits directement par le tirage de nombres aléatoires identiques (par exemple des nonces). Une autre possibilité est l'obtention de nombres aléatoires peu éloignés entre eux qui vont donner des paramètres identiques. Par exemple, pour produire des nombres premiers (utilisés dans divers cryptosystèmes) une méthode est de prendre ceux les plus

proches des nombres aléatoires générés. En conséquence, un même nombre premier peut être produit par plusieurs nombres aléatoires peu différents (seuls les bits de poids les plus faibles varient).

Avec l'algorithme DSA, il est possible de retrouver une clef privée à partir de deux messages chiffrés en utilisant des nombres aléatoires identiques. Pour cela il faut tout d'abord calculer la valeur du nombre aléatoire utilisé (équation 1.7) puis en déduire la clef privée (équation 1.8).

Dans le cryptosystème RSA, de telles occurrences permettraient à plusieurs personnes de partager le même module RSA ($N = pq$) ou un facteur premier (p ou q). Ces personnes seraient alors par exemple à même de déchiffrer des messages ne leur étant pas adressés. Une étude a cherché à vérifier que des nombres aléatoires différents sont bien utilisés pour générer des clefs dans des certificats [LHA+12]. Elle montre qu'un certain pourcentage des clefs et nombres premiers se recouvrent, en particulier 0,2% des premiers générant des modules RSA.

4.4.3.2 Bits identiques

La connaissance partielle des nombres aléatoires peut aboutir à la même connaissance de paramètres secrets mais les efforts à fournir sont plus importants.

Dans le cas de DSA il est possible de retrouver une clef privée en connaissant quelques bits des nombres aléatoires [NS00, NS03]. La contrepartie est qu'il faut utiliser plus de deux messages chiffrés par comparaison au cas où les nombres aléatoires sont entièrement connus.

Pour le cryptosystème RSA le but est de réussir à factoriser le module N en connaissant quelques bits des nombres premiers p et q. Cette factorisation est difficile à cause de la taille des nombres manipulés. La connaissance d'un nombre suffisant de bits de poids fort permet de la rendre faisable en temps polynomial. Dans [RS86] il est dit que la connaissance des $\frac{n}{3}$ bits de poids fort de p (avec n la taille de N) sont nécessaires à cette opération. Ce résultat a été amélioré par l'utilisation de la méthode de Coppersmith [Cop96] pour laquelle la connaissance de $\frac{n}{4}$ bits de poids fort est suffisante. Il existe un autre type de factorisation appelée implicite [SM09, MR09, BNT10, FMR10] : les méthodes utilisées considèrent deux modules avec des nombres premiers p ayant un certain nombre de bits identiques non connus. Ces techniques sont applicables quand les premiers p et q ont une certaine différence de taille, ce qui n'est pas recommandé pour la génération des modules RSA. Dans [BNT10], p et q sont toujours de différentes tailles mais la connaissance d'un seul module RSA est nécessaire. L'attaque se base sur la connaissance de $2 \times \log_2(q)$ bits successifs.

4.4.4 Clefs connues

Cette classe d'attaque s'appuie sur le fait que des clefs soient connues afin d'analyser le déroulement d'un algorithme et sa sécurité. L'attaque a par exemple été décrite dans le cadre d'un AES simplifié [KR07]. Elle consiste à vérifier que les occurrences de chaque octet d'un message en clair et de son chiffré suit une distribution uniforme.

Chapitre 5

Bancs de test

Ce chapitre introduit les différents bancs de test utilisés lors de cette thèse afin de perturber physiquement des générateurs de nombres aléatoires. Ceux-ci sont considérés dans leur environnement physique, c'est-à-dire la plateforme dans laquelle ils sont implantés. Cela signifie que la perturbation visant un TRNG va généralement être appliquée plus globalement au composant le contenant.

Nous décrivons ici les bancs de test que nous avons mis en place pour générer les perturbations suivantes :

- des variations positives et négatives de température pour sortir du fonctionnement normal des générateurs,
- l'application d'un faisceau laser, une technique couramment utilisée afin de perturber le fonctionnement d'un circuit électronique dans le domaine de la sécurité,
- l'injection d'un signal à une fréquence pour induire un verrouillage de la phase de plusieurs oscillateurs,
- l'utilisation de rayonnements ionisants induisant des modifications de fonctionnement des composants électroniques.

5.1 Perturbation par variations de température

Le but de cette expérience est de plonger les composants étudiés dans un environnement ayant une certaine température. Nous avons pour cela développé trois montages permettant de les chauffer ainsi que de les refroidir.

La figure 5.1 décrit le schéma de principe général des montages développés. Un PC pilote le fonctionnement du composant sous test, ainsi que la température appliquée. Un capteur mesure la température de l'environnement via un multimètre. Le composant est introduit dans l'environnement avec le capteur.

Un cas particulier que nous avons rencontré est un composant se chauffant par lui-même à cause de son activité. En temps normal ce composant est plaqué contre un radiateur afin de dissiper la chaleur. Pour notre expérience nous remplaçons le radiateur par un ventilateur auquel nous faisons varier la vitesse de rotation et ainsi sa température. De plus ce composant possède un capteur de température intégré dans son circuit électronique.

5.1.1 Résistance chauffante

Le chauffage de la puce se fait grâce à une résistance chauffante. En effet, selon la tension appliquée à ses bornes celle-ci délivre une certaine puissance qui est dissipée par effet Joule. La résistance

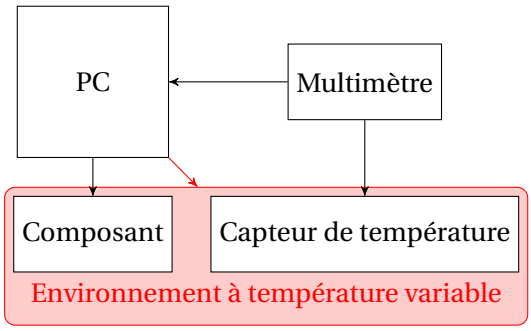


FIGURE 5.1 – Schéma de principe d'une perturbation par variation de température.

est maintenue contre un crochet dans lequel se propage la chaleur créée comme le montre la figure 5.2. Enfin, les composants étudiés sont soit glissés à l'intérieur du crochet soit plaqués contre lui pour être chauffés.

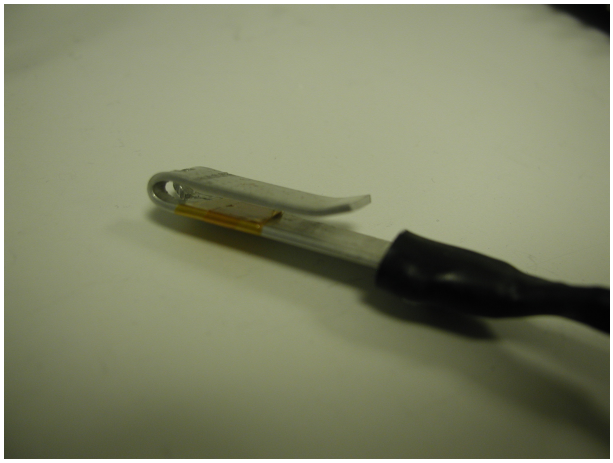


FIGURE 5.2 – Montage pour le chauffage d'un composant : une résistance chauffante est maintenue contre un crochet et diffuse ainsi la chaleur qu'elle dégage.

5.1.2 Module Peltier

Le module Peltier exploite l'effet Peltier [Gol10] : selon le sens du courant traversant deux matériaux semi-conducteurs différents il s'effectue une absorption ou un dégagement de chaleur à leur jonction. Le module est lui constitué de plusieurs de ces thermocouples interconnectés et placés entre deux plaques de céramique. En fonctionnement cela se traduit par l'obtention d'une face dégageant de la chaleur (face chaude) et d'une face l'absorbant (face froide). Selon le courant traversant le module et la tension à ses bornes il s'établit une certaine différence de température entre ses deux faces comme indiqué sur la figure 5.3. La figure montre l'évolution de la différence de température à courant constant. Pour différentes valeurs possibles du courant nous observons qu'une augmentation de la tension se répercute sur la différence de température et vice-versa.

Dans le montage, observable sur la figure 5.4a, un ventilateur est maintenu contre la face chaude du module Peltier afin d'évacuer la chaleur. En conséquence cela permet une plus grande diminution de la température sur la face froide. L'utilisation de pâte thermique permet d'améliorer la transmission de la chaleur entre le module Peltier et le ventilateur. Le composant est plaqué contre la face

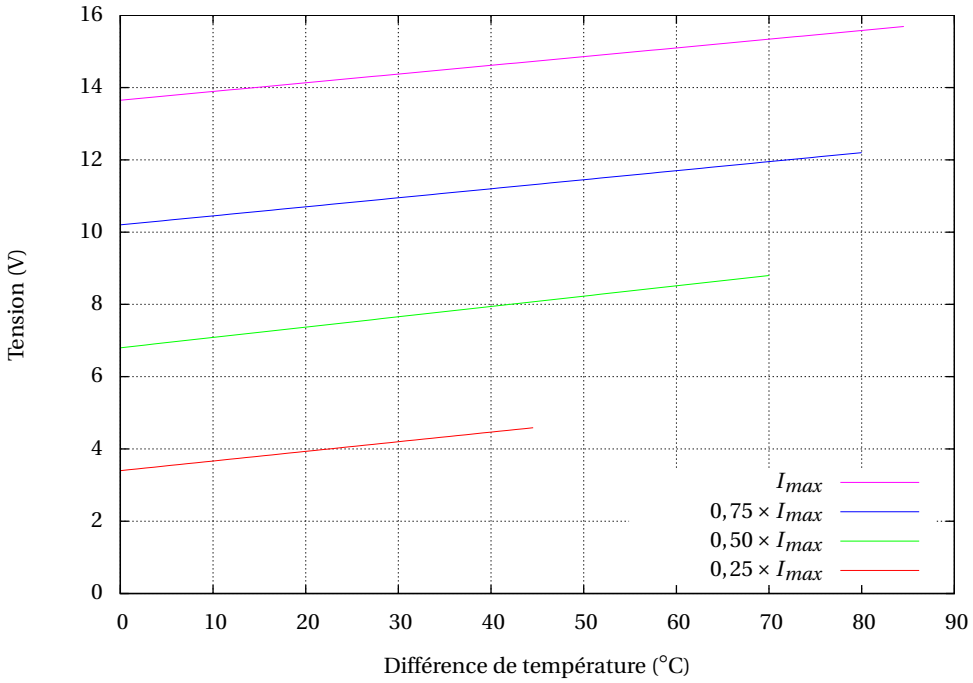


FIGURE 5.3 – Variation de la différence de température entre les faces du module Peltier en fonction de la tension et du courant appliqués.

froide. Ce montage nous permet d'atteindre des températures jusqu'à environ −30 °C.

Pour certains générateurs, les temps d'acquisition des nombres aléatoires se comptent en heures. Au cours du temps la condensation apparaît, entraînant la formation de givre à la surface du module et autour du composant. La conséquence est que la température se stabilise aux alentours de 0 °C. Une amélioration consiste donc à mettre ce montage dans un environnement sous vide (comme représenté dans la figure 5.4b) afin de supprimer la condensation et minimiser la température sur la face froide.

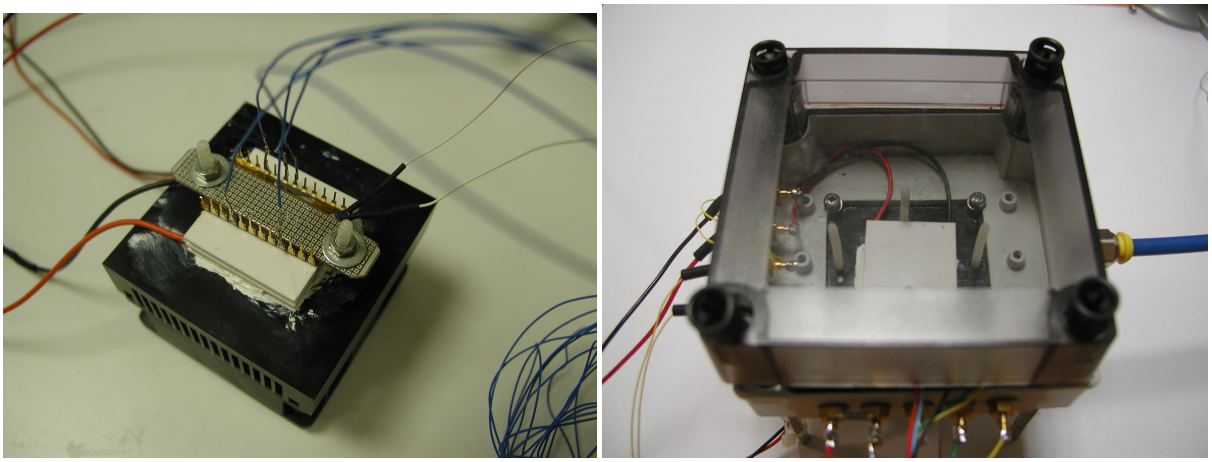
5.1.3 Azote liquide

Afin d'atteindre des températures encore plus faibles nous utilisons de l'azote liquide. En effet l'azote liquide a une température d'environ -196 °C et les vapeurs qu'il dégage ont donc une température échelonnée entre cette valeur et la température ambiante selon leur proximité à leur source.

Nous utilisons de l'azote liquide contenu dans un dewar (ou vase de Dewar). En effet, en contact avec l'air ambiant l'azote est en constante évaporation. Le dewar est un récipient ayant une très bonne isolation thermique nous permettant de conserver l'azote en minimisant les pertes.

Selon la taille du composant étudié nous pouvons l'introduire dans le dewar ou non. Si le composant est suffisamment petit, il est fixé sur une tige qui est ensuite introduite dans le vase. En réglant la hauteur de la tige dans les vapeurs d'azote, il est possible d'atteindre la température désirée jusqu'à -196 °C. Ce montage est représenté sur la figure 5.5. Au contraire, s'il n'est pas possible d'introduire le composant dans le vase alors une tige est insérée dedans jusqu'à en toucher le fond. La tige est en aluminium afin de bien conduire la chaleur. Le composant est plaqué à l'extrémité de la tige sortant du vase et est ainsi refroidi.

Ces montages permettent d'atteindre des températures très basses mais ont quelques inconvénients :



(a) Montage à l'air libre.

(b) Montage sous vide.

FIGURE 5.4 – Montage pour le refroidissement d'un composant : un module Peltier est maintenu contre un ventilateur pour évacuer la chaleur de sa face chaude et un composant est fixé sur sa face froide.

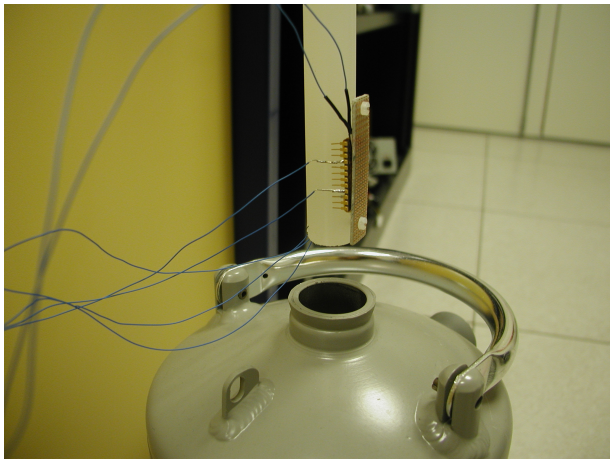


FIGURE 5.5 – Refroidissement de la puce grâce à des vapeurs d'azote.

- la thermalisation du composant est assez longue (plusieurs dizaines de minutes),
- au moindre déplacement de la tige la température varie beaucoup,
- l'azote ne peut pas être maintenu indéfiniment dans le dewar (la température reste stable quelques heures).

De plus, durant nos expériences nous avons pu observer un effet sur nos composants et plus précisément le silicium, appelé «freeze-out» [Sze81]. En effet les semi-conducteurs ont trois régimes de fonctionnement distincts dépendant de la température, que nous pouvons observer sur la figure 5.6. Ceci est dû en partie au dopage du silicium qui crée des impuretés en ajoutant des électrons ou des trous autour de ses atomes. À basse température, l'énergie thermique n'est pas suffisante pour ioniser toutes les impuretés et des électrons sont «gelés»(région de freeze-out). Avec le réchauffement du semi-conducteur il y a une augmentation des ionisations. Une fois toutes les impuretés ionisées, la concentration en porteur de charges reste stable (région extrinsèque). En chauffant encore le semi-conducteur, il y a une augmentation des porteurs de charge générés par l'excitation thermique (ré-

gion intrinsèque). En pratique nous avons observé un arrêt du fonctionnement des composants aux alentours de -150 °C (environ 123 K) à cause du gel des électrons.

–

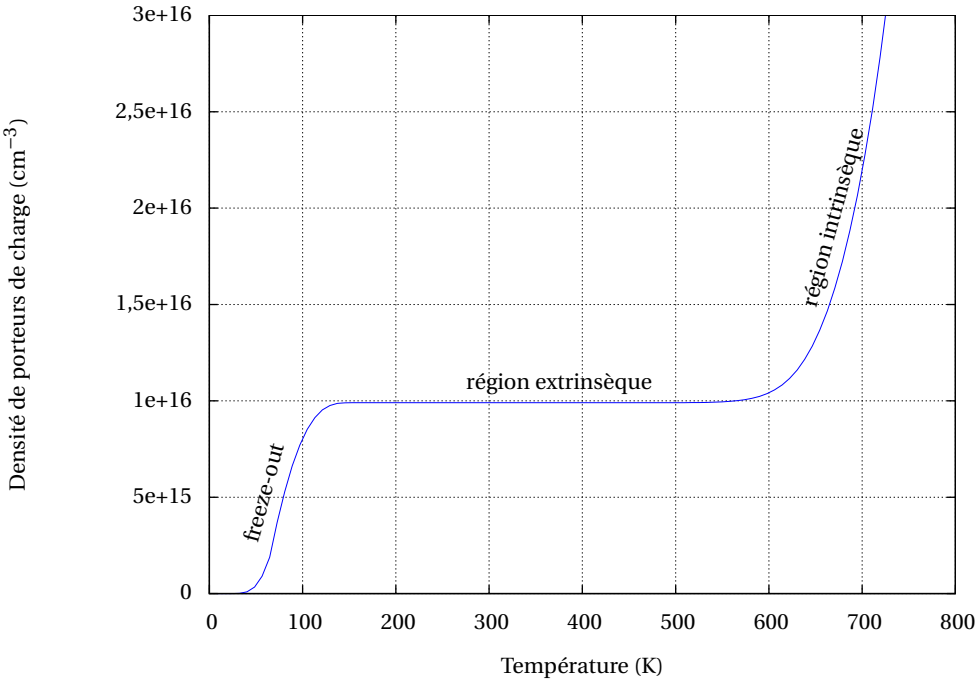


FIGURE 5.6 – Régimes de fonctionnement du silicium suivant sa température.

5.1.4 Capteurs de température

Deux types de capteurs ont été utilisés afin de mesurer la température appliquée à nos composants. Le choix du capteur dépend de la facilité de sa mise en œuvre pour un montage en particulier. Par exemple, avec la résistance chauffante la température peut être calculée simplement à partir de la tension appliquée à ses bornes, d'où l'utilisation d'un thermocouple. En revanche cette manipulation n'est pas possible avec le module Peltier et l'azote liquide, nous utilisons donc des capteurs de température à résistance de platine.

5.1.4.1 Le thermocouple K

Les thermocouples utilisent les effets thermoélectriques, c'est-à-dire que leur température est liée à la tension à leurs bornes. Le thermocouple est relié à la résistance chauffante et la tension à ses bornes donne la température de la résistance grâce à l'équation suivante :

$$T = c_0 + c_1 v + c_2 v^2 + c_3 v^3 + ... + c_n v^n \tag{5.1}$$

où T est la température en °C, v est la tension thermoélectrique en V, c sont les coefficients polynomiaux spécifiques au type de thermocouple utilisé et n est l'ordre maximum de l'équation.

Le standard ITS 90 (International Temperature Scale of 1990) [Bur93] fixe les valeurs des coefficients c. Pour un thermocouple K mesurant une température entre 0 et 500 °C, ils sont donnés dans le tableau 5.1.

c_i	Valeur
c_0	0,0
c_1	$2{,}508\,355 \times 10^{-2}$
c_2	$7{,}860\,106 \times 10^{-8}$
c_3	$-2{,}503\,131 \times 10^{-10}$
c_4	$8{,}315\,270 \times 10^{-14}$
c_5	$-1{,}228\,034 \times 10^{-17}$
c_6	$9{,}804\,036 \times 10^{-22}$
c_7	$-4{,}413\,030 \times 10^{-26}$
c_8	$1{,}057\,734 \times 10^{-30}$
c_9	$-1{,}052\,755 \times 10^{-35}$

TABLE 5.1 – Coefficients c pour le thermocouple K.

Dans le montage utilisant la résistance chauffante le thermocouple est directement intégré au fil d'alimentation de la résistance. Il est de plus branché à un voltmètre, permettant ainsi le calcul de la température.

5.1.4.2 Le capteur de température à résistance de platine

Le principe de fonctionnement d'un RTD (Resistance Thermometer Detector) est basé sur le fait que la résistivité d'un matériau dépend de sa température. La relation reliant ces deux grandeurs est la suivante :

$$R_T = R_0[1 + AT + BT^2 - C(T-100)T^3] \tag{5.2}$$

$$T = \frac{-A + \sqrt{A - 4B\left(1 - \frac{R_T}{R_0}\right)}}{2B} \tag{5.3}$$

où T est la température du RTD, R_0 est la résistance du RTD à 0 °C, R_T est la résistance du RTD à température T et A, B et C sont des coefficients dépendants du matériau utilisé.

Le RTD utilisé se nomme Pt100, il est composé d'un film métallique en platine de résistance 100 Ω à 0 °C et ses coefficients A, B et C sont indiqués dans le tableau 5.2. Ces coefficients sont fixés par une norme de la Commission Électrotechnique Internationale [Com08].

Coefficient	Valeur
A	$3{,}908\,30 \times 10^{-3}$
B	$-5{,}775\,00 \times 10^{-7}$
C	$-4{,}183\,01 \times 10^{-12}$

TABLE 5.2 – Coefficients A, B et C pour le capteur de température Pt100.

Dans le montage composé du module Peltier, le Pt100 est maintenu soit contre le composant étudié soit contre la face froide du module en fonction de l'accessibilité du composant. Pour le montage utilisant de l'azote liquide le Pt100 est toujours plaqué contre le composant. Le capteur est branché à un ohmmètre afin de calculer la température.

5.2 Perturbation par une source laser

Afin d'appliquer les effets produits par des faisceaux lasers aux générateurs de nombres aléatoires, nous avons réalisé un montage exploitant une source laser focalisée représenté sur la figure 5.7. Nous utilisons pour cela une diode laser ayant une puissance maximale de 25 W à 25 °C et une longueur d'onde de 979 nm. La puissance peut être réglée en modifiant les valeurs de tension et courant de l'alimentation. La relation entre la puissance en sortie et le courant appliqué est quasiment linéaire. La diode est fixée à un radiateur et une fibre la relie à la focalisation. Le faisceau laser est focalisé avec une lentille ayant un grossissement de 50 fois, en sortie duquel la puissance transmise (atténuation de 15 dB) atteint 3,75 W maximum. En conséquence, le diamètre du faisceau sur le composant est de 30 µm. Le PC pilote l'alimentation de la diode afin d'enchaîner plusieurs tirs lasers. Il pilote aussi une table XY déplaçant le faisceau laser au dessus du composant. Ceci permet d'impacter le composant à diverses positions puis de générer des cartographies à partir des résultats obtenus. Enfin le PC dialogue avec le composant pour lancer la génération des nombres aléatoires, récupérer des informations sur son fonctionnement et le réinitialiser en cas de plantage lors d'un tir laser.

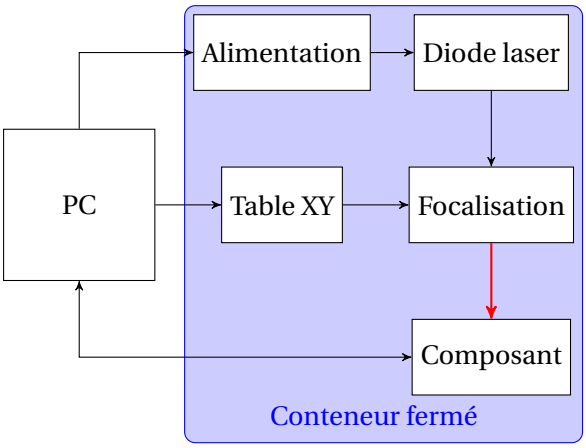


FIGURE 5.7 – Schéma de principe d'une perturbation par laser.

Afin d'obtenir une meilleur attaque sur un composant, celui-ci doit être préparé au préalable. Cette préparation consiste à retirer une certaine épaisseur de silicium (amincissement) afin d'optimiser la pénétration du faisceau dans le composant et d'atteindre les zones actives au plus près. Il s'agit d'une opération délicate, qui se réalise avec une micro-fraiseuse.

5.3 Perturbation par injection d'une fréquence

Cette attaque vise des TRNGs composés d'oscillateurs à anneaux. Son principe consiste à verrouiller la fréquence de ces oscillateurs. Plus précisément, les oscillateurs ont une sortie instable due à leur gigue de phase. À l'injection d'une certaine fréquence nous pouvons obtenir le contrôle de cette gigue. En particulier, cette manipulation conduit à l'élimination de la variation de phase entre plusieurs oscillateurs tournant en parallèle.

L'injection de la fréquence peut se faire soit directement en perturbant les signaux entrants du composant testé soit via un champ électromagnétique créant des signaux parasites dans le composant.

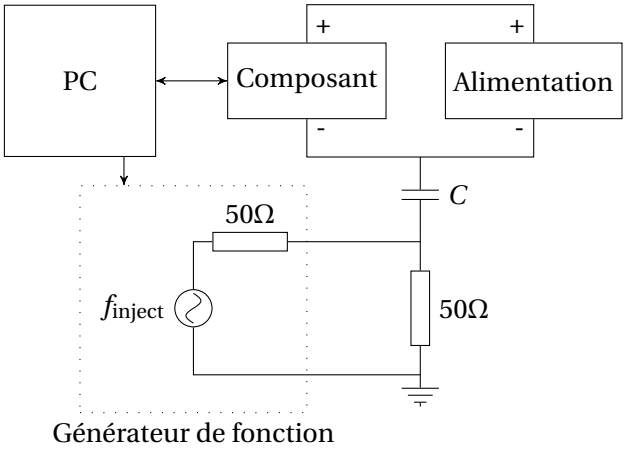


FIGURE 5.8 – Schéma de principe d'une perturbation par injection d'un signal sinusoïdal sur la masse d'un composant.

5.3.1 Injection directe dans un circuit

Cette méthode consiste à injecter un signal sinusoïdal à fréquence variable directement sur l'alimentation du composant testé. Dans la figure 5.8 nous expliquons le montage pour une injection sur sa masse. Nous ajoutons une résistance et une capacité afin de protéger le circuit et la fréquence est injectée par le biais d'un signal sinusoïdal produit par un générateur de fonction. Un PC commande le générateur de fonction et nous permet de faire varier la fréquence injectée durant le fonctionnement du composant sous test.

5.3.2 Injection via un champ électromagnétique

Notre montage pour perturber par rayonnement électromagnétique est présenté sur la figure 5.9. La perturbation est générée par une sonde constituée d'une tige avec une boucle à l'une de ses extrémités. Un signal sinusoïdal dont la puissance est amplifiée jusqu'à 2 W passe dans la boucle. Cette dernière est placée au dessus du composant sous test, sur une table XY afin de permettre son déplacement et générer des cartographies. Un PC pilote la génération du signal dans la boucle, sa position et les informations en provenance et vers le composant.

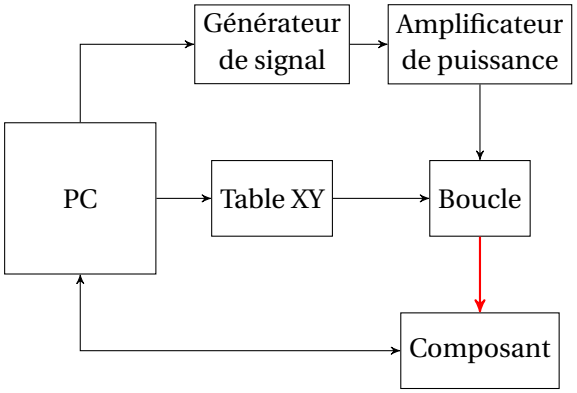


FIGURE 5.9 – Schéma de principe d'une perturbation par rayonnement EM.

Lors de nos expériences, la fréquence du signal EM généré et sa position au dessus du composant sont variées.

5.4 Perturbation par des rayonnements ionisants

Pour nos tests de perturbation des générateurs de nombres aléatoires nous avons utilisé deux bancs exploitant les rayonnements ionisants provenant de sources radioactives (rayons γ) et d'un tube à rayons X. En effet ces rayonnements sont très énergétiques et ont donc un grand pouvoir de pénétration dans la matière. Au contraire, les rayonnements α et β sont émis à des vitesses plus faibles et sont donc arrêtés plus facilement (par exemple une feuille de papier suffit pour les rayons α). Ils auraient donc nécessité de se placer au plus près des TRNGs ce qui n'était pas réalisable.

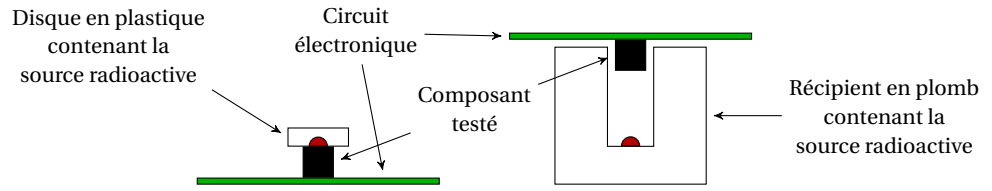
5.4.1 Rayons gamma

Quatre sources radioactives différentes sont utilisées et nous nous intéressons à leur émission de rayons γ. Chaque source est caractérisée par un niveau d'énergie (en électron-volts) et une activité (nombre de désintégrations par seconde en Becquerels) dépendant de la quantité de matière utilisée. Les sources utilisées et leurs caractéristiques sont données dans le tableau 5.3.

Source	Énergie (keV)	Activité (MBq)
Américium 241	60	3,7
Américium 241	60	518
Baryum 133	356	2,5
Césium 137	660	26

TABLE 5.3 – Caractéristiques des sources radioactives utilisées.

Le principe du montage est de mettre des sources radioactives à proximité du circuit électronique nous intéressant. Les sources sont donc placées au plus près possible du composant contenant le RNG testé. Elles sont intégrées dans des disques de plastique (figure 5.10a), à part le césium qui est contenu dans un vase en plomb (figure 5.10b).



(a) Perturbation avec des sources d'Américium et de Baryum.

(b) Perturbation avec une source de Césium.

FIGURE 5.10 – Montages pour des perturbations par sources radioactives.

5.4.2 Rayons X

Un générateur à rayons X est constitué d'un transformateur haute tension alimentant un tube à rayons X. A l'établissement d'une grande tension entre deux électrodes, un courant est créé entre la cathode et l'anode (cible). En arrivant sur la cible les électrons sont freinés par ses atomes, les

excitant. Les atomes réémettent alors un rayonnement X. Ce rayonnement est caractérisé par son intensité (proportionnelle au courant généré) et son énergie (en électron-volts, elle est fixée par la tension choisie).

La figure 5.11 décrit le montage pour la mise en œuvre des tests avec des rayons X. Le circuit électronique est placé de telle manière que le composant soit perpendiculaire aux rayons X émis par le tube. Des plaques en plomb permettent de protéger le circuit et d'impacter uniquement le composant.

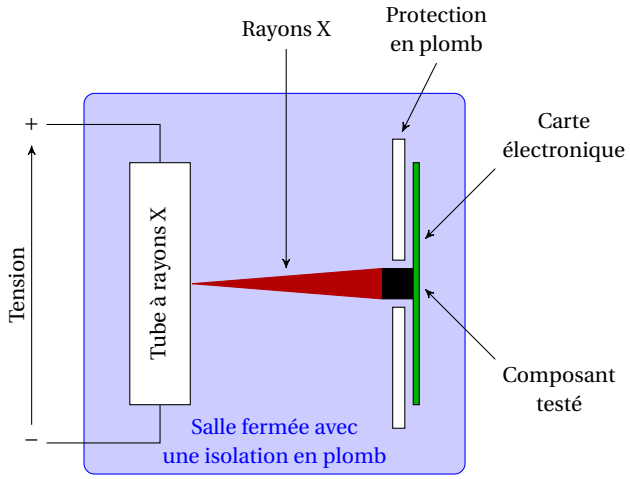


FIGURE 5.11 – Montage pour des perturbation par sources radioactives.

