Table des matières

Introduction

Candidat "Officier de carrière" à l'***École Royale Militaire*** (ERM), je suis actuellement ma formation académique à l'***École Centrale des Arts et Métiers*** (ECAM) en option électronique.

Durant notre 2ème année de Master, les étudiants doivent réaliser un stage d'immersion en entreprise d'une durée de 6 semaines. Ce stage consiste, entre autres, à s'insérer dans une entreprise afin d'y découvrir différents aspects tels que l'organisation générale d'une entreprise, son management, son contexte social, son insertion économique, ses aspects techniques et ses produits. Il a également pour but de se familiariser au travail quotidien de l'ingénieur en participant à diverses activités.

Ayant réalisé mon stage de 3ème Bachelier chez ***AIRBUS DS SLC*** sur le site de Diegem et de Elancourt, il était important pour moi de saisir la chance et l'opportunité de découvrir une nouvelle entreprise renommée à travers le monde. C'est ainsi que je décidai de réaliser mon stage chez ***THALES*** *Telecommunications Belgium* sur le site de Tubize.

1 Environnement de stage

Cette section a pour objectif de décrire l'entreprise à différents points de vue. Tout d'abord, une description de la structure de l'entreprise (dont l'équipe dans laquelle se trouve le stagiaire) et de ses objectifs est reprise. Ensuite, sont détaillées succontement la gestion de production mais aussi la gestion marketing, la gestion financière et la gestion des ressources humaines. Enfin, un descriptif de la cellule qualité clôturera ce premier point.

1.1 Structure de l'entreprise

C'est donc chez ***Thales*** *Telecommunication Belgium* que je me suis rendu pour réaliser mon stage d'immersion en entreprise.

1.2 Objectifs de l'entreprise

1.3 Equipe du stagiaire

1.4 Gestion de production

1.5 Gestion marketing

1.6 Gestion financière

1.7 Gestion des ressources humaines

1.8 Politique QHSE

2 Objectifs du stage

L'objectif de ce stage était d'introduire l'ensemble des notions élémentaires, nécessaires pour la réalisation du *Travail de Fin d'Étude* (TFE). Ce travail de fin d'étude qui allait se poursuivre durant 6 mois à compter du mois de Novembre 2018. Dans un premier temps, une définition des notions théoriques est abordée. Ensuite, un exemple ou une simulation est mise en oeuvre afin d'appuyer le concept théorique.

La liste ci-dessous reprend l'ensemble des objectifs fixés et réalisés durant les 6 semaines de stage :

1. **Chiffrement**
2. **L'algorithme AES (*Advanced Encryption Standard*)**
3. ***Side-Channel Attacks* (attaques par canal auxiliaire)**
4. **Simulation d'un tracé et d'une attaque par canal auxiliaire sur MATLAB**
5. **Statistiques et t-test**
6. **Simulation d'un t-test sur MATLAB**

Ces différents objectifs sont décrits dans la section 3 *"Projet du stage"*.

3 Projet du stage

Cette section décrit l'ensemble des objectifs, cités à la section 2, fixés pour le stage.

3.1 Notions de chiffrement

Les systèmes de sécurité modernes utilisent des algorithmes de chiffrement pour assurer la disponibilité, la confidentialité et l'intégrité de données. Ces algorithmes de chiffrement sont en réalité des fonctions mathématiques qui prennent typiquement :

— 2 paramètres en entrée : un ***message clair*** (nommé *plaintext* en anglais) et une ***clé de chiffrement*** (nommée *key* en anglais).
— 1 paramètre en sortie : le ***message chiffré*** (nommé *ciphertext* en anglais).

Le procédé transformant les données claires en entrée en données chiffrées en sortie est appelé le ***chiffrement***. Ce procédé est réalisé grâce à un ***algorithme de chiffrement*** utilisant une clé de chiffrement et diverses opérations mathématiques. Il est important de préciser que tous les détails décrivant le fonctionnement d'un algorithme sont disponibles publiquement, seule la clé de chiffrement doit rester secrète. En effet, la sécurité offerte par un algorithme de chiffrement ne doit pas dépendre du secret de son implémentation. Un bon algorithme est un algorithme dont on ne parviendra pas à déchiffrer les données chiffrées. Lorsque la clé d'un algorithme est trouvée, le déchiffrement des données confidentielles peut être réalisé. On dit que l'algorithme de chiffrement est ***cassé***.

