Faculté des Sciences
Aix*Marseille Université

RAPPORT DE STAGE

Élaboration d'outils statistiques pour la détection de chevaux de Troie matériels

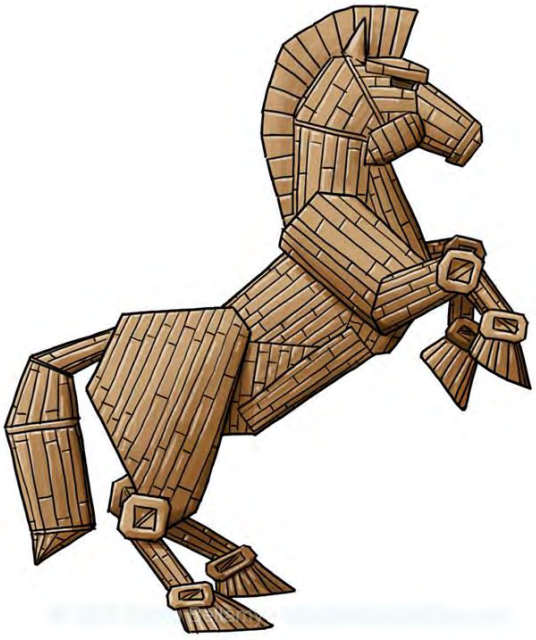


Table des matières

Introduction

Nous vivons dans un monde où la cryptographie est omniprésente pour l'échange de données informatiques.

L'assurance que ces données cryptées ne puissent être lues uniquement par le destinataire est critique, en particulier dans le milieu bancaire.

Même si les algorithmes mathématiques sont extrêmement fiables, il existe toujours un risque dans leur implémentation, logicielle ou matérielle au sein des puces électroniques.

L'une des attaques possibles est la modification matérielle de ces puces de cryptage, afin de contourner l'algorithme mathématique et de récupérer les données secrètes. Ce type d'attaque est appelé cheval de Troie matériel.

Une des missions de recherche du laboratoire S.A.S. est de développer des méthodes de détection de ces chevaux de Troie matériels.

Pour cela les chercheurs utilisent des circuits logiques reprogrammables (FPGA) et modélisent sur ces circuits des chevaux de Troie pour par la suite tenter de les détecter.

Ce stage en laboratoire de recherche entre dans le cadre du Cursus Master Ingénierie (CMI), ma mission a été de mettre en place des outils statistiques dans un programme informatique, permettant la détection de ces chevaux de Troie matériels.

Ce rapport présentera dans un premier temps les différents types d'attaques possibles sur des circuits intégrés (puce) puis les différentes méthodes de détection des chevaux de Troie matériels.

Et enfin il présentera ma mission et le travail que j'ai réalisé.

II. Présentation des chevaux de Troie matériels

Qu'est-ce qu'un circuit intégré ?

Un circuit intégré (ou puce) est un circuit contenant différents composants: transistors, diodes, condensateurs, résistances, gravés sur un morceau de silicium selon une configuration donnée, pour répondre à un besoin spécifique.

Figure 1 - Exemple d'un circuit intégré : une carte a puce

2.1 Les différents types d'attaques physiques

Des attaques physiques sont possibles sur un circuit intégré dans le but de comprendre son fonctionnement interne et tenter de récupérer une information comme une clé secrète par exemple.

Il existe différents types d'attaques physiques :

- **Inspection physique**

En ouvrant le boitier de la puce, il est possible de faire de la rétro-conception et d'étudier la structure de la puce à l'aide d'outils d'imageries.

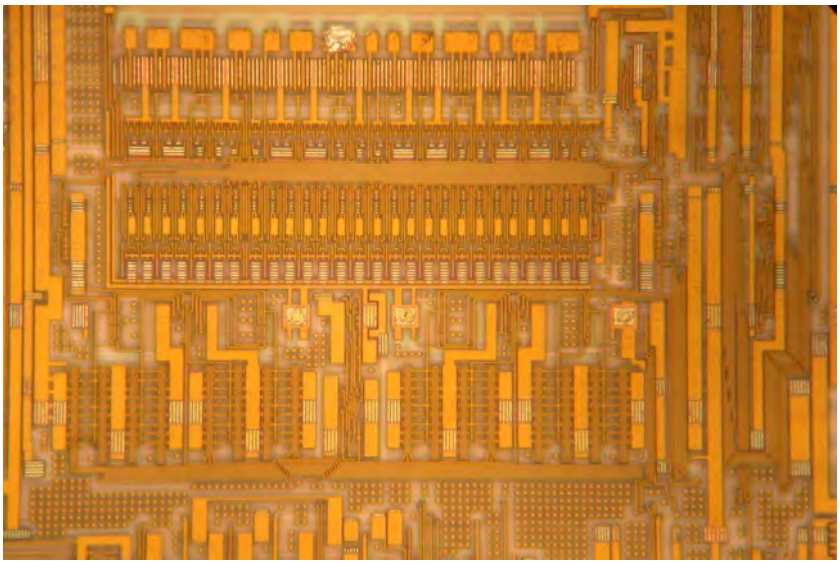


Figure 2 - Photo du circuit intégré à l'intérieur d'une puce

- **Observation du comportement (attaque par canal auxiliaire)**

Une attaque par canal auxiliaire désigne une attaque qui, sans remettre en cause la robustesse théorique des méthodes et procédures de sécurité, recherche et exploite des vulnérabilités dans leur implémentation, logicielle ou matérielle. En effet, une sécurité « mathématique » ne garantit pas forcément une sécurité lors de l'utilisation « en pratique ».

En observant le comportement de la puce, par exemple le temps de réponse, la consommation électrique, l'émission électromagnétique, il est possible de déduire l'opération en cours et les données traitées par cette opération (nommé fonction de fuite de donnée) car ces paramètres physiques sont corrélés a l'action que la puce est en train d'effectuer.

- **Injection de fautes**

En utilisant des perturbations, par exemple des impulsions lasers, des impulsions électromagnétiques, des modifications de la fréquence d'horloge de la puce ou de sa tension d'alimentation, il est possible de modifier l'opération en cours et les données traitées par cette opération (nommé fonction d'erreur de donnée).

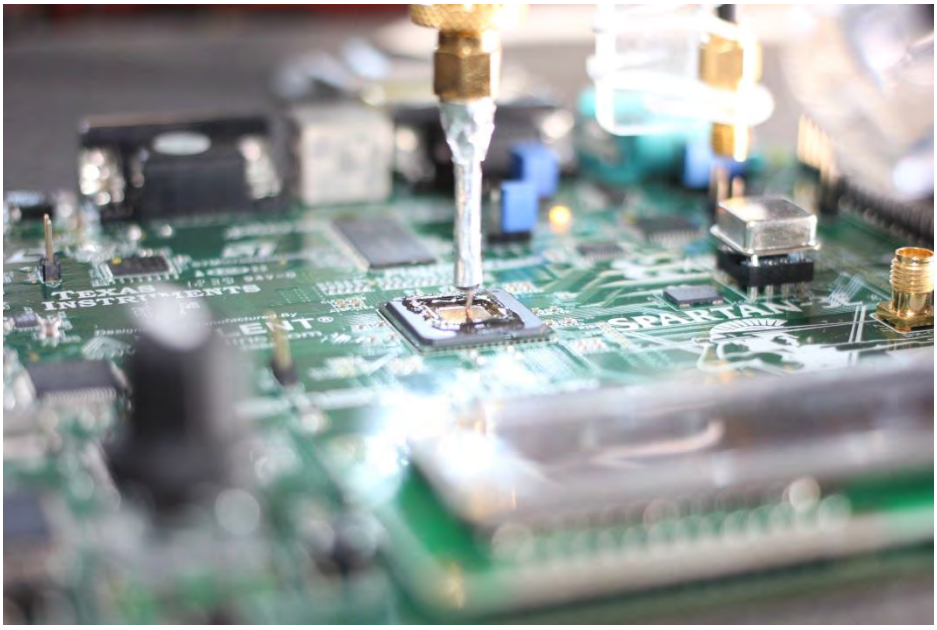


Figure 3 - Photo d'une sonde électromagnétique au-dessus d'une puce décapsulée

Il est même possible en forçant des données connues en entrée d'un système interne au circuit intégré, en lisant les données de sortie, de déduire le traitement qu'effectue ce système.

2.2 Les chevaux de Troie matériels (Hardware Trojan)

En informatique, le cheval de Troie est un logiciel en apparence légitime, mais qui contient une malveillance. Le rôle du cheval de Troie est de faire entrer ce parasite sur l'ordinateur et de l'y installer à l'insu de l'utilisateur.

Dans notre cas, ce cheval de Troie est matériel, c'est-à-dire que ce n'est pas une entité logicielle installée mais une modification malveillante de la structure physique du circuit intégré.

Ce cheval de Troie matériel peut servir à transmettre des informations confidentielles telles que les clefs de chiffrement par exemple. Il peut aussi changer le fonctionnement de l'appareil visé ou le rendre inutilisable pour en tirer profit (déni de service).

De nos jours, à cause de la délocalisation de la fabrication des circuit intégrés, il ne s'agit pas des même entités qui conçoivent et fabriquent un circuit intégré. Par exemple une entreprise de conception sans usine (fabless) européenne peut concevoir un circuit et l'envoyer pour fabrication en Asie, il se peut donc que des personnes mal intentionnée modifient le tracé du circuit qu'on leur envoie et y intègre un cheval de Troie.

2.3 Détection des chevaux de Troie matériels

Il est possible de détecter les chevaux de Troie matériels, voici les différentes techniques :

- **Inspection physique**

Il s'agit d'ouvrir le boitier de la puce et d'utiliser des techniques d'imagerie pour étudier la structure du circuit intégré et tenter de repérer le cheval de Troie matériel.