Lors des tests, on fait varier l'intensité et l'énergie des rayons ainsi que la distance du composant au tube.

Chapitre 6

Étude d'un TRNG exploitant du bruit thermique

Dans ce chapitre, nous étudions un TRNG implanté dans une puce. Nous voulons en particulier observer les effets de différentes perturbations sur son fonctionnement. Cela nous permettra de comparer leur influence et d'en déduire les vulnérabilités potentielles sur la conception du TNRG.

6.1 Principe de fonctionnement

La conception de ce TRNG est basée sur l'amplification différentielle du bruit thermique provenant de deux résistances, comme expliqué dans la section 2.2.1.2. La figure 6.1 montre le principe de fonctionnement de ce TRNG. Du bruit thermique est généré par les résistances R_1 et R_2. Celles-ci sont reliées à la masse via des condensateurs C_1 et C_2, ainsi les tensions en entrée de l'amplificateur ont leurs composantes continues supprimées pour ne garder que les variations dues au bruit thermique. La différence entre ces deux signaux est amplifiée successivement par plusieurs amplificateurs. Les composantes continues des signaux générés à chaque étape sont éliminées par des condensateurs C. Le dernier amplificateur génère un signal numérique en amplifiant la différence de ses deux entrées. Si le signal analogique différentiel est plus près de la tension à la masse ou de la tension d'alimentation des amplificateurs alors le signal numérique sera respectivement à 0 ou à 1. Le signal se maintient à l'un ou l'autre de ces états de manière irrégulière. L'échantillonnage de ce signal par une horloge produit donc une suite de bits aléatoire.

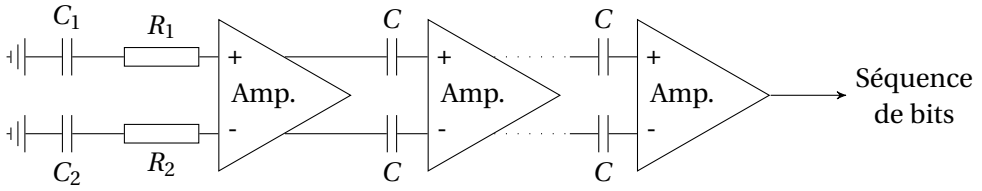


FIGURE 6.1 – Amplification différentielle du bruit thermique.

La conception de l'amplification avec une différenciation permet normalement de supprimer les signaux dus au bruit ambiant. En effet les deux résistances sont dans le même environnement et donc influencées identiquement par ce bruit.

A la suite de la génération physique de l'aléa nous ajoutons un retraitement simple afin de corriger les défauts les plus «grossiers» qui pourraient être présents. Deux retraitements différents sont

utilisés, soit un correcteur de von Neumann soit un LFSR.

6.2 Variations de température

Le TRNG que nous étudions étant basé sur l'amplification de bruit thermique de deux résistances, un changement de température devrait avoir un impact sur son comportement. En théorie il ne serait pas souhaitable de voir une telle influence se répercuter dans les séquences de bits générés. Nous étudions donc la sortie du TRNG soumis à différentes températures et plus particulièrement par comparaison avec son état normal.

La température ambiante est de 25 °C et le TRNG est testé à −85 °C, −55 °C, −30 °C, 35 °C, 70 °C et 125 °C. Pour cela les différents montages décrits dans le section 5.1 sont utilisés. La puce est constituée d'un micromodule inséré dans un boîtier DIL24 pour communiquer avec un PC.

Des séquences de 2,4 Mo sont générées à chaque température avec des temps d'acquisitions d'environ 8 heures. Ces acquisitions se font sur la sortie directe du TRNG, ainsi qu'après le passage des bits dans le correcteur de von Neumann ou le LFSR.

6.2.1 Résultats des tests statistiques

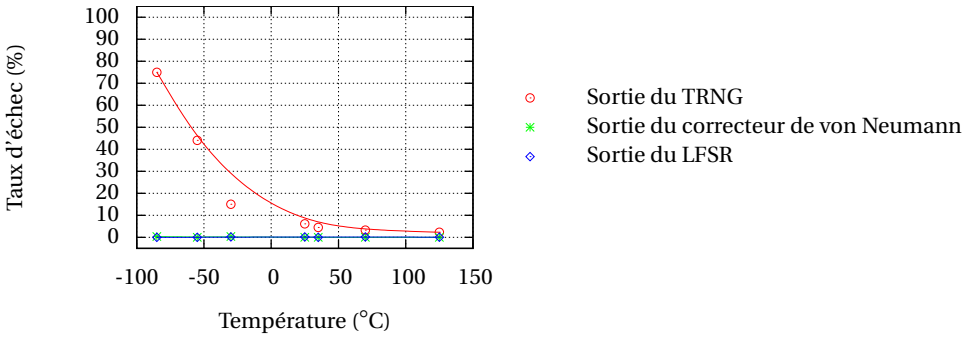
Les séquences générées sont examinées via leurs performances aux différents tests statistiques. La figure 6.2 représente les taux d'échec des séquences aux différents tests.

Nous observons tout d'abord que la qualité des bits générés par le TRNG sans retraitement et dans des conditions normales (sans perturbation) est assez médiocre. En effet tous les tests, à part le FIPS, présentent d'importants pourcentages d'échec. Avec l'application de la perturbation en variant la température, ces mêmes tests ont de légères variations de leur performances. Bien que des échecs soient attendus en fonctionnement normal, leur évolution est ici suspecte. Nous constatons l'aggravation du taux d'échec de des mêmes tests avec la diminution de la température et vice-versa. Ceci se vérifie plus clairement avec les tests FIPS où le nombre de sous-séquences échouant les tests passe de moins de 10% pour des températures ambiantes et supérieures à des proportions de près de 80% quand la température diminue.

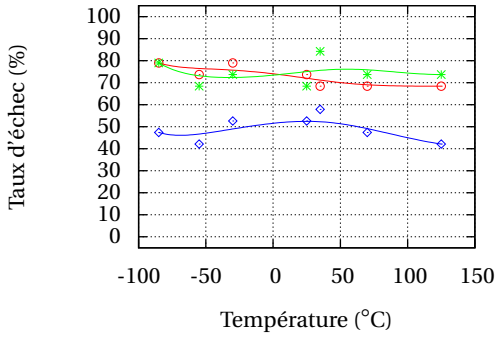
Après application des retraitements, le taux d'échec aux tests s'annule pour presque toutes les batteries qu'il s'agisse du correcteur de von Neumann et le LFSR. Les résultats des tests Diehard (figure 6.2b) montrent un taux d'échec toujours aussi important avec le correcteur de von Neumann alors qu'il diminue aux alentours de 45% avec le LFSR. Ces retraitements semblent donc être suffisants pour corriger la perturbation du TRNG, cependant les tests Diehard indiquent la présence de défauts. De plus, le LFSR est plus efficace que le correcteur de von Neumann, ce qui est attendu car ce dernier ne corrige que le biais dans le distribution des bits alors que le LFSR a potentiellement plus de pouvoir d'action corrective.

6.2.2 Évolution du biais dans la distribution des bits

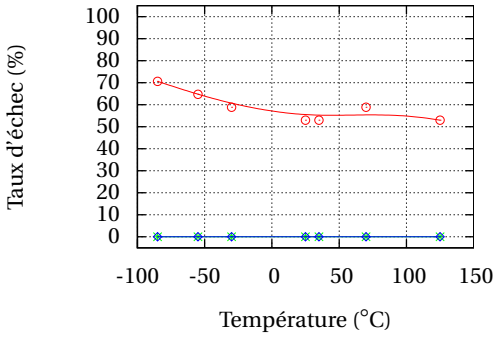
En examinant les tests du FIPS nous précisons les observations précédentes. Le tableau 6.1 donne les résultats à chaque test du FIPS. Le pourcentage total de sous-séquences échouant au moins à un test est principalement dû au test du monobit. La perturbation par la température se traduit donc par un biais dans la distribution des bits. La figure 6.3a montre la distribution des bits dans les sous-séquences. En théorie il devrait y avoir autant de bits à 0 qu'à 1 et la distribution du nombre de bits à 0 dans les sous-séquences devrait suivre une loi normale de moyenne 10 000 (chaque sous-séquence étant composée de 20 000 bits). À température ambiante un biais d'environ 2% est déjà présent dans la distribution des bits. Ceci est dû à la génération physique de l'aléa qui n'est pas «parfaite». De plus



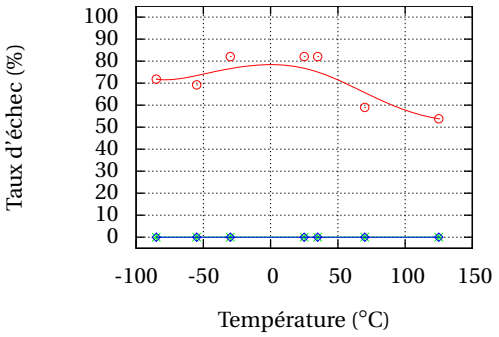
(a) Résultat des tests FIPS.



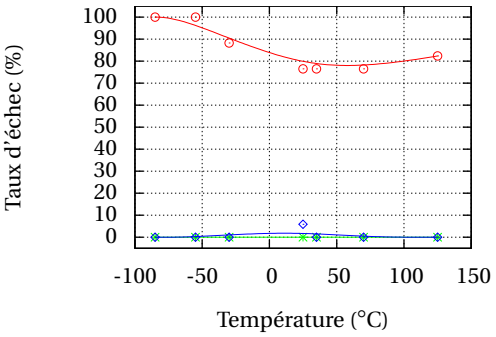
(b) Résultat des tests Diehard.



(c) Résultat des tests NIST.



(d) Résultat des tests Rabbit (TestU01).



(e) Résultat des tests Alphabit (TestU01).

FIGURE 6.2 – Résultat des tests statistiques pour le TRNG perturbé en température.

le biais évolue avec la température. Plus elle diminue, plus le biais augmente et vice-versa. Ainsi à −85 °C il atteint 3,2% et à 125 °C 1,3%.

La figure 6.3b et la figure 6.3c montrent la distribution des bits dans les sous-séquences après retraitement. Qu'il s'agisse du correcteur de von Neumann ou du LFSR, le biais disparaît. Le but même du correcteur de von Neumann étant l'élimination d'une telle déviation ce résultat n'est pas étonnant. De même, le LFSR a pour propriété de générer des séquences ayant autant de bits à 0 qu'à 1.

La figure 6.4 représente la déviation de la distribution des mots de 8 bits par rapport à la théorie et la séquence testée est considérée comme une suite de tels mots. Deux distributions sont utilisées : celle des valeurs des mots et celle de leur poids de Hamming. La déviation est calculée de la manière

Température (°C)	Monobit	Poker	Runs	Long Runs	Total
-85	74,97	17,88	5,96	0,09	74,97
-55	43,94	5,37	1,56	0,00	44,04
-30	14,84	0,58	0,58	0,00	15,03
25	5,86	0,29	0,48	0,00	6,15
35	4,49	0,19	0,19	0,00	4,49
70	3,22	0,00	0,09	0,00	3,32
125	2,24	0,00	0,19	0,00	2,34

TABLE 6.1 – Pourcentage de sous-séquences échouant aux tests FIPS.

suivante :

$$D_i = 100 * \frac{P_i - Pt_i}{Pt_i} \tag{6.1}$$

où i est le mot considéré, P_i est sa probabilité d'apparition observée, Pt_i est sa probabilité d'apparition théorique et le résultat est exprimé en pourcentage de cette dernière.

La figure montre l'évolution de ces déviations avec la variation de température. Nous constatons encore une déviation pour des températures négatives qui disparaît quand le composant se réchauffe. Cette déviation, due au biais, est vérifiée par la distribution du poids de Hamming des mots montrant qu'il existe plus de bits à 0. Enfin, en appliquant les retraitements cette déviation disparaît.

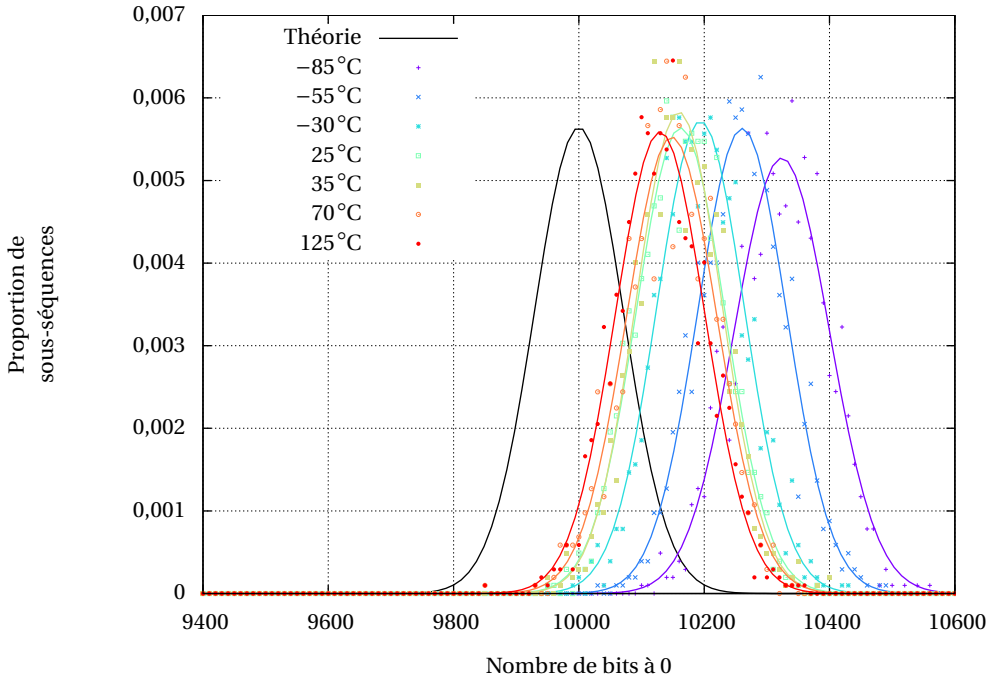
6.2.3 Impact de la variation de température sur la puce

Cette évolution du biais s'explique par le mode de génération du bruit physique. En effet, l'amplitude du bruit thermique est directement liée à la température des résistances. La diminution de la température implique donc aussi celle du bruit thermique et les signaux amplifiés ont donc des amplitudes plus faibles. En conséquence, leur différence résulte aussi en un signal de faible amplitude. La numérisation se faisant en comparant ce signal à un seuil (bit à 0 s'il le signal est inférieur au seuil, à 1 sinon), il y a donc plus de chances de générer des bits à 0 qu'à 1, d'où la présence d'un biais. Plus la température diminue, plus ce biais s'aggrave.

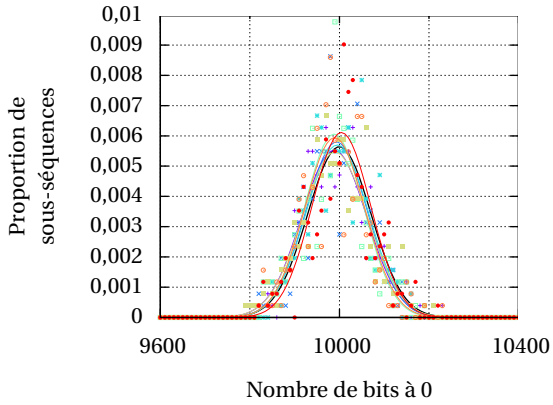
Au contraire quand la température augmente, le bruit thermique aussi. Les signaux ont ainsi de plus grandes amplitudes et après amplification ont plus de chances de prendre des valeurs de manière uniforme parmi toutes celles possibles en sortie des amplificateurs. Ceci se répercute sur la différence des deux signaux et en ayant les mêmes probabilités d'avoir le signal en dessous et au dessus du seuil, nous obtenons donc une séquence avec une distribution des bits se rapprochant de la distribution théorique uniforme après numérisation. La conséquence est une diminution du biais.

6.3 Attaque laser

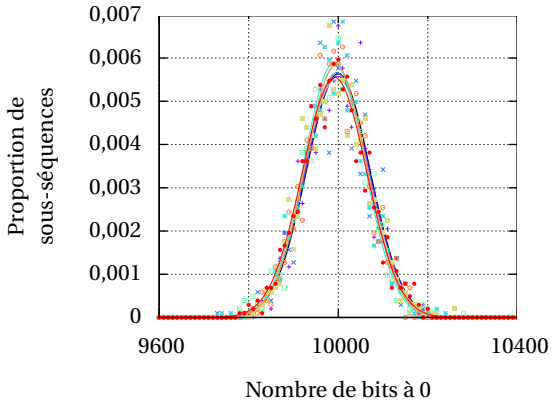
Les perturbations par variation de température impactent globalement le composant contenant le TRNG. Avec l'application d'un faisceau laser nous voulons focaliser la perturbation sur le TRNG lui-même. Nous utilisons pour cela la puce dans une configuration de carte à puce. Cette dernière est glissée dans un lecteur de carte avec une ouverture pour l'accès du laser à la face arrière de la puce. Durant la préparation de la puce, l'accès au silicium est dégagé et elle est amincie afin d'améliorer l'impact du laser sur ses fonctions.



(a) TRNG sans retraitement.



(b) Sortie du correcteur de von Neumann.



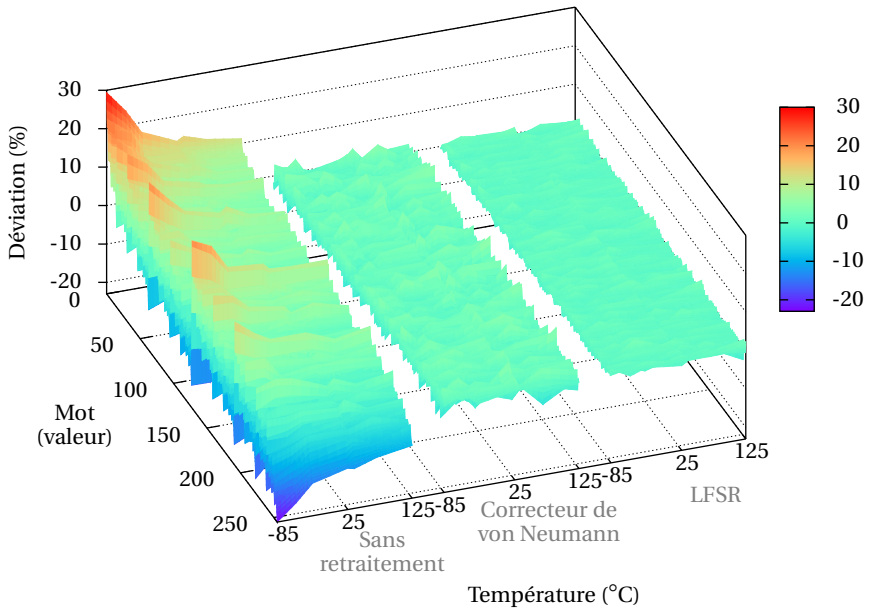
(c) Sortie du LFSR.

FIGURE 6.3 – Distribution du nombre de bits à 0 dans les sous-séquences de 20 000 bits avec le TRNG perturbé en température.

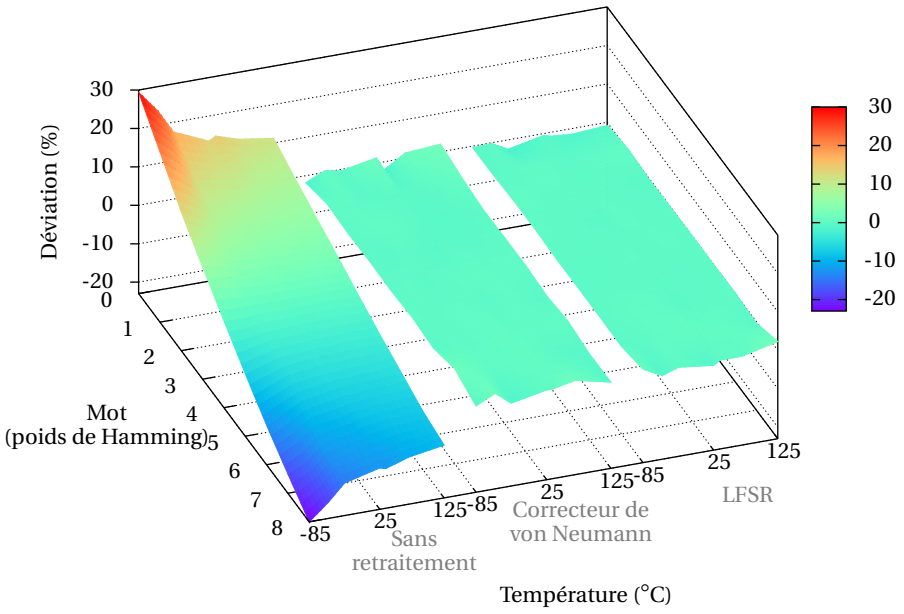
L'attaque se fait sans connaître l'emplacement exact du TRNG. En conséquence nous balayons les zones analogiques de la puce, dans lesquelles doit se trouver la partie physique de la génération de l'aléa.

6.3.1 Cartographie des zones logiques de la puce

Le principe est de positionner le laser au dessus de la puce, d'envoyer un faisceau lumineux durant 5µs pendant la génération d'aléa, ceci une dizaine de fois, puis d'avancer d'un pas (50µm), etc. jusqu'à avoir testé toute la zone visée. À chaque position le PC envoie une demande pour des nombres aléatoires. La puce répond en générant 256 octets puis en les envoyant au PC. En analysant cette ré-



(a) Déviations dans la distribution des mots selon leur valeur.

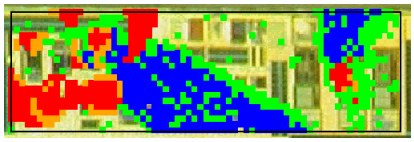


(b) Déviations dans la distribution des mots selon leur poids de Hamming.

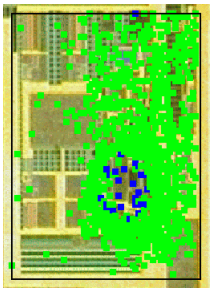
FIGURE 6.4 – Évolution des déviations dans la distribution des mots avec la température.

ponse nous déduisons un type d'effet associé à la position sur laquelle se trouve le laser. En faisant l'expérience plusieurs fois au même endroit nous augmentons les chances de bien détecter un effet dû à la perturbation. Ainsi nous construisons des cartographies de la puce illustrées sur la figure 6.5. Plusieurs effets sont observés :

– le reset de la puce (en rouge et orange),
– la mise à 0 de deux octets (en vert),
– la mise à 0 d'une suite d'octet (en bleu).



(a) Première partie analogique attaquée.



(b) Seconde partie analogique attaquée.

FIGURE 6.5 – Cartographie de la puce sous attaque laser.

6.3.2 Mise à 0 de deux octets

Un des deux effets intéressants d'un coup de laser sur la puce est la mise à 0 de deux octets parmi les 256 générés. L'exemple suivant (figure 6.6) présente une génération d'aléa avec les octets en hexadécimal et pour laquelle deux octets sont mis à zéro (en rouge). À première vue un tel effet ne semble pas remarquable car nous nous attendons à obtenir toutes les combinaisons de bits possibles, il a donc une certaine probabilité d'occurrence. En revanche, en répétant l'expérience un grand nombre de fois nous pourrions remarquer que le nombre d'occurrences du motif «00 00» s'éloigne de plus en plus de ce qui est attendu car le motif est présent à chaque génération.

La figure 6.6 montre les différents signaux provenant du laser et de la puce affichés sur un oscilloscope. La communication entre le PC et la puce est en rose, la consommation de la puce en jaune et celle du laser en bleu. La figure montre le déroulement du scénario : la commande de génération de l'aléa par le PC, le coup de laser peu après et la réponse de la puce avec les octets produits (au centre de l'écran). De plus, en modifiant l'instant auquel se produit le coup de laser nous voyons la place des octets mis à 0 varier en conséquence.

6.3.3 Mise à 0 d'une suite d'octets

Sur l'exemple suivant (figure 6.7) nous observons l'apparition d'une suite d'octets à 0 se fait à partir du coup de laser. La différence avec l'effet décrit précédemment est que la réponse de la puce intervient directement après le coup de laser ce qui explique que la suite d'octets à 0 est en fait due à l'absence de génération de l'aléa.

6.3.4 Impact du coup de laser sur la puce

Comme observé précédemment, la mise à 0 d'une suite d'octets n'est pas due à une génération de bits à 0. Cela signifie que ce n'est pas le TRNG lui-même qui est perturbé mais plutôt le déroulement du programme de génération de l'aléa. De même la mise à 0 de deux octets peut s'expliquer par le déroutement du programme. En effet, dans ce cas au lieu que la génération des octets ne n'interrompe en plein milieu, le programme saute une instruction de génération sachant que celles-ci se font sur des registres de 16 bits d'où les deux octets à 0.

```
42 B1 5E 13 AE 64 21 FE 3F 0E D0 E8 C5 8D C2 F6 09 1A F2 C3 B9 B7 5D 97 76 DC DB B9
E7 E9 00 00 19 CF 15 CA 65 98 7A 6A 79 6F 74 26 03 5C D0 64 1C 07 52 5D 6F F3 09 86
2E 0D 27 6D 28 AF 83 98 5C 7F 6D E0 00 00 95 D1 20 4E BE 9E AB 7C EC D6 AC 57 7D 88
DD E5 B1 AB 33 6A 89 9E 59 74 94 C0 3A 76 87 68 4B 67 25 F2 D0 0A 47 B8 A0 69 25 48
E2 76 D6 09 A0 9A 48 3D 1A 01 9D 60 81 A8 DA B7 0B D0 7E 10 94 69 78 18 9A E7 19 A6
EB 0A 6E 87 AC 80 4B 44 06 4C F7 7C 70 B5 99 D6 8C F0 A6 16 0E BE AD 98 25 17 4E 96
4D C8 9C C4 6E 30 E2 FE 84 E5 1A 2E C4 79 30 17 F3 6C 65 B0 B0 21 6B 4E 96 9C DB 26
BC 6D 85 13 44 E7 71 3A 74 FF 37 D4 75 B5 42 2E 1D 4F F4 53 19 B7 BB 32 4F F4 FA 9A
8A BE F2 64 5C 84 E0 5C 49 64 67 02 B5 64 14 55 16 99 4C B8 47 2A C0 40 D2 32 7B 50
CE EE B4 9D
```

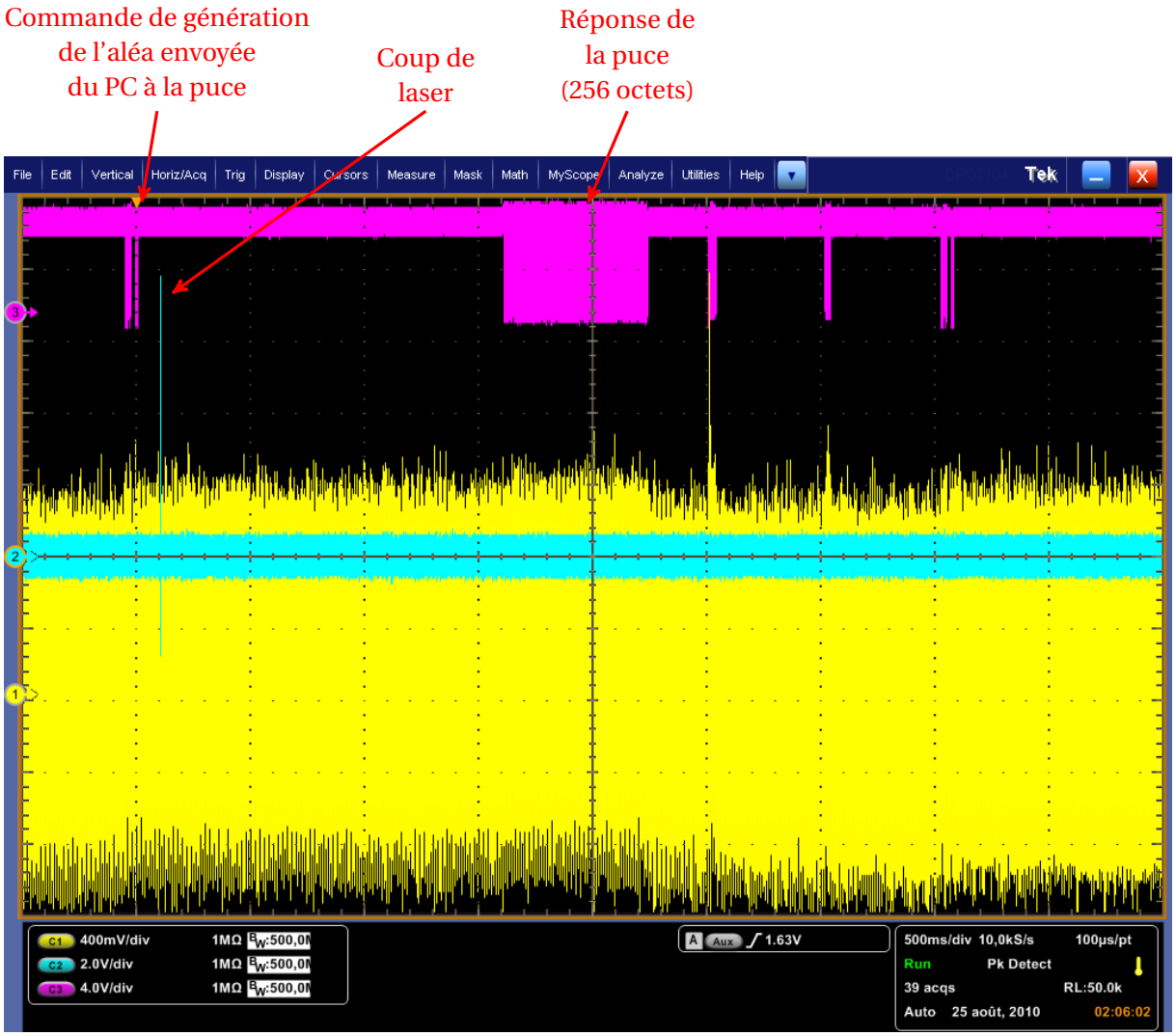


FIGURE 6.6 – Visualisation de la réponse de la puce et des différents signaux lors d'un coup de laser entrainant l'apparition de deux octets à 0.

Ce comportement est vérifié en effectuant la même expérience et en ajoutant le correcteur de von Neumann à la suite de la génération de l'aléa. En effet nous retrouvons les mêmes effets (mise à 0 d'octets) alors que dans le cas où la source physique d'aléa avait été perturbé, les octets à 0 auraient

```
8F B2 88 F4 1D AE F4 BA 58 A8 2B 50 FB 99 E9 DF 3D 56 17 D7 D5 31 C1 AD 66 38 44 F6
89 4A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00
```

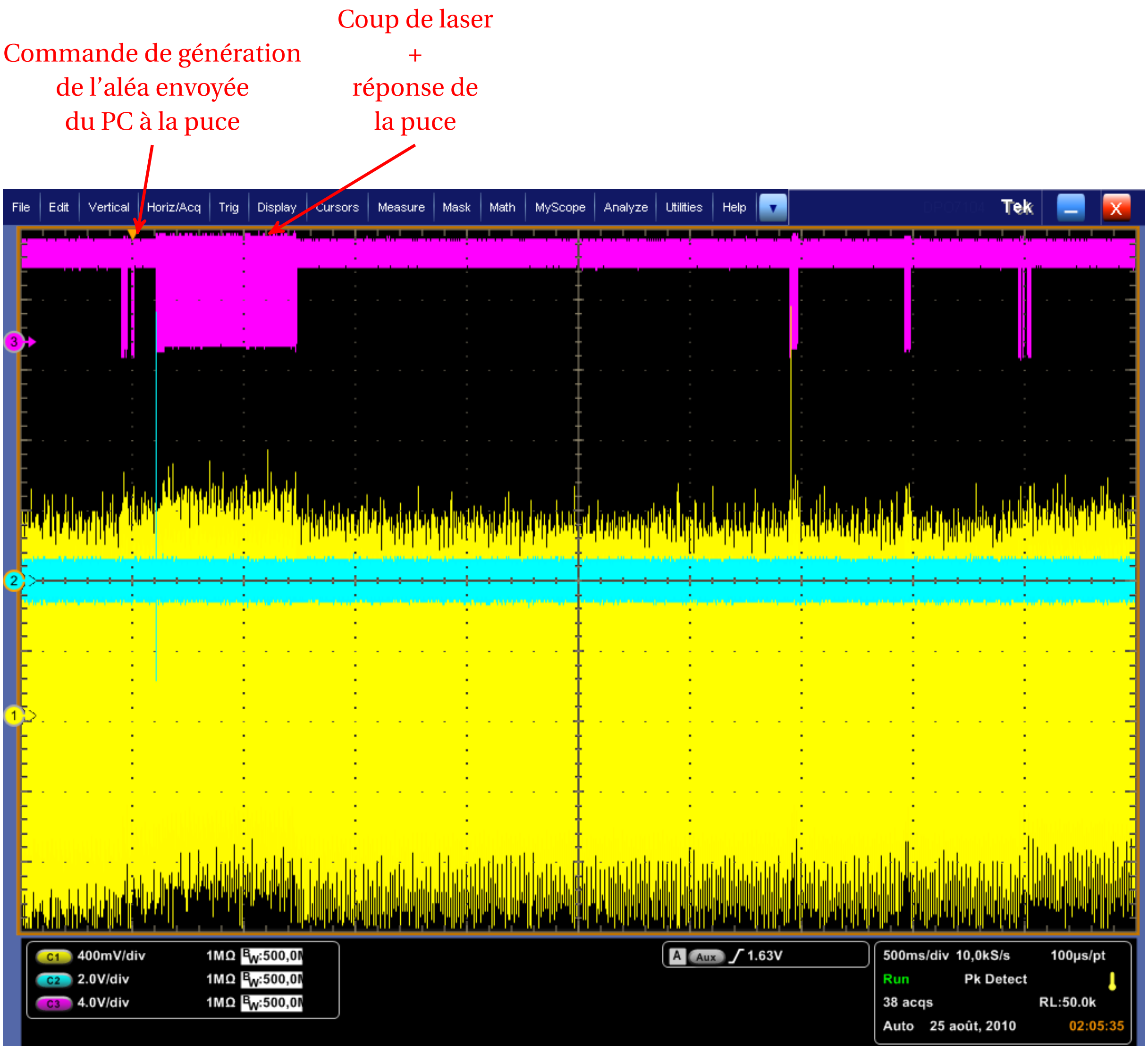


FIGURE 6.7 – Visualisation de la réponse de la puce et des différents signaux lors d'un coup de laser entrainant l'apparition d'une suite d'octets à 0.