La figure 1 ci-dessous présente le principe de fonctionnement d'un algorithme de chiffrement.

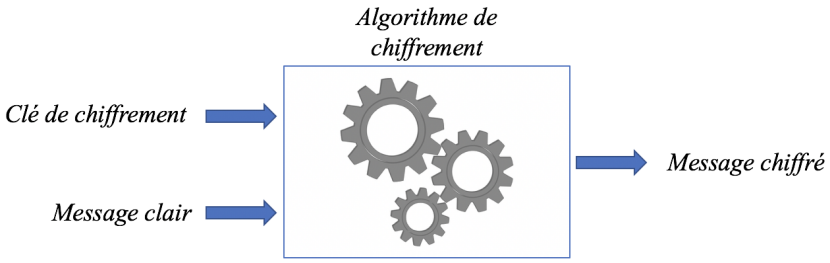

Figure 1 : L'algorithme de chiffrement, caractérisé par diverses opérations mathématiques, utilise une clé de chiffrement en entrée pour chiffrer un message clair. Cela produit un message chiffré, non compréhensible pour une personne ne connaissant pas la clé de chiffrement.

Nous distinguons 2 types d'algorithmes de chiffrements :

— ***Chiffrement symétrique*** : Le chiffrement est dit symétrique lorsque le procédé de chiffrement (algorithme) utilise une seule clé, appelée *clé secrète*. Par convention, ce type de chiffrement permet à la fois de chiffrer et de déchiffrer des messages à partir d'une seule et unique clé. Le désavantage de ce typer de chiffrement est que si une personne parvient à subtiliser la clé, elle sera en mesure de déchiffrer tout message qu'elle intercepte.
Exemple : L'algorithme AES. Une explication plus détaillée de cet algorithme est reprise à la section 3.2

— ***Chiffrement asymétrique*** : Le chiffrement est dit asymétrique lorsque le procédé de chiffrement (algorithme) utilise 2 clés : une *clé publique* et une *clé privée*. Par convention, la clé publique est la clé de chiffrement du message clair, elle peut être communiquée sans aucune restriction tandis que la clé privée est la clé de déchiffrement du message chiffré, elle ne doit être communiquée sous aucun prétexte. Le fonctionnement est le suivant : Avec une clé publique, l'expéditeur code, dans un algorithme de chiffrement donné, un message. Ce message, une fois transmis, ne pourra être déchiffré que par le destinataire, détenteur de la clé privée.
Exemple : L'algorithme RSA.

Les figures 2 et 3 ci-dessous présentent les principes de fonctionnement des chiffrements symétriques et asymétriques respectivement.

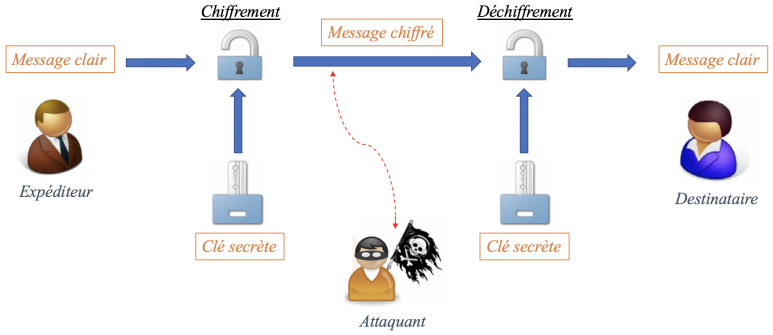

Figure 2 : Chiffrement symétrique : Une seule clé est utilisée pour chiffrer et déchiffrer les messages.