Ces techniques peuvent êtres couteuse et sont destructives pour le circuit intégré.

- **Auto test intégré (Built in self-test)**

Ce sont des tests exécutés par le programme implanté dans la puce elle-même qui permettent de vérifier son bon fonctionnement. Ils testent pour cela toutes les fonctions de la puce, comme les ports d'entrée/sortie et vérifient les valeurs de sortie.

- **Analyse par canal auxiliaire**

Il s'agit de mesurer la réponse aux tests décrits dans le paragraphe 2.1 (Observation du comportement) d'abord pour une puce que nous savons saine, que nous appellerons puce Golden, puis de recommencer pour une puce similaire dont nous souhaitons tester la présence d'un cheval de Troie matériel. Ceci permettra alors d'établir des différences, qui, si elles sont suffisamment importantes, conduirons à la conclusion que la puce cible est infectée.

La difficulté est de savoir si ces différences sont réellement dues au cheval de Troie, ou sont une conséquence des paramètres environnementaux de la mesure.

Pour cela, on cherche à mettre en place ce que l'on appelle un distingueur dont le rôle est d'attester de la présence ou non d'un cheval de Troie matériel.

Un distingueur pour un cheval de Troie matériel est un outil statistique déterminant avec une forte probabilité la présence ou non de celui-ci.

III. Activité réalisée

3.1 Introduction

Ma mission a été d'implémenter différents tests statistiques dans un programme en langage C, permettant de distinguer les différences entre deux mesures de consommation électrique faites sur une puce cible et une puce golden comme vu dans le paragraphe 2.1 (Observation du comportement).

Ces deux puces sont des FPGA (circuit logique reprogrammable, utile dans le cas d'une expérimentation en laboratoire) programmées dans le but de réaliser un cryptage de données.

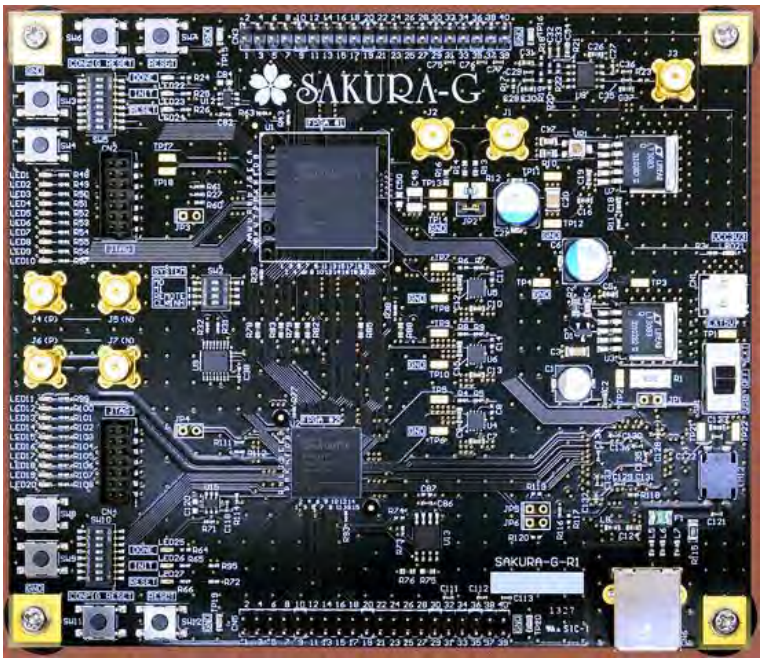


Figure 4 - Photo de la carte FPGA sur laquelle sont effectuées les mesures de consommation

3.2 Tests statistiques

3.2.1 Présentation

Nous cherchons à savoir si une puce est infectée par un cheval de Troie matériel. Pour cela nous utilisons une seconde puce que nous savons saine (puce Golden), et nous lançons une même opération de cryptage successivement sur ces deux cartes (la cible et la Golden).

Durant cette opération les consommations électriques des deux cartes sont surveillées et enregistrées.

Cette mesure est effectuée un grand nombre de fois (1000 fois par exemple) pour réduire les erreurs possibles.

La consommation électrique est relevée par un oscilloscope numérique, connecté à un ordinateur qui enregistre les données.

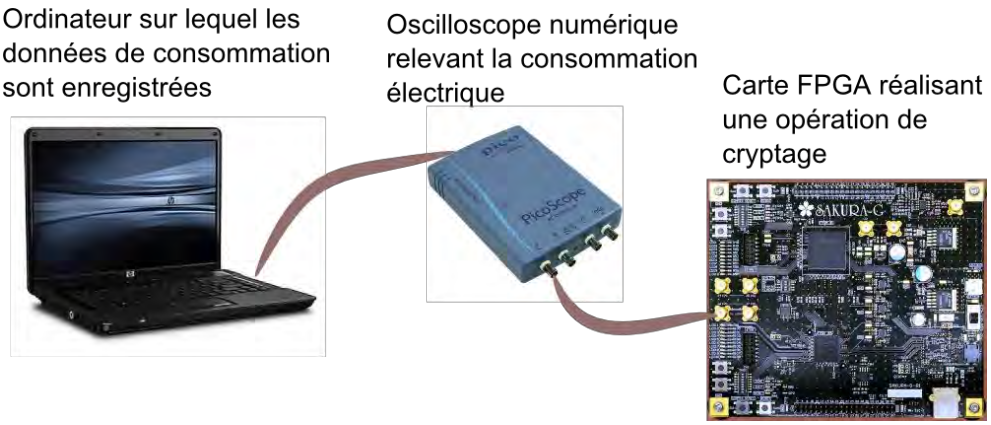


Figure 5 - Schéma de l'expérience

Plus tard, nous appliquerons différents tests statistiques sur ces données pour essayer d'en déduire s'il y a présence d'un cheval de Troie ou non. En effet, une carte saine et une carte infectée ont des réponses différentes à certain tests, ce qui permet de les différencier.

Qu'est-ce qu'un test statistique ?

Un test statistique est un test permettant d'accepter ou de rejeter une hypothèse H0 avec une probabilité α de se tromper.

Il faut alors établir une hypothèse alternative H1 dans le cas où notre hypothèse H0 est rejetée, il s'agit généralement du contraire de notre hypothèse.

Dans notre cas, ces hypothèses sont les suivantes :

- H0 : les deux courbes de mesure se correspondent (donc la carte cible est identique à la golden)
- H1 : les deux courbes de mesure ne se correspondent pas (donc la carte cible est différente de la golden)

Quatre issues sont possibles :

- On accepte H0 et H0 est vraie.
- On rejette H0 et H0 est fausse.
- On accepte H0 alors que H0 est fausse : c'est une erreur de première espèce (notée α).
- On rejette H0 alors que H0 est vraie : c'est une erreur de seconde espèce (notée β).

Nous aurons le choix entre un test paramétrique ou un test non paramétrique. Les tests paramétriques sont généralement plus performants mais ils requièrent des contraintes au niveau des conditions initiales pour être appliqués.

3.2.1 Test de Student (T-test)

a) Théorie

Le T-test est un test paramétrique qui permet de vérifier l'égalité de la moyenne de deux échantillons suivant une loi normale. Pour cela, on calcule un rapport correspondant à la variation des différences par rapport à la moyenne.

Les conditions pour passer ce test sont que les courbes suivent une distribution normale (gaussienne) et que leurs variances soient égales.

Déroulement du test pour deux groupes de n-échantillons :

- On calcule les différences d_i pour chaque valeur des deux n-échantillons.
- On calcule la somme des carrés d'erreur (SCE) : $SCE = \sum_{i=1}^{n} {d_i}^2 - \frac{(\sum_{i=1}^{n} d_i)^2}{n}$
- On calcule t, la valeur statistique du t-test : $t = \frac{\frac{\sum_{i=1}^{n} d_i}{n}}{\sqrt{\frac{SCE}{n(n-1)}}}$
- On cherche dans la table de Student la valeur critique $t_{critique}$ de t pour un degré de liberté $ddl = n - 1$, pour α choisi (1% ou 5% le plus souvent).
- Si $|t| > t_{critique}$ alors on peut rejeter l'hypothèse H0 avec un seuil de confiance α.

b) Mise en œuvre

La mise en œuvre du test de Student se fait en langage C, avec l'aide de la bibliothèque de fonctions mathématiques open source « GSL ».

La première étape est celle du chargement des données à analyser. L'oscilloscope numérique fournit des fichiers texte avec une mesure de tension par ligne, notre premier objectif est de recopier toutes les mesures dans une grande matrice. Cette opération doit être réalisée pour les deux groupes de courbes (cible et golden).

Dès que les courbes de données sont importées dans notre programme, nous pouvons travailler dessus et mettre en œuvre les calculs exposés dans la partie « a) Théorie ».

Ces calculs se font avec les fonctions GSL « gsl_vector_sub » pour la soustraction, « gsl_vector_add » pour l'addition, « gsl_vector_mul » pour le produit vectoriel et « gsl_vector_scale » pour le produit scalaire. Toutes ces fonctions agissent directement sur des tableaux d'une dimension (vecteurs), on peut alors réaliser ces aditions pour tous les points de ces tableaux avec une seule ligne de code.

La figure ci-dessous est une représentation logique des actions qu'effectue le programme sur les deux matrices contenant les mesures de la cible et de la golden.

Sur chaque ligne de ces matrices il y a une courbe de mesure dont les points (échantillons) sont placés dans la largeur de ces matrices.