été éliminés par le correcteur.

La perturbation induite par le laser est donc due à la modification d'une valeur dans un registre,

entrainant la modification du déroulement normal du programme de génération d'aléa. Ce type d'attaque est typiquement utilisé sur des algorithmes cryptographiques afin de révéler des données normalement secrètes.

Bien que les coups de laser ne semblent pas impacter le TRNG, les effets obtenus sur la génération de l'aléa peuvent poser problème. En effet, non seulement la perturbation est praticable en de nombreux points de la puce mais en plus elle n'est pas forcément détectable par des tests statistiques (cas des deux octets mis à 0) car trop localisée. Enfin un attaquant réalisant cette perturbation aurait la maîtrise précise des octets mis à 0. Il pourrait en effet viser des octets de manière précise (position dans le groupe de 256 octets) et aussi répéter l'attaque plusieurs fois.

6.3.5 Une exploitation

Nous détaillons ici une conséquence possible du résultat obtenu précédemment, la mise à 0 de deux octets avec une maîtrise de leur localisation dans la génération d'aléa.

Nous imaginons maintenant que cet aléa soit utilisé comme vecteur d'initialisation dans un algorithme de chiffrement par bloc en mode CBC (Cipher Bloc Chaining). Ce mode d'opération a pour principe le chaînage des calculs, c'est-à-dire que le résultat obtenu sur un bloc d'un message est réutilisé sur le suivant. Le chiffrement d'un bloc dépend donc du bloc de message en clair correspondant ainsi que de tous les blocs précédents. La figure 6.8 montre ce fonctionnement : le bloc chiffré $\mathscr{C}_1$ du 1er bloc en clair $\mathscr{M}_1$ est XORé avec le 2e bloc en clair, etc.

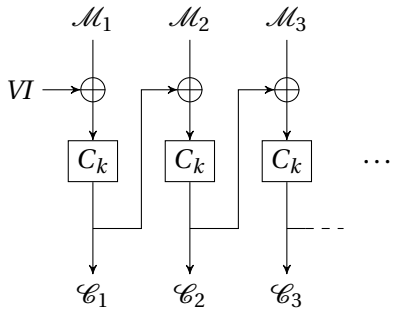


FIGURE 6.8 – Principe de fonctionnement du chiffrement pour un algorithme de chiffrement par blocs en mode CBC.

En outre, le premier bloc en clair n'a pas de prédécesseur chiffré et est donc XORé à un vecteur d'initialisation (IV : Initialisation Vector). Ceci permet d'éviter que le chiffrement de deux messages identiques avec une clef k donne le même message chiffré. Plus particulièrement, cela empêche le chiffrement de deux messages différents mais ayant des blocs en clair $\mathscr{M}_1$ identiques d'avoir le même bloc chiffré $\mathscr{C}_1$. Le vecteur d'initialisation est donc un nombre aléatoire et rend chaque message unique.

Dans le cas de notre attaque par perturbation, le vecteur d'initialisation est généré par le RNG et nous savons donc que deux octets sont à 0. Les messages ne sont plus correctement randomisés et nous pouvons ainsi gagner de l'information sur leur contenu.

6.4 Conclusion

L'attaque de ce TRNG implanté dans une carte à puce a permis de montrer des vulnérabilités. En effet la variation de température, qui se réalise grâce à des moyens relativement simples, a montré que

la source de bruit physique est sensible à un bruit extérieur. La différenciation des signaux de bruit thermique est censée éliminer tout bruit extérieur, ce qui n'est pas le cas et est dû au fait qu'il s'agit de la même source de bruit sur laquelle repose le fonctionnement du TRNG. Le biais ainsi généré dans la distribution des bits peut être corrigé avec des retraitements simples.

L'attaque laser a montré qu'en plus de la source de bruit physique, d'autres chemins d'attaque sont possibles sur le TRNG et peuvent poser problème. Il est en effet tout aussi dangereux d'avoir un programme vulnérable à un déroutement de code comme nous l'avons observé. De plus cette technique nous permet la maîtrise précise de l'erreur induite (localisation et durée).

Chapitre 7

Étude d'un TRNG exploitant des oscillateurs

Dans ce chapitre, nous étudions un second TRNG toujours implanté dans une puce. Nous observons en particulier les effets de différentes perturbations sur son fonctionnement et en comparaison au précédent TRNG étudié. De plus, la conception de ce TRNG est basée sur l'utilisation d'oscillateurs qui est couramment utilisée comme source d'entropie.

7.1 Principe de fonctionnement

La conception de ce TRNG est basée sur l'échantillonnage d'oscillateurs tournant en parallèle, comme expliqué dans la section 2.2.2.2. La figure 7.1 montre le principe de fonctionnement de ce TRNG. Il comporte six oscillateurs : cinq d'entre eux tournent à des fréquences fixes toutes différentes, le dernier tourne librement donc sa fréquence varie et sa phase se décale par rapport aux autres oscillateurs. Les oscillateurs sont XORés deux à deux et le signal résultant est échantillonné dans une bascule D grâce à un signal de reset. Cette dernière opération permet d'éviter tout problème de métastabilité, la bascule garantissant la mémorisation d'un état stabilisé à son entrée. En effet avec les transitions des oscillateurs et le passage des signaux dans les portes XOR, des fluctuations très rapides des signaux peuvent apparaître. Les fonctions XOR et inverseur mettent les signaux intermédiaires à 0 ou à 1 selon que les paires d'oscillateurs ont des états différents ou similaires. Nous obtenons une séquence de bits aléatoires à la sortie de la bascule.

Nous ajoutons un retraitement à la suite de la source physique d'entropie (correcteur de von Neumann et LFSR), de manière identique au TRNG étudié dans le chapitre précédent. Ceci nous permet de comparer leur performance, ainsi qu'avec le précédent générateur exploitant le bruit thermique de résistances.

7.2 Variations de température

Des études ont montré que des TRNGs du même type que celui que nous utilisons ici (oscillateurs tournant en parallèles puis XORés) sont sensibles à des variations de température [SSR09, YKBS10, ŠDF11]. Cette expérience nous permet donc de mettre en évidence un tel phénomène sur notre TRNG, ainsi que l'influence des retraitements sur l'élimination des perturbations induites et de comparer les performances obtenues au précédent TRNG étudié.

Le TRNG est testé à trois températures : 25 °C (température ambiante), −30 °C et 70 °C. Nous utilisons pour cela les montages de réchauffement grâce à une résistance et de refroidissement par mo-

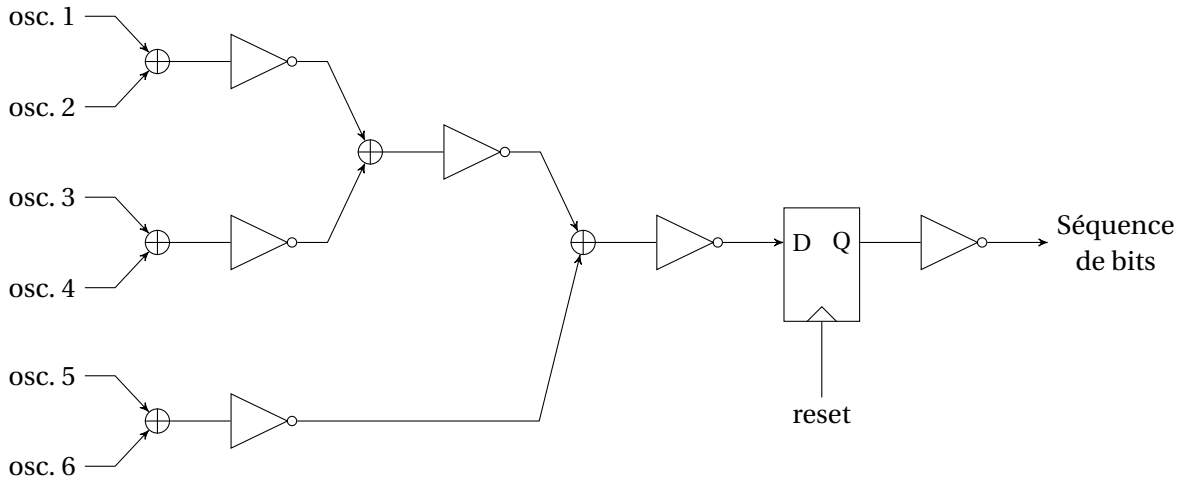


FIGURE 7.1 – Échantillonage d'oscillateurs tournant en parallèle.

dule Peltier présentés dans le section 5.1. La puce est contenue dans un micromodule inséré dans un boîtier DIL24 afin de communiquer avec un PC.

Des séquences de 102,4 Mo sont générées à chaque température pour des temps d'acquisitions d'environ 80 heures. Ces acquisitions se font sur la sortie directe du TRNG, ainsi qu'après le passage des bits dans le correcteur de von Neumann ou le LFSR, de manière similaire à l'étude du précédent TRNG.

7.2.1 Résultats des tests statistiques

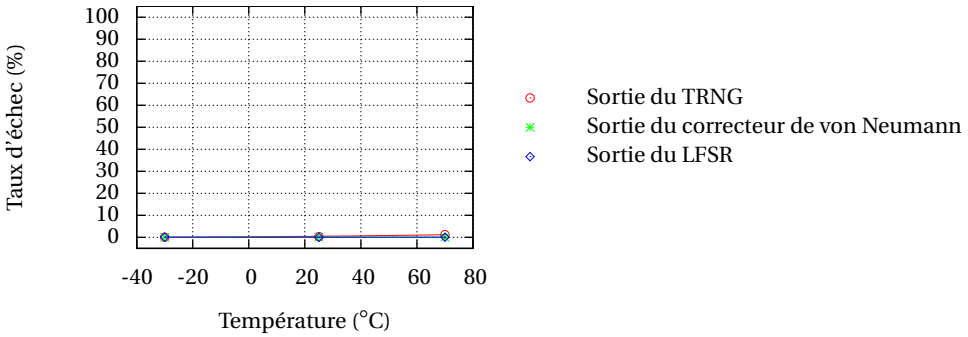
La figure 7.2 montre les taux d'échec aux tests statistiques des séquences générées à chaque température. Nous observons tout d'abord que la qualité des bits générés par le TRNG sans retraitement et dans des conditions normales (sans perturbation) est plutôt bonne et en tout cas meilleure que celle du TRNG étudié précédemment. En effet les tests FIPS et Diehard ont des taux d'échec très proches de zéro tandis que ceux des batteries du NIST et TestU01 ne dépassent pas 30%.

L'application de la perturbation en variant la température entraîne de légères variations de la performance du TRNG sur ces derniers tests. Il y a plus particulièrement une faible aggravation des taux d'échec avec le chauffage du composant, en revanche en diminuant la température nous observons plus clairement leur amélioration.

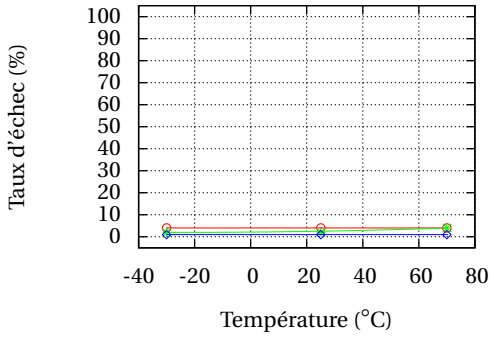
L'application du retraitement avec le LFSR supprime tous les défauts détectés par les tests statistiques et ceci pour toutes les températures. Le correcteur de von Neumann améliore aussi les performances du TRNG mais avec une efficacité moindre. En effet, il semble suivre l'évolution de la sortie brute du TRNG en ne corrigeant qu'un pourcentage à peu près stable de sous-séquences. Dans le cas où la température est à 70 °C cela veut dire que l'effet du retraitement n'est pas assez important pour annuler le taux d'échec alors qu'à −30 °C il y arrive presque.

7.2.2 Évolution du biais dans la distribution des bits

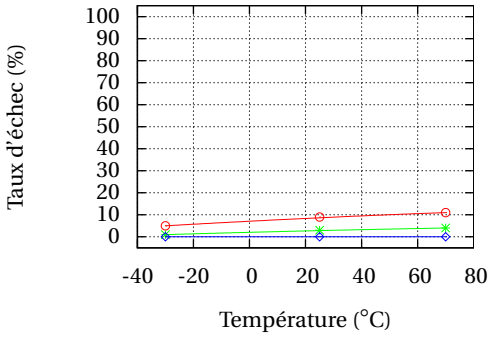
Le tableau 7.1 donne le détail des résultats des tests FIPS pour chaque température et sans retraitement. Le taux d'échec total est principalement dû aux tests Monobit et Runs. Il existe donc un biais dans la distribution des bits. La figure 7.3a montre cette distribution en examinant le nombre de bits dans les subdivisions des séquences testées. À température ambiante nous observons un biais d'en-



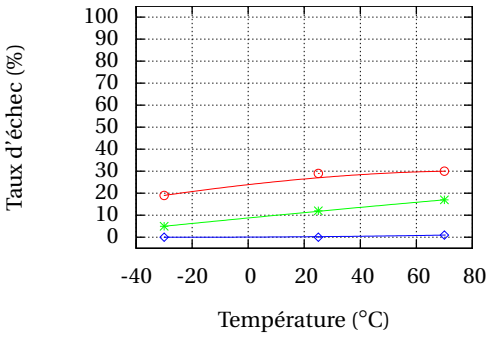
(a) Résultat des tests FIPS.



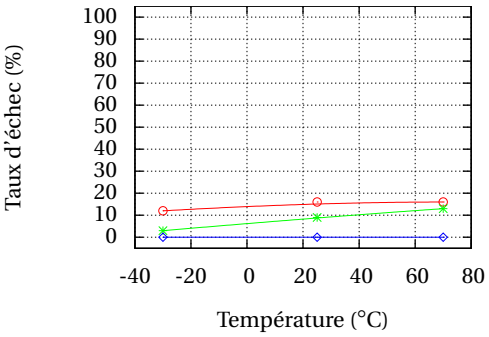
(b) Résultat des tests Diehard.



(c) Résultat des tests NIST.



(d) Résultat des tests Rabbit (TestU01).



(e) Résultat des tests Alphabit (TestU01).

FIGURE 7.2 – Résultat des tests statistiques pour le TRNG perturbé en température.

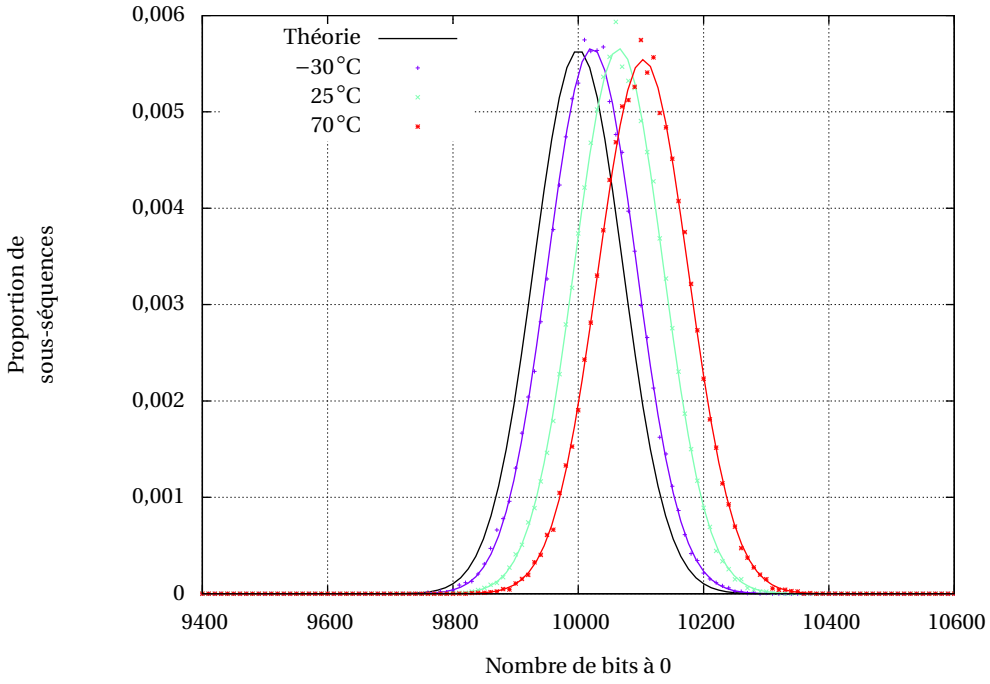
viron 0,65% dû au fonctionnement de la source de bruit physique. Ce biais augmente et diminue avec la température de manière contraire au précédent TRNG étudié. En conséquence, à −30 °C et 70 °C il atteint respectivement 0,22% et 1,04%.

La figure 6.3b et la figure 6.3c montrent la distribution des bits dans les sous-séquences après retraitement. Sans surprise, le biais est éliminé que ce soit par l'application du correcteur de von Neumann ou par l'utilisation du LFSR.

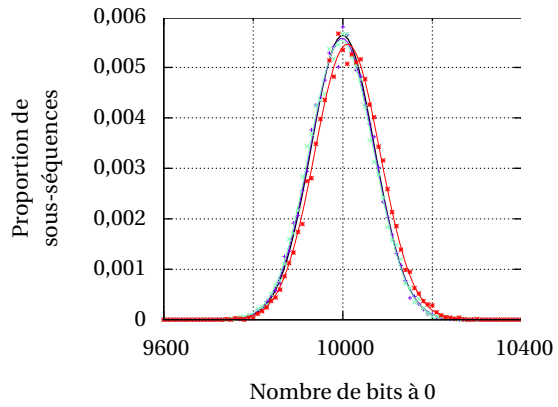
La figure 7.4 représente la distribution des mots de 8 bits des séquences testées en considérant soit la valeur des mots soit leur poids de Hamming. L'évolution de la distribution des mots en fonction de leur poids de Hamming montre l'évolution du biais sans utilisation de retraitement : à 70 °C les mots de faible poids de Hamming sont plus nombreux que ceux de grand poids de Hamming et donc il y

Température (°C)	Monobit	Poker	Runs	Long Runs	Total
-30	0,03	0,01	0,05	0,02	0,11
25	0,14	0,03	0,10	0,04	0,30
70	1,01	0,03	0,16	0,03	1,22

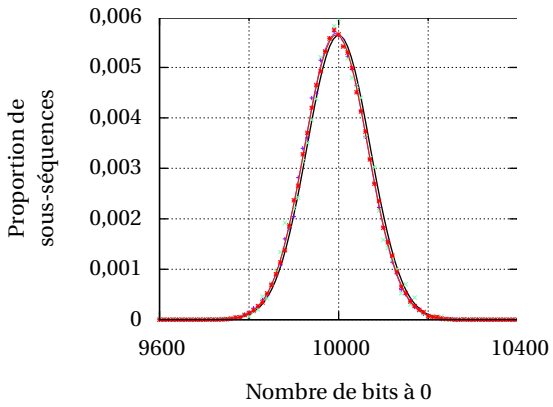
TABLE 7.1 – Pourcentage de sous-séquences échouant aux tests FIPS.



(a) TRNG sans retraitement.



(b) Sortie du correcteur de von Neumann.



(c) Sortie du LFSR.

FIGURE 7.3 – Distribution du nombre de bits à 0 dans les sous-séquences de 20 000 bits avec le TRNG perturbé en température.

plus de bits à 0 qu'à 1. Ce déséquilibre disparaît quand la température diminue.

Avec l'application du LFSR nous observons une distribution homogène des mots pour toutes les

températures. Le correcteur de von Neumann en revanche apparaît moins efficace pour les températures positives. En effet, à 25 °C il transforme le biais sur les bits en biais sur les mots : les mots de faibles et grands poids de Hamming sont plus importants que ceux avec des poids de Hamming intermédiaires. À 70 °C le correcteur produit l'effet contraire avec plus de poids de Hamming moyens.

Ces observations vérifient donc le fonctionnement des retraitements car le correcteur de von Neumann élimine uniquement le biais dans la distribution des bits alors que le LFSR a une action plus importante sur la suppression des défauts.

7.2.3 Effet de la température sur le fonctionnement du TRNG

Les temps de propagation dans des portes logiques sont dépendant de la température. Plus elle est importante et plus ils s'allongent. Sur des oscillateurs à anneaux, cela signifie que leur période est plus grande et leur fréquence diminue donc de manière inverse à la température.

Une conséquence de cet effet peut être la perte des «bonnes» caractéristiques du générateur avec l'apparition de fréquences parentes à celle du signal d'échantillonnage. Un autre effet peut être le couplage de deux oscillateurs entre eux avec l'élimination de leur différence de phase. Dans les deux cas nous obtenons une perte d'entropie en sortie du générateur.

7.3 Injection d'une fréquence variable

Le TRNG que nous étudions ici étant composé d'oscillateurs tournant en parallèle, il est un bon candidat pour les perturbations par injection de fréquence. Les deux expérimentations suivantes mettent en œuvre les bancs décrits dans la section 5.3. Ils nous permettent d'étudier la sortie du TRNG durant des injections à fréquence variable.

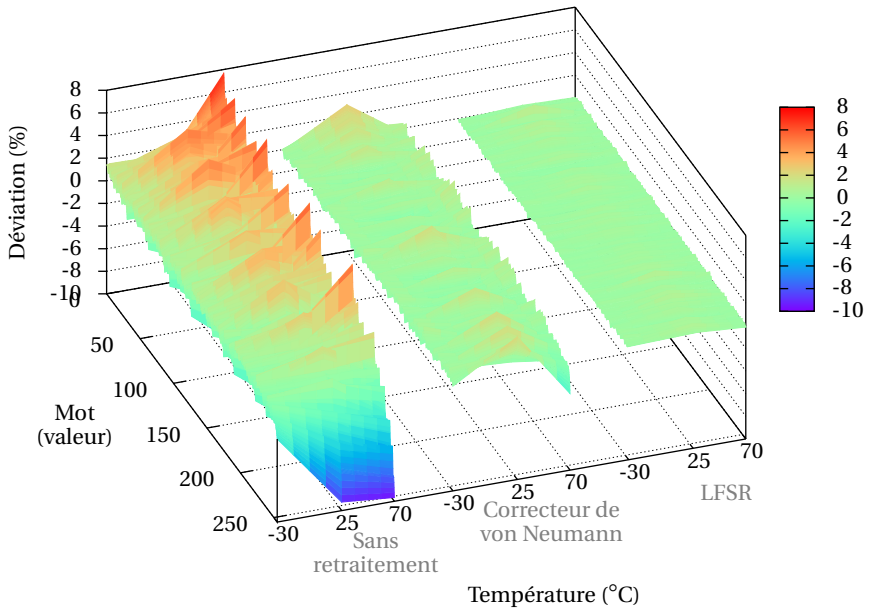
7.3.1 Injection sur la masse de la puce

L'injection directe de fréquence se fait sur la masse de la puce contenant le TRNG testé avec un signal sinusoïdal comme décrit dans la section 5.3.1. La puce est alimentée en 5 V, nous fixons l'amplitude du signal sinusoïdal à 1 Vpp ce qui permet à la puce de continuer à bien fonctionner tout en ayant un effet le plus important possible. Nous incrémentons la fréquence du signal par pas de 1 MHz. À chaque fréquence nous faisons des acquisitions de 10 Mo durant environ 8 heures à chaque fois.

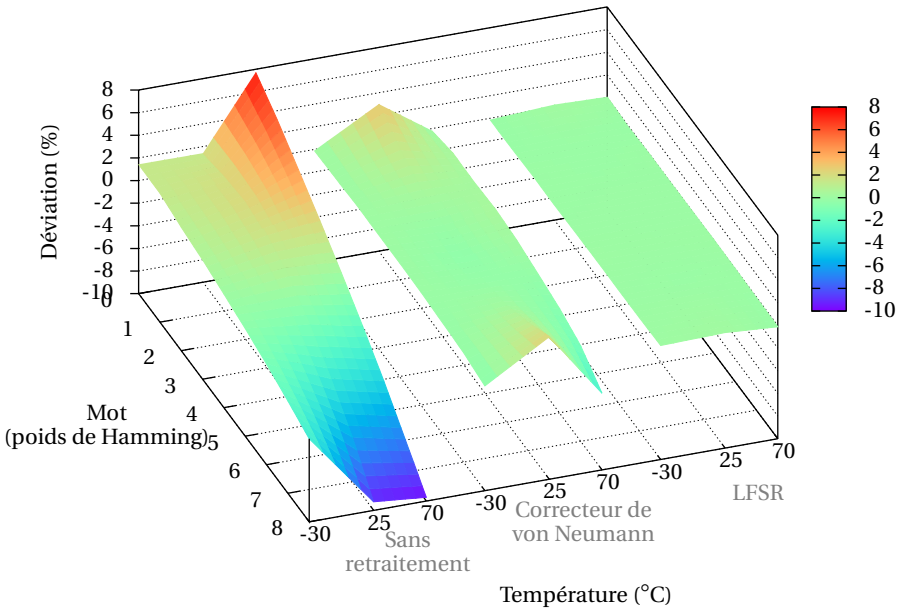
La figure 7.5 donne les résultats des tests statistiques pour chaque fréquence. L'observation des taux d'échec ne montre de variation importante pour aucune fréquence. Nous avons plus particulièrement étudié les tests se rapportant à la détection de répétitions de suites de bits dans les différentes batteries (tests calculant la transformée de Fourier, le rang de matrices, la corrélation, etc.). En effet, en cas de verrouillage des oscillateurs leurs variations de phase disparaissent et la sortie du TRNG devient périodique. Ceci ne nous a pas non plus permis de conclure sur une éventuelle influence de l'injection de fréquence car les tests se sont révélés réussis.

7.3.2 Signal électromagnétique

L'injection se fait ici via un champ électromagnétique comme décrit dans la section 5.3.2. Nous voulons faire des essais plus précis en fréquence en diminuant le pas à 0,1 MHz. Ceci implique que nos acquisitions sont de plus petite taille (100 ko) pour des temps d'acquisition plus faibles (d'environ 2 minutes). Précédemment nous étudions le résultat des tests statistiques pour conclure sur l'influence de l'injection de fréquence. Dans ce cas les fichiers contenant les séquences de bits sont trop



(a) Déviations dans la distribution des mots selon leur valeur.



(b) Déviations dans la distribution des mots selon leur poids de Hamming.

FIGURE 7.4 – Évolution des déviations dans la distribution des mots avec la température.

petits pour être traités par les tests ou pour que leurs résultats aient une signification. Nous calculons donc simplement les autocorrélations des séquences avec différents décalages.

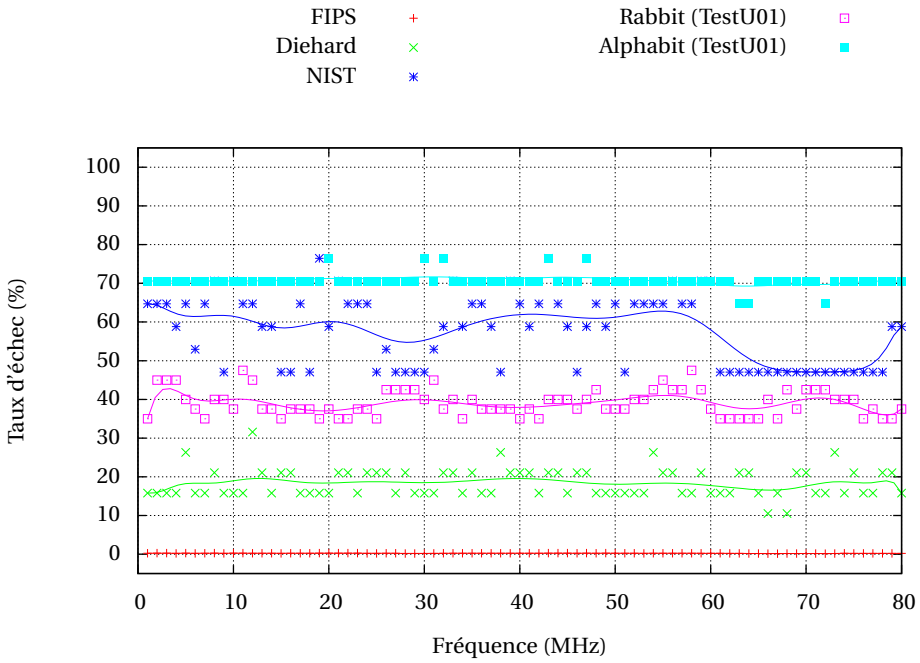


FIGURE 7.5 – Résultat des tests statistiques pour le TRNG perturbé en fréquence.

Nous utilisons la formule suivante de l'autocorrélation :

$$A(\tau) = \frac{\sum_{i=1}^{N-\tau} b_i b_{i+\tau}}{\sum_{i=1}^{N-\tau} b_i \sum_{i=1}^{N-\tau} b_{i+\tau}} \tag{7.1}$$

où τ est le décalage de la séquence par rapport à elle-même, b_i est la valeur du bit d'index i ($-0{,}5$ si le bit est à 0 et 0,5 s'il est à 1) et N est la taille de la séquence. La valeur de A est comprise entre -1 et 1. Une autocorrélation de 1 signifie qu'il n'y a pas de différence entre la séquence et sa version décalée d'où une périodicité des bits. Une autocorrélation de -1 signifie que la séquence étudiée et sa version décalée sont exactement l'inverse l'une de l'autre. Enfin une autocorrélation de 0 signifie qu'il n'y a aucune similarité dans la séquence. Pour accélérer le calcul du numérateur nous utilisons une transformée de Fourier rapide (FFT : Fast Fourier Transform) ainsi que son inverse (IFFT : Inverse Fast Fourier Transform).

Nous avons calculé l'autocorrélation des séquences à chaque fréquence injectée différente et pour des décalages τ allant de 0 à 10 000. Pour comparaison nous avons généré autant de séquences sans injection et aussi calculé leur autocorrélation. Ceci nous a donné des valeurs d'autocorrélations distribuées autour de 0 et ne dépassant pas $-0{,}006$ d'un côté et 0,006 de l'autre, que ce soit avec ou sans injection. Ces valeurs sont faibles et ne montrent donc pas de périodicité dans les séquences étudiées et en conséquence l'injection de fréquence n'a pas eu d'effet visible sur le fonctionnement du TRNG.

7.4 Conclusion

Dans ce chapitre nous avons étudié un TRNG de conception classique utilisant des oscillateurs en parallèle. Les oscillateurs étant sensibles à des fréquences externes à leur fonctionnement normal, nous avons tenté d'attaquer le TRNG par injection de fréquence. Les résultats n'ont pas montré d'influence sur le générateur pour aucun des chemins d'injection utilisés.

Des perturbations par variation de température ont elles mis en évidence des conséquence sur le TRNG avec un biais dans la distribution des bits augmentant avec la température. Cet effet va dans le sens inverse de ce que nous avons observé avec le premier TRNG étudié, il a de plus une amplitude plus faible.