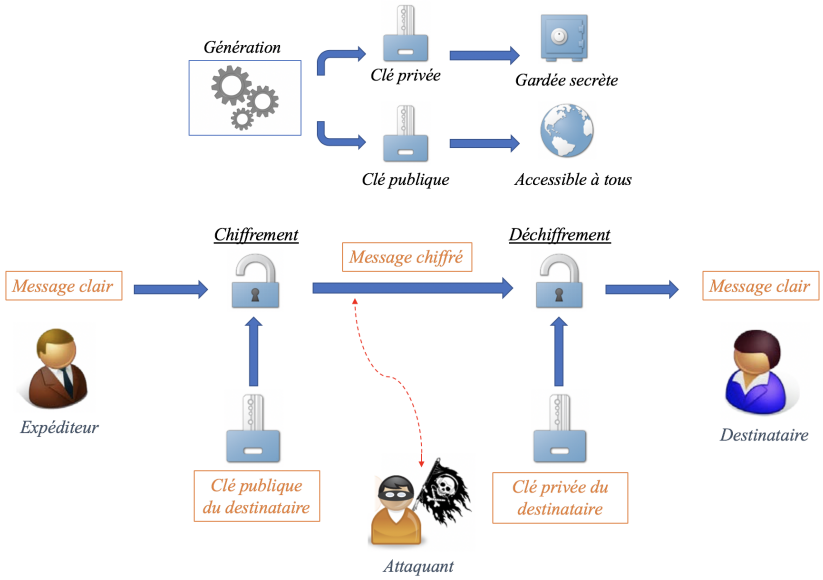

Figure 3 : Chiffrement asymétrique : Un clé publique est utilisée pour chiffrer le message et une clé privée est utilisée pour le déchiffrer.

Que ce soit pour un algorithme de chiffrement symétrique ou asymétrique, la clé de chiffrement doit être stockée sur un support physique. Ce support, appelé *device cryptographique*, doit être suffisamment sécurisé que pour contenir de manière protégée la clé. Ainsi, un ***device cryptographique*** est un device qui implémente des algorithmes de chiffrement et qui stocke des clés de chiffrement.

3.2 Algorithme *AES* (*Advanced Encryption Standard*)

En 1997, le NIST (*National Institute of Standards and Technology*) décida qu'il était temps de développer un nouveau standard d'algorithme de chiffrement. Ce nouveau standard, nommé **AES** (pour *Advanced Encryption Standard*), était appelé à remplacer l'ancien standard de chiffrement, l'algorithme DES (pour *Data Encryption Standard*). Pour ce faire, le NIST organisa un concours cryptographique, les chercheurs du monde entier furent invités à soumettre leurs propositions. En Octobre 2000, Le NIST annonça le vainqueur du concours : l'algorithme de Rijndael, du nom de ses concepteurs Joan Daemen et Vincent Rijmen, tous deux de nationalité belge.

L'algorithme de Rijndael, désormais plus connu sous le nom d'algorithme AES, est un **algorithme de chiffrement symétrique par blocs**. C'est-à-dire que les données sont traitées par blocs de 128 bits. La clé secrète peut posséder différentes tailles : 128 bits (AES-128), 192 bits (AES-192) ou encore 256 bits (AES-256). À noter qu'en théorie, plus la taille de la clé est élevée, moins il y a de chance de casser l'algorithme cependant, comme nous le verrons par la suite, avec les attaques par canal auxiliaire (section 3.3), le problème peut vite être contourné. La description qui suit est basée sur l'algorithme AES-128 bits, c'est-à-dire que la clé de chiffrement a une taille de 128 bits.

L'AES-128 a donc pour rôle de chiffrer des blocs de données de 128 bits avec une clé de 128 bits. Les données et la clé sont représentés par une matrice où chaque élément de la matrice correspond à un byte (un octet, i.e 8 bits). Étant donné que 128 bits correspond à 16 bytes, la matrice de données, au même titre que la matrice de clé, correspond à une matrice de 4 lignes et 4 colonnes (formant ainsi les 4x4 soit 16 bytes). Une matrice particulière (de taille 4x4 également) appelé STATE contient l'ensemble des résultats intermédiaires résultant des diverses opérations que subissent les données (depuis leur état initial).

La figure 4 présente les 3 matrices qui viennent d'être citées : la matrice de donnée (message clair initial de 128 bits), la matrice STATE (qui va contenir les résultats intermédiaires des données suite aux différentes opérations) et la matrice clé (clé de 128 bits).