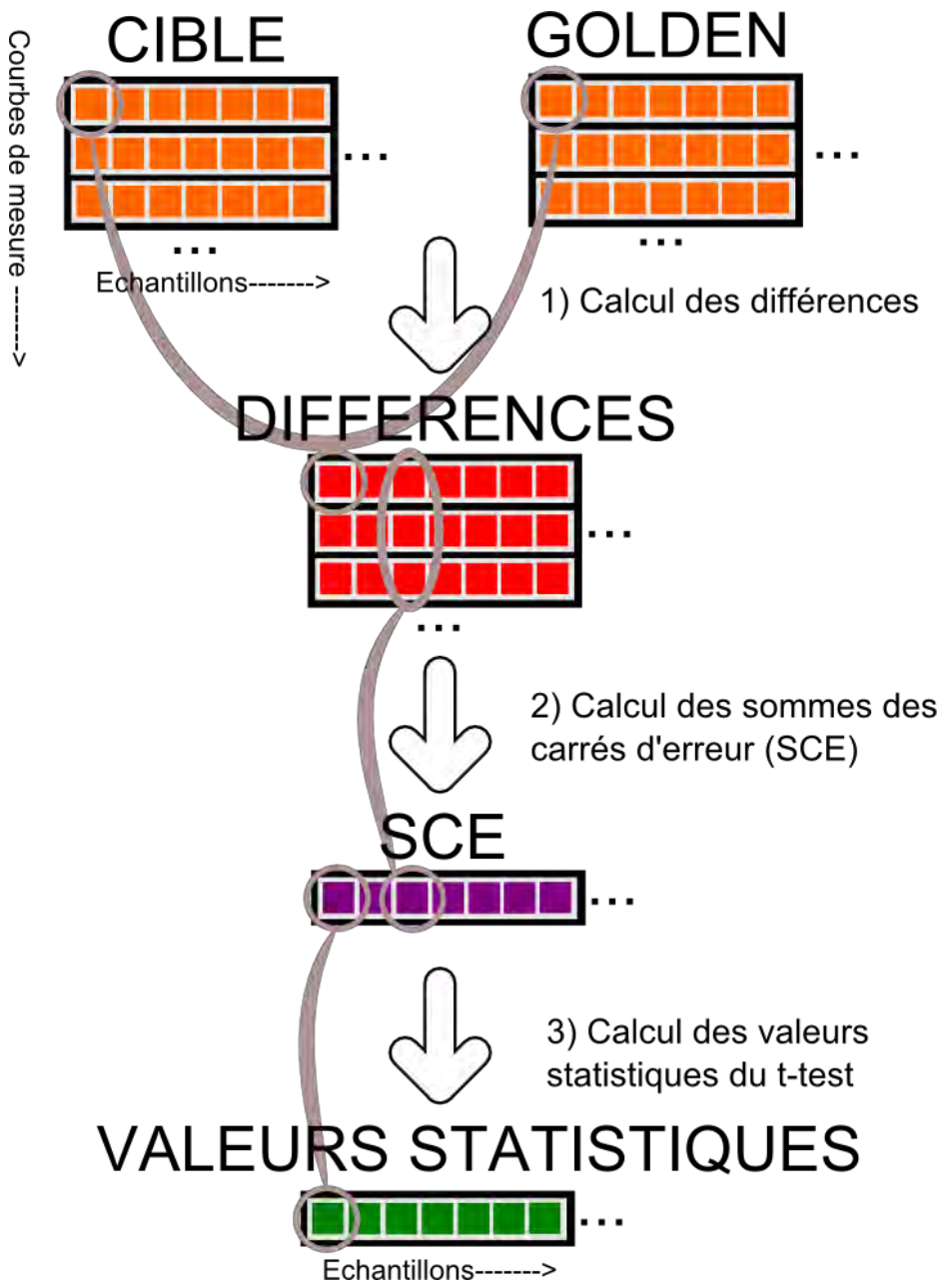


Figure 6 - Schéma logique du test de Student (T-test)

Pour conclure ce test, nous comptons le nombre d'échantillons dont la valeur t respecte $|t| < 1{,}96$. Il s'agit de la valeur critique à 5 % pour un nombre de degré de liberté infini.

La dernière étape consiste à tracer sur un graphe, les valeurs statistiques pour tous les échantillons.

Pour tracer cette courbe nous utilisons le programme GNU Plot, voici un exemple de résultat du test de Student :

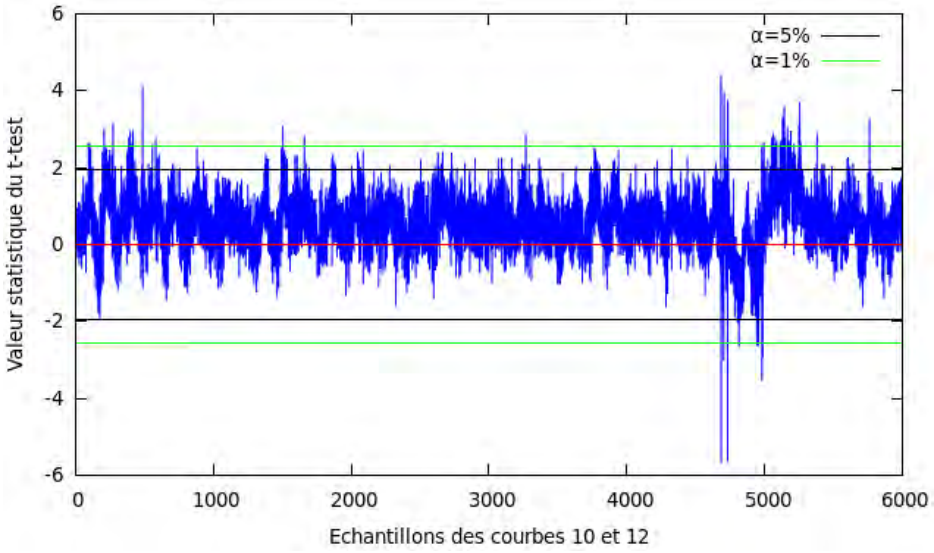


Figure 7 - Courbe résultat du test de Student (T-test)

A partir de ce graphe et de l'information du pourcentage d'échantillons étant dans notre intervalle critique à 5%, nous pouvons nous prononcer sur la similitude de la carte cible avec la carte golden et donc savoir si elle pourrait contenir un cheval de Troie matériel.

3.2.2 Vérification de la variance

a) Théorie

Comme expliqué précédemment, pour effectuer le test de Student (ou T-test), il est nécessaire que les séries de mesures que nous comparons aient des variances proches.

La variance est une mesure servant à caractériser la dispersion d'un échantillon. Elle indique de quelle manière la série statistique se disperse autour de sa moyenne. Une variance nulle signale que toutes les valeurs sont identiques. Une petite variance montre que les valeurs sont proches les unes des autres alors qu'une variance élevée montre que celles-ci sont très écartées.

Pour effectuer cette vérification, nous calculons les variances de chaque groupe des n-ièmes échantillons, et ce, pour les deux groupes (cible et golden). Puis nous effectuons le rapport entre chaque variance cible et chaque variance golden. Nous avons donc au final n rapports de variance (n étant le nombre de points par courbe).

Nous considérons que les variances de deux groupes d'échantillons sont proches lorsque leur rapport est compris dans l'intervalle [0,90 ; 1,10]. Nous comptons ensuite le nombre de groupe d'échantillons dont la variance est proche.

b) Mise en œuvre

Comme pour le test de Student, la première étape est celle du chargement des données à analyser. Notre premier objectif est de recopier tous les points de tous les fichiers texte correspondant chacun à une mesure, dans une grande matrice. Cette opération doit être réalisée pour les deux groupes de courbes (cible et golden).

La quantité de donnée pouvant être grande (jusqu'à 2Go dans mes expérimentations) il est nécessaire d'utiliser une fonction C particulière : la fonction malloc. Cette fonction permet d'allouer de la mémoire vive (RAM) dynamiquement et de créer des variables dont la taille n'est pas connue avant l'exécution du programme.

Nous créons donc deux matrices : MesureGolden, contenant les mesures de la carte considérée comme saine et MesureCible, contenant les mesures de la carte que nous testons.

Nous calculons ensuite la variance de chaque colonne (chaque groupe des n-ièmes échantillons) de ces deux matrices, de façon à obtenir deux vecteurs contenant les variances de chacun des échantillons. Le calcul de variance se fait avec la fonction « gsl_stats_variance » de la bibliothèque GSL.

Finalement, pour comparer les puces (cible et golden) nous effectuons le rapport de la variance des mesures venant de la carte cible sur la variance des mesures de la carte Golden.