Chapitre 8

Étude d'un TRNG implanté dans un processeur

Dans ce chapitre, nous étudions un TRNG implanté dans un processeur Via Nano. Pour cela nous utilisons une carte mère VIA EITX-3000 attachée à un radiateur à partir de laquelle nous pouvons nous connecter à distance par SSH. Le système d'exploitation que nous avons installé est Linux 2.6.32 et des APIs développées par VIA [VIA09] nous permettent d'accéder au TRNG.

L'intérêt d'avoir un générateur de nombres aléatoires dans un processeur est multiple, il est par exemple très utile pour l'exécution de simulations numériques, les jeux vidéos et bien sûr la cryptographie. En particulier, dans ce dernier cas il peut être exploité pour la génération de clefs dans OpenSSL.

8.1 Principe de fonctionnement

La conception de ce TRNG est basée sur l'échantillonnage d'un oscillateur de grande fréquence par un oscillateur de plus faible fréquence comme expliqué dans la section 2.2.2.1. La figure 8.1 montre le principe de fonctionnement de ce TRNG qui est décrit dans [Cry03]. Il comporte quatre oscillateurs : l'oscillateur 1 est échantillonné par une combinaison des trois autres de fréquence plus faible. Deux oscillateurs ont leur fréquence divisée par huit, ils sont XORés et le signal ainsi formé agit sur le dernier oscillateur dont la sortie sert d'horloge à une bascule D. Le signal de l'oscillateur 1 entre dans la bascule D et à sa sortie nous obtenons une séquence de bits.

Les oscillateurs sont des oscillateurs à anneaux tournant librement, leur fréquence est donnée

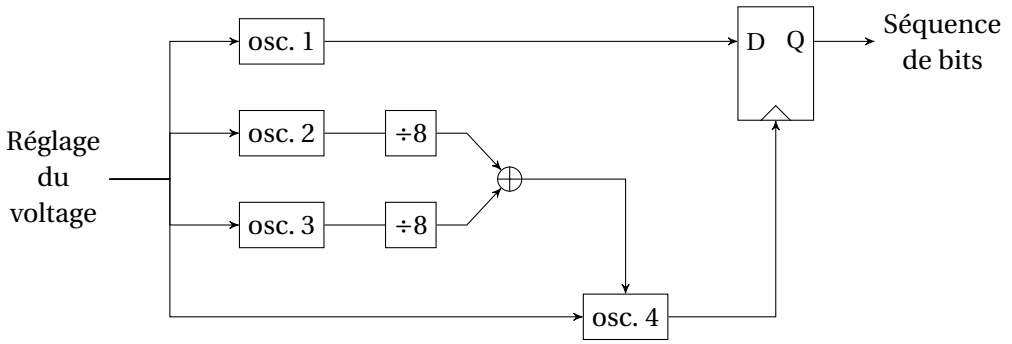


FIGURE 8.1 – Conception du TRNG du Via Nano constitué d'oscillateurs.

dans une gamme de 450-810 MHz à part l'oscillateur 4 qui a une fréquence plus faible aux alentours de 20-68 MHz.

La documentation de VIA [VIA08] indique que nous pouvons choisir deux sorties différentes du TRNG. Il est en effet possible soit d'accéder directement à la sortie de la source d'entropie soit d'appliquer un retraitement par AES. Dans ce dernier cas, la séquence brute est divisée en mots de seize octets où huit octets constituent la clef et les huit autres le message en clair. A la sortie de l'AES, les premiers huit octets générés sont gardés.

8.2 Émissions électromagnétiques

Avant d'appliquer des perturbations au TRNG nous avons essayé de déterminer si de l'information pouvait être récupérée via l'émission électromagnétique du processeur. En effet, en fonctionnement ce dernier consomme un courant dépendant des opérations qu'il effectue et se retrouvant sur le signal électromagnétique (EM) émis. En plaçant une sonde au dessus du processeur nous pouvons observer et enregistrer ce signal puis potentiellement faire de la SPA et/ou DPA.

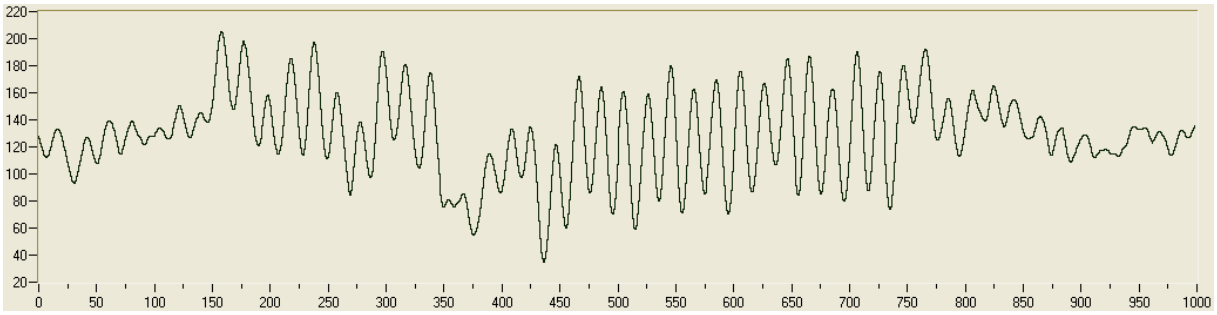
Le TRNG a à sa sortie quatre mémoires tampon de 8 octets. Si ces mémoires ne sont pas toutes remplies, le TRNG est allumé (oscillateurs en fonctionnement) jusqu'à temps qu'elles le soient et il est ensuite éteint. La génération d'aléa du côté de l'utilisateur se fait grâce à l'instruction «XSTORE»du processeur. Celle-ci transfère les données d'une mémoire (8 octets), et rien si elles sont toutes vides.

Nous avons tout d'abord observé l'émission électromagnétique du processeur lors d'un appel en boucle à XSTORE. Pour cela, nous avons déplacé une sonde reliée à un oscilloscope au dessus du processeur et trouvé une position où cette boucle était visible avec le motif de la figure 8.2a correspondant à l'appel à XSTORE. Sur l'axe des x nous avons l'index des échantillons et sur l'axe des y une amplitude proportionnelle au champ EM mesuré. L'oscilloscope est réglé avec une fréquence d'échantillonnage de 20 GS/s et sachant qu'il y a 20 points entre deux oscillations à l'intérieur du motif, nous vérifions donc que le processeur tourne à une fréquence de 1 GHz. Nous avons ensuite synchronisé l'appel à XSTORE avec le signal EM observé en générant des impulsions sur le port RS-232 de la carte mère. En étudiant le signal obtenu nous repérons le motif correspondant à l'appel à XSTORE (figure 8.2b et figure 8.2c).

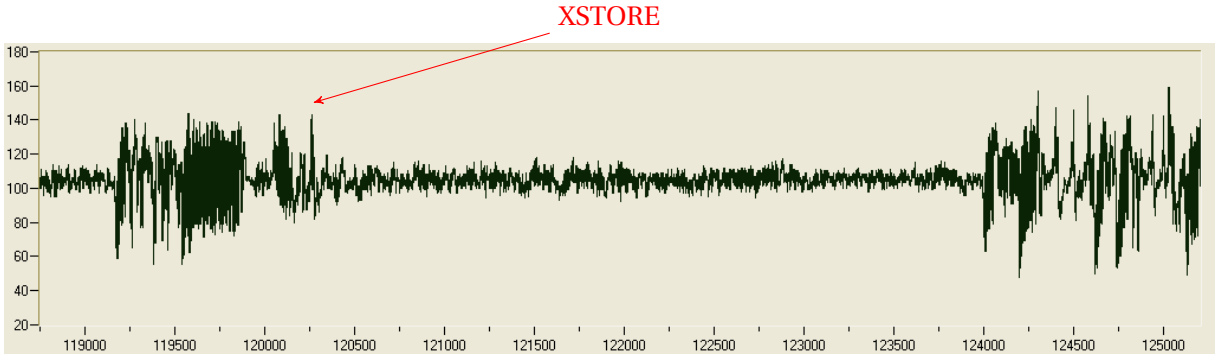
Nous avons ensuite tenté de faire ressortir des corrélations entre la connaissance des nombres aléatoires récupérés par l'appel à XSTORE et son motif correspondant. Pour cela nous faisons un moyennage des courbes contenant les motifs, pondérées par la valeur ainsi que le poids de Hamming des nombres aléatoires générés. En effet le champ EM du processeur est dépendant des données qu'il manipule et cette opération doit donc faire ressortir les instants durant lesquels les nombres aléatoires sont mis en jeu. Cependant, dans ce cas une telle analyse statistique des courbes n'a rien pu donner et implique que les signaux observés sont décorrelés des nombres générés. Ceci est expliqué par le fait que le TRNG ait un fonctionnement asynchrone avec les appels à XSTORE, ce qui ne nous permet pas d'analyser les signaux EM car nous ne savons pas quand il est allumé.

8.3 Variations de température

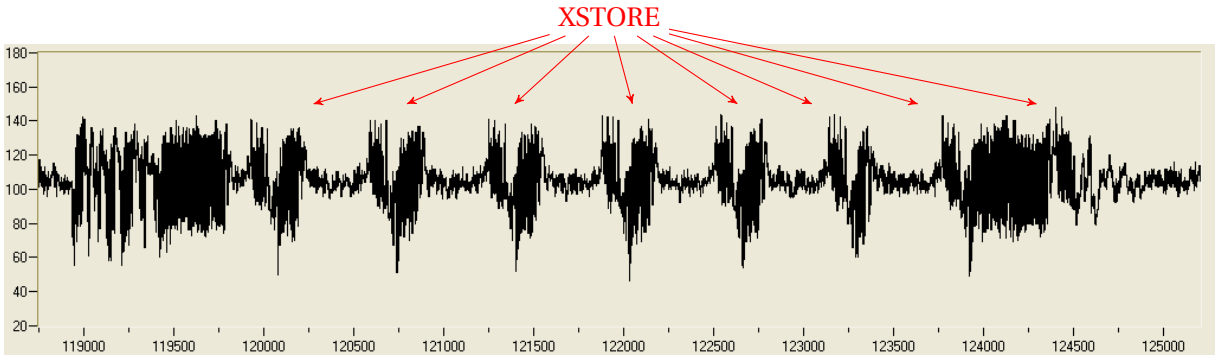
Comme pour les précédents TRNGs étudiés nous avons soumis ce générateur à des variations de températures. Le TRNG étant implanté dans un processeur attaché à une carte mère, il s'est posé des problèmes d'accessibilité pour l'utilisation des bancs de tests que nous avons mis en place. Nous avons donc dû modifier nos méthodes de test : pour le chauffage nous avons détaché la carte mère de son radiateur et utilisé à la place un ventilateur avec une distance variable. Pour le refroidissement, l'utilisation du module Peltier était impraticable et nous avons donc seulement pu plaquer



(a) Zoom sur le motif de l'appel à XSTORE.



(b) Appel à XSTORE une fois.



(c) Appel à XSTORE huit fois.

FIGURE 8.2 – Signaux EM lors d'appels à XSTORE.

le processeur à une tige en aluminium plongée dans de l'azote liquide. La température est mesurée directement par un capteur intégré dans le processeur.

Les températures de test obtenues sont 0 °C, 36 °C (carte mère collée à son radiateur), 80 °C, 90 °C et 100 °C. Des séquences de 307,2 Mo sont générées à chaque température en environ 3 minutes. Ces acquisitions sont réalisées sur la sortie directe de la source d'entropie.

8.3.1 Résultats des tests statistiques

La figure 8.3 montre les taux d'échec aux tests statistiques des séquences générées à chaque température.

La qualité des bits générés par le TRNG dans des conditions normales (à 36 °C) n'est déjà pas très bonne. En effet, tous les tests présentent des taux d'échec au moins supérieurs à 40%. Comme pour

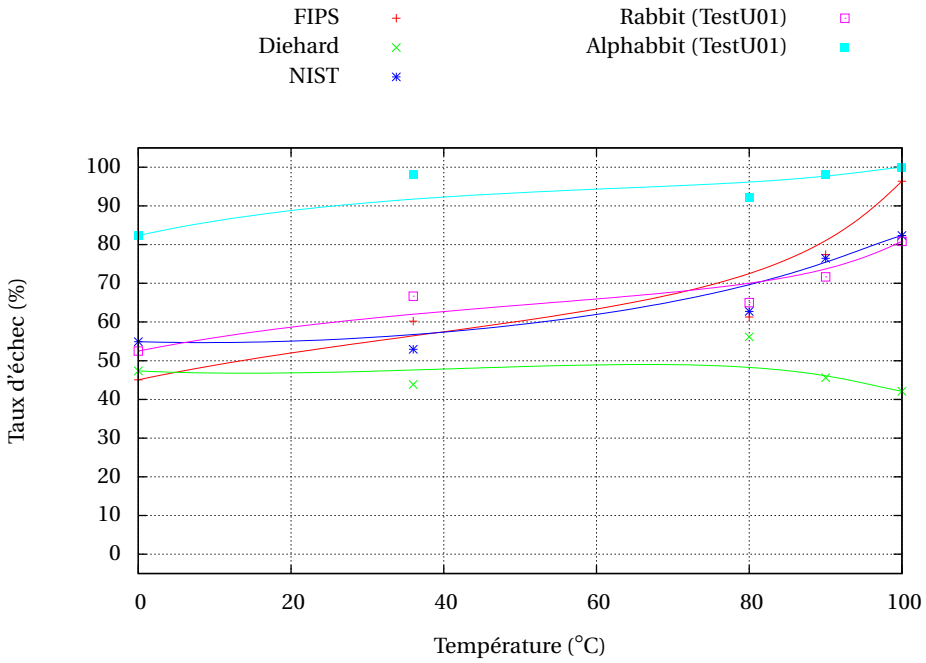


FIGURE 8.3 – Résultat des tests statistiques pour le TRNG perturbé en température.

les TRNGs étudiés précédemment il est donc nécessaire d'utiliser des retraitements afin d'éliminer les défauts présents dans les séquences de bits générées. La source de bruit sert donc à extraire de l'entropie d'un phénomène physique aléatoire, mais qui n'est pas utilisable directement.

L'application des variations de température entraîne d'importantes variations de la performance du TRNG, visibles sur tous les tests à part la batterie Diehard. Nous observons une aggravation des taux d'échec avec l'augmentation de la température et vice-versa.

8.3.2 Évolution du biais dans la distribution des mots

Le tableau 8.1 donne le détail des résultats des tests FIPS pour chaque température et sans retraitement. Contrairement aux TRNGs précédents, il n'y a pas de biais dans la distribution des bits. En revanche, le taux de sous-séquences échouant aux tests provient principalement des tests Poker et Runs. Le test Poker est un test de khi deux mesurant la ressemblance de la distribution des mots de 4 bits dans les sous-séquences étudiées à leur distribution théorique. Le fort taux d'échec que nous constatons indique donc qu'il existe un biais dans la distribution de ces mots, tout en gardant l'équilibre entre les bits à 0 et à 1. De même, les résultats du test Runs montre que les bits ne sont pas «bien» distribués et il apparaît des suites de bits à 0 et à 1 de façon disproportionnée.

La figure 8.4 représente la distribution des mots de 8 bits des séquences testées en considérant la valeur des mots ou leur poids de Hamming. Ces graphes confirment les résultats des tests FIPS : avec l'augmentation de la température nous voyons apparaître des déviations importantes des occurrences des mots par rapport à ce qui est attendu d'une séquence aléatoire. De plus, la distribution des mots selon leur poids de Hamming a un déséquilibre symétrique. En effet, les proportions de mots avec le plus de bits à 0 sont les mêmes que celles des mots avec le plus de bits à 1, cela signifie qu'il n'y a pas de biais dans le nombre de bits à 0 et à 1.

Température (°C)	Monobit	Poker	Runs	Long Runs	Total
0	0,01	33,60	41,97	0,04	45,05
36	0,01	41,95	46,81	0,03	60,21
80	0,02	55,96	55,56	0,07	61,31
90	0,01	72,77	71,75	0,03	77,42
100	0,06	95,34	95,33	0,25	96,38

TABLE 8.1 – Pourcentage de sous-séquences échouant aux tests FIPS.

8.4 Exposition à des rayonnements ionisants

8.4.1 Utilisation de sources radioactives

Cette expérience a consisté à approcher différentes sources radioactives du processeur et à générer des nombres aléatoires en même temps. Le but de cette manipulation est d'envoyer des rayons γ sur le processeur. Nous avons pour cela utilisé les sources et les bancs de tests décrits dans la section 5.4.1. L'application des sources radioactives devant se faire au plus près de la cible, nous utilisons la carte mère avec le processeur découvert donc sans radiateur. Ceci nécessite l'utilisation d'un ventilateur afin de maintenir le processeur en état de marche. L'efficacité du ventilateur étant inférieure à celle du radiateur, le processeur n'est pas complètement refroidi et fonctionne à une température d'environ 80 °C quand il génère des nombres aléatoires sans perturbation.

Nous distinguons les deux sources d'américium par Am^1 et Am^2 d'activités respectives 3,7 MBq et 518 MBq, le baryum est représenté par Ba et le césium par Cs. Pour chaque source, nous avons acquis plusieurs séquences de 102,4 Mo en 1 minute chacune. Nous avons de plus fait des acquisitions sans application de source radioactive afin de pouvoir effectuer des comparaisons.

La figure 8.5 montre les taux d'échec aux tests statistiques dans tous les cas de figure.

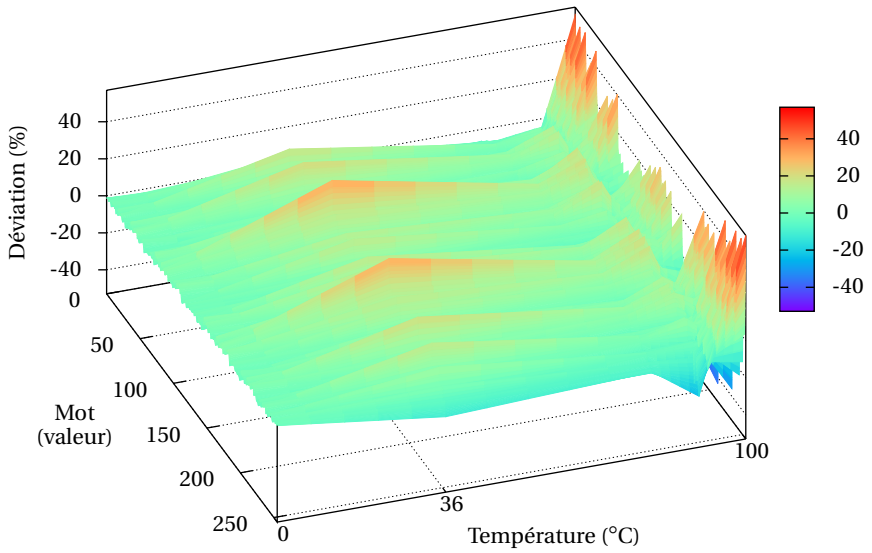
L'observation des résultats des tests statistiques ne nous permet pas de conclure qu'il existe une influence des source radioactives. En effet, en comparant la performance du TRNG pour les différentes sources nous de voyons pas de variation évidente des taux d'échec par rapport à son état normal ou à l'application d'autres sources.

8.4.2 Utilisation de rayons X

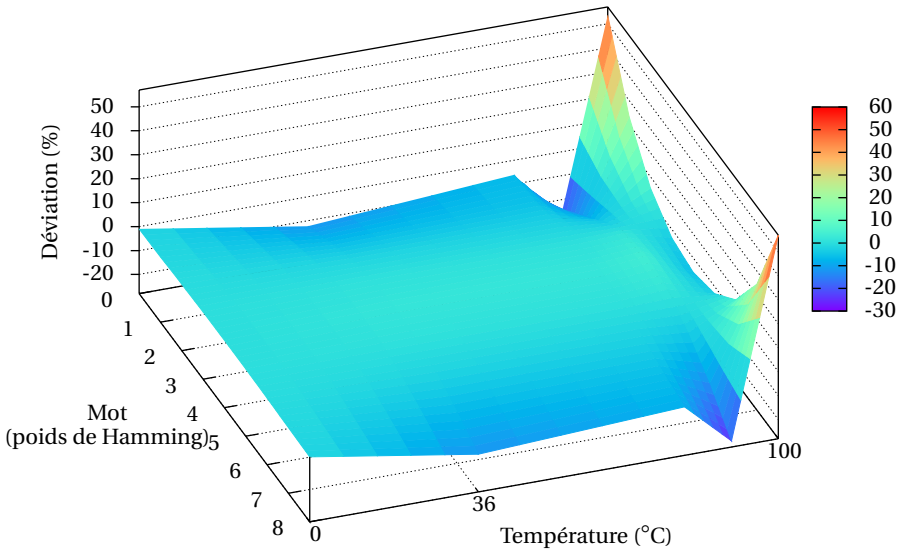
L'utilisation de sources radioactives n'ayant pas montré d'influence particulière de ces dernières sur le TRNG, nous avons ensuite utilisé des rayons X. Nous avons pour cela exploité le banc de test décrit dans la section 5.4.2 en variant le courant circulant dans le tube à rayons X, la tension appliquée (donc l'énergie du rayonnement) ainsi que la distance du tube au processeur. Les différentes configurations ainsi mises en place sont détaillées dans le tableau 8.2. L'état normal du TRNG (sans perturbation) est désigné par la configuration 0.

Configuration	Tension (kV)	Courant (mA)	Distance (cm)
1	140	21	65
2	70	40	65
3	70	40	25
4	160	18,75	25

TABLE 8.2 – Configurations du banc de test utilisant les rayons X.



(a) Déviations dans la distribution des mots selon leur valeur.



(b) Déviations dans la distribution des mots selon leur poids de Hamming.

FIGURE 8.4 – Évolution des déviations dans la distribution des mots avec la température.

De manière similaire à l'expérience précédente, ce banc de test nécessite un accès au processeur. Le radiateur de la carte mère est donc remplacé par un ventilateur et la température du processeur avoisine les 80 °C en fonctionnement normal.

La figure 8.6 montre les taux d'échec aux tests statistiques pour chacune des configurations. Les

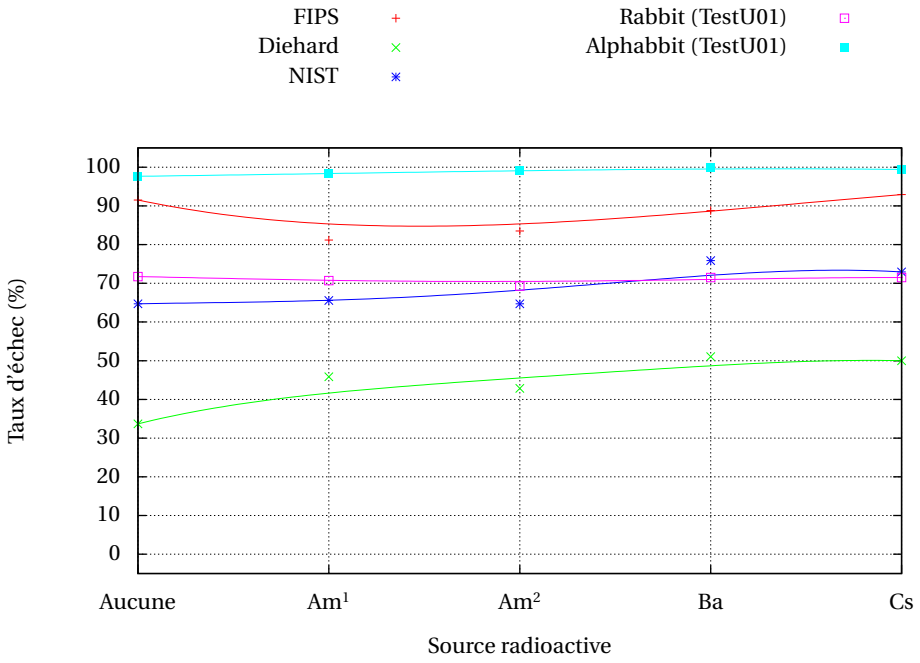


FIGURE 8.5 – Résultat des tests statistiques pour le TRNG perturbé par des rayons gamma.

résultats obtenus ne montrent pas de variations selon les configurations utilisées. En les comparant avec la configuration sans perturbation, nous n'observons pas non plus de différence flagrante de comportement. Ces résultat sont donc très similaires au cas précédent mettant en œuvre des sources radioactives.

Durant le bombardement des rayons X sur le processeur dans la configuration 4, nous avons cependant observé un fonctionnement anormal. En effet, après quelques minutes de l'application des rayons X, le processeur s'est éteint et essayait de se rallumer sans y parvenir. Après avoir arrêté le rayonnement et en attendant quelques minutes nous avons réussi à le démarrer normalement. En essayant de recommencer l'expérience nous avons obtenu la même anomalie. Nous en concluons que les rayons X ont bien un impact sur le processeur qui leur est complètement exposé. En revanche, le TRNG ne semble pas perturbé ou pas assez pour montrer des déviations dans les résultats des tests statistiques.

8.5 Attaque laser

Dans cette expérience, nous mettons en œuvre le banc de test décrit dans la section 5.2 et usuellement utilisé pour des attaques d'algorithmes cryptographiques ou des modifications de registres par exemple. Nous voulons ici déterminer si le laser peut avoir une influence sur le TRNG et en déduire sa propre localisation ou celle de signaux qui lui sont liés. Nous avons besoin pour cela de l'accès au processeur et plus précisément de positionner le laser au dessus de sa surface afin de pouvoir se focaliser sur tous ses points. Nous retirons donc le radiateur le refroidissant et nous le remplaçons par un ventilateur. Afin d'améliorer l'impact du faisceau laser il est nécessaire de préparer le processeur en l'amincissant. Nous avons pour cela retiré une couche de 570µm de silicium à partir des 900µm initialement présents. Puis nous avons poli la surface afin d'éviter des déviations intempestives du faisceau quand il pénètre le matériau. Cette phase a réduit le processeur de 80µm de silicium en

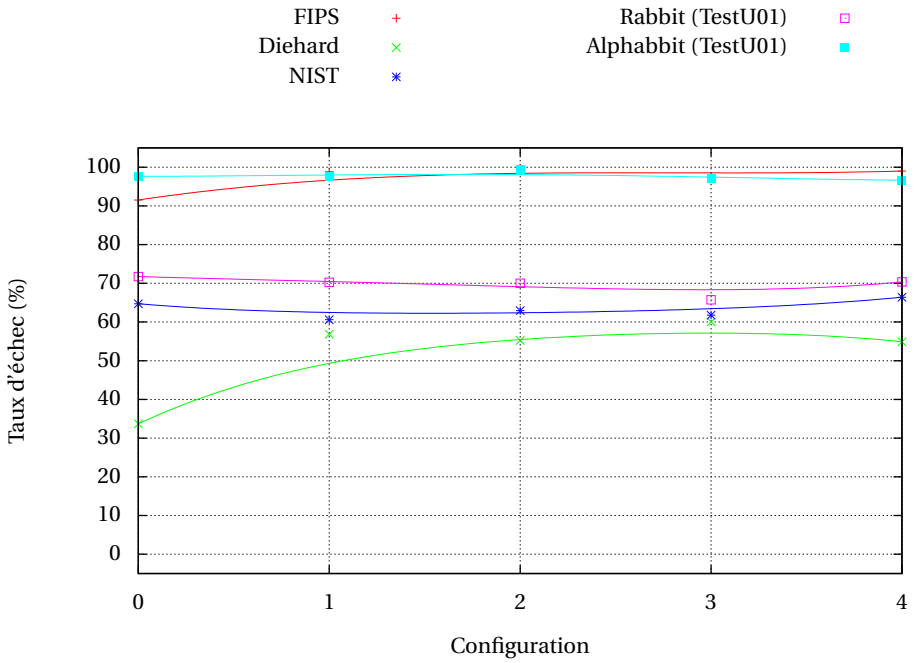


FIGURE 8.6 – Résultat des tests statistiques pour le TRNG perturbé par des rayons X.

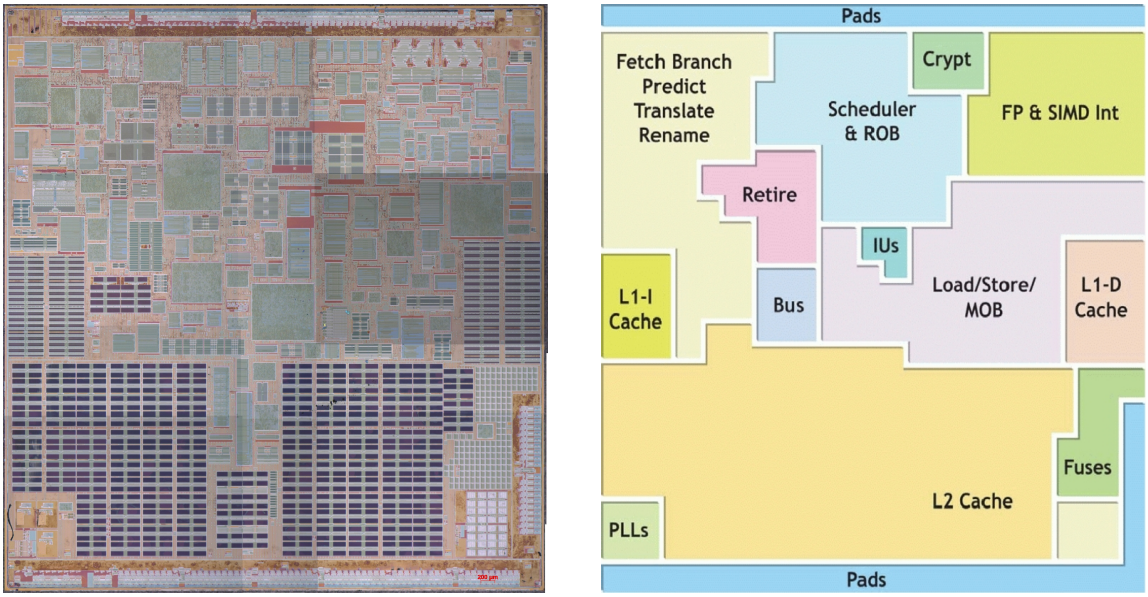
plus, au final il reste une épaisseur d'environ 150 µm pour une largeur de 7,65 mm et une longueur de 8,275 mm.

Comme nous ignorons où se trouve le TRNG dans le processeur, nous décidons de le balayer entièrement par des impulsions laser. Ceci nous permet d'obtenir la réponse du processeur à une telle perturbation et d'identifier d'éventuelles positions montrant une sensibilité à l'attaque. À chaque position et durant l'impulsion laser nous faisons une acquisition de 4 Mo de la sortie brute du TRNG en environ 2 secondes. La taille des acquisitions est limitée par celle du balayage ainsi que la tolérance du processeur au laser. En effet, celui-ci a tendance à cesser de fonctionner au bout de quelques secondes d'une impulsion laser.

Pour chaque acquisition nous étudions la réponse du processeur via les tests statistiques sur les acquisitions. Les tests usuels ne sont pas applicables dans ce cas car les acquisitions sont trop petites pour donner des résultats significatifs. Nous utilisons donc le programme ENT qui donne plutôt des caractéristiques générales sur les nombres aléatoires enregistrés. Le programme calcule ainsi six propriétés et pour chacune d'elles nous construisons une cartographie du processeur.

La figure 8.7 montre une photographie du processeur effectuée via un microscope. Elle est en fait composée de plusieurs photographies combinées ensemble. La figure donne aussi un schéma correspondant des différentes fonctionnalités du processeur.

La figure 8.8 illustre les résultats obtenus par le balayage du processeur avec des impulsions laser. La propriété représentée est l'entropie de la source de nombres aléatoires. Elle est calculée grâce aux occurrences des octets dans les séquences testées et devrait donc en théorie être égale à 8 bits pour des nombres vraiment aléatoires. Une valeur inférieure signifie que des octets ont des proportions d'apparitions qui dévient de leurs occurrences idéales. La figure 8.8a présente une cartographie du processeur en entier. Elle a été effectuée en réglant le pas du balayage à 50 µm et avec une puissance du laser de 1,05 W. Les points en noir signifient que le processeur a eu un dysfonctionnement durant l'impulsion envoyée sur ces localisations. Dans ces cas de figure, un redémarrage du processeur était nécessaire. De plus, la cartographie a été faite en plusieurs fois d'où des limites visibles sur l'image



(a) Photographie du processeur au microscope.

(b) Représentation schématique des principales composantes du processeur.

FIGURE 8.7 – Processeur Via Nano.

finale.