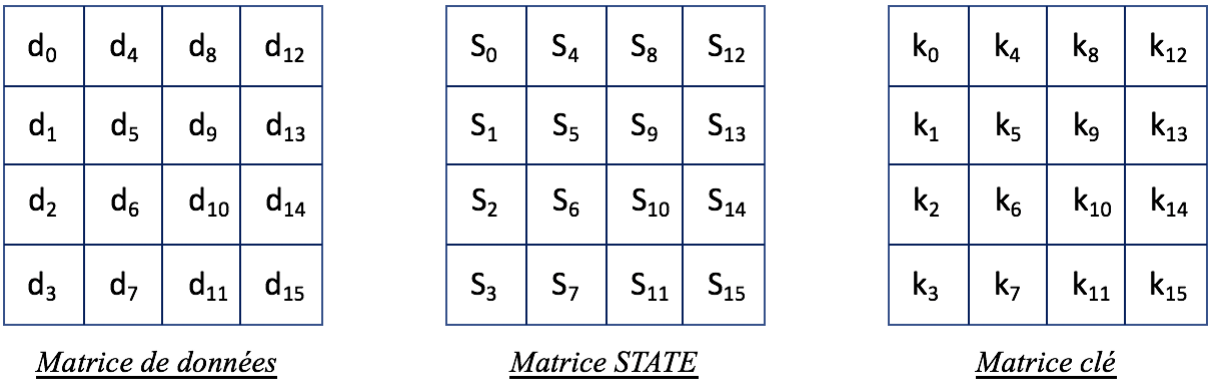

Figure 4 : Les 3 matrices utilisées par l'algorithme AES.

Remarque : En pratique, la matrice de données est directement confondue avec la matrice STATE. Autrement dit, les premiers éléments à être placés dans la matrice STATE représentent les bytes de données. Ainsi, on n'utilise que deux matrices durant le fonctionnement de l'algorithme AES : la matrice STATE et la matrice clé.

Par ailleurs, l'algorithme AES est caractérisé par une série de tours (*rounds* en anglais) dépendant de la taille de la clé. Pour une clé dont la taille est 128 bits, on dénombre 10 tours (12 tours pour une clé de 192 bits et 14 tours pour une clé de 256 bits). Un tour est définit par 4 opérations appliquées succinctement sur la matrice STATE. Ces 4 opérations sont : *AddRoundKey*, *SubBytes*, *ShiftRows* et *MixColumns*. Elles sont appliquées à divers instants dans l'exécution de l'algorithme AES. La figure 5 (page suivante) permet de visualiser l'ordre d'exécution chronologique de ces 4 opérations.

La figure 5 ci-dessous présente le principe de fonctionnement général de l'algorithme AES-128.

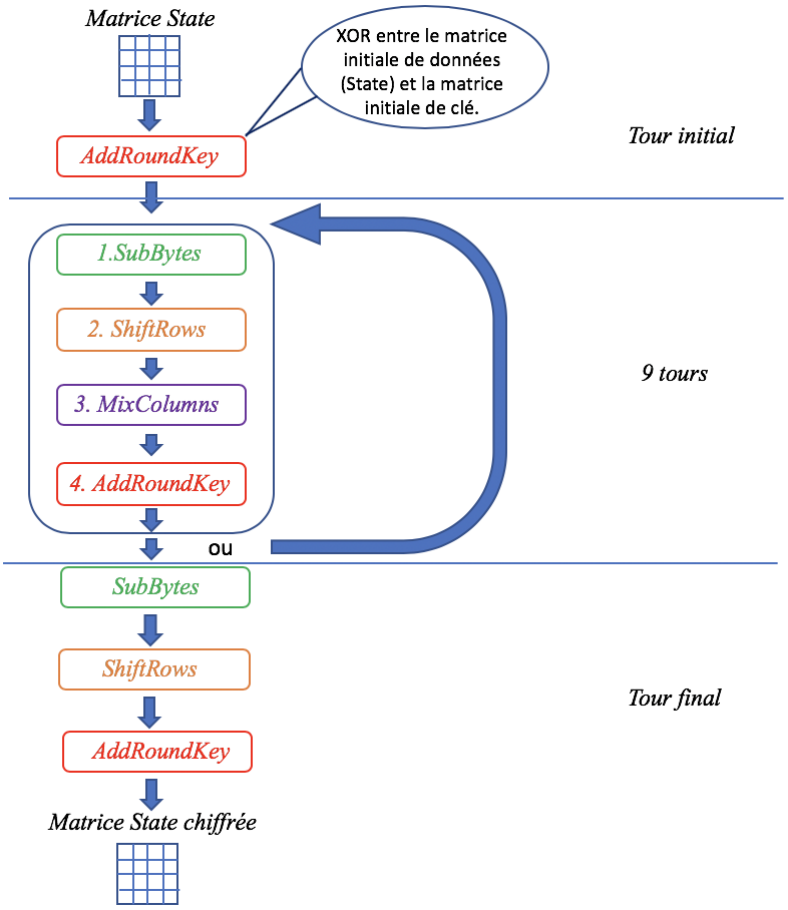

Figure 5 : Principe de fonctionnement de l'algorithme AES-128.