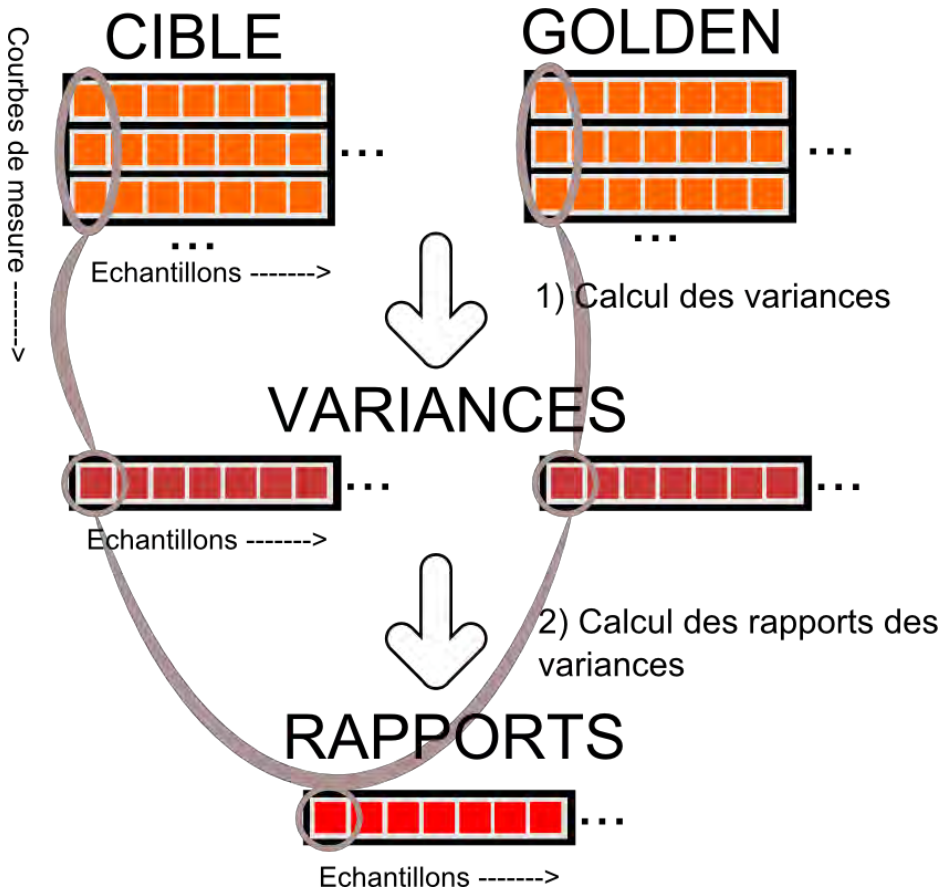


Figure 8 - Schéma logique de la vérification de la variance

Si le rapport des variances est contenu dans l'intervalle [0,90 ; 1,10] alors nous considérons que les variances sont proches et que les tests paramétriques sont possibles sur ces données.

Nous comptons également le nombre d'échantillons dont le rapport de variance respecte cette condition.

La dernière étape consiste à tracer sur un graphe, tous les rapports de variances.

Pour tracer cette courbe nous utilisons le programme GNU Plot, voici un exemple de résultat du test de variance :

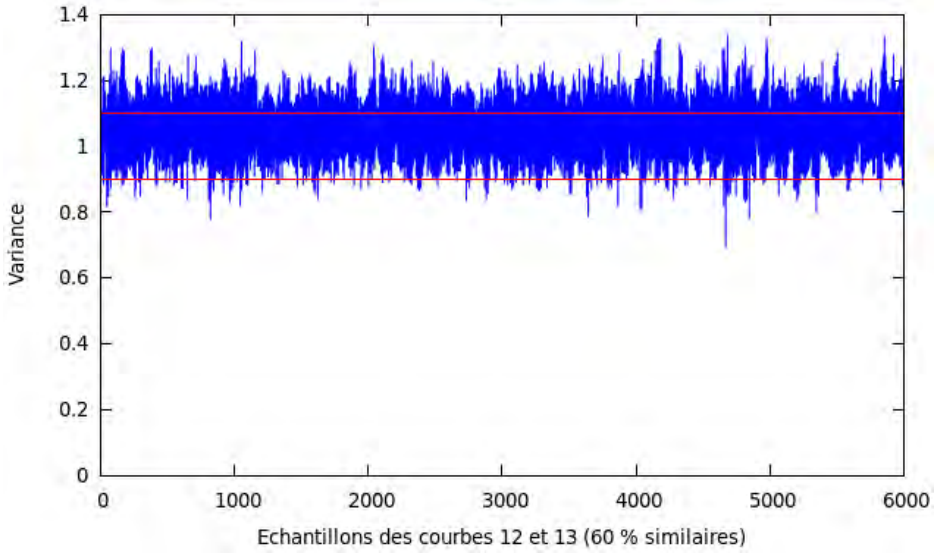


Figure 9 - Courbe résultat du test de variance

3.2.3 Vérification de la normalité

a) Théorie

Pour effectuer le test de Student (ou T-test), il est nécessaire que les échantillons des mesures que nous comparons suivent une loi gaussienne.

Une fonction gaussienne est une fonction en exponentielle de l'opposé du carré de l'abscisse (une fonction en $\exp(-x^2)$). Elle a une forme caractéristique de courbe en cloche.

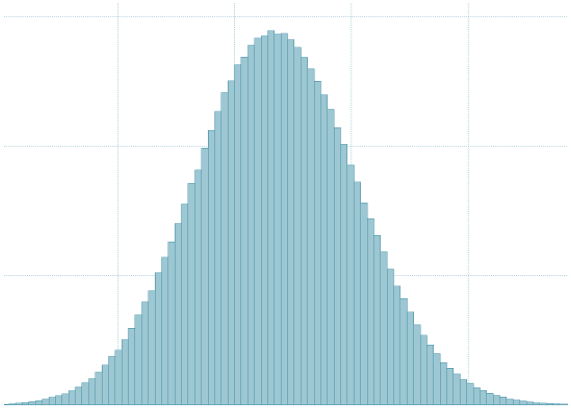


Figure 10 - Allure de la distribution normale (gaussienne)

Pour tester qu'une série soit bien de forme gaussienne, nous allons utiliser le test de Jarque Bera. Ce test se base sur deux caractéristiques d'une fonction gaussienne :

- Une symétrie (Skew)
- Un aplatissement (Kurtosis)

Le paramètre de skew est nul pour une loi normale standard (centrée réduite), à l'opposé un skew trop négatif ou trop élevé signifierait que « la cloche » de la gaussienne n'est pas symétrique :

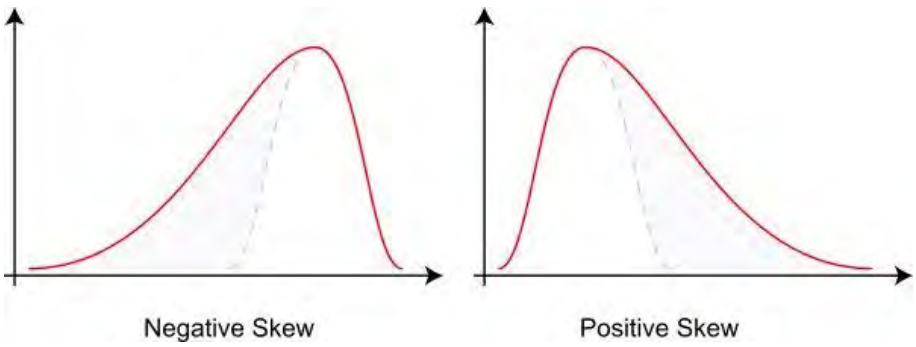


Figure 11 - Allure d'une mauvaise symétrie (Skew)

Le paramètre de kurtosis est égal à 3 pour une loi normale standard (Mesokurtic), à l'opposé un kurtosis trop faible signifierait que « la cloche » de la gaussienne est trop plate (Platykurtic) et un kurtosis trop élevé signifierait que « la cloche » de la gaussienne est trop pointue (Leptokurtic).

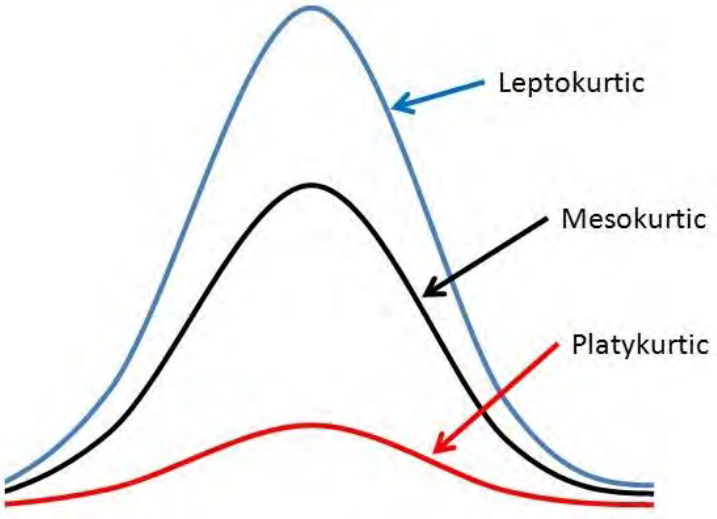


Figure 12 - Allure d'un mauvais aplatissement (Kurtosis)

L'objectif du test de normalité est donc de s'assurer que le coefficient de symétrie des n-ième échantillons de chaque mesure est nul, et que le kurtosis des n-ième échantillons de chaque mesure est égal à 3.

Le résultat de ce test est simplement affiché dans la console :

```
sas@sgc-pc-sas02: ~/stage_Devoge/Programmes/verif_normale
0
Fin :
499

Entrez un intervalle d'echantillons sur lequel effectuer le test.
Début :
0
Fin :
5999

Nombre de courbes/mesures : 500
Nombre d'échantillons : 6000

1) Creation des matrices...

2) Chargement des données...

3) Calcul des skew et kurtosis...

La mesure cible contient 5933 sur 6000 echantillons gaussiens
La mesure golden contient 5980 sur 6000 echantillons gaussiens
```

Figure 13 - Console affichant le résultat du test de normalité

b) Mise en œuvre

Comme précédemment, la première étape est celle du chargement des données à analyser dans deux matrices.

Puis nous calculons le skew et le kurtosis de chaque colonne de nos deux matrices de mesure (chaque groupe des n-ièmes échantillons).

Pour calculer ses deux paramètres, nous utilisons la bibliothèque GSL et ses fonctions « gsl_stats_skew » et « gsl_stats_kurtosis ». Il faut savoir que cette dernière nous donne le kurtosis excessif, celui-ci est nul pour une loi normale standard (centré réduite).