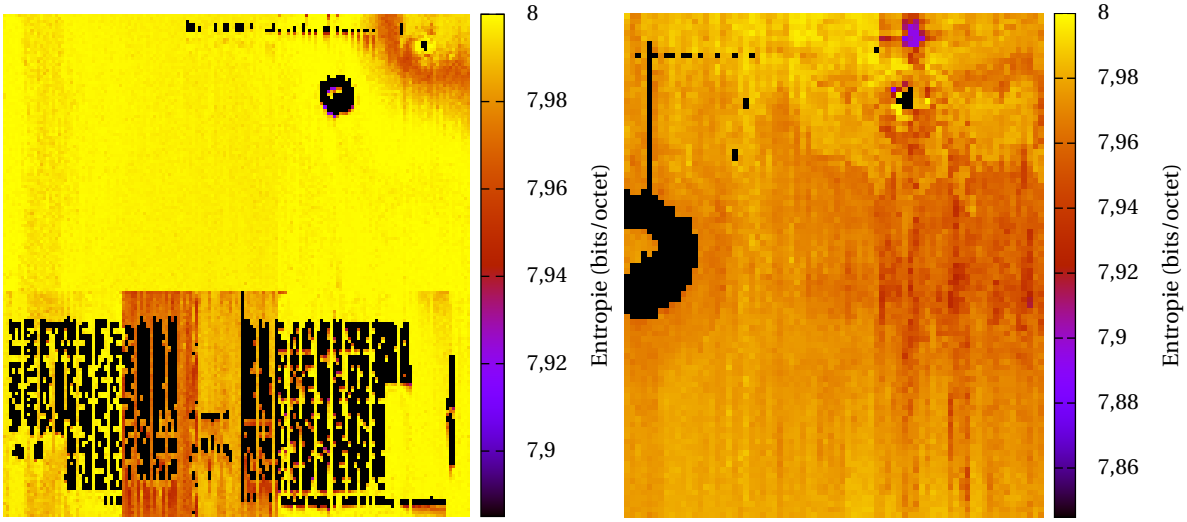
L'observation importante que nous pouvons faire sur cette cartographie est la présence d'un arc de cercle dans le coin supérieur droit. À cet endroit, l'entropie diminue pour atteindre environ 7.96 bits. Ce phénomène a été vérifié sur plusieurs autres balayages.

La figure 8.8b montre ce qu'il advient de cette perturbation en augmentant la puissance du laser à 3,75 W et en diminuant le pas à 30 µm. L'arc de cercle semble s'être diffusé dans son voisinage et il apparaît un petit ensemble de points au bord du processeur pour lequel l'entropie a encore diminué jusqu'à environ 7,9 bits. D'après la figure 8.7b ces points sont situés dans une zone appelée «pads» qui contient les plots d'interconnexion entre le processeur et le substrat sur lequel il est posé. Il regroupe donc les bus des signaux entrants et sortants du processeur.

8.6 Combinaison des effets de la température et du laser

Les expériences précédentes ont permis de dégager l'influence de la température et du laser sur le fonctionnement du TRNG étudié. Nous voulons ici étudier plus précisément ces effets et les combiner afin d'essayer d'amplifier les perturbations obtenues séparément.

Pour profiter de l'effet du laser au maximum nous plaçons le processeur de telle manière que le faisceau atteigne l'endroit précédemment identifié où la perte d'entropie est la plus importante. Nous pouvons varier la température du processeur en commandant la vitesse de rotation du ventilateur. Nous la fixons à environ 90 °C avant le commencement de chaque acquisition. La méthode d'acquisition est la suivante : comme le temps d'application des impulsions laser est limité pour éviter tout dégât au processeur, nous effectuons 1000 acquisitions de 4 Mo (en 2 secondes) séparées par plusieurs secondes de repos afin que le processeur puisse refroidir entre temps. À titre de comparaison nous faisons des acquisitions similaires sans le laser mais avec le processeur chaud ainsi que sans aucune perturbation et avec la carte mère attachée à son radiateur.



(a) Cartographie du processeur entier.

(b) Cartographie d'un coin du processeur.

FIGURE 8.8 – Cartographie du processeur grâce au calcul de l'entropie du TRNG sur des mots de 8 bits.

8.6.1 Résultats du programme ENT

Nous avons appliqué le programme ENT à chacune des acquisitions. Ceci nous permet d'avoir une vision globale des caractéristiques des séquences de nombres aléatoires générées dans les différentes configurations de test. La figure 8.9 présente des statistiques qui ont ainsi été calculées en considérant les acquisitions comme des suites d'octets, il s'agit de l'entropie et de la moyenne.

Notre première observation concerne les acquisitions sans application de perturbation. Dans ce cas nous obtenons des statistiques qui se rapprochent le plus de leur valeur théorique. En effet la moyenne est centrée autour de 127,5 mais l'entropie atteint 7,985 bits. De plus, les statistiques semblent avoir besoin d'un certain temps avant de se stabiliser. Ceci est dû à la source de bruit physique qui a ici un comportement assez variable.

En laissant le processeur chauffer, nous voyons une dégradation des performances avec une baisse de l'entropie donc une augmentation du déséquilibre entre les octets de différentes valeurs. La moyenne semble, elle, rester centrée sur sa valeur théorique. Les différences d'occurrences des octets conservent donc l'équilibre entre le nombre de bits à 0 et à 1.

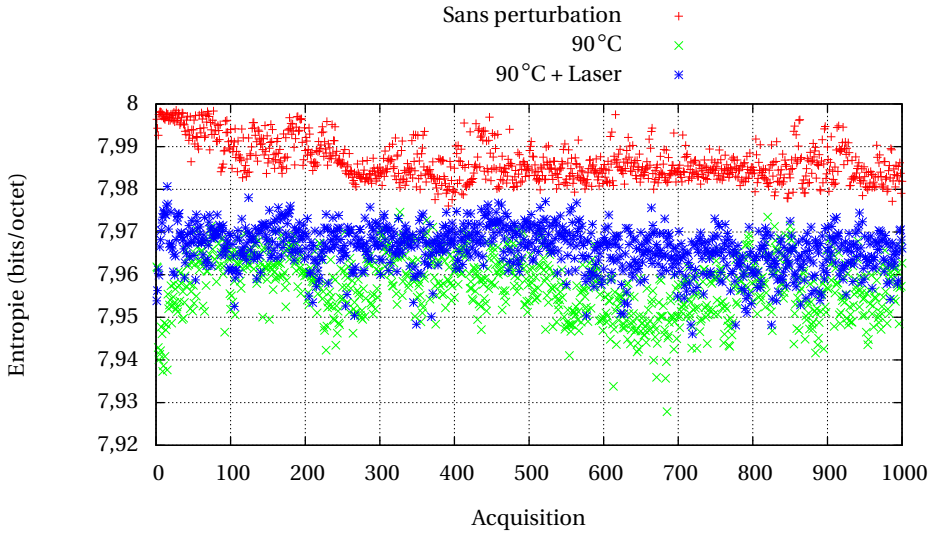
En ajoutant des impulsions laser, cet équilibre est brisé au profit de l'apparition de plus de bits à 1. Nous pourrions alors nous attendre à une baisse encore plus prononcée de l'entropie mais ce n'est pas le cas, elle augmente au contraire légèrement.

En conséquence, au lieu de l'effet recherché d'une amplification des variations produites par la température nous obtenons plutôt deux phénomènes indépendants.

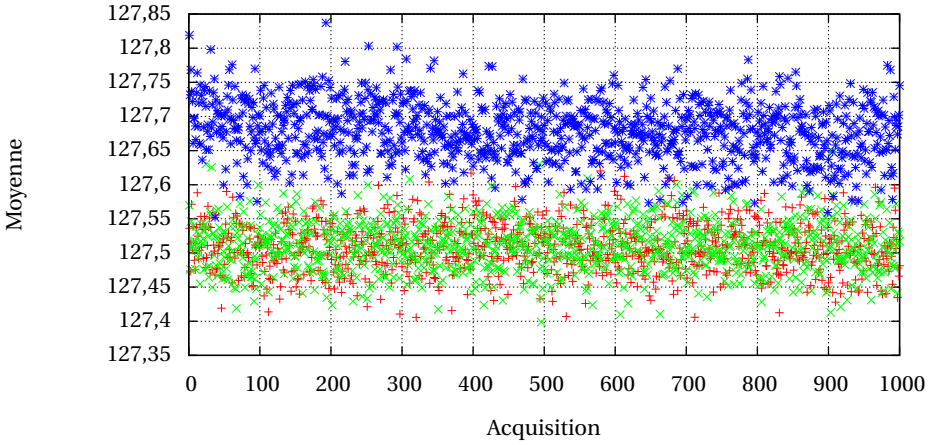
8.6.2 Biais dans la distribution des bits

La figure 8.10 nous aide à comprendre comment le déséquilibre dans la distribution des bits s'effectue. Les acquisitions sont considérées comme des suites de 128 bits. Pour chacune des positions dans les suites, nous comptabilisons les probabilités d'apparition des bits à 1.

La figure montre logiquement que dans les cas où nous n'appliquons pas de perturbation et avec seulement la température, chaque bit des suites de 128 bits a la même chance d'être à 0 ou à 1. En revanche, en ajoutant le laser un déséquilibre apparaît avec une périodicité de 8 bits. Cela signifie



(a) Calcul de l'entropie.



(b) Calcul de la moyenne.

FIGURE 8.9 – Résultats de tests ENT sur les acquisitions constituées de suites d'octets.

que les bits d'index $i+8k$ pour $0 \leqslant i \leqslant 7$ et $0 \leqslant k \leqslant 15$ ont les mêmes probabilités d'apparition si i est constant. De plus, le déséquilibre est clairement en faveur d'un biais total avec plus de bits à 1 et ceci à cause des bits d'index $7+8k$ qui ont une déviation par rapport à leur probabilité théorique d'environ 3%.

8.6.3 Évolution temporelle des déviations

Nous voulons maintenant étudier plus précisément l'impact des perturbations sur le TRNG et leur progression dans le temps. L'intervalle de temps que nous considérons s'étend entre le début et la fin des perturbations. Avec notre méthode d'acquisition des nombres aléatoires cela correspond à une impulsion laser (de durée d'environ 2 s). Pour prendre en compte toutes nos acquisitions nous les moyennons donc. De plus, l'observation dans le temps se fait en divisant chaque acquisition par un

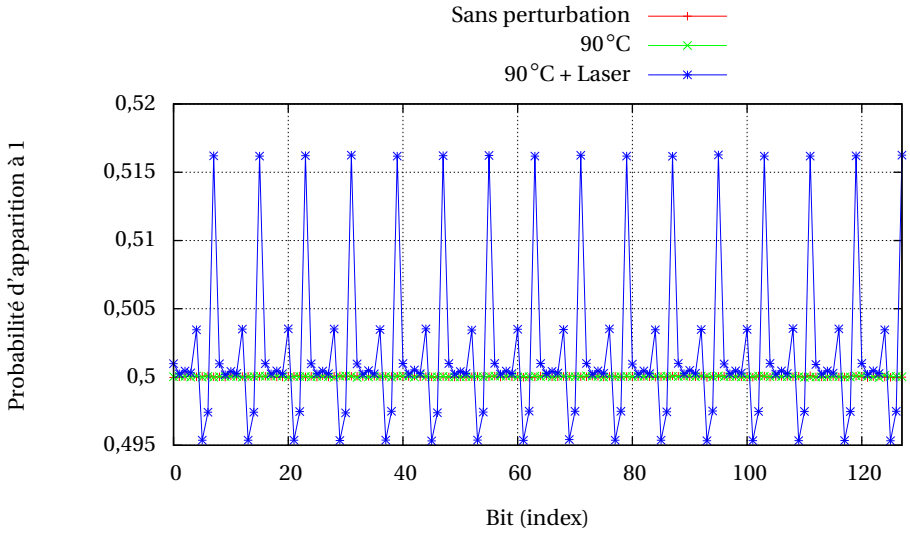


FIGURE 8.10 – Occurrences des bits à 1 pour des suites de 128 bits.

certain nombre de tronçons et en observant les caractéristiques des nombres aléatoires par tronçon.

La figure 8.11 montre l'évolution du biais dans les trois cas que nous étudions ici : sans perturbation, à 90 °C et à 90 °C plus le laser. Nous pouvons ainsi comparer leur progression.

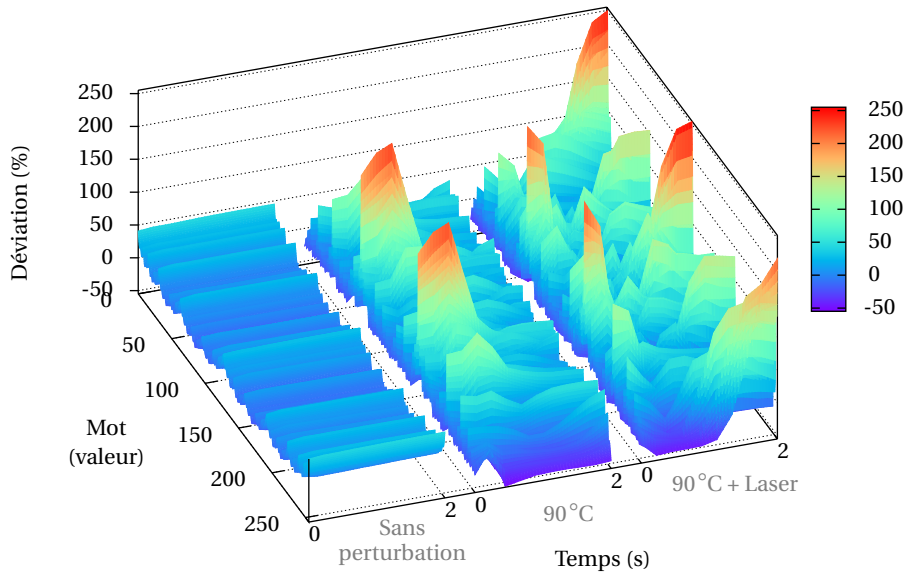
Sans perturbation, les octets ne sont pas tous équiprobables d'où la présence d'un motif dans la distribution. Au cours du temps ce motif ne se modifie pas, nous avons donc un système stable.

En chauffant le processeur nous voyons apparaître des déviations plus importantes au commencement de la génération des nombres aléatoires. Ensuite une distribution plus équilibrée se met en place. En étudiant la répartition des mots selon leur poids de Hamming nous faisons une observation inverse : les déviations semblent initialement plus près de zéro puis s'aggravent. Dans la deuxième partie des acquisitions les déviations rejoignent des amplitudes comparables à celles observées sans perturbation.

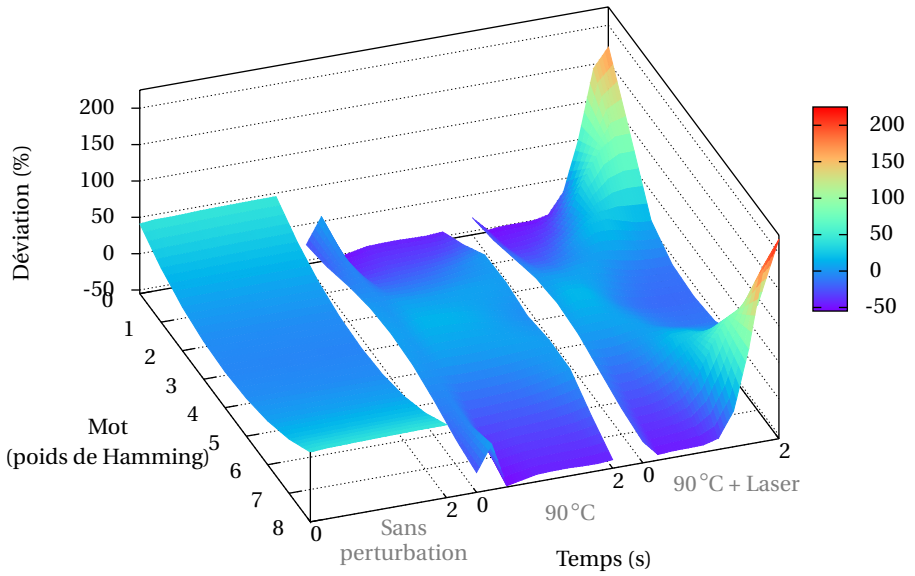
En ajoutant les impulsions laser, nous distinguons trois fonctionnements différents. Nous retrouvons ainsi une phase identique au cas précédent quand le processeur est seulement chauffé avec des déviations similaires dans la distribution des octets selon leurs valeurs et poids de Hamming. Ensuite, le TRNG semble retrouver son mode de fonctionnement normal avec la diminution des déviations. Enfin, il apparaît une nouvelle distribution des octets ayant d'importantes déviations par rapport à la théorie et qui serait due au laser car elle n'est pas observable quand nous appliquons seulement le chauffage du processeur. Contrairement aux déviations induites au début des acquisitions, cette dernière distribution est déséquilibrée à la fois sur les valeurs des octets ainsi que sur leur poids de Hamming. Elle montre en effet des biais en faveur de l'augmentation de l'apparition des octets à très faibles et très forts poids de Hamming.

8.6.4 Effets de divers retraitements sur les déviations

Cette dernière expérience nous a permis de générer des déviations dans la production des bits. Ces écarts par rapport à la distribution attendue se manifestent de différentes façons. Il n'est donc pas évident qu'un retraitement puisse corriger tous les défauts présents dans les séquences. Il est même possible qu'en traitant ainsi les séquences produites, des défauts soient mis en évidence. Nous



(a) Déviations dans la distribution des mots selon leur valeur.



(b) Déviations dans la distribution des mots selon leur poids de Hamming.

FIGURE 8.11 – Progression temporelle des déviations dans la distribution des mots.

appliquons donc plusieurs retraitement et observons leur performance.

8.6.4.1 Ou exclusif (XOR)

L'opération de ou exclusif prend des paires de bits et rend 0 s'il sont identiques et 1 dans le cas contraire. Il a pour effet d'éliminer grossièrement un biais dans la distribution des bits comme expliqué dans la section 2.4.1. La figure 8.12 montre l'effet du XOR appliqué aux trois cas étudiés précédemment (pas de perturbation, chauffage à 90 °C puis en ajoutant le laser).

Sans perturbation et sans retraitement, il apparait plus d'octets avec de faibles et forts poids de Hamming. En traitant les séquence avec un XOR, cette déviation se transfère sur les poids de Hamming faibles. Cela implique que dans les séquences d'origine, les bits ont tendance à se suivre avec des valeurs identiques, il y aurait donc une certaine corrélation.

Sur les séquences avec perturbation, nous voyons le biais au début des acquisitions qui est mis en évidence par le retraitement tandis qu'à la fin il a tendance a être éliminé. En effet, l'effet initial se transforme et devient visible sur la distribution des poids de Hamming avec plus de poids de Hamming importants par rapport à la normale et moins de poids de Hamming faibles. Il y aurait donc plutôt des oscillations dans les séquences produites, contrairement au cas sans perturbation.

En revanche, le biais sur la fin des acquisitions est transformé de manière similaire à ce qui est obtenu sans perturbation ce qui montre un certain retour à la normale. Cependant, dans le cas où nous utilisons à la fois la température et le laser il y a des déviations fortes dans les parties finales des séquences. Ceci montre que ces déviations sont similaires à celles présentes sans perturbation mais amplifiées.

8.6.4.2 Correcteur de von Neumann

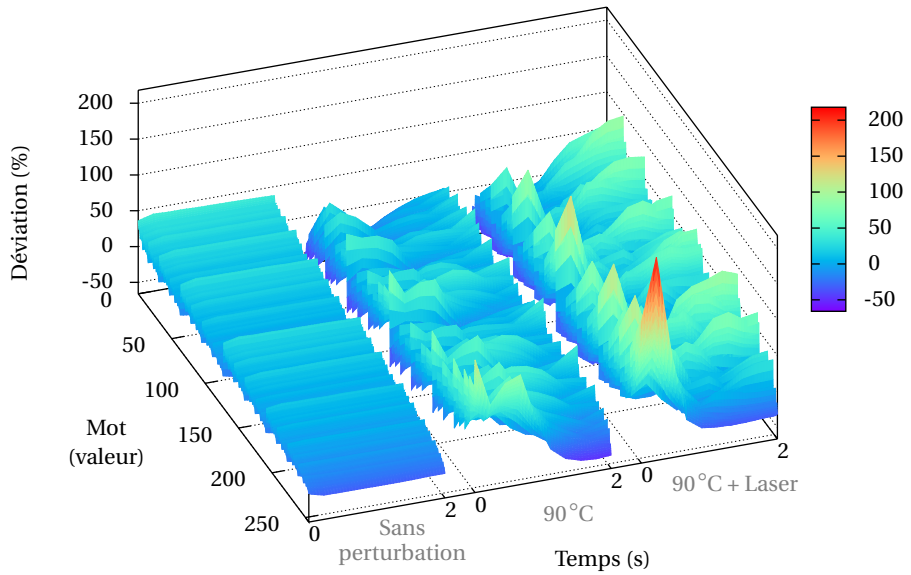
Le correcteur de von Neumann traite des paires de bits de manière similaire au XOR, mais il ne génère rien si les bits sont similaires et sinon un 0 pour la paire 01 et un 1 pour 10. Il élimine complètement un biais dans la distribution des bits comme expliqué dans la section 2.4.1. La figure 8.13 montre l'effet du correcteur sur les déviations obtenues.

Nous pouvons tout d'abord faire une observation générale sur les distributions obtenues quant aux séquences d'origine. En effet, le correcteur de von Neumann éliminant totalement le biais dans la distribution des bits, alors ce défaut n'est clairement pas le seul problème généré par les perturbations au vu des déviations obtenues après retraitement. Le retraitement met donc en évidence les corrélations entre bits à la sortie du TRNG.

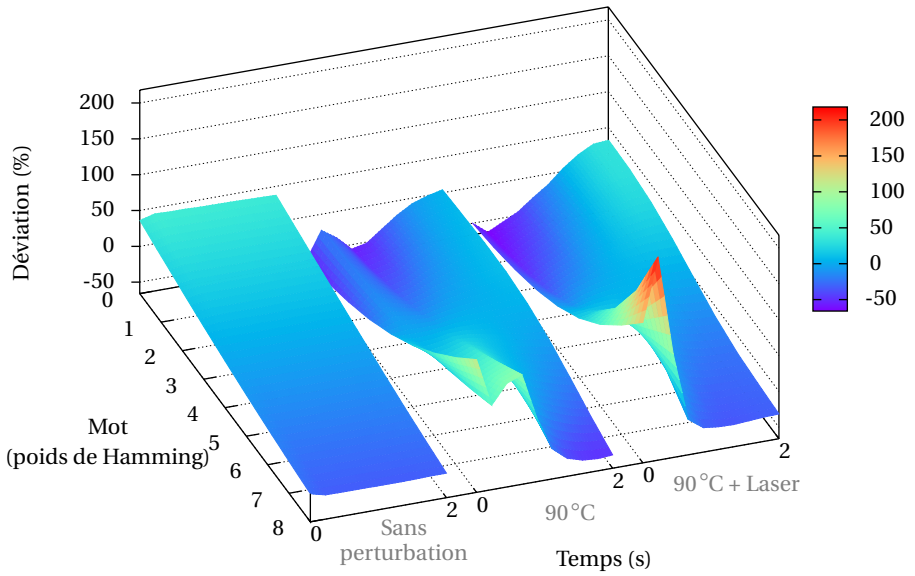
Sur le début des séquences perturbées nous retrouvons la déviation initiale. Celle-ci étant due à des oscillations de la valeur des bits consécutifs, elle se retrouve ici dans la prépondérance des octets de poids de Hamming faibles et forts.

L'observation des déviations sur la partie finale des acquisitions est plus difficile à expliquer. En effet, nous avons déduit du retraitement effectué grâce à l'opération XOR que les bits ont plutôt tendance à se succéder avec des valeurs similaires. Cependant, avec le correcteur de von Neumann nous observons une déviation en faveur des poids de Hamming faibles et forts bien que plus faible qu'en début d'acquisition. Le résultat obtenu caractériserait donc plutôt une périodicité des successions de suites de bits identiques dans la séquence.

Enfin, ce retraitement met en évidence un phénomène apparaissant environ à la moitié des acquisitions en appliquant seulement le chauffage du processeur. La déviation observée en sortie du correcteur est identique à celles détaillées précédemment alors qu'avant retraitement rien de particulier ne semble ressortir.



(a) Déviations dans la distribution des mots selon leur valeur.

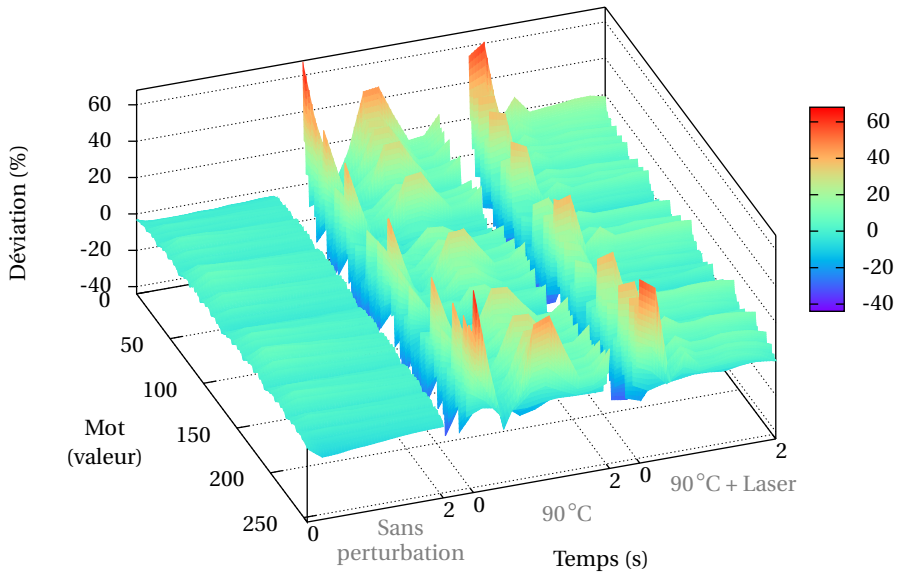


(b) Déviations dans la distribution des mots selon leur poids de Hamming.

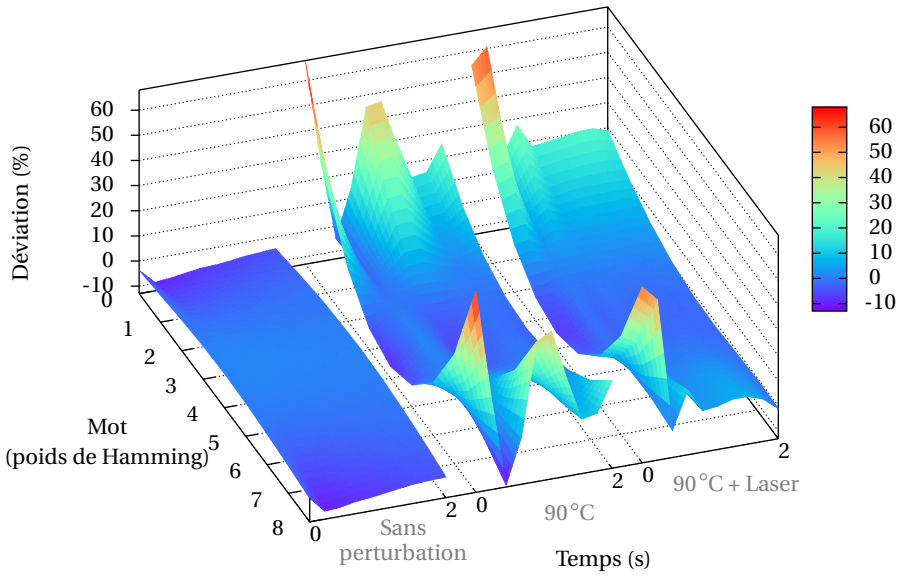
FIGURE 8.12 – Effet de l'opération XOR sur les déviations dans la distribution des mots.

8.6.4.3 SHA-256

Les fonctions de hachage construisent le condensé (hash) d'un message. Pour cela elles mélangent les bits qu'elles reçoivent en entrée de telle façon à éliminer toute corrélation entre eux. Nous utilisons ici l'algorithme SHA-256 (Secure Hash Algorithm) tel que décrit dans la publication FIPS



(a) Déviations dans la distribution des mots selon leur valeur.

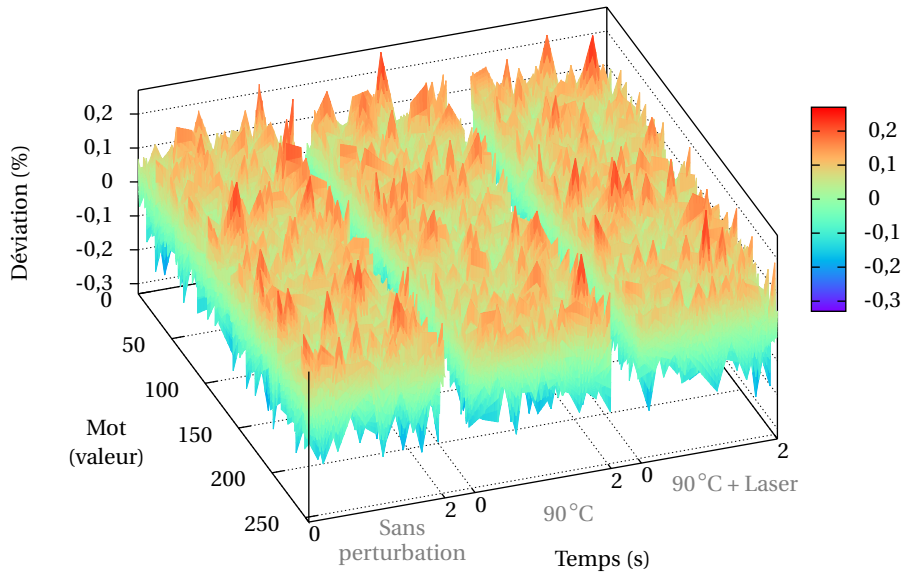


(b) Déviations dans la distribution des mots selon leur poids de Hamming.

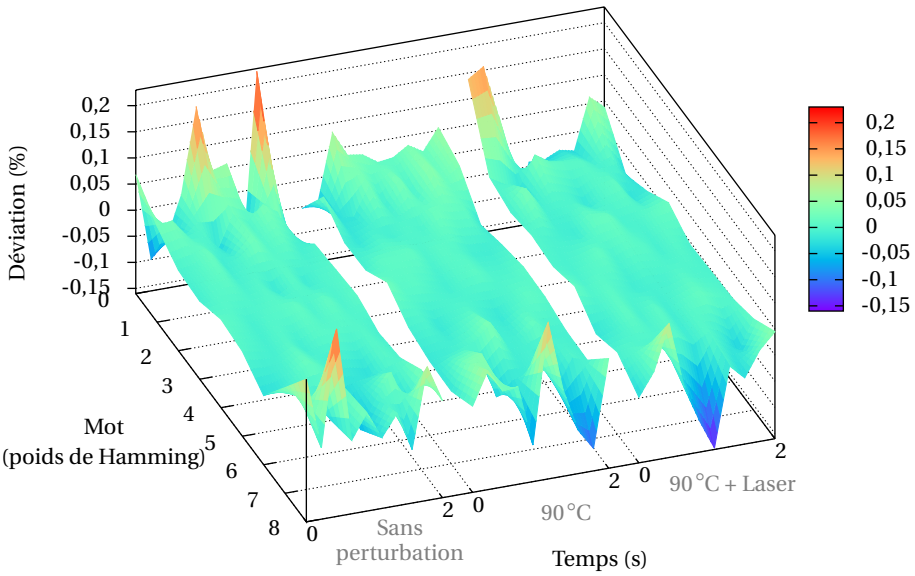
FIGURE 8.13 – Effet du correcteur de von Neumann sur les déviations dans la distribution des mots.

180-2 [FIP02]. Les données sont traitées par paquets de 32 octets. La figure 8.14 montre la distribution des octets après retraitement par SHA-256.

L'effet de la fonction de hachage est évident sur les graphiques : premièrement l'échelle des déviations est petite par rapport à ce que nous avons obtenu précédemment, ensuite elle est la même



(a) Déviations dans la distribution des mots selon leur valeur.



(b) Déviations dans la distribution des mots selon leur poids de Hamming.

FIGURE 8.14 – Déviations dans la distribution des octets après hachage des séquences.

avec ou sans perturbation, enfin il ne ressort pas de pic particulier et ceci dans tous les cas de figure.

Les représentations par poids de Hamming semblent avoir des variations qui pourraient être significatives (pics) pour les poids 0 et 8 mais des résultats similaires sont obtenus pour des séquences de bits vraiment aléatoires.

8.6.4.4 AES-128

L'Advanced Encryption Standard est un algorithme pour le chiffrement et déchiffrement de messages. Le retraitement consiste à chiffrer les séquences de bits aléatoires tel que décrit dans la publication FIPS 197 [FIP01]. Les données sont traitées par paquets de 128 bits (AES-128) et en mode Electronic Code Book (ECB). Dans ce mode, les paquets sont chiffrés de manière indépendante et nous avons fixé la clef à 0. La figure 8.15 montre la distribution des octets après retraitement par AES-128.