Fonctionnement :

Initialement, deux matrices vont être utilisées : la matrice STATE, contenant les données claires, et la matrice clé, contenant la clé secrète initiale. En effet, une première opération (*AddRoundKey*) est appliquée sur ces deux matrices. Plus précisément, cette opération réalise un XOR (symbole $\oplus$) entre chaque élément de la matrice STATE et chaque élément de la matrice clé. Le résultat est ré-écrit dans la matrice STATE. La figure 6 ci-dessous présente le principe de fonctionnement de cette première opération :

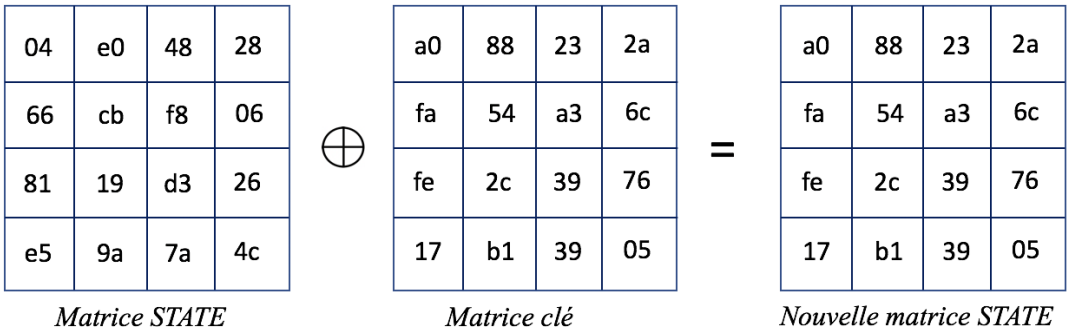

Figure 6 : Opération XOR entre la matrice STATE et la matrice clé.

Ensuite, une série de 4 opérations se répétant 9 fois (9 tours cycliques) est exécutée. Ces 4 opérations sont appliquées dans l'ordre suivant : *SubBytes*, *ShiftRows*, *MixColumns*, *AddRoundKey*.
Enfin, une fois les 9 tours exécutés, le dernier *round* se lance exécutant 3 opérations : *SubBytes*, *ShiftRows* et *AddRoundKey*. À la fin de ces 3 dernières opérations, une matrice de taille 4x4 présente le message chiffré de 128 bits.

Description des 4 opérations :

1. **SubBytes** : La figure 7 ci-dessous présente le principe de fonctionnement de l'opération *SubBytes* :

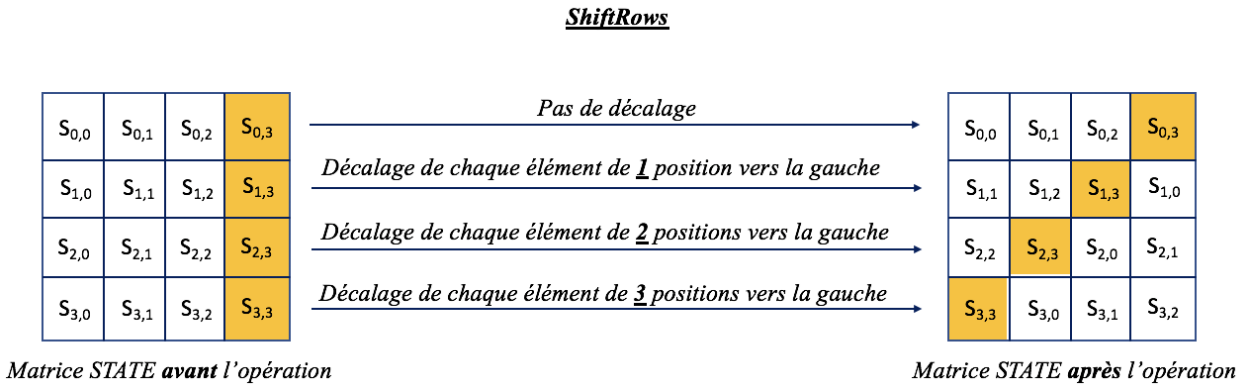

Figure 7 : Opération *subBytes* exécutée sur la matrice STATE.