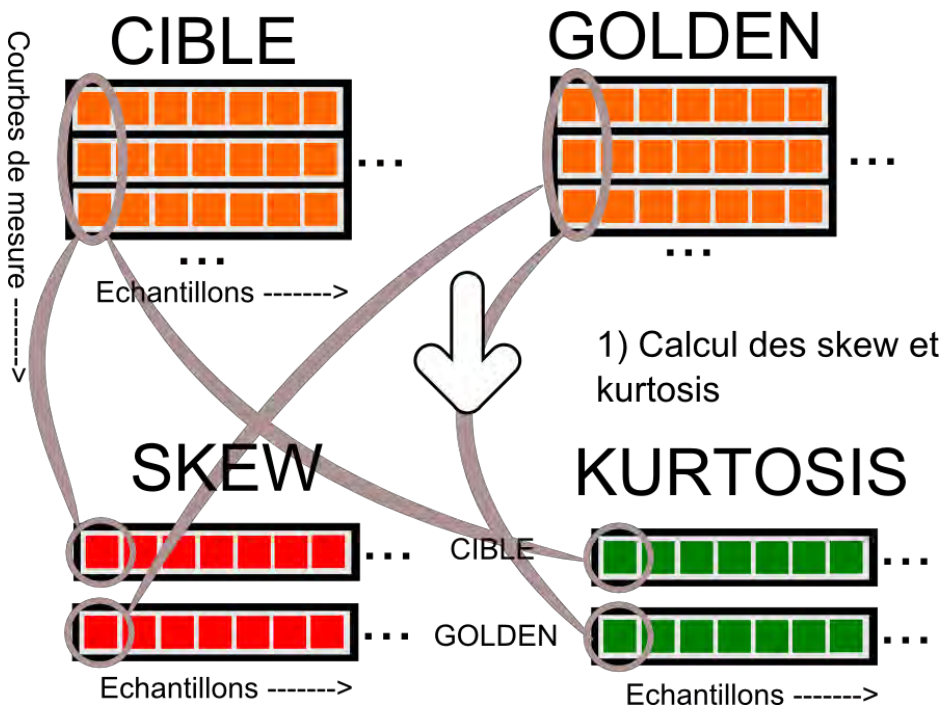


Figure 14 - Schéma logique de la vérification de normalité

Nous considérons que la série d'échantillons étudiée est gaussienne si son skew et son kurtosis excessif sont compris dans l'intervalle [-2;+2].

Nous comptons ensuite le nombre d'échantillons qui suivent une loi gaussienne parmi tous les échantillons.

3.2.4 Test de Wilcoxon

a) Théorie

Le test de Wilcoxon est un test non paramétrique, il correspond au t-test lorsque les conditions ne sont pas vérifiés (normalité et variance).

L'hypothèse de ce test est : la différence moyenne entre deux mesures d'un même phénomène est nulle.

Dans l'hypothèse où les deux échantillons proviennent d'une même population (et donc que la cible et la golden sont équivalents), il doit y avoir autant de différences négative que de différences de positive et leurs valeurs absolues doivent être, en moyenne, les mêmes.

Déroulement du test de Wilcoxon pour échantillons appariés :

- On calcule les différences entre tous les échantillons cible et golden.
- On ordonne ces différences par rangs, et, si une différence est nulle on attribue 0 à son rang et si deux différences sont identiques, alors leurs rangs deviennent la moyenne de leurs deux rangs.
- On calcule ensuite la somme des rangs des différences positives W+, et la somme des rangs des différences négatives W-.
- On calcule enfin t, la valeur statistique du test pour tous les échantillons :

  $t = \frac{W_{Min} - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}}$ Avec $W_{Min} = min(W+ ; W-)$ et n le nombre d'échantillons.
- On cherche dans la table de Student la valeur critique $t_{critique}$ de t pour un degré de liberté $ddl = n - 1$, pour α choisi (1% ou 5% le plus souvent).
- Si $|t| > t_{critique}$ alors on peut rejeter l'hypothèse H0 avec un seuil de confiance α.

b) Mise en œuvre

Comme précédemment, la première étape est celle du chargement des données à analyser dans deux matrices.

Nous créons ensuite une troisième matrice avec la fonction malloc, et nous la remplissons avec les différences des matrices cible et golden.

La prochaine étape est de trier cette matrice « Différence » selon les colonnes (les échantillons) par ordre croissant en valeur absolue.

Nous créons ensuite une quatrième matrice « rangs » avec la fonction malloc. Nous attribuons à chacun de ses éléments un rang correspondant à la valeur dans la matrice « Différence ».

Si la différence est nulle alors on attribue 0 à son rang et si deux différences sont identiques, alors on applique une moyenne sur leurs rangs.

On calcule ensuite la somme des rangs des différences positives W+ et la somme des rangs des différences négatives W-.

On calcule enfin t, la valeur statistique du test pour tous les échantillons :

$$t = \frac{W_{Min} - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}}$$ Avec $W_{Min} = min(W+ ; W-)$ et n le nombre d'échantillons.

Pour conclure ce test, nous comptons le nombre d'échantillons dont la valeur t respecte $|t| < 1{,}96$. Il s'agit de la valeur critique à 5 % pour un nombre de degré de liberté infini.

Pour tracer cette courbe nous utilisons le programme GNU Plot, voici un exemple de résultat du test de Wilcoxon :

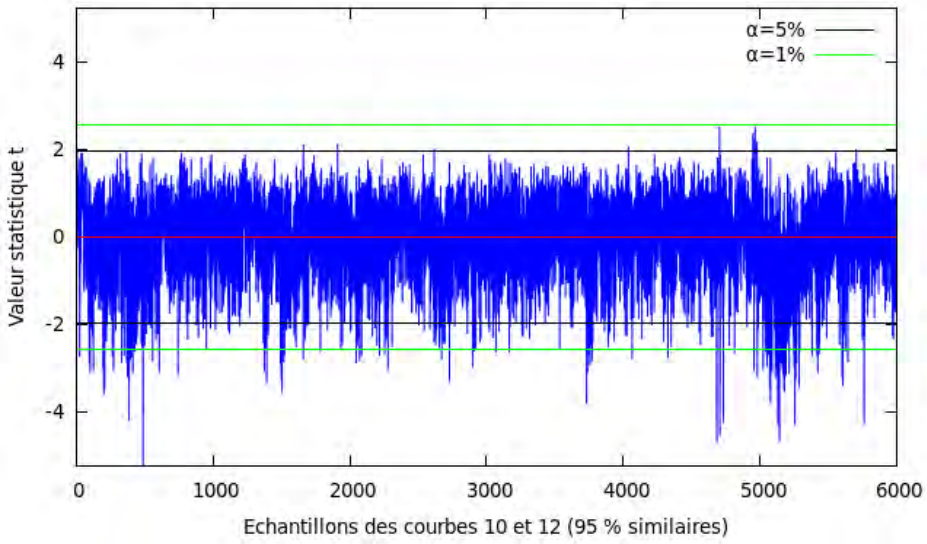
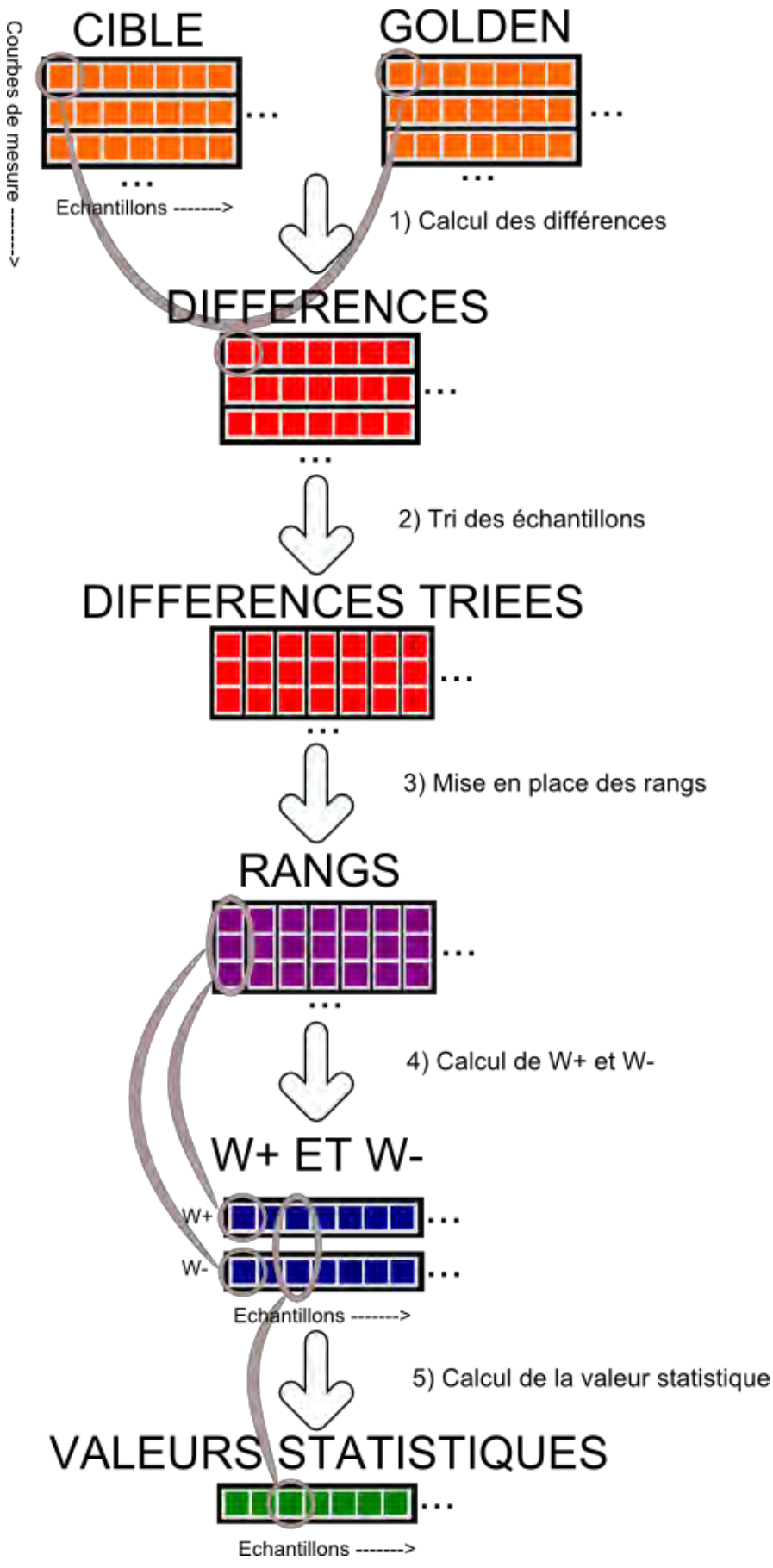