Les résultats obtenus sont similaires aux précédents avec SHA-256. Nous avons en effet une distribution uniforme des octets et nous ne distinguons pas les cas avec et sans perturbation.

8.6.5 Impact du maximum de déviation sur la génération de nombres premiers

Nous étudions ici une conséquence possible des déviations obtenues par notre attaque en température combinée avec le laser. En effet, les nombres générés par ce TRNG peuvent être directement exploités par une personne possédant un processeur Via Nano. En particulier, en installant une distribution Linux il est possible qu'une telle personne utilise OpenSSL qui contient une librairie de cryptographie. De plus, OpenSSL offre la possibilité de choisir notre propre source d'aléa et propose l'accès à des générateurs de nombres aléatoires déjà implantés dans le matériel utilisé.

Par ailleurs, OpenSSL offre la possibilité d'utiliser divers algorithmes cryptographiques. La génération de nombres premiers étant couramment utilisée dans ce contexte, nous nous penchons donc sur les conséquences possible de notre attaque sur le résultat de cette opération. Notre version d'OpenSSL est la 0.9.8g.

L'algorithme de génération de nombres premiers fonctionne en tirant tout d'abord un nombre aléatoire puis en cherchant le nombre premier le plus près. La différence entre ces deux nombres se trouve donc dans les bits de poids les plus faibles. Nous faisons cette opération avec des nombres aléatoires issus de nos acquisitions ainsi qu'en utilisant la fonction «rand» dans un programme c afin de pouvoir effectuer une comparaison. De plus, nous utilisons seulement la partie finale de nos acquisitions, à l'endroit où la déviation est la plus importante. Nous choisissons une taille de 40 bits pour les nombres premiers générés.

Le tableau 8.3 montre le résultat de la génération des nombres premiers. À partir des nombres ainsi produits nous calculons la proportion de nombres apparaissant plusieurs fois et en déduisons une certaine redondance d'après la formule suivante :

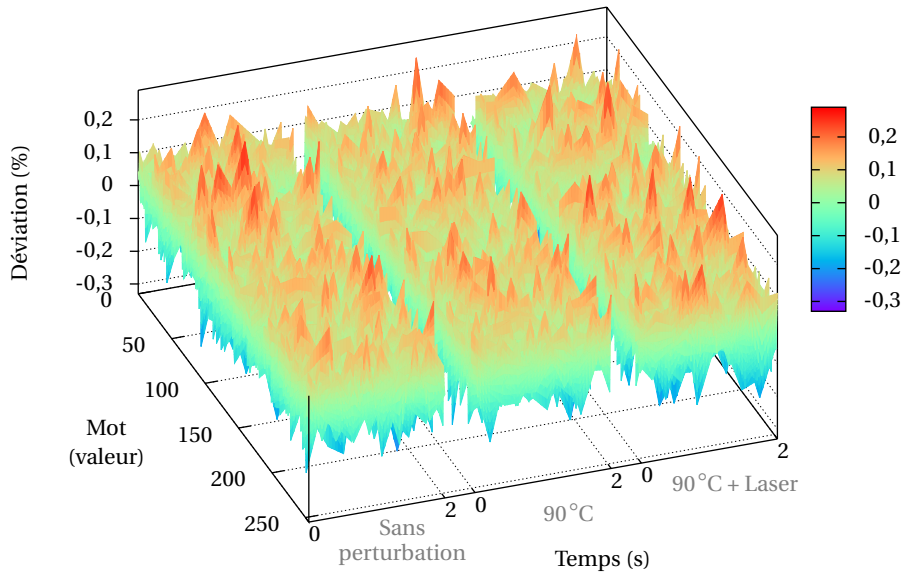
$$R = 100 \times \frac{N_{\text{tot}} - N_{\text{diff}}}{N_{\text{tot}}} \tag{8.1}$$

où N_{tot} est le nombre total de premiers générés et N_{diff} est le nombre de premiers n'apparaissant qu'une seule fois.

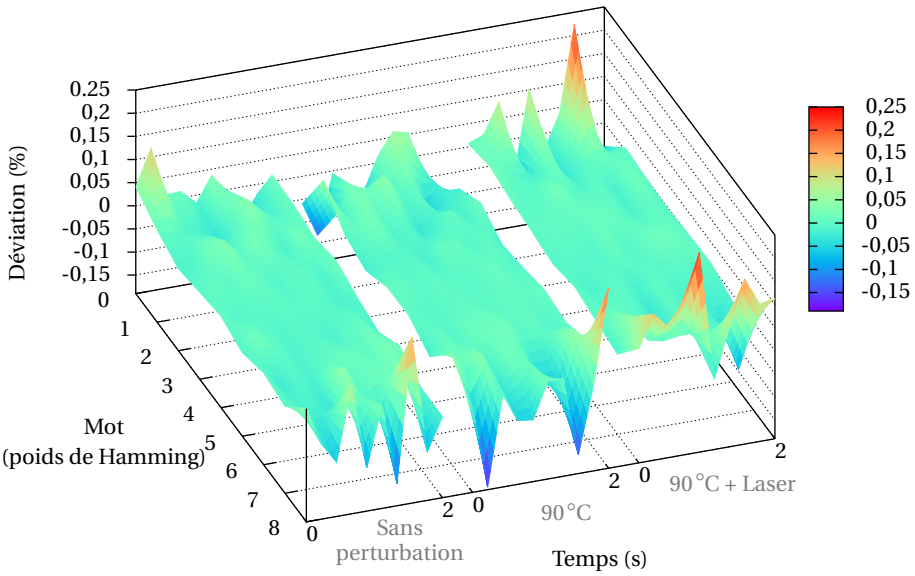
Source d'aléa	Nombres premiers générés	Nombres premiers différents	Redondance (%)
rand	4927486	4897510	0,6
acquisitions	4791510	4714515	1,6

TABLE 8.3 – Génération de nombres premiers avec le TRNG attaqué.

Les résultats montrent que parmi les nombres premiers générés il y en a plus étant identiques dans le cas où nous utilisons nos acquisitions. Ces résultats ne sont pas surprenants, en effet le biais existant dans la distribution des nombres aléatoires se retrouve dans celle des nombres premiers, leur



(a) Déviations dans la distribution des mots selon leur valeur.



(b) Déviations dans la distribution des mots selon leur poids de Hamming.

FIGURE 8.15 – Déviations dans la distribution des octets après retraitement par un AES.

différence consistant en quelques bits de poids faibles. Au final, il y a donc plus de chances de générer des nombres premiers identiques.

8.7 Conclusion

Dans ce chapitre nous avons expérimenté différentes perturbations sur un TRNG accessible via un processeur Via Nano. Sa conception est basée sur un principe classique d'échantillonnage d'un oscillateur de grande fréquence par un autre de plus petite fréquence.

L'étude des émanations électromagnétiques du processeur n'a pas permis de gagner de l'information sur le fonctionnement du TRNG. En effet, il n'a pas été possible de se synchroniser sur ses signaux internes.

Les oscillateurs ont été montrés sensibles aux variations de température et nous avons donc pu étudier l'influence sur ce TRNG. Notamment ses performances s'améliorent en diminuant la température et vice-versa.

Nous avons ensuite attaqué le processeur au laser, ce qui nous as permis de localiser des zones sensibles avec une variation visible du comportement du TRNG. Nous avons obtenu une baisse d'entropie du générateur.

Enfin, en combinant ces deux expériences nous avons mis en évidence l'évolution du comportement du TRNG au cours du temps avec différentes déviations par rapport à ce que nous en attendons. L'utilisation de plusieurs retraitements a permis d'approfondir les résultats obtenus et montré leur performances respectives. Finalement, nous avons montré que la génération de nombres premiers par l'algorithme implémenté dans OpenSSL garde les déviations présentes dans les nombres aléatoires et a donc une influence sur leur distribution.

Chapitre 9

Étude d'un TRNG basé sur un phénomène de physique quantique

Dans ce chapitre, nous étudions un QRNG développé par ID Quantique et dénommé Quantis. Il se présente sous la forme d'un composant pouvant s'intégrer sur une carte électronique et est ainsi commercialisé sur différentes plateformes permettant des connexions USB (Universal Serial Bus) ou PCI (Peripheral Component Interconnect). Un tel composant peut donc facilement être ajouté à un système informatique (PC, serveur) pour des applications nécessitant des nombres vraiment aléatoires (loteries, jeux en ligne, clefs secrètes, identifiants uniques pour un vote par internet, etc.).

9.1 Principe de fonctionnement

9.1.1 Source d'entropie

La source de bruit physique de ce générateur est basée sur un phénomène d'optique quantique et peut donc être désigné sous le terme de QRNG. Son fonctionnement est comparable à celui décrit dans la section 2.2.3 avec l'expérience du miroir semi-réfléchissant. Contrairement à ce système il n'utilise cependant pas de miroir mais est basé sur les probabilités de détection des photons sur chaque détecteur [RG07]. La figure 9.1 illustre le fonctionnement de ce QRNG. Des impulsions sur la tension d'une LED commande son état (allumée ou éteinte). Quand elle est allumée, un faisceau de lumière est généré. En face de la LED se trouvent deux photodiodes à avalanche pour détecter les photons. Selon que l'un ou l'autre des détecteurs est activé, un bit à 0 ou à 1 est produit. La succession de ces détections forme une séquence de bits aléatoires. Selon la documentation du générateur [ID 10a], le débit à sa sortie est de 4 Mbit/s.

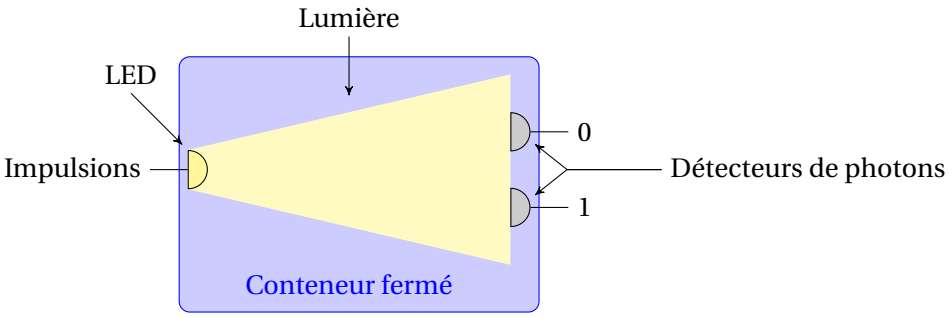


FIGURE 9.1 – Trajectoire des photons générés par une LED.

Dans le modèle sur lequel se base cette expérience, l'aléa provient du fait que les probabilités qu'un photon soit réfléchi ou transmis en atteignant un miroir semi-réfléchissant sont égales. Le QRNG que nous étudions ne contient pas de miroir, l'aléa est ici produit par la répartition des photons dans l'espace. En effet, nous devons considérer la lumière sous sa dualité onde-particule : sa propagation est caractérisée grâce aux propriétés des ondes et son interaction avec la matière grâce à celles des particules.

Ainsi, un photon représente une quantité d'énergie échangée avec la matière avec laquelle elle interagit. L'absorption d'un photon entraîne la création d'une paire électron-trou qui est multipliée par l'effet avalanche de la photodiode. Un courant est ainsi généré et permet la détection du photon. En pratique, l'énergie va être absorbée (et un photon détecté) avec une certaine probabilité. De plus, une proportion d'erreur est attendue dans la détection. Le principal bruit perturbant le fonctionnement théorique des photodiodes est le bruit thermique. Ce dernier est présent de manière générale dans tous les semiconducteurs et a pour effet de générer des paires électrons-trous comparables à celles créées par la détection d'un photon et qui entraîne par conséquent la création d'un courant dit d'obscurité. De plus, après une détection une photodiode est mise dans un mode lui permettant d'évacuer les charges pour revenir à son état initial et elle ne peut détecter aucun photon pendant un certain temps appelé temps mort (ici de 100 ns). Pour permettre une génération aléatoire des bits, l'allumage de la LED est arrêté pendant la durée du temps mort après une détection. Plus la température d'une photodiode est faible plus les charges s'éliminent lentement. Si la durée du temps mort n'est pas modifiée cela augmente les chances qu'une avalanche soit de nouveau déclenchée à cause des charges piégées dans la photodiode.

La propagation des photons est caractérisée par la fonction d'onde de la lumière, qui dépend de la source utilisée. Elle permet de calculer la probabilité de présence d'un photon à un endroit donné. La figure 9.2 montre la densité de probabilité dans le cas où la distribution des photons est gaussienne. Nous observons le plan perpendiculaire à la LED, sur lequel se trouvent les deux détecteurs. En plaçant les détecteurs de manière symétrique par rapport au centre du faisceau de lumière comme indiqué sur la figure, les bits générés ont autant de chance d'être à 0 qu'à 1.

Trois phénomènes différents peuvent faire apparaître des biais dans la génération des bits : un placement non symétrique des détecteurs par rapport à la LED, le bruit thermique dans les détecteurs et la durée de leur temps mort. L'élimination du biais dû à un mauvais placement des détecteurs est réalisée en alternant les valeurs correspondant à chaque détecteur. À la réception des photons, ces derniers vont donc générer des bits à 0 pendant un certain temps puis des bits à 1 pendant le même intervalle de temps, et ainsi de suite. De plus les bits détectés sans envoi de photon comptent pour moins de 1% du nombre total de bits selon la documentation du générateur [ID 10a]. Une vérification de ce pourcentage pendant la génération permet d'arrêter le QRNG si ce seuil est dépassé.

9.1.2 Signaux d'entrée/sortie

Le module basique composant le QRNG est relié à une carte électronique par plusieurs signaux [ID 10a]. Le détail de ces signaux est donné dans la figure 9.3a. Le signal DATA_OUT est en fait la séquence de bits générée par le QRNG. L'horloge DATA_CLOCK signale la détection valide d'un bit et ce dernier est envoyé dans un registre de 8 bits. Le signal DATA_STROBE envoie une impulsion tous les huit bits afin de transmettre le contenu du registre et pouvoir le remplir à nouveau. STATUS est à un état haut en temps normal et passe à l'état bas dans le cas d'une malfonction du module et la séquence de bits est inhibée. SHUTDOWN permet d'arrêter le fonctionnement du module et de le réinitialiser quand le signal STATUS est à l'état bas. MODULE_DETECTION permet la détection de plusieurs modules sur une même carte électronique.

Le fonctionnement du QRNG est illustré sur la figure 9.3b. Nous avons pour cela relié les signaux

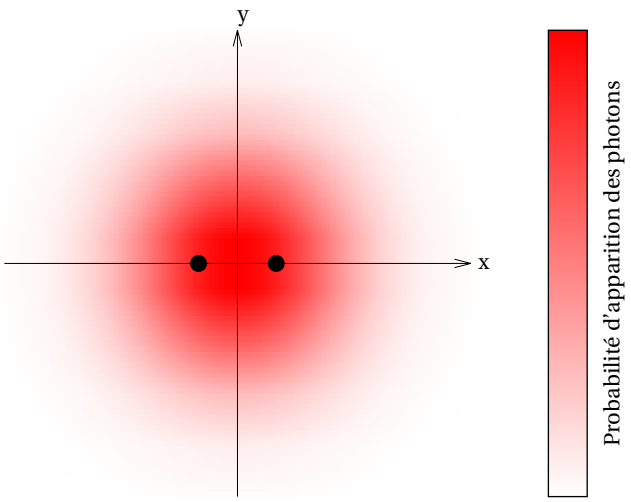
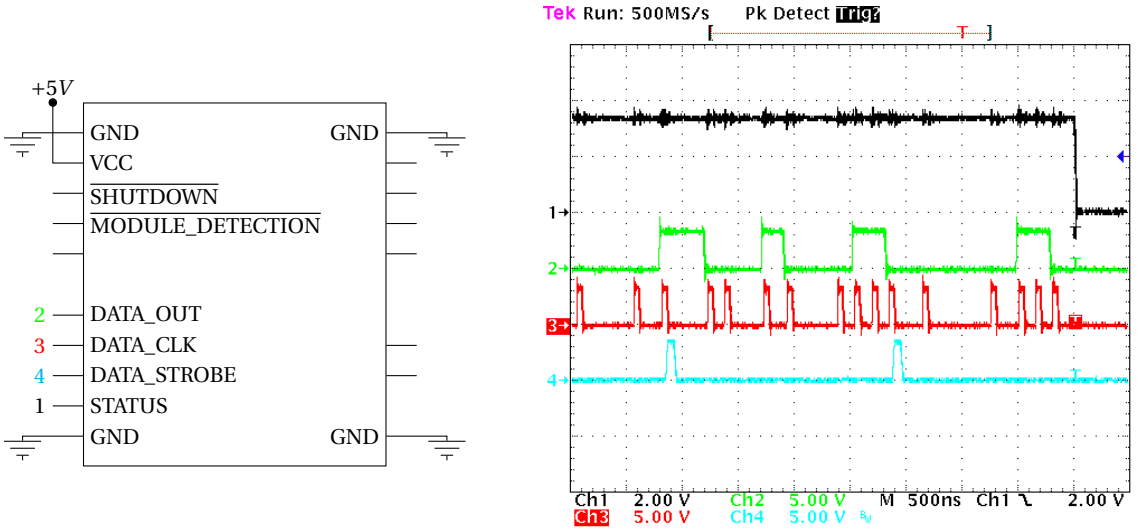


FIGURE 9.2 – Distribution des photons sur le plan opposé à la LED.

STATUS (1), DATA_OUT (2), DATA_CLK (3) et DATA_STROBE (4) à un oscilloscope. Nous observons en particulier que DATA_STROBE a une impulsion tous les huit coups d'horloge. De plus, en créant un évènement perturbateur et en triggant dessus nous voyons le signal STATUS passer à un état bas comme prévu et les autres signaux sont mis à 0.

Certains signaux sortant du QRNG n'ont pas de fonction définie. Nous les avons observés à l'oscilloscope et en avons trouvé quatre intéressants. La figure 9.4a montre ces signaux vus sur l'oscilloscope et durant 5 ms. Trois d'entre eux semblent être liés au signal d'horloge DATA_CLK tandis que le quatrième apparaît plus rarement et pourrait indiquer un bit non valide. La figure 9.4b montre DATA_CLK (en noir) et les trois signaux qui lui sont associés (voies 2, 3 et 4 de l'oscilloscope). Nous déduisons de cette observation que DATA_CLK est activé quand les signaux sur les voies 2 et 3 sont à l'état haut ou quand le signal sur la voie 2 est à l'état haut.

Nous avons ensuite accédé à l'intérieur même du module Quantis et pu observer des signaux directement en sortie des détecteurs ainsi que l'alimentation de la LED. La figure 9.5a montre quatre de ces signaux : celui de la LED (en noir) et trois autres nous donnant des informations intéressantes sur le fonctionnement du générateur. En particulier, il semble que les signaux sur les voies 2 et 3 correspondent aux sorties des détecteurs et que sur la voie 4 nous ayons leur combinaison par une opération logique OU. Les détecteurs ayant une certaine probabilité de détection, toutes les impulsions de la LED ne sont pas suivies d'une détection. Dans le cas où cette dernière a lieu alors la prochaine impulsion de la LED intervient après le temps mort. La figure 9.5b montre la relation entre le signal de la LED, celui enregistrant le total des détections et le signal en sortie du module qui apparaît peu souvent. Cela nous permet de confirmer que ce dernier indique la détection d'un bit sans activation de lumière donc non valide. Un tel phénomène peut être dû au bruit thermique dans les détecteurs comme nous l'avons expliqué précédemment.



(a) Détail des signaux entrant et sortant du QRNG.

(b) Fonctionnement du QRNG.

FIGURE 9.3 – Signaux d'entrée/sortie du QRNG.

9.2 Perturbation de la propagation des photons

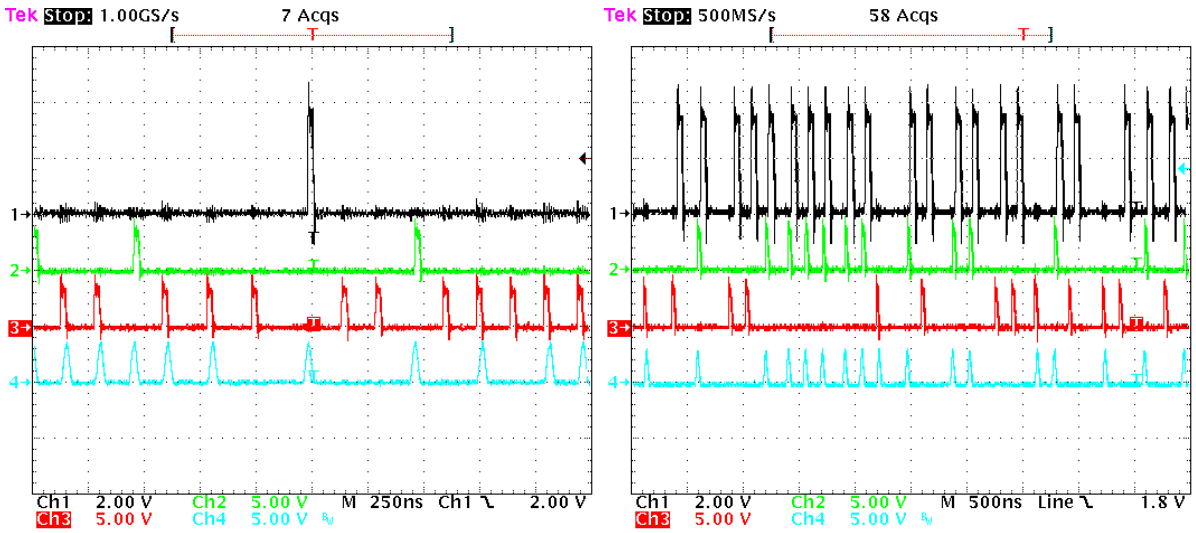
Dans cette expérience, nous nous sommes approchés au plus près du fonctionnement du générateur. Pour cela nous avons percé un trou dans le module entre la LED et les détecteurs. Nous avons ainsi pu injecter de la lumière additionnelle ou bloquer le chemin des photons envoyés par la LED.

Sur la figure 9.6 nous observons les signaux de la LED en noir et du total des détections en vert, dans l'état normal et perturbé du générateur. Sans perturbation les détections correspondent bien aux envois de photons, qui sont de plus effectués de manière régulière. En ajoutant de la lumière, les détections sont plus nombreuses que les envois et ces derniers semblent plus espacés. Enfin, en bloquant le chemin de la lumière entre la LED et les détecteurs avec un papier, les détections sont moins nombreuses que les envois de photons et ceux-ci ont un débit plus rapide.

Les perturbations ont donc un impact direct sur le débit des détections et par conséquent sur la vitesse d'allumage de la LED. En effet, dans le cas normal il y a le temps mort entre une détection et le rallumage de la LED pour permettre aux détecteurs de revenir à leur état initial. En ajoutant de la lumière, nous obtenons plus de détections dont certaines durant le temps mort, l'allumage de la LED est donc repoussé et intervient au final moins souvent. Au contraire, en insérant un papier nous avons moins de détections et la LED est allumé plus souvent car il y a moins de temps mort. De plus, les observations en perturbation ont été réalisées alors que le signal STATUS était à l'état bas, le module a donc détecté notre intrusion dans le système et désactivé la sortie DATA_OUT. Il examine pour cela le taux de détections sans envoi de photons et le débit de sortie du générateur.

La figure 9.7 montre le signal d'erreur qui est généré lors de l'ajout de lumière. Nous voyons ici qu'il gagne une impulsion dès qu'il y a une détection sans envoi de photon. Le cas contraire, un envoi sans réception n'est pas considéré comme une erreur car les détections se font avec une certaine probabilité. De plus, les pics arrivent plus souvent dans le signal d'erreur en comparaison au cas normal représenté sur la figure 9.5b.

En revanche, quand nous bloquons le chemin des photons avec un papier, le signal d'erreur n'est pas différent. En fait, au niveau des détecteurs cela revient à avoir une probabilité de détection plus faible.



(a) Relation entre les quatre signaux inconnus.

(b) Comparaison de trois signaux à DATA_CLK.

FIGURE 9.4 – Visualisation des signaux inconnus.

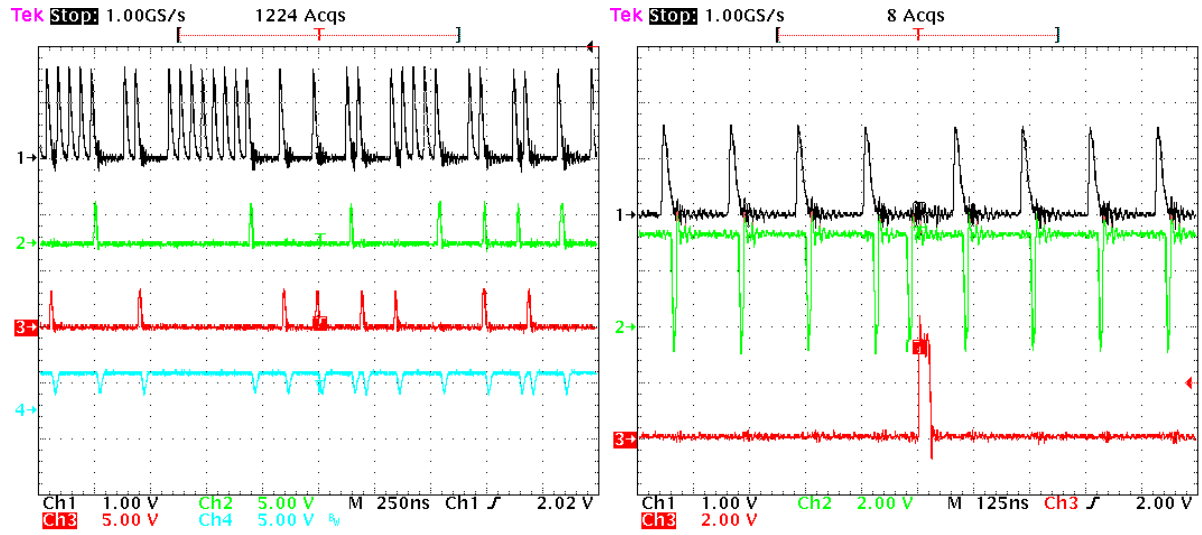
9.3 Variations de température

Nous appliquons des variations de température à ce QRNG, de manière similaire aux précédents TRNG étudiés. La carte sur laquelle est intégrée le générateur ne peut pas être insérée dans le dewar contenant l'azote liquide. La méthode consistant à utiliser une tige en aluminium plongée dans de l'azote liquide n'étant ici pas plus efficace qu'un module Peltier, nous avons donc appliqué ce dernier. Pour le chauffage nous avons exploité le banc décrit dans le section 5.1.1, en plaquant le crochet contre le module contenant la LED et les détecteurs.

Les températures de test obtenues sont −20 °C, 0 °C, 10 °C, 25 °C (température ambiante), 60 °C et 100 °C Des séquences de 307,2 Mo sont générées à chaque température en environ 10 minutes. Ces acquisitions sont réalisées sur la sortie après retraitement (inversion des valeurs attribuées aux détecteurs).

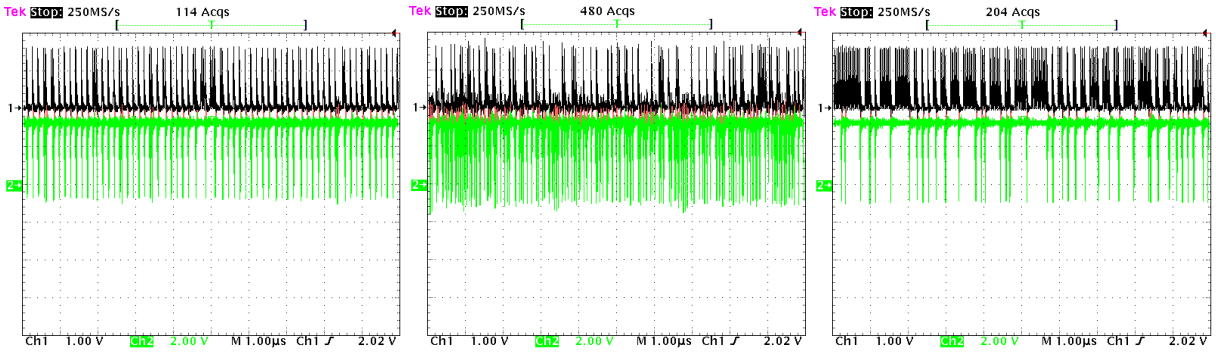
9.3.1 Résultats des tests statistiques

La figure 9.8 montre les taux d'échec aux tests statistiques des séquences générées à chaque température. Les taux d'échec des tests à température ambiante sont très faibles et montrent la qualité des bits aléatoires par cette méthode de génération quantique. Bien évidemment, nous observons ici la séquence de bits après le retraitement inversant la valeur attribuée à chaque détecteur et les défauts se rapportant à la mauvaise distribution des bits à 0 et à 1 sont éliminés. En augmentant la température nous obtenons une très légère hausse des taux d'échec (NIST et Alphabit). La diminution de la température a elle un effet plus marqué avec la quasi annulation de tous les taux d'échec à −20 °C. D'après le fonctionnement des photodiodes, en diminuant encore la température nous pourrions observer un effet dû à l'allongement du temps nécessaire à la suppression des charges après une détection. En effet, les charges qui n'ont pas eu le temps d'être évacuées durant le temps mort augmentent les chances de déclencher un effet avalanche intempestif quand une photodiode retourne en mode de détection de photons. Nous aurions alors des corrélations entre les bits générés. Dans notre expérience, la température atteinte n'est pas assez basse pour observer un tel effet car il s'agit en fait de rejoindre le mode de fonctionnement de gel des électrons sur un semiconducteur (figure 5.6)



(a) Signaux aux bornes de la LED et des détecteurs. (b) Activation d'un signal de fausse détection.

FIGURE 9.5 – Activation du signal d'erreur.



(a) Sans perturbation. (b) Injection de lumière. (c) Insertion d'un papier.

FIGURE 9.6 – Perturbation de la propagation des photons.

à approximativement −150 °C.

La figure 9.9 représente la distribution des mots de 8 bits des séquences testées en considérant la valeur des mots ou leur poids de Hamming. Ces graphes confirment nos observations précédentes : à 25 °C il existe de faibles déviations dans la distribution des mots, qui sont légèrement amplifiées avec l'augmentation de la température. La diminution de la température les fait disparaître jusqu'à avoir une distribution uniforme. De plus, la distribution des mots selon leur poids de Hamming est symétrique pour toutes les températures ce qui signifie que les occurrences de bits à 0 et à 1 sont égales et la correction consistant à inverser les valeurs attribuées aux détecteurs est effective. L'évolution de cette distribution avec la température est conforme à nos précédentes observations avec des déviations qui s'annulent à −20 °C et qui s'accentuent avec le chauffage.

9.3.2 Impact de la variation de température sur le QRNG

Dans la description du fonctionnement de ce générateur nous avons évoqué des biais pouvant être provoqués à basse et haute température. Dans le premier cas, des corrélations devraient appa-

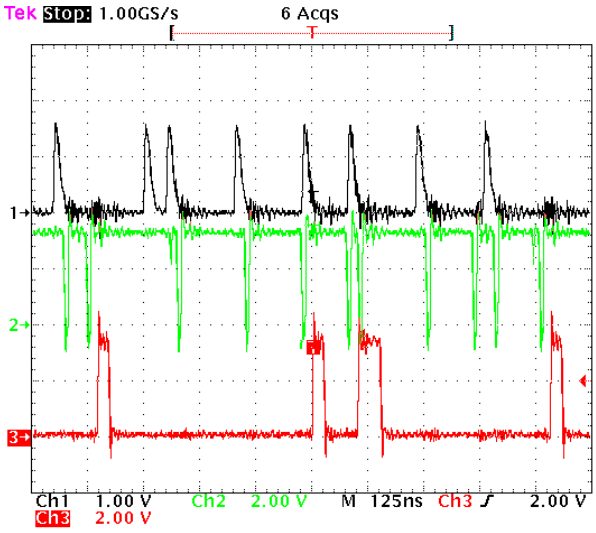


FIGURE 9.7 – Signal d'erreur lors de l'injection de lumière.