Figure 15 - Courbe résultat du test de Wilcoxon


CIBLE
GOLDEN
Courbes de mesure ------->
Echantillons ------->
1) Calcul des différences
DIFFERENCES
2) Tri des échantillons
DIFFERENCES TRIEES
3) Mise en place des rangs
RANGS
4) Calcul de W+ et W-
W+ ET W-
W+
W-
Echantillons ------->
5) Calcul de la valeur statistique
VALEURS STATISTIQUES
Echantillons ------->
Figure 16 - Schéma logique du test de Wilcoxon

3.3 La programmation

Tous les programmes qui viennent de vous être présentés ont été réalisés en langage C. Le C est un langage de programmation inventé au début des années 1970, il est devenu un des langages les plus utilisés. De nombreux langages plus modernes comme C++, Java et PHP reprennent des aspects de C.

Le langage C est beaucoup utilisé dans le cadre de la recherche scientifique, et il nous est enseigné à l'université.

Pour programmer en langage C, il nous faut un compilateur, ce programme va transcrire notre code C en langage machine. J'utilise le compilateur « GCC » car celui-ci peut être utilisé très simplement en ligne de commande sous Linux.

En complément j'utilise la bibliothèque de fonctions « GSL ». C'est une bibliothèque de fonctions mathématique gratuite et libre de droit (open source).

Voici un exemple de l'utilisation du langage C et de la bibliothèque GSL :

```
double VarianceCible = gsl_stats_variance(MesureCible, 1, LargeurMesure);
double VarianceGolden = gsl_stats_variance(MesureGolden, 1, LargeurMesure);
```

Dans le morceau de code ci-dessus, nous calculons les variances des séries « MesureCible » et « MesureGolden » de largeur « LargeurMesure », et les enregistrons dans les variables « VarianceCible » et « VarianceGolden » qui sont de type « double ».

Vous pouvez aussi consulter en Annexe, le programme complet de vérification de la variance.

Dans certains programmes, il est nécessaire, pour bien visualiser les résultats du test, de tracer les données dans un graphique. Pour cela nous utilisons le logiciel « GNU Plot ». Il s'agit d'un logiciel gratuit permettant de tracer n'importe quel type de données.

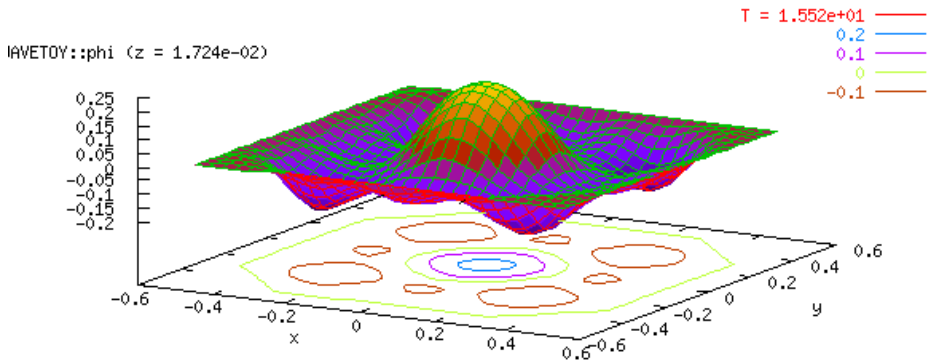


Figure 17 - Démonstration des capacités du logiciel GNU Plot

3.4 Résultats

Nous avons pu tester les programmes avec des données réelles, dans cette partie nous présenterons les résultats obtenus.

3.4.1 Tests entre une puce saine et une puce Golden

Afin d'avoir une base, nous effectuons en premiers les tests sur deux groupes de mesures provenant de cartes saines.

Commençons déjà par vérifier si les données respectent les conditions du test de Student.

- Test de variance

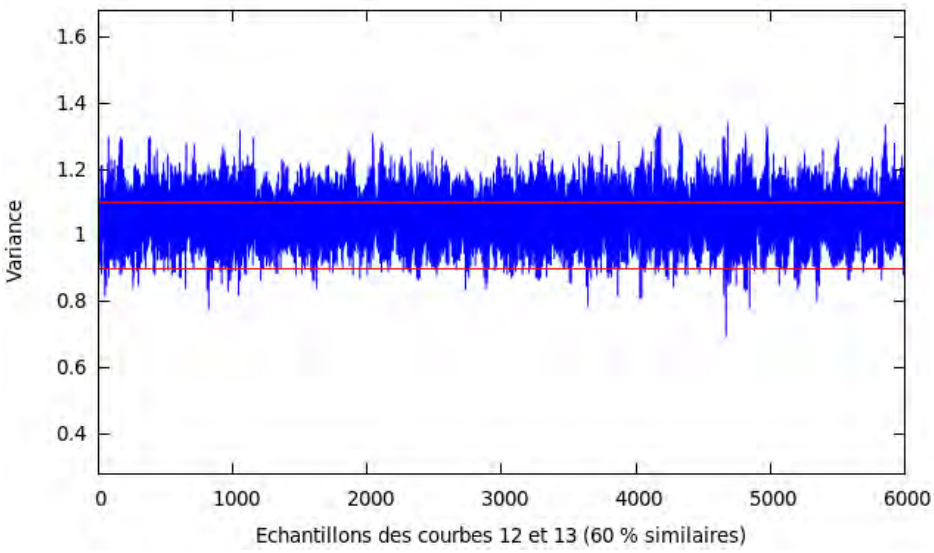


Figure 18 - Courbe du test de variance entre deux puces golden

Sur cette courbe nous voyons que le rapport des variances est centré en 0 (60% des échantillons ont le rapport de leur variance dans l'intervalle [0,90 ; 1,10]. Nous pouvons donc considérer que les variances sont proches.

- Test de normalité

Les resultat du test de normalité sont : 5980 sur 6000 échantillons gausiens pour la puce cible saine et 5990 sur 6000 pour la golden. Les conditions pour le test de Student sont donc validées.

- Test de Student

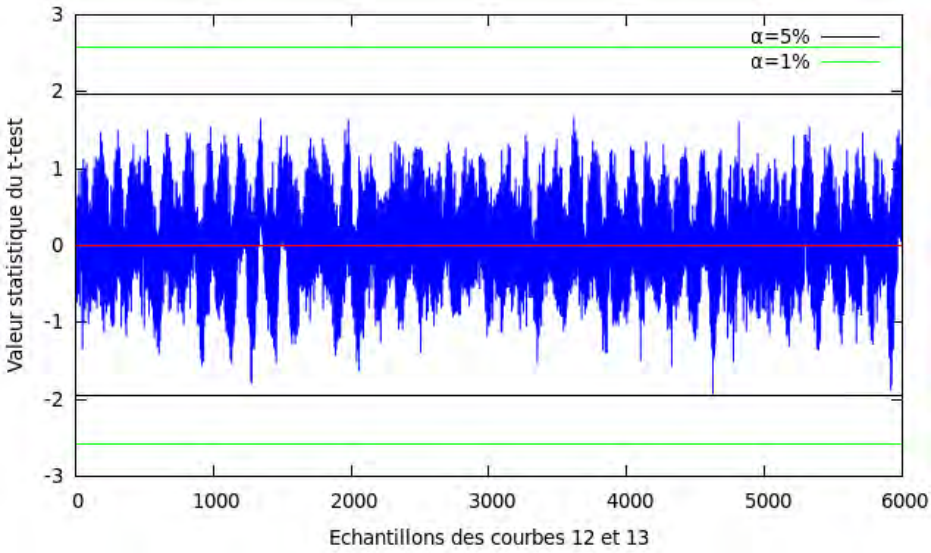


Figure 19 - Courbe du test de Student entre deux puces golden

On voit sur cette courbe que tous les échantillons sont dans l'intervalle critique à 5%, les deux mesures sont donc extrêmement proches.

- Test de Wilcoxon

Les données autorisaient un test de Student, plus performant, néanmoins nous pouvons tout de même regarder si le test de Wilcoxon confirme bien les résultats.

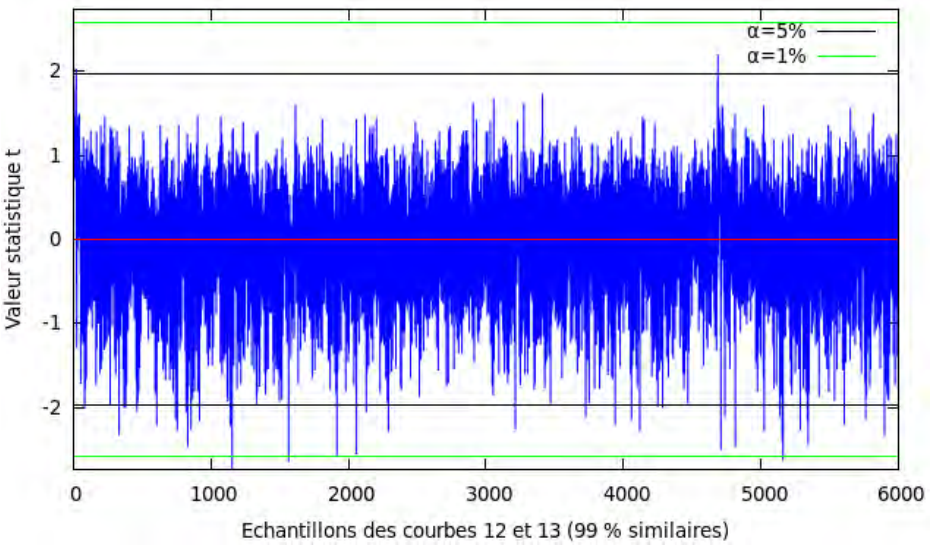


Figure 20 - Courbe du test de Wilcoxon entre deux puces golden

Sur la courbe ci-dessus, on peut voir que presque tous les échantillons sont dans l'intervalle critique à 5% (99% des échantillons).

Conclusion : Ce test révèle que les deux courbes de mesures testées proviennent de deux puces très similaires.

En effet le relevé de la consommation électrique a été fait dans ce cas sur deux cartes saines.

3.4.2 Tests entre une puce infectée et une puce Golden

Nous comparons ensuite une puce saine et une puce infectée par un cheval de Troie. Ces deux puces doivent effectuer la même opération de cryptage mais la puce infectée doit exécuter sa malveillance avant de lancer l'opération.

- Test de variance

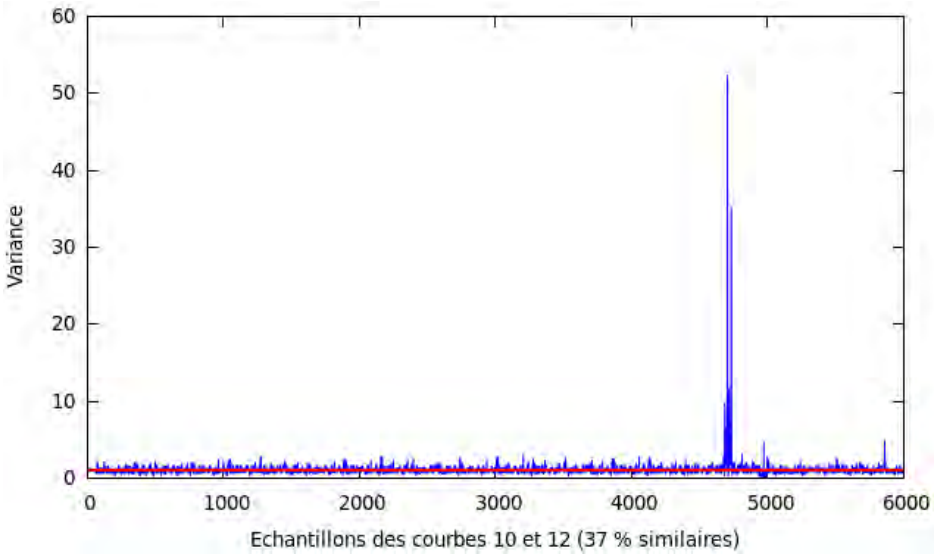


Figure 21 - Courbe du test de Student entre une puce infectée et une golden

Sur cette courbe on voit que le rapport des variances tourne autour de 0 (37% des échantillons sont dans l'intervalle [0,90 ; 1,10]) mais on voit surtout un pic un peu avant le 5000ieme échantillon.

Nous pouvons également comparer cette courbe avec celle que nous avions eue au test entre deux puces golden. Nous centrons la fenêtre autour de 1 en ordonnée pour une meilleure visibilité.

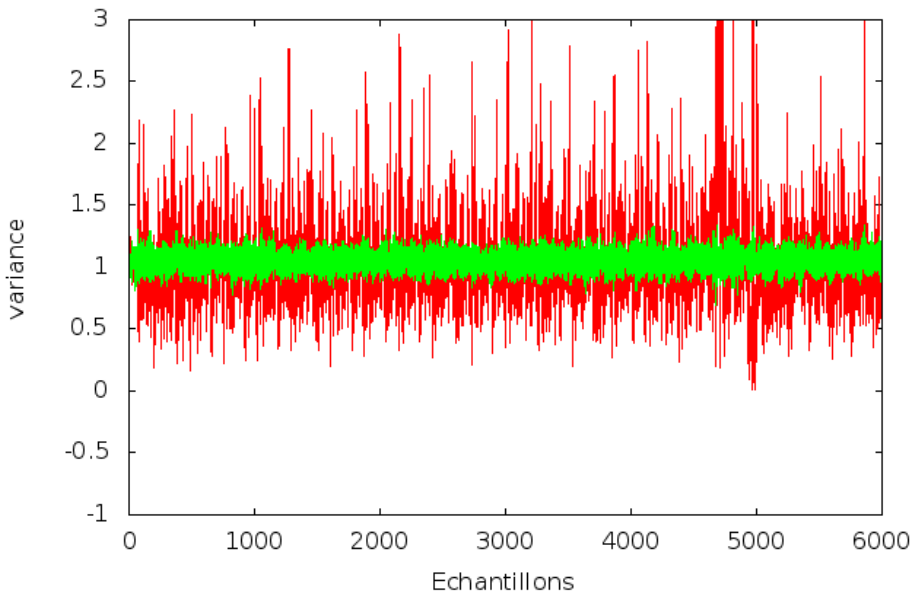


Figure 22 - Courbe des tests de variance entre deux golden en vert et entre une golden et une puce infectée en rouge

On voit nettement que la courbe rouge, correspondant au test entre une puce infectée par un cheval de Troie et une puce saine, est bien plus dispersée autour de 1 que la courbe verte, correspondant au test entre deux puces saines.

- Test de normalité

Les résultats du test de normalité sont : 5933 sur 6000 échantillons gausiens pour la cible infectée et 5980 sur 6000 pour la golden. Les conditions pour le test de Student sont validées.

- Test de Student

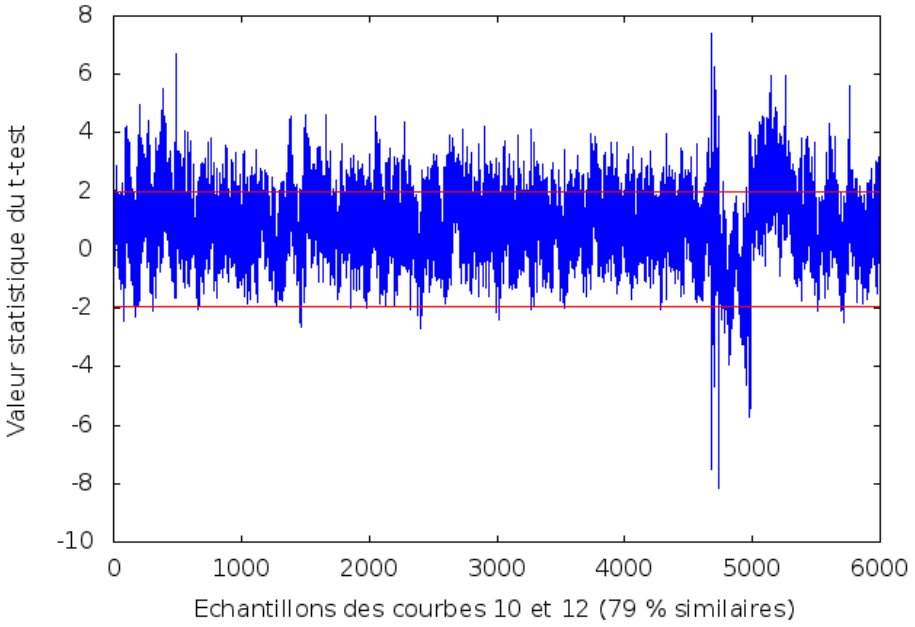


Figure 23 - Courbe du test de Student entre une puce infectée et une golden

Sur la courbe ci-dessus, nous pouvons observer une difformité de la courbe au niveau du 5000ieme échantillon, elle correspond au pic que nous avions vu lors du test de variance.

Nous pouvons également comparer cette courbe avec celle que nous avions eue au test entre deux puces golden.

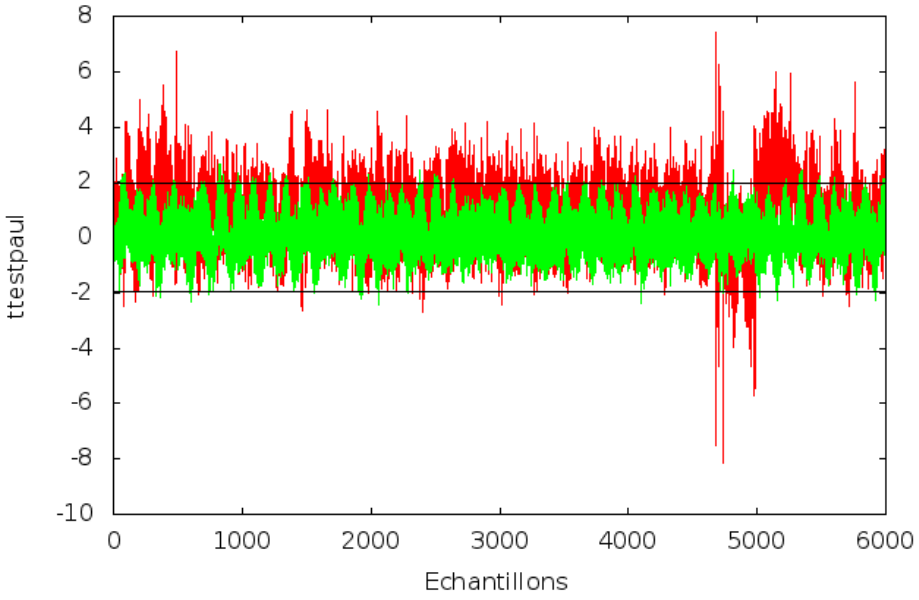


Figure 24 - Courbe des tests de Student entre deux golden en vert et entre une golden et une puce infectée en rouge

Nous voyons que la courbe verte est parfaitement dans l'intervalle critique à 5%, tandis que la courbe rouge dépasse fortement.