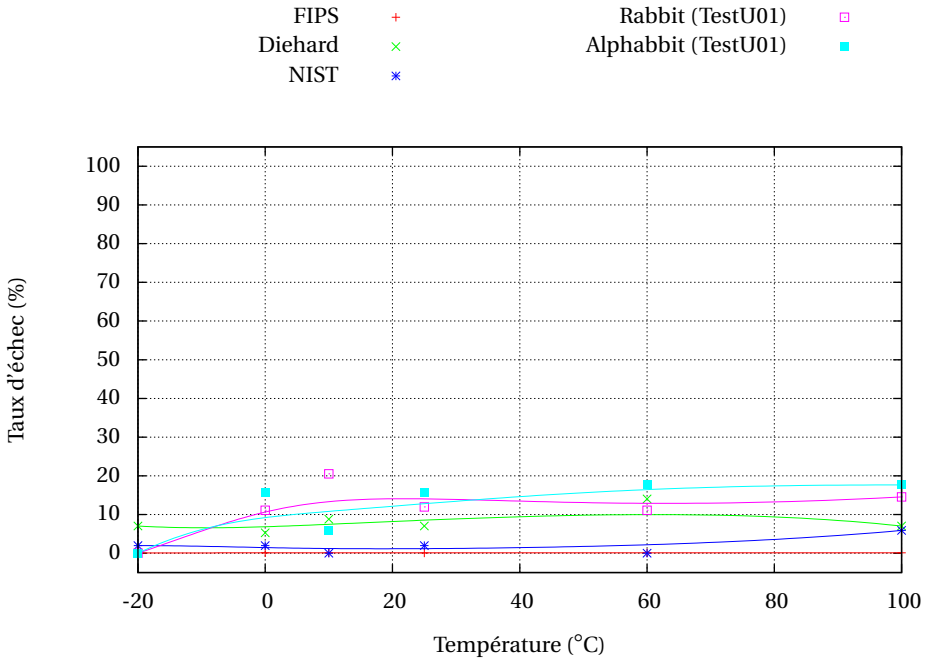
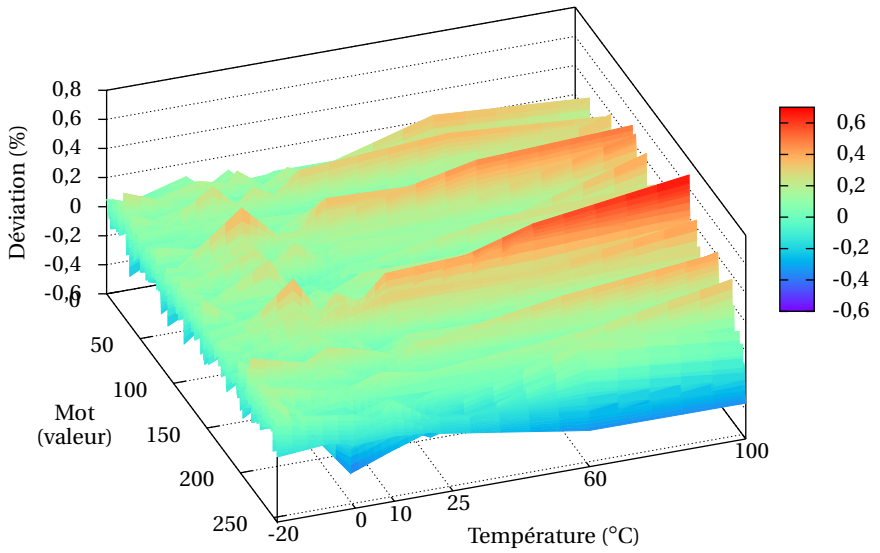


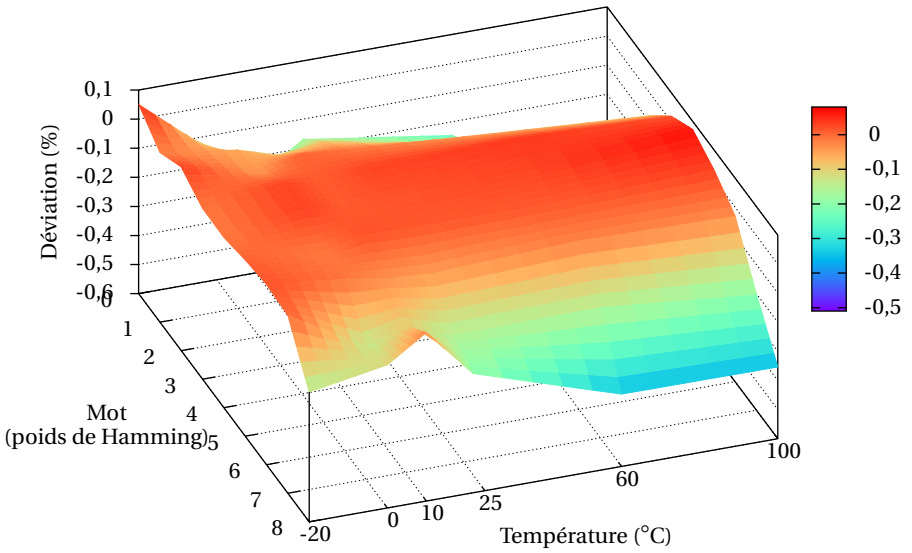
FIGURE 9.8 – Résultat des tests statistiques pour le QRNG perturbé en température.

raître entre les bits générés (bits identiques se suivant). Dans nos expérimentations nous n'avons pas observé ce phénomène, probablement car la température appliquée n'était pas assez basse.

En augmentant la température, le bruit thermique est aggravé et donc les chances d'avoir une détection sans envoi de photon sont plus importantes. Cela peut induire un biais si la détection se fait durant un temps mort, donc avec un détecteur désactivé. Dans ce cas nous aurons tendance à obtenir des suites de bits avec des valeurs qui sont alternées. Dans nos expériences nous retrouvons bien ce phénomène, en particulier à 100 °C les octets ayant les plus grandes déviations sont ceux de valeur 170 et 85 (respectivement 10101010 et 01010101 en binaire).



(a) Déviations dans la distribution des mots selon leur valeur.



(b) Déviations dans la distribution des mots selon leur poids de Hamming.

FIGURE 9.9 – Déviations dans la distribution des mots avec des variations de température.

9.4 Exposition à des rayonnements ionisants

Comme expliqué précédemment, le principe de la détection des photons pour ce QRNG passe par la création de paires électrons-trous. Les rayonnements ionisants mettent eux aussi en œuvre des

photons et peuvent potentiellement générer un effet similaire.

9.4.1 Utilisation de sources radioactives

Nous décrivons ici les tests que nous avons effectués sur le QRNG en manipulant les sources radioactives décrites dans la section 5.4. Elles sont positionnées sur le conteneur du système de génération quantique.

Nous distinguons les deux sources d'américium par Am^1 et Am^2 d'activités respectives 3,7 MBq et 518 MBq, le baryum est représenté par Ba et le césium par Cs. Nous avons acquis des séquences de 102,4 Mo (en 2,5 minutes) avec chaque source ainsi qu'à l'état normal (sans source) afin d'effectuer des comparaisons.

La figure 9.10 montre les taux d'échec aux tests statistiques dans tous les cas de figure.

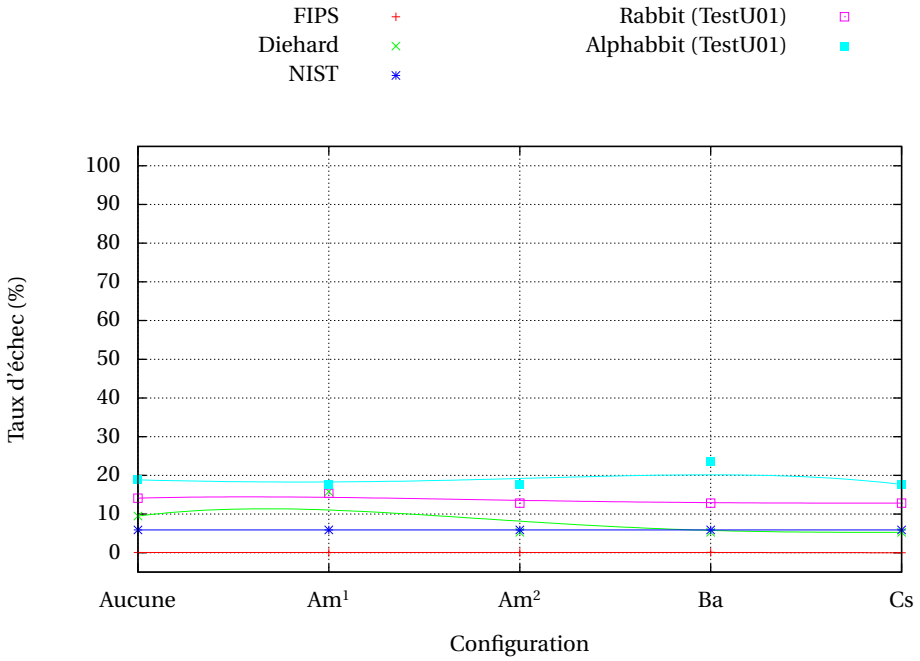


FIGURE 9.10 – Résultat des tests statistiques pour le QRNG perturbé par des rayons gamma.

L'observation des résultats des tests statistiques ne nous permet pas de conclure qu'il existe une influence des source radioactives. En effet, en comparant la performance du QRNG pour les différentes sources nous de voyons pas de variation évidente des taux d'échec par rapport à son état normal ou à l'application d'autres sources. Il est possible que les sources ne soient pas assez énergétiques pour pénétrer au travers du conteneur et influencer le système quantique.

9.4.2 Utilisation de rayons X

L'utilisation de sources radioactives n'ayant pas montré d'influence particulière de ces dernières sur le QRNG, nous avons ensuite utilisé des rayons X. Nous avons pour cela exploité le banc de test décrit dans la section 5.4.2 en variant le courant circulant dans le tube à rayons X et la tension appliquée (donc l'énergie du rayonnement).

Le QRNG est placé à environ 70 cm du tube. Les différentes configurations ainsi mises en place sont détaillées dans le tableau 9.1. L'état normal du QRNG (sans perturbation) est désigné par la

configuration 0. Nous avons ici aussi acquis des séquences de 102,4 Mo (en 2,5 minutes) pour chaque configuration ainsi qu'à l'état normal.

Configuration	Tension (kV)	Courant (mA)
1	140	21,42
2	70	40
3	40	45
4	60	45
5	160	16
6	160	18

TABLE 9.1 – Configurations du banc de test utilisant les rayons X.

La figure 9.11 montre les taux d'échec aux tests statistiques pour chacune des configurations. Les

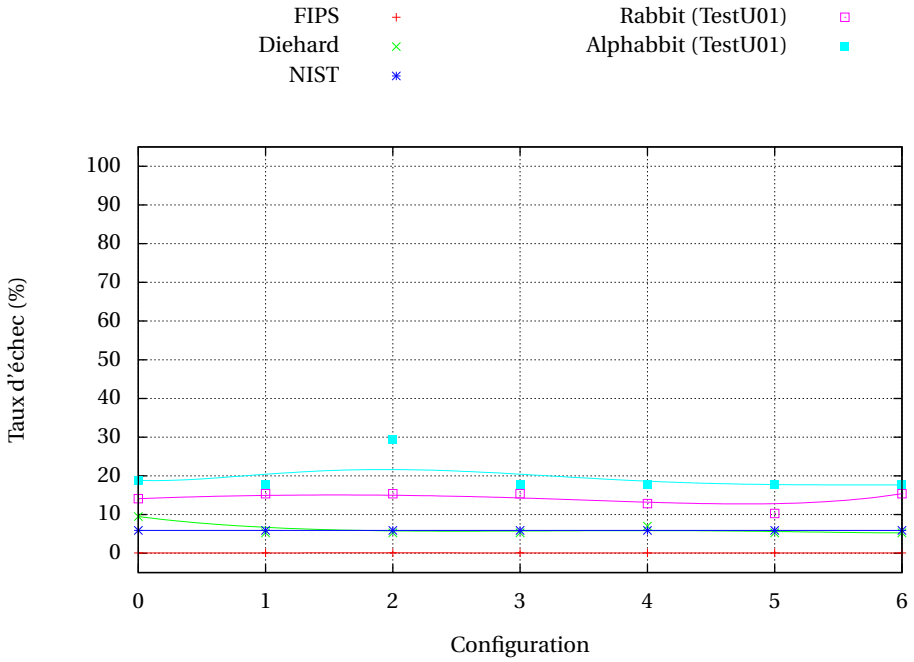


FIGURE 9.11 – Résultat des tests statistiques pour le QRNG perturbé par des rayons X.

résultats obtenus ne montrent pas de variations selon les configurations utilisées ou par rapport à l'état normal du QRNG. Ces résultats sont donc très similaires au cas précédent mettant en œuvre des sources radioactives.

Dans cette expérience aussi, le conteneur a pu enrayer la pénétration des rayons X et empêcher l'obtention d'effets différents en comparaison du comportement attendu. Cependant, il n'est pas improbable que des perturbations aient réellement été générées mais sans répercussion sur la qualité des bits produit. En effet, l'injection de rayons X a elle-même un caractère aléatoire quant à son impact sur les détecteurs de photons. En revanche, si un effet avait réellement été produit alors nous aurions eu des détections sans envoi de photons et le module aurait dû cesser de fonctionner, ce qui n'a pas été le cas.

9.5 Conclusion

Dans cette partie nous avons étudié un système de génération de nombres aléatoires basé sur la physique de l'optique quantique. Un tel système exploite un phénomène aléatoire «parfait», les défauts provenant des détecteurs de photons, et il a donc de bonnes performances dans son fonctionnement normal.

En faisant varier la température du QRNG, nous avons pu observer l'aggravation des erreurs de détections déjà présentes dans le système dans son état initial et dues au bruit thermique dans les détecteurs. La perturbation est cependant restée très faible (perte d'entropie inférieure à 5×10^{-6} bits/octet) et sans danger pour l'utilisation des nombres aléatoires en résultant.

En revanche, l'utilisation de sources radioactives et de rayons X n'a pas révélé de perturbation du générateur que ce soit sur la qualité de l'aléa généré ou des détections intempestives de photons. Ces rayonnements n'ont donc pas eu d'influence sur le QRNG.

Enfin, la perturbation directe de la détection des photons nous a permis une meilleure compréhension du fonctionnement du système. Cependant, notre intrusion a été détectée dans tous les cas et la génération des nombres aléatoires arrêtée.

Les vérifications implantées dans le système sur la validité des bits et de leur distribution ont donc réussi à contrer nos attaques et semblent adéquates.

Chapitre 10

Étude d'un CSPRNG : Blum-Blum-Shub

Dans ce chapitre, nous nous éloignons de l'aspect physique de la génération d'aléa pour nous pencher sur l'algorithme BBS qui est un CSPRNG. Nous voulons étudier ici des moyens de récupérer des informations sur son fonctionnement par des chemins détournés.

10.1 Principe de fonctionnement

Nous avons présenté l'algorithme de Blum-Blum-Shub comme un CSPRNG dans la section 2.3.3.2. Nous rappelons sa construction ici. Soit $N = pq$, p et q deux premiers distincts congruents à 3 modulo 4. Soit X_0 un résidu quadratique dans Z_N^*, $X_0 = s^2 \bmod N$ avec $s \in [1; N-1]$ une graine aléatoire. La séquence de bits générée par BBS ($b_0 b_1 \cdots$) est obtenue par les opérations suivantes :

$$X_{i+1} = X_i \bmod N \tag{10.1}$$

$$b_i = \text{parité}(X_i) \tag{10.2}$$

Les X_i peuvent être calculés directement par $X_i = {X_0}^{2^i} \bmod N$.

La sécurité de ce PRNG repose sur le problème de résiduosité quadratique qui consiste à décider pour un $X \in Z_N^*$ si celui-ci est un résidu quadratique. BBS est un CSPRNG imprédictible, c'est-à-dire qu'une séquence de bits qu'il génère ne peut pas être distinguée d'une séquence de bits vraiment aléatoire par un test statistique en temps polynomial ($\log(N)$).

Il a plus tard été prouvé que BBS est sûr tant que factoriser N est difficile [VV84, ACGS84]. Sous cette condition il est aussi possible d'extraire non plus un bit mais les $\log_2(\log_2(N))$ plus petits bits des X_i en toute sécurité.

10.2 Observation de l'implantation dans une puce

Nous avons tout d'abord implémenté l'algorithme BBS dans une puce afin d'en étudier son fonctionnement. Pour cela, il est possible d'optimiser les calculs en utilisant la multiplication de Montgomery (MM). Le but de cette expérimentation est de retrouver des informations sur les données manipulées par DPA.

10.2.1 La multiplication de Montgomery

Présenté par Peter Montgomery en 1985 [Mon85], cet algorithme effectue une multiplication modulaire de manière efficace. Il est particulièrement adapté pour le calcul de plusieurs multiplications

modulaires de même modulo à la suite. En conséquence, il est souvent utilisé en cryptographie pour effectuer des calculs d'exponentiation modulaire (pour RSA par exemple).

En réalité, la multiplication de Montgomery (MM) effectue le calcul $AB \bmod N$ en modifiant la base de représentation des opérandes :

$$MM(A,B,N) = ABr^{-t} \bmod N \tag{10.3}$$

où $A = \sum_{k=0}^{t-1} a(k)r^k$, $B = \sum_{k=0}^{t-1} b(k)r^k$ et $N = \sum_{k=0}^{t-1} n(k)r^k$ avec $t = \lceil \log_r(N) \rceil$. Les variables en minuscule avec leur rang entre parenthèses sont la suite des mots de base r constituant les opérandes.

Le détail des opérations constituant la multiplication $MM(A,B,N)$ est expliqué dans l'algorithme 1. Les indices des variables correspondent aux itérations de l'algorithme.

Algorithme 1 Détail de la multiplication de Montgomery $MM(A,B,N)$

1: $n'(0) \leftarrow -n(0)^{-1}$
2: $P \leftarrow 0$
3: **for** $k = 0$ **to** $t-1$ **do**
4: $\quad q_k \leftarrow ((p_0 + a(k)b(0)) \bmod r)n'(0) \bmod r$
5: $\quad P \leftarrow \dfrac{(P + a(k)B + q_k M)}{r}$
6: **end for**
7: **return** P

Le changement de représentation des opérandes à l'origine de la multiplication de Montgomery est avantageux quand r est un multiple de 2 et qu'une division par r revient à faire des décalages à droite dans un registre.

10.2.2 Détail de l'implantation

Nous voulons maintenant appliquer la multiplication au calcul de BBS. Pour une itération de BBS, il suffit d'appliquer l'opération $MM(X_i, X_i, N) = X_i^2 r^{-t} \bmod N$ une fois puis de multiplier le résultat par r^t. Cette dernière multiplication peut être optimisée en utilisant encore une multiplication de Montgomery comme suit :

$$\begin{aligned} MM(X_i^2 r^{-t}, r^{2t}, N) &= X_i^2 r^{-t} r^{2t} r^{-t} \bmod N \\ &= X_i^2 \bmod N \end{aligned} \tag{10.4}$$

L'efficacité de l'algorithme de Montgomery se trouvant dans l'enchainement des multiplications, nous pouvons adapter les opérations pour produire les itérations du générateur BBS. L'algorithme 2 présente l'algorithme que nous implantons dans la puce. Le changement de base de représentation R est calculé en dehors des itérations car il reste constant. Ici r est égal à 16, les mots composants les opérandes sont donc constitués de 4 bits. La normalisation finale (ligne 8) permet de ramener le résultat entre 0 et N si jamais il est négatif.

10.2.3 Analyse statistique du signal de consommation de la puce

Nous observons maintenant le déroulement de l'algorithme sur la puce. Nous posons pour cela des sondes sur le signal d'entrée/sortie du lecteur de puce, ainsi que sur son alimentation et un signal de synchronisation envoyé par la commande du PC à la puce. Nous observons ces signaux avec un oscilloscope.

Algorithme 2 Algorithme de BBS par la multiplication de Montgomery pour *ni* itérations

1: $R \leftarrow r^{2t}$
2: $C \leftarrow ni$
3: $A \leftarrow X_0$
4: **while** $C > 0$ **do**
5: $\quad B \leftarrow MM(A, A, N)$
6: $\quad A \leftarrow MM(B, R, N)$
7: $\quad$ **if** $A < 0$ **then**
8: $\quad\quad A \leftarrow A + N$
9: $\quad$ **end if**
10: $\quad C \leftarrow C - 1$
11: **end while**

Nous fixons la taille de N à 1024 bits, le nombre d'itérations à 10000 et nous générons un X_0 (résidu quadratique) aléatoire. Un signal de synchronisation nous permet d'observer la consommation de la puce lors des itérations de l'algorithme. La figure 10.1 montre cette consommation pendant une itération de BBS avec l'amplitude du signal qui lui est proportionnelle et sur l'axe des x les échantillons successifs. En particulier, nous pouvons voir deux blocs distincts avec une consommation plus importante. Ils correspondent au calcul des deux multiplications de Montgomery (lignes 5 et 6 de l'algorithme 2).

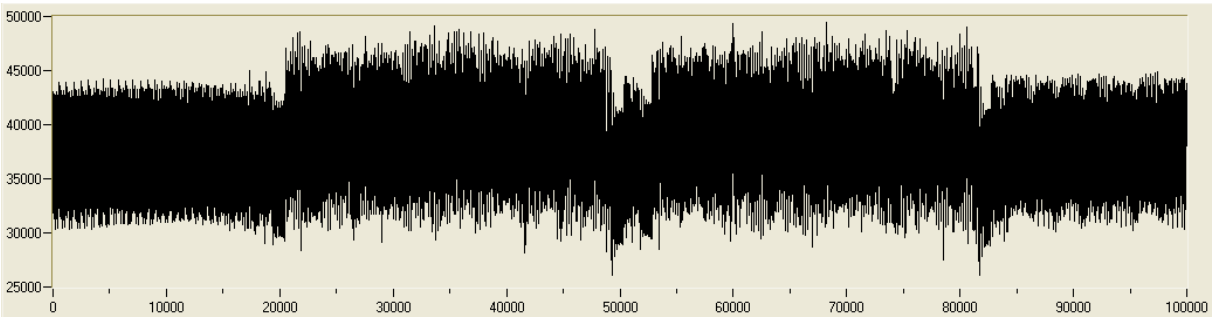
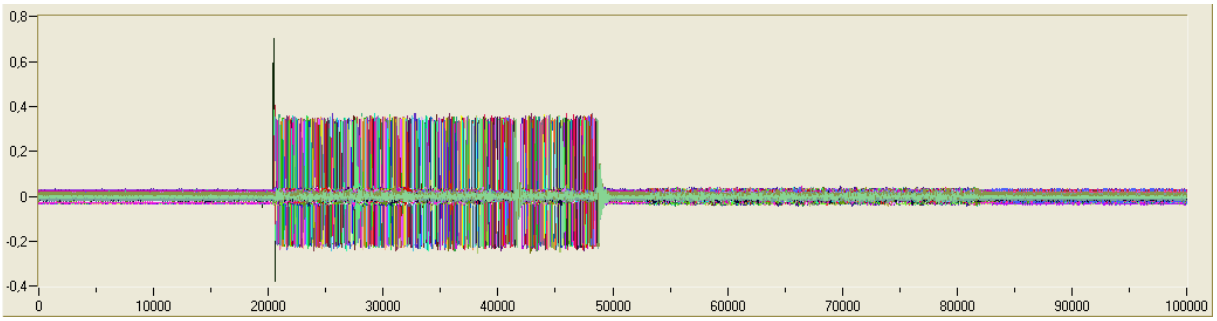


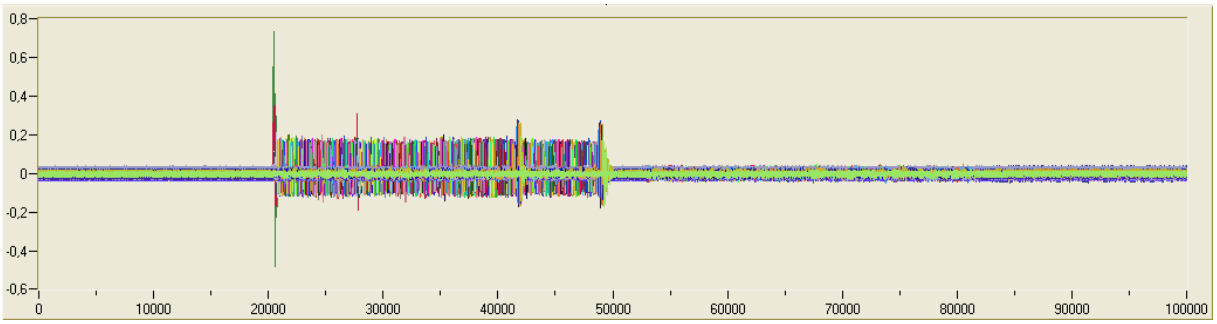
FIGURE 10.1 – Une itération de BBS.

En pratique le calcul de la multiplication de Montgomery est implémenté selon l'algorithme 1. Nous voulons récupérer de l'information sur les variables utilisées dans l'algorithme grâce aux signaux de consommation. En particulier, à chaque itération de l'algorithme il y a une mise à jour des variables $p(0), a(k), q_k$ et P. Ceci passe par le remplacement de leurs anciennes valeurs par celles nouvellement calculées. L'énergie dépensée pour réaliser cette opération va être dépendante du nombre de bit écrasés dans les registres contenant les variables. Elle sera donc proportionnelle à leur distance de Hamming (nombre de bits différents entre deux variables lors d'une comparaison bit à bit).

L'objectif de notre expérimentation est de vérifier si cette énergie se retouve dans la consommation de la puce. À l'instant où une variable est mise à jour, nous devrions avoir une consommation proportionnelle à la distance de Hamming entre son ancienne et sa nouvelle valeur. Nous combinons donc les courbes de manière à faire ressortir cette relation. Pour cela nous associons à chaque courbe une pondération lui correspondant. Cette pondération est égale à la distance de Hamming normalisée, c'est-à-dire en lui retranchant 8 (les variables étant sur 16 bits). Les pondérations sont donc des valeurs dans $[-8;8]$ au lieu de $[0;16]$. Nous faisons ensuite la moyenne de toutes les courbes pondérées (10000 itérations). Le résultat est ainsi composé de bruit avec un pic à l'instant où la variable est



(a) Pics aux instants où $a(k)$ est mis à jour pour $0 \leqslant k \leqslant t-1$.



(b) Pics aux instants où q_k est mis à jour pour $0 \leqslant k \leqslant t-1$

FIGURE 10.2 – Corrélations entre les valeurs successives des variables $a(k)$ et q_k.

écrasée, le moyenne annulant tout le reste. La figure 10.2 montre le résultat de cette opération sur les variables $a(k)$ et q_k durant la première multiplication de Hamming. Chaque courbe correspond à une itération k, d'où t courbes de couleurs différentes. Les signaux ainsi calculés représentent des corrélations et sont donc compris entre 0 et 1 en absolu. Ici, nous voyons des pics de corrélations importantes à chaque mise à jour des variables considérées. Ces pics s'enchainent sur différentes courbes selon les itérations successives de l'algorithme.

10.2.4 Exploitation des corrélations observées

La variable $a(k)$ étudiée précédemment dans la première multiplication de Montgomery correspond en fait aux mots successifs constituant un X_i. La connaissance des distances de Hamming entre les $a(k)$ et $a(k+1)$ nous donne de l'information sur la valeur de X_i et peut être la possibilité de le reconstruire complètement. Par exemple, au lieu d'avoir N combinaisons possibles pour la valeur de X_i, il est possible de construire un ensemble plus réduit de possibilités en déduisant les différentes valeurs $x(k+1)$ à partir de leur distance de Hamming avec x_k. Les autres variables fuyant aussi de l'information sur la consommation de la puce peuvent apporter de l'information supplémentaire.

De plus, si nous connaissons des X_i successifs, il est possible de calculer N de la façon suivante :

$$N = pgcd(X_{i+1}^2 - X_i, X_i^2 - X_{i-1}, \cdots) \tag{10.5}$$

La connaissance d'un X_i et de N nous permet de calculer ensuite toutes les itérations suivantes de X.

10.3 Simulation à partir de graines biaisées

Nous examinons maintenant le comportement de BBS en fonctionnement anormal. Nous étudions pour cela les mots générés par l'algorithme dans le cas où la graine est biaisée. En particulier,

nous cherchons à savoir si un biais va se propager à travers les itérations successives.

Pour cela, nous étudions la distribution des mots Z_i générés à chaque itération en prenant les $\log_2(\log_2(N))$ plus petits bits des X_i ($Z_i = X_i \bmod \log_2(\log_2(N))$. En fixant la taille de N à 256 bits, les mots sont eux sur 8 bits. Nous choisissons un biais pour les X_0 en vérifiant qu'ils sont bien des résidus quadratiques dans Z_N^*. Nous tirons 30 N différents et pour chaque N nous faisons tourner l'algorithme 100 000 fois (avec autant de X_0 distincts) sur 10 itérations pour observer l'évolution de la distribution des mots extraits. Nous calculons ensuite les occurrences de ces mots pour chaque itération.

La figure 10.3 présente la déviation dans la distribution des mots quand les X_0 sont générés à partir d'une distribution uniforme. Elle montre de plus cette distribution pour les 10 premières itérations de l'algorithme. Il s'agit du cas de fonctionnement normal de BBS, et nous ne voyons donc pas de biais particulier, toutes les combinaisons possibles des mots sont présentes avec autant de chance d'apparition.

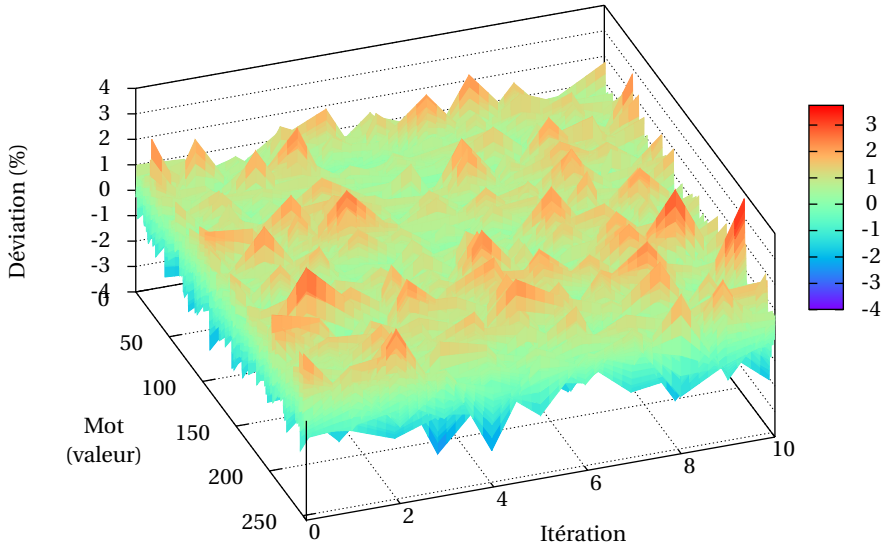


FIGURE 10.3 – Déviations dans la distribution des mots sans biais initial.

La figure 10.4 montre le cas où les X_0 sont constitués de longs runs de 0 et de 1. L'extraction de Z_0 montre seulement quelques combinaisons parmis toutes celles possibles. En effet, nous obtenons en grande majorité des mots «0»et «1». Après une itération de BBS, ce biais disparaît complètement.

La figure 10.4 montre le cas où les X_0 sont constitués de bits à 90% à 0 et à 10% à 1. Malgré cette forte déviation, il n'apparaît pas de biais à l'itération suivante.

La figure 10.4 montre le dernier cas pour lequel nous avons généré les X_0 à partir de la plus forte déviation obtenue lors de l'expérience combinant le chauffage du processeur Via Nano avec l'application d'un faisceau laser à sa surface. Les X_0 montrent encore de fortes déviations qui ne sont pas transmises aux itérations suivantes.

En dépit des fortes déviations que nous avons appliquées aux distributions de X_0, il semble que l'opération de mise au carré modulaire redistribue les X_i de manière uniforme dès la première itération. En implémentant BBS de la manière décrite au début du chapitre avec une graine s puis $X_0 = s^2 \bmod N$ à partir duquel nous commençons à extraire les nombres aléatoires, nous serions

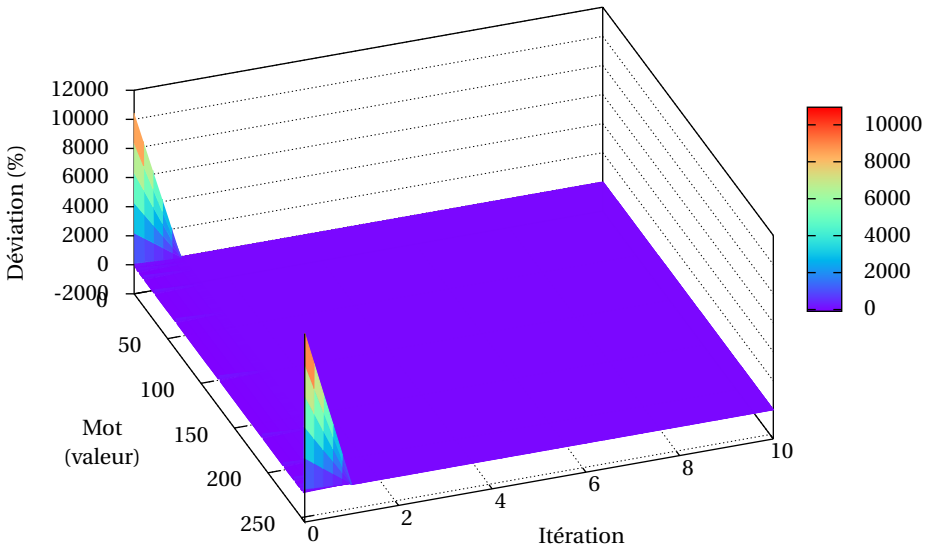


FIGURE 10.4 – Déviations dans la distribution des mots avec X_0 constitués de longs runs.

assurés d'éviter toute déviation en sortie du générateur.