- Test de Wilcoxon

Encore une fois les données permettaient un test de Student, mais nous pouvons tous de même regarder les résultats du test de Wilcoxon.

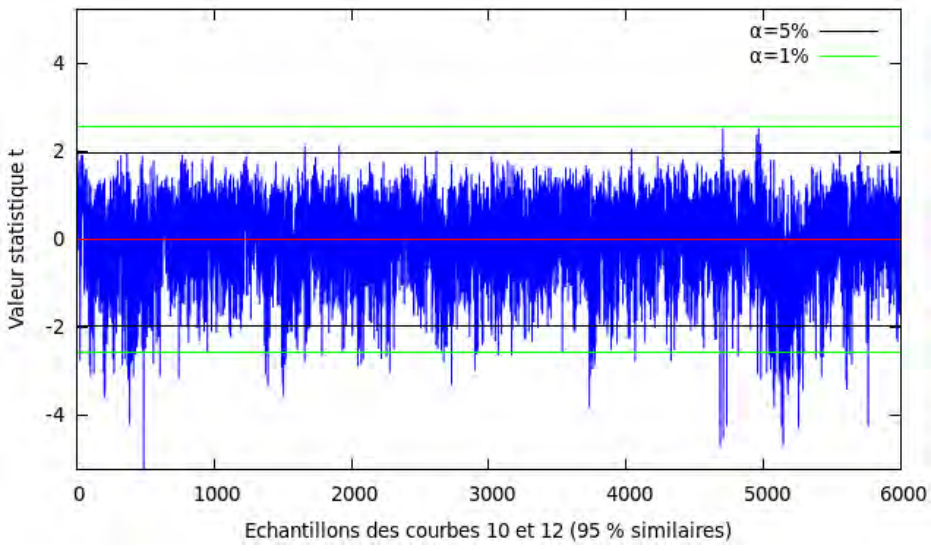


Figure 25 - Courbe du test de Wilcoxon entre une puce infectée et une golden

Sur la courbe ci-dessus, même si la plupart des échantillons sont dans l'intervalle critique, on observe encore la même difformité autour du 5000ieme échantillon.

Nous pouvons également comparer cette courbe avec celle que nous avions eue au test entre deux puces golden.

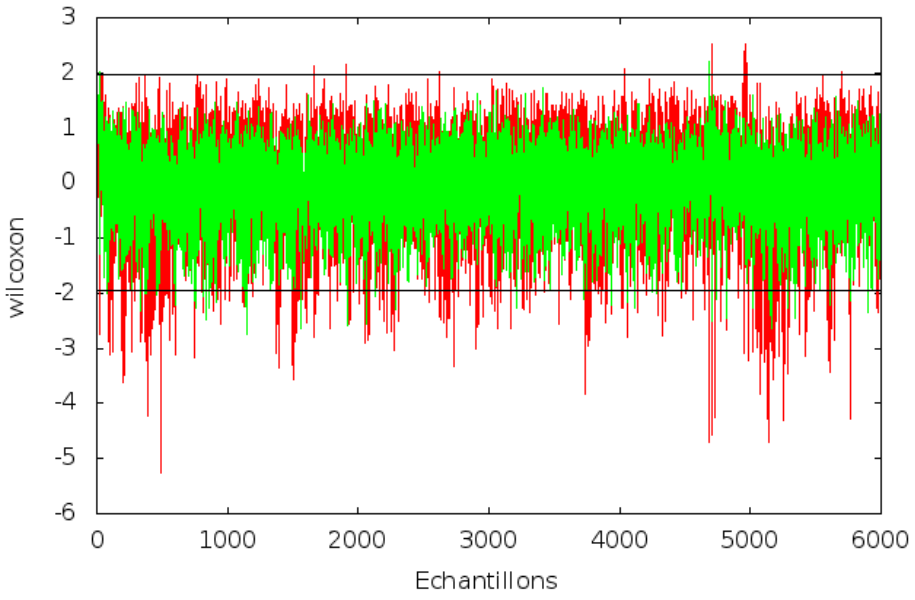


Figure 26 - Courbe des tests de Wilcoxon entre deux golden en vert et entre une golden et une puce infectée en rouge

Conclusion : Sur toutes ces courbes, on a pu observer une anomalie au niveau du 5000ieme échantillon.

En réalité nous savons que l'anomalie est dû au fait que la carte infectée ne démarre pas l'opération de cryptage demandé en même temps que la puce golden, donc elles ne finissent pas en même temps. Il y a donc une différence au niveau de la consommation électrique.

IV. Conclusion

Le sujet de détection des chevaux de Troie matériel est d'actualité car de telles menaces sont conçues chaque jour, de plus en plus performantes, cela nécessite donc un renouvellement continu des recherches en laboratoire afin d'empêcher que ces malveillances parviennent à leur fin.

La principale difficulté de ce stage a été pour moi l'apprentissage des notions mathématiques et des méthodes de programmation nécessaires à l'avancement du projet. Pour pallier cette difficulté, j'ai pu bénéficier de l'expérience des chercheurs du laboratoire et j'ai également effectué plusieurs recherches bibliographiques.

Ce stage m'a permis de découvrir un nouveau domaine liant électronique, informatique et mathématique.

Il m'a permis d'acquérir une bonne expérience en programmation en langage C ainsi que d'enrichir mes connaissances mathématiques sur les outils statistiques qui ont été utilisés pour la détection des chevaux de Troie.

Au final je suis très satisfait par ce stage durant lequel j'ai pu découvrir le monde de la recherche scientifique.

VI. Lexique

- Circuit intégré : circuit contenant différents composants: transistors, diodes, condensateurs, résistances, gravés sur un morceau de silicium selon une configuration donnée, pour répondre à un besoin spécifique.
- Cryptographie : ensemble des techniques permettant de chiffrer des messages.
- Cheval de Troie logiciel : programme en apparence légitime, mais qui contient une malveillance. Le rôle du cheval de Troie est de faire entrer ce parasite sur l'ordinateur et de l'y installer à l'insu de l'utilisateur.
- Cheval de Troie matériel : modification malveillante des fonctions physiques du circuit intégré dans le but transmettre des informations confidentielles telles que les clefs de cryptage par exemple
- FPGA (Field-Programmable Gate Array): circuit intégré logique qui peut être reprogrammé après sa fabrication. Notons qu'il serait impropre de parler de programmation au sens logiciel (contrairement à un microprocesseur, il n'exécute aucune ligne de code). Ici, mieux vaudrait parler de « reconfiguration (on modifie des connexions ou le comportement du composant, on connecte des portes logiques entre elles, etc.).
- Oscilloscope : instrument de mesure destiné à visualiser un signal électrique variable au cours du temps.
- Distingueur : outil statistique déterminant avec une forte probabilité la véracité d'une hypothèse.
- Aplatissement (Kurtosis) : mesure de l'aplatissement de la distribution d'une variable aléatoire réelle.
- Symétrie (Skew) : mesure de l'asymétrie de la distribution d'une variable aléatoire réelle.
- Echantillon : point d'une courbe discrète.
- Gaussienne : fonction en exponentielle de l'opposé du carré de l'abscisse (une fonction en $\exp(-x^2)$. Elle a une forme caractéristique de courbe en cloche.
- Matrice : tableau de nombres à deux dimensions.
- Vecteur : tableau de nombres à une dimension.
- Variance : mesure servant à caractériser la dispersion d'un échantillon. Elle indique de quelle manière la série statistique se disperse autour de sa moyenne.

VII. Bibliographie

- LIN, Lang, KASPER, Markus, GÜNEYSU, Tim, *et al.* Trojan side-channels: lightweight hardware trojans through side-channel engineering. In : *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer Berlin Heidelberg, 2009. p. 382-395.
- « Loi de Student. » *Wikipédia, l'encyclopédie libre*. <https://fr.wikipedia.org/w/index.php?title=Loi_de_Student>.
- Saporta. *Probabilités, analyse des données et statistiques.* Edition Technip, 2006.
- « Test de Wilcoxon » Statisticien.fr. <http://www.statisticien.fr/tests-de-rangs/test-wilcoxon-paires.html>
- Wilcoxon, F. "TEST DE STUDENT." *Dictionnaire encyclopédique*: 544.
- LE, Thanh-Ha, CLÉDIÈRE, Jessy, CANOVAS, Cécile, *et al.* A proposition for correlation power analysis enhancement. In : *Cryptographic Hardware and Embedded Systems-CHES 2006*. Springer Berlin Heidelberg, 2006. p. 174-186.
- DEHBAOUI, Amine, DUTERTRE, Jean-Max, ROBISSON, Bruno, *et al.* Injection of transient faults using electromagnetic pulses-Practical results on a cryptographic system-. *IACR Cryptology ePrint Archive*, 2012, vol. 2012, p. 123.
- « Loi normale. » *Wikipédia, l'encyclopédie libre*. <https://fr.wikipedia.org/w/index.php?title=Loi_normale >.