10.4 Conclusion

Dans cette partie nous avons étudié un système de génération de nombres aléatoires basé sur un algorithme déterministe. Nous avons étudié le fonctionnement de ce générateur pour en tirer des informations normalement secrètes.

En implémentant BBS dans une puce nous avons pu étudier sa consommation lors des opérations de l'algorithme. Nous avons ainsi pu gagner de l'information sur les variables manipulées. Cependant, cette technique classique d'attaque d'un système cryptographique peut être évitée en masquant ces variables avec des nombre aléatoires.

Dans une deuxième partie nous avons étudié la réponse de BBS à une entrée biaisée. Cette expérience a montré qu'aucune des déviations appliquées n'est transmise lors des itérations de l'algorithme.

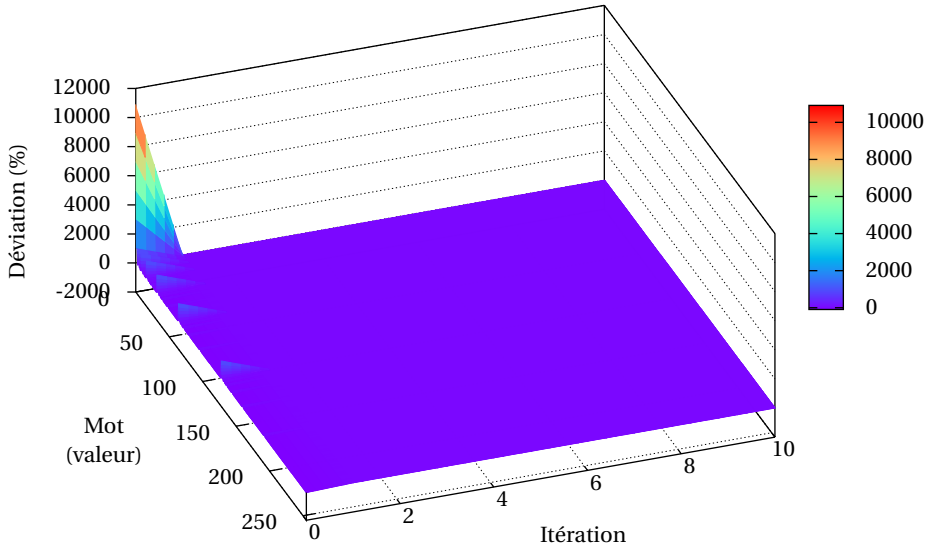


FIGURE 10.5 – Déviations dans la distribution des mots avec un gros biais initial.

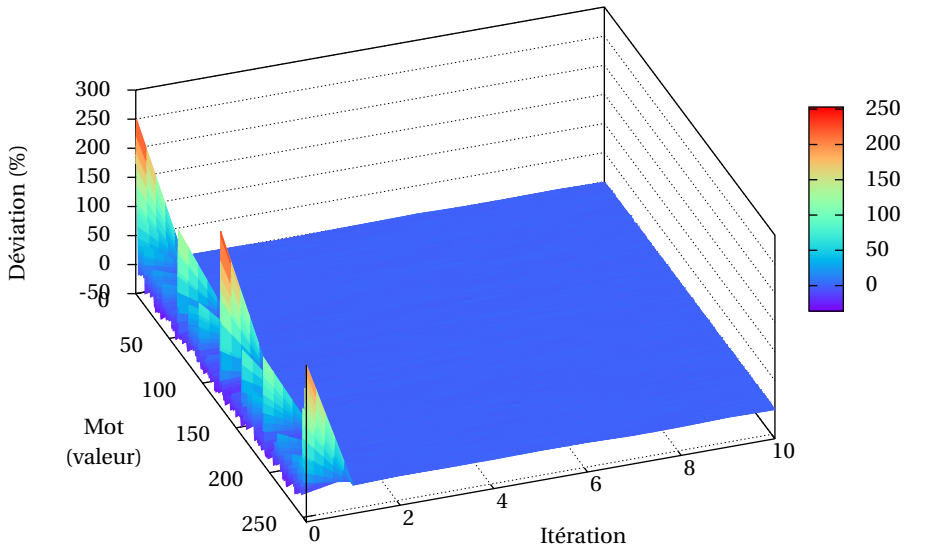


FIGURE 10.6 – Déviations dans la distribution des mots avec un biais initial similaire à celui observé sur le processeur Via Nano.

Chapitre 11

Conclusion

Les travaux présentés dans ce mémoire regroupent des observations sur le fonctionnement de divers RNGs et l'analyse de leurs vulnérabilités. Ils se placent dans le cadre très strict de la sécurisation de systèmes électroniques. Ces composants manipulent en effet des données sensibles devant être gardées secrètes. Dans ce contexte, il est impératif de pouvoir garantir la sécurité des composants et plus précisément des différentes parties les constituant. Les générateurs de nombres aléatoires sont, entre autres, une des fonctions importantes de ces composants. Le caractère bien aléatoire des nombres qu'ils génèrent peut en effet être une condition nécessaire dans les preuves de sécurité de certains algorithmes de cryptographie. De manière similaire à ces algorithmes qui sont soumis à des cryptanalyses afin de trouver leurs failles, les RNGs doivent être étudiés d'un point de vue «attaquant».

Dans une première partie nous avons recensé les différents types de conceptions existantes des RNGs. Ces derniers sont classés dans deux grandes familles, une basée sur l'échantillonnage de bruit physique, l'autre sur des algorithmes déterministes. Dans chaque classe des mécanismes différents sont mis en jeu et ouvrent ainsi la voie à divers chemins d'attaques. De manière générale les propriétés des PRNGs ont été longuement étudiées, alors que pour les TRNGs le caractère aléatoire de la génération des nombres est plutôt assumé. Il existe donc peu d'études sur des attaques de TRNG.

Nous introduisons ensuite les outils nous permettant d'apprécier la qualité aléatoire des nombres issus de RNGs. Ceux-ci sont composés de tests statistiques et leur principe est de comparer une séquence de test à une séquence parfaite de nombres aléatoires. Durant nos expériences nous avons mis en évidence des limitations à ces tests, notamment le sens que l'on peut tirer de leurs résultats ou l'adéquation entre ces derniers et les caractéristiques réelles des nombres générés. Ces tests ne semblent dont pas parfaitement adaptés à nos besoins et il est donc nécessaire de les étudier plus profondément et de les améliorer.

Dans le chapitre suivant nous avons décrit des attaques existantes sur des systèmes cryptographiques ainsi que plus spécifiquement sur des RNGs. Nous nous sommes inspirés de ces travaux pour mettre en place nos expériences et pour décrire des exploitations possibles des perturbations ainsi produites. En effet notre but est de pouvoir mettre en défaut des systèmes cryptographiques par le biais de leurs RNGs.

Nous décrivons ensuite les bancs de test de TRNGs que nous avons mis en place. Ceux-ci peuvent être très simples et nécessiter peu de matériel, par exemple pour des variations de température. En revanche nous avons utilisé des bancs plus élaborés, comme le banc laser qui est utilisé par ailleurs pour des attaques classiques de cryptographie.

L'essentiel de nos travaux s'est porté sur le test de quatre TRNGs. Le but des manipulations concernant les TRNGs étaient de perturber leur génération de nombres afin que ces derniers ne suivent plus leur distribution théorique. Nous avons pour cela exploité les différents bancs de test mis en place.

Ces expériences nous ont permis, pour certaines, d'influencer le fonctionnement des TRNGs et de générer des biais dans la distribution des nombres produits. Certains retraitements censés corriger les déviations dues au bruit physique n'étaient pas adaptés pour prendre en compte de telles influences. Il semble que l'implantation des TRNGs se fasse parfois au détriment de la qualité de leur performance. La conception des TRNGs devrait mettre en œuvre plus de moyens pour étudier ses failles potentielles afin de les corriger au plus près de leur source. De même, les retraitements mériteraient d'être plus étudiés afin de ne pas sacrifier leur efficacité au profit de leur coût et rapidité. Les travaux effectués sur deux de ces TRNGs implantés dans des puces ont fait l'objet d'une présentation à HOST 2011 (IEEE International Symposium on Hardware-Oriented Security and Trust), et l'étude d'un troisième implanté dans un processeur Via Nano a été soumis à JETTA en 2012 (Journal of Electronic Testing - Theory and Applications). Par ailleurs cette dernière étude fera l'objet d'une présentation au forum Chip-to-Cloud ainsi qu'à YACC en septembre 2012.

Enfin nous avons étudié le fonctionnement d'un PRNG. Ce générateur s'apparente à un algorithme cryptographique et semble donc plus résistant aux perturbations. Ses propriétés lui ont en effet permis de complètement diffuser des déviations sur son entrée et nous avons mis en évidence un chemin permettant d'obtenir des informations sur les données qu'il manipule. Il est cependant facile à de remédier à ce problème.

Ce travail nous a permis de mettre en évidence des faiblesses potentiellement exploitables dans des TRNGs. Au delà du sujet de cette thèse nous avons pu rencontrer des problèmes connexes encore peu étudié et pourtant très importants dans ce domaine de la génération d'aléa. En premier lieu, la manière de tester les performances des TRNGs ne semble pas optimale et en tout cas laisse le doute s'installer quant à la véracité des conclusions que l'on en tire. Ensuite, l'exploitation de nombres aléatoires biaisés par des retraitements ou algorithmes cryptographiques a été peu étudiée bien qu'il soit très important de s'assurer de l'absence de failles dans un système où l'on cherche à mettre en place une sécurité absolue. Enfin, avec l'évolution des conceptions des RNGs il est nécessaire d'adapter de nouveaux chemins d'attaque.

Bibliographie

[ACGS84] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA/Rabin bits are 1/2 + 1 poly (log N) secure. *Foundations of Computer Science, Annual IEEE Symposium on*, 0 :449–457, 1984.

[ADN+10] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. When clocks fail : On critical paths and clock faults. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application*, volume 6035 of *Lecture Notes in Computer Science*, pages 182–193. Springer Berlin / Heidelberg, 2010.

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance : a cautionary note. In *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*, volume 2 of *WOEC'96*, pages 1–1. USENIX Association, 1996.

[AK98] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 125–136. Springer-Verlag, 1998.

[ANS98] Digital signatures using reversible public key cryptography for the financial services industry (rDSA), 1998.

[BBS86] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo random number generator. *SIAM Journal on Computing*, 15 :364–383, 1986.

[BDK+10] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In *Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 299–319. Springer-Verlag, 2010.

[BECN+04] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. Cryptology ePrint Archive, Report 2004/100, 2004.

[BGL+03a] Marco Bucci, Lucia Germani, Raimondo Luzzi, Pasquale Tommasino, Alessandro Trifiletti, and Mario Varanonuovo. A high-speed IC random-number source for smartcard microcontrollers. *IEEE Transactions on Circuits and Systems I : Fundamental Theory and Applications*, 50(11) :1373–1380, 2003.

[BGL+03b] Marco Bucci, Lucia Germani, Raimondo Luzzi, Alessandro Trifiletti, and Mario Varanonuovo. A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC. *IEEE Transactions on Computers*, 52(4) :403–409, 2003.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security : Advances in Cryptology*, ASIACRYPT '09, pages 1–18. Springer-Verlag, 2009.

[BK12] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators, 2012.

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '09, pages 231–249. Springer-Verlag, 2009.

[BLMT10] Mathieu Baudet, David Lubicz, Julien Micolod, and André Tassiaux. On the security of oscillator-based random number generators. *Journal of Cryptology*, pages 1–28, 2010.

[BNT10] Eric Brier, David Naccache, and Mehdi Tibouchi. Factoring unbalanced moduli with known bits. In *Proceedings of the 12th international conference on Information security and cryptology*, ICISC'09, pages 65–72. Springer-Verlag, 2010.

[Bou73] J. C. Bourgoin. Production des défauts par irradiation dans les semi-conducteurs. *J. Phys. Colloques*, 34(C5) :49–60, 1973.

[Boy89] Joan Boyar. Inferring sequences produced by pseudo-random number generators. *J. ACM*, 36 :129–141, 1989.

[BRC60] R.C. Bose and D.K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1) :68–79, 1960.

[Bur93] George W. Burns. *Temperature electromotive force reference functions and tables for the letter-designated thermocouple types based on the ITS-90*. NIST monograph ; 175. U.S. Dept. of Commerce, National Institute of Standards and Technology, 1993.

[CAAC08] C Cid, M Albrecht, D Augot, and A Canteaut. ECRYPT European network of excellence in cryptology algebraic cryptanalysis of symmetric primitives. *ReVision*, 2008.

[CH10] Yves Crama and Peter L. Hammer. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Cambridge University Press, 1st edition, 2010.

[CM03] Nicolas T. Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03, pages 345–359. Springer-Verlag, 2003.

[Com08] Industrial platinum resistance thermometers and platinum temperature sensors, 2008.

[Cop96] Don Coppersmith. Finding a small root of a bivariate integer equation ; factoring with high bits known. In *Proceedings of the 15th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'96, pages 178–189. Springer-Verlag, 1996.

[Cry03] Cryptography Research. Evaluation of VIA C3 Nehemiah RNG. `http://www.cryptography.com/public/pdf/VIA_rng.pdf`, 2003.

[Dav02] Robert B . Davies. Exclusive OR (XOR) and hardware random number generators. `http://www.robertnz.net/pdf/xor2.pdf`, 2002.

[DG07] Markus Dichtl and Jovan Dj. Golić. High-speed true random number generation with logic gates only. In *Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems*, CHES '07, pages 45–62. Springer-Verlag, 2007.

[Dic07] Markus Dichtl. Bad and good ways of post-processing biased physical random numbers. In *FSE'07*, pages 137–152, 2007.

[DIF94] Don Davis, Ross Ihaka, and Philip Fenstermacher. Cryptographic randomness from air turbulence in disk drives. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 114–120, 1994.

[DJ00] Markus Dichtl and Norbert Janssen. A high quality physical random number generator. In *Eurosmart Security Conference Proceedings*, 2000.

[Dub09] Tristan Dubois. *Etude de l'effet d'ondes électromagnétiques sur le fonctionnement de circuits électroniques – Mise en place d'une méthode de test des systèmes.* électronique, Université Montpellier 2, 2009.

[Fai10] Fail 0verflow. Console hacking 2010 : PS3 epic fail. In *27th Chaos Communication Congress*, December 2010.

[FD03] Viktor Fischer and Miloš Drutarovský. True random number generator embedded in reconfigurable hardware. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 415–430. Springer-Verlag, 2003.

[FIP00] Digital signature standard (DSS), January 2000.

[FIP01] Advanced Encryption Standard (AES), 2001.

[FIP02] Secure Hash Standard (SHS), 2002.

[FMR10] Jean-Charles Faugère, Raphaël Marinier, and Guénaël Renault. Implicit factoring with shared most significant and middle bits. In *Proceedings of the 13th international conference on Practice and Theory in Public Key Cryptography*, PKC'10, pages 70–87. Springer-Verlag, 2010.

[FMS01] Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, SAC '01, pages 1–24. Springer-Verlag, 2001.

[Fou90] P. Fouillat. *Contribution à l'étude de l'interaction entre un faisceau laser et un milieu semiconducteur, Applications à l'étude du Latchup et à l'analyse d'états logiques dans les circuits intégrés en technologie CMOS.* PhD thesis, Université de Bordeaux I, 1990.

[FS03] Niels Ferguson and Bruce Schneier. *Practical cryptography.* Wiley, 2003.

[FSS+06] M. Fiorentino, C ; M. Santori, S. M. Spillane, W. J. Munro, and R. G. Beausoleil. Secure self-calibrating quantum random bit generator. *Physical Review A*, 75(3) :5, 2006.

[FWN+10] Martin Fürst, Henning Weier, Sebastian Nauerth, Davide G. Marangon, Christian Kurtsiefer, and Harald Weinfurter. High speed optical quantum random number generation. *Opt. Express*, 18(12) :13029–13037, Jun 2010.

[GA03] Sudhakar Govindavajhala and Andrew W. Appel. Using memory errors to attack a virtual machine. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, page 154. IEEE Computer Society, 2003.

[Gen04] James E. Gentle. *Random Number Generation and Monte Carlo Methods.* Springer, 2nd edition, 2004.

[GFK+06] N. Gisin, S. Fasel, B. Kraus, H. Zbinden, and G. Ribordy. Trojan-horse attacks on quantum-key-distribution systems. *Phys. Rev. A*, 73 :022320, 2006.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis : Concrete results. In *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '01, pages 251–261. Springer-Verlag, 2001.

[Gol06] Jovan Dj. Golić. New methods for digital generation and postprocessing of random data. *IEEE Transactions on Computers*, 55(10) :1217–1229, 2006.

[Gol10] Hiroshi Julian Goldsmid. *Introduction to thermoelectricity*. Springer series in materials science ; 121. Springer, Heidelberg, 2010.

[Haa98] Mads Haahr. Random.org true random number service. `http://www.random.org`, 1998.

[Hab65] D. H. Habing. The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. *IEEE Transactions on Nuclear Science*, 12(5) :91–100, 1965.

[HCD97] W.Timothy Holman, J. Alvin Connelly, and Ahmad B. Dowlatabadi. An integrated analog/digital random noise source. *Circuits and Systems I : Fundamental Theory and Applications, IEEE Transactions on*, 44(6) :521–528, 1997.

[Hen12] Nadia Heninger. There's no need to panic over factorable keys - just mind your Ps and Qs. `https://freedom-to-tinker.com/blog/nadiah/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs`, 2012.

[HLWZ09] Yue Hu, Xiaofeng Liao, Kwok-wo Wong, and Qing Zhou. A true random number generator based on mouse movement and chaotic cryptography. *Chaos Solitons Fractals*, 40(5) :2286–2293, 2009.

[Hoc59] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, page 147–156, 1959.

[HQSMD+04] Ma Hai-Qiang, Wang Su-Mei, Zhang Da, Chang Jun-Tao, Ji Ling-Ling, Hou Yan-Xue, and Wu Ling-An. A random number generator based on quantum entangled photon pairs. *Chinese Physics Letters*, 21(10) :1961, 2004.

[ID 10a] ID Quantique. Quantis-oem application note. `http://www.idquantique.com/images/stories/PDF/quantis-random-generator/quantis-appnote.pdf`, 2010. v2.0.

[ID 10b] ID Quantique. Random number generation using quantum physics. `http://www.idquantique.com/images/stories/PDF/quantis-random-generator/quantis-whitepaper.pdf`, 2010.

[JAW+99] Thomas Jennewein, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, and Anton Zeilinger. A fast and compact quantum random number generator. *Review of Scientific Instruments*, 71(4) :23, 1999.

[JK99] Benjamin Jun and Paul Kocher. The intel random number generator. `http://www.cryptography.com/public/pdf/IntelRNG.pdf`, 1999.

[Joh28] J. B. Johnson. Thermal agitation of electricity in conductors. *Physical Review*, 32(1) :97, 1928.

[JSHJ98] Markus Jakobsson, Elizabeth Shriver, Bruce K. Hillyer, and Ari Juels. A practical secure physical random bit generator. In *Proceedings of the 5th ACM conference on Computer and communications security*, CCS '98, pages 103–111. ACM, 1998.

[Kel05] Sharon S. Keller. NIST-recommended random number generator based on ANSI X9.31 appendix A.2.4 using the 3-key triple DES and AES algorithms, 2005.

[KG04] Paul Kohlbrenner and Kris Gaj. An embedded true random number generator for FPGAs. In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, FPGA '04, pages 71–78. ACM, 2004.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397. Springer-Verlag, 1999.

[KK99] Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, WOST'99, pages 2–2. USENIX Association, 1999.

[Knu97] Donald E. Knuth. *The art of computer programming : seminumerical algorithms*, volume 2. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113. Springer-Verlag, 1996.

[KR07] Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In *Proceedings of the Advances in Crypotology 13th international conference on Theory and application of cryptology and information security*, ASIACRYPT'07, pages 315–324. Springer-Verlag, 2007.

[KS08] Wolfgang Killmann and Werner Schindler. A design for a physical RNG with robust entropy estimators. In *Proceedings of the 10th international workshop on Cryptographic Hardware and Embedded Systems*, CHES '08, pages 146—-163. Springer-Verlag, 2008.

[Leh51] D. H. Lehmer. Mathematical methods in large-scale computing units. In Mass. : Harvard University Press Cambridge, editor, *Proc. Sec. Symp. on Large-scale Digital Calculating Machinery*, pages 141–146, 1951.

[LHA+12] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, whit is right. Cryptology ePrint Archive, Report 2012/064, 2012.

[LM05] Chengxin Liu and John McNeill. A digital-pll-based true random number generator. In *Proc. PhD Research in Microelectronics and Electronics*, volume 1, pages 113–116, 2005.

[LS07] Pierre L'Ecuyer and Richard Simard. TestU01 : A c library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 33, 2007.

[LWW+10a] Lars Lydersen, Carlos Wiechers, Christoffer Wittmann, Dominique Elser, Johannes Skaar, and Vadim Makarov. Hacking commercial quantum cryptography systems by tailored bright illumination. *Nature Photonics*, 4 :686–689, 2010.

[LWW+10b] Lars Lydersen, Carlos Wiechers, Christoffer Wittmann, Dominique Elser, Johannes Skaar, and Vadim Makarov. Thermal blinding of gated detectors in quantum cryptography. *Opt. Express*, 18(26) :27938–27954, 2010.

[Mak09] Vadim Makarov. Controlling passively quenched single photon detectors by bright light. *New Journal of Physics*, 11(6) :065003, 2009.

[Mar95] George Marsaglia. The Marsaglia random number CDROM including the Diehard battery of tests. `http://stat.fsu.edu/pub/diehard/`, 1995.

[Mas69] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1) :122–127, 1969.

[MM09] A. Theodore Markettos and Simon W. Moore. The frequency injection attack on ring-oscillator-based true random number generators. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '09, pages 317–331. Springer-Verlag, 2009.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170) :519–521, 1985.

[MR09] Alexander May and Maike Ritzenhofen. Implicit factoring : On polynomial time factoring given only an implicit hint. In *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography : PKC '09*, Irvine, pages 1–14. Springer-Verlag, 2009.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., 1st edition, 1996.

[MW78] Timothy C. May and Murray H. Woods. A new physical mechanism for soft errors in dynamic memories. In *Proc. 16th Annual Reliability Physics Symp*, pages 33–40, 1978.

[NCP01] Landon Curt Noll, Simon Cooper, and Mel Pleasant. Lavarnd. `http://www.lavarnd.org`, 2001.

[NIS01] Security requirements for cryptographic modules, May 2001.

[NS00] Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15 :151–176, 2000.

[NS03] Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptography*, 30(2) :201–217, 2003.

[Nyq28] H. Nyquist. Thermal agitation of electric charge in conductors. *Physical Review*, 32(1) :110, 1928.

[PC00] Craig S. Petrie and J. Alvin Connelly. A noise-based IC random number generator for applications in cryptography. *Circuits and Systems I : Fundamental Theory and Applications, IEEE Transactions on*, 47(5) :615—-621, 2000.

[Pou00] V. Pouget. *Simulation expérimentale par impulsions laser ultra-courtes des effets des radiations ionisantes sur les circuits intégrés*. PhD thesis, Université de Bordeaux I, 2000.

[PTL+11] F. Poucheret, K. Tobich, M. Lisart, L. Chusseau, B. Robisson, and P. Maurine. Local and direct EM injection of power into CMOS integrated circuits. In *FDTC'11*, pages 100–104, 2011.

[QS00] J-J. Quisquater and D. Samyde. A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions, the sema and dema methods. 2000. Presented at the rump session of EUROCRYPT'2000.

[RG07] Gregoire Ribordy and Olivie Guinnard. Method and apparatus for generating true random numbers by way of a quantum optics process, 2007.

[RS86] R L Rivest and A Shamir. Efficient factoring based on partial information. In *Proc. of a workshop on the theory and application of cryptographic techniques on Advances in cryptology—EUROCRYPT '85*, pages 31–34. Springer-Verlag, 1986.

[RSN+01] Andrew Rukhin, Juan Soto, James Nechvatal, Elaine Barker, Stefan Leigh, Mark Levenson, David Banks, Alan Heckert, James Dray, San Vo, Andrew Rukhin, Juan Soto, Miles Smid, Stefan Leigh, Mark Vangel, Alan Heckert, James Dray, and Lawrence E. Bassham

III. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2001.

[SA03] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 2–12. Springer-Verlag, 2003.

[ŠDF11] Martin Šimka, Miloš Drutarovský, and Viktor Fischer. Testing of pll-based true random number generator in changing working conditions. *Radioengineering*, 20(1) :94–101, 2011.

[SDFF06] M. Simka, M. Drutarovsky, V. Fischer, and J. Fayolle. Model of a true random number generator aimed at cryptographic applications. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, 2006.

[Sel66] Ernst S. Selmer. *Linear recurrence relations over finite fields.* Department of Mathematics, University of Bergen, 1966.

[SGG+00] André Stefanov, Nicolas Gisin, Olivier Guinnard, Laurent Guinnard, and Hugo Zbinden. Optical quantum random number generator. *Journal of Modern Optics*, 47 :595–598, 2000.

[Sie84] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications (corresp.). *IEEE Transactions on Information Theory*, 30(5) :776–780, 1984.

[SK03] Werner Schindler and Wolfgang Killmann. Evaluation criteria for true (physical) random number generators used in cryptographic applications. In Burton Kaliski, Cetin KoC, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 431–449. Springer Berlin / Heidelberg, 2003.

[Sko05] Sergei P. Skorobogatov. *Semi-invasive attacks - A new approach to hardware security analysis.* PhD thesis, University of Cambridge, 2005.

[SM09] Santanu Sarkar and Subhamoy Maitra. Further results on implicit factoring in polynomial time. *Advances in Mathematics of Communications*, 3(2) :205–217, 2009.

[SMS07] B. Sunar, W. J. Martin, and D. R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers*, 56(1) :109–119, 2007.

[SSAQ02] David Samyde, Sergei Skorobogatov, Ross Anderson, and Jean-Jacques Quisquater. On a new way to read data from memory. In *Proceedings of the First International IEEE Security in Storage Workshop*, SISW '02, page 65. IEEE Computer Society, 2002.

[SSR09] Renaud Santoro, Olivier Sentieys, and Sébastien Roy. On-the-fly evaluation of FPGA-based true random number generator. In *VLSI, 2009. ISVLSI '09. IEEE Computer Society Annual Symposium on*, pages 55–60, 2009.

[Sze81] Simon M. Sze. *Physics of Semiconductor Devices.* John Wiley & Sons, 2nd edition, 1981.

[TBDSL01] Elena Trichina, Marco Bucci, Domenico De Seta, and Raimondo Luzzi. Supplemental cryptographic hardware for smart cards. *IEEE Micro*, 21 :26–35, 2001.

[Ter04] Terrazon Semiconductor. Soft errors in electronic memory. `http://www.tezzaron.com/about/papers/soft_errors_1_1_secure.pdf`, 2004.

[TLL03] K. H. Tsoi, K. H. Leung, and P. H. W. Leong. Compact fpga-based true and pseudo random number generators. In *Proc. 11th Annual IEEE Symp. Field-Programmable Custom Computing Machines FCCM 2003*, pages 51–61, 2003.

[UNS00] Sources and effects of ionizing radiation. Report to the General Assembly Volume 1, United Nations Scientific Committee on the Effects of Atomic Radiation, 2000.

[VHKK08] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. Fast digital TRNG based on metastable ring oscillator. In *Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems*, CHES '08, pages 164–180. Springer-Verlag, 2008.

[VIA08] VIA. Padlock quick reference, 2008.

[VIA09] VIA Padlock software development kit. `http://www.viaarena.com/Download/PadlockSDK_3.1_Release_20090121.zip`, 2009.

[VMH01] Artem Vakhitov, Vadim Makarov, and Dag R. Hjelme. Large pulse attack as a method of conventional optical eavesdropping in quantum cryptography. *Journal of Modern Optics*, 48(13) :2023–2038, 2001.

[vN63] John von Neumann. Various techniques used in connection with random digits. *Pergamon Press*, Collected Works(5) :758–770, 1963.

[VV84] U. V. Vazirani and V. V. Vazirani. Efficient and secure pseudo-random number generation. In *Proc. 25th Annual Symp. Foundations of Computer Science*, pages 458–463, 1984.

[Wal96] John Walker. HotBits : Genuine random numbers, generated by radioactive decay. `https://www.fourmilab.ch/hotbits/`, 1996.

[Wal08] John Walker. Pseudorandom number sequence test program. `http://www.fourmilab.ch/random/`, 2008.

[WG09] Wei Wei and Hong Guo. Bias-free true random-number generator. *Opt. Lett.*, 34(12) :1876–1878, 2009.

[WK10] Michael A. Wayne and Paul G. Kwiat. Low-bias high-speed quantum random number generator via shaped optical pulses. *Opt. Express*, 18(9) :9351–9357, 2010.

[WT08] Knut Wold and Chik How Tan. Analysis and enhancement of random number generator in fpga based on oscillator rings. In *Proc. Int. Conf. Reconfigurable Computing and FPGAs ReConFig '08*, pages 385–390, 2008.

[WZSL05] Yu-Hua Wang, Huan-Guo Zhang, Zhi-Dong Shen, and Kang-Shun Li. Thermal noise random number generator based on SHA-2 (512). In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 7, pages 3970—-3974, 2005.

[YKBS10] Sang-Kyung Yoo, Deniz Karakoyunlu, Berk Birand, and Berk Sunar. Improving the robustness of ring oscillator trngs. *ACM Trans. Reconfigurable Technol. Syst.*, 3 :9 :1–9 :30, May 2010.

[ZH01] Huang Zhun and Chen Hongyi. A truly random number generator based on thermal noise. In *ASIC, 2001. Proceedings. 4th International Conference on*, pages 862—-864, 2001.

[ZL79] J. F. Ziegler and W. A. Lanford. Effect of cosmic rays on computer memories. *Science*, 206(4420) :776–788, 1979.

Résumé

Les nombres aléatoires ont été de tous temps utilisés pour des jeux de hasard, plus récemment pour créer des codes secrets et ils sont aujourd'hui nécessaire à l'exécution de programmes informatiques. Les générateurs de nombres aléatoires sont maintenant bien éloignés de simples dés à lancer et sont constitués de circuits électroniques ou d'algorithmes. Ceci pose des problèmes quant à la reconnaissance du caractère aléatoire des nombres générés. De plus, de la même manière ou autrefois les dés étaient pipés pour augmenter les chances de gagner, il est aujourd'hui possible d'influencer la sortie des générateurs de nombres aléatoires.

Ce sujet est donc toujours d'actualité avec des exemples récents très médiatisés. Ceci concernait en effet la console de jeu PS3 qui génère un nombre aléatoire constant où la distribution de clefs secrètes redondantes sur internet.

Ce mémoire présente l'étude de plusieurs générateurs ainsi que diverses manières de les perturber. Il montre ainsi des faiblesses inhérentes à leurs conceptions et des conséquences possibles de leur défaillance sur des composants de sécurité. Ces travaux ont de plus permis de mettre en évidence l'importance des problématiques concernant le test des nombres aléatoires ainsi que des retraitements corrigeant des biais dans ces nombres.

Mots-clés: Générateur de Nombres Aléatoires • Tests statistiques • Perturbation • Température • Laser • Injection de fréquence • Rayonnements ionisants

Abstract

Random numbers have been used through the ages for games of chance, more recently for secret codes and today they are necessary to the execution of computer programs. Random number generators have now evolved from simple dices to electronic circuits and algorithms. Accordingly, the ability to distinguish between random and non-random numbers has become more difficult. Furthemore, whereas in the past dices were loaded in order to increase winning chances, it is now possible to influence the outcome of random number generators.

In consequence, this subject is still very much an issue and has recently made the headlines. Indeed, there was talks about the PS3 game console which generates constant random numbers and redundant distribution of secret keys on the internet.

This thesis presents a study of several generators as well as different means to perturb them. It shows the inherent defects of their conceptions and possible consequences of their failure when they are embedded inside security components. Moreover, this work highlights problems yet to be solved concerning the testing of random numbers and the post-processing eliminating bias in these numbers distribution.

Keywords: Random Number Generator • Statistical tests • Perturbation • Temperature • Laser • Frequency injection • Ionising radiations