2. **ShiftRows** : Comme son nom l'indique, cette opération concerne les lignes de la matrice STATE. Cette opération réalise une permutation cyclique des octets sur les lignes de la matrice STATE. Plus précisément, **pour la *i-ième* ligne, on décalera chaque élément de la matrice STATE de *i* positions vers la gauche**, en considérant que la première a pour indice 0. La figure 8 ci-dessous présente le principe de fonctionnement de l'opération *ShiftRows* :

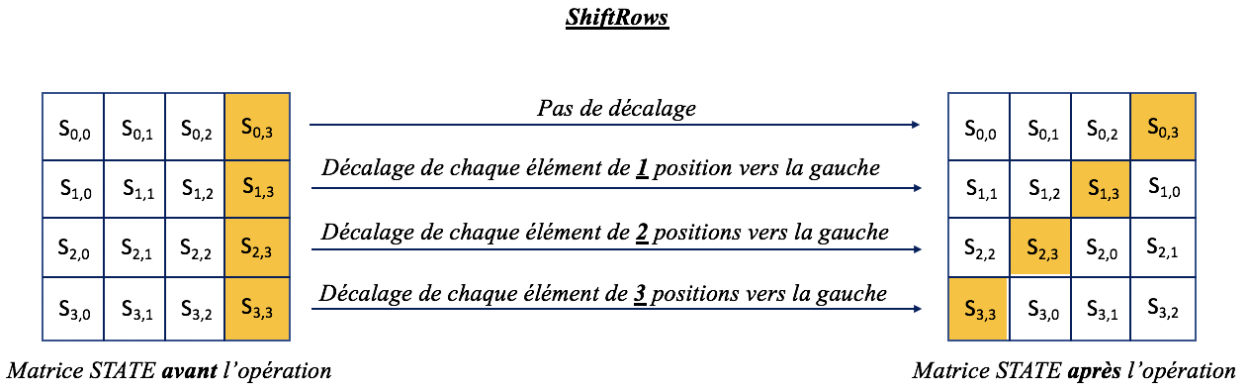

Figure 8 : Opération *ShiftRows* exécutée sur la matrice STATE.

3. **MixColumns** : Comme son nom nom l'indique, cette opération concerne les colonnes de la matrice STATE. **Cette opération réalise un produit matriciel entre une matrice fixée (taille 4x4) définie ci-dessous (figure 9) et un vecteur colonne (taille 4x1) de la matrice STATE**. Cela produit un nouveau vecteur colonne (taille 4x1) permettant de définir la nouvelle matrice STATE. La figure 9 ci-dessous présente le principe de fonctionnement de l'opération *MixColumns* :
4. **AddRoundKey** : La gestion des clés se fait au travers des fonctions addRoundKey et keyExpansion. AddRounkey réalise un XOR entre STATE et la clé courante (roundkey 8). Le résultat de ce XOR est placé dans STATE. Cette roundkey est dérivée de la clé secrète. On change de clé courante à chaque tour pour éviter d'utiliser toujours la même valeur ; cela rend la tâche d'attaquants plus difficile. KeyExpansion est là pour étendre la clé afin, comme nous l'avons dit plus haut, de ne pas utiliser la même clé à chaque tour. La taille de la clé, après avoir été étendue, est égale à la longueur d'un bloc multipliée par le nombre de tours plus un. En effet, il nous faut une clé différente par tour ; de plus, au début de l'algorithme, on exécute un addRounKey initial avant le premier tour. Pour AES-128 (10 tours et clé de 16 octets), la clé, après avoir été étendue, aura 176 octets. $16(10+1) = 176$ octets. Le fonctionnement de KeyExpansion étant le même pour la construction de chaque sous-clé, nous allons voir comment on crée la première sous-clé à

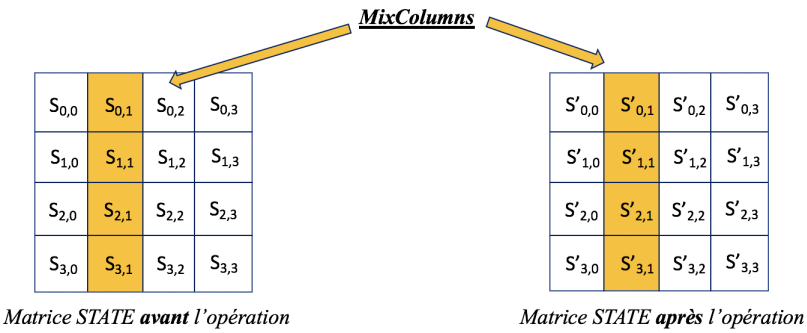

Figure 9 : Opération *MixColumns* exécutée sur la matrice STATE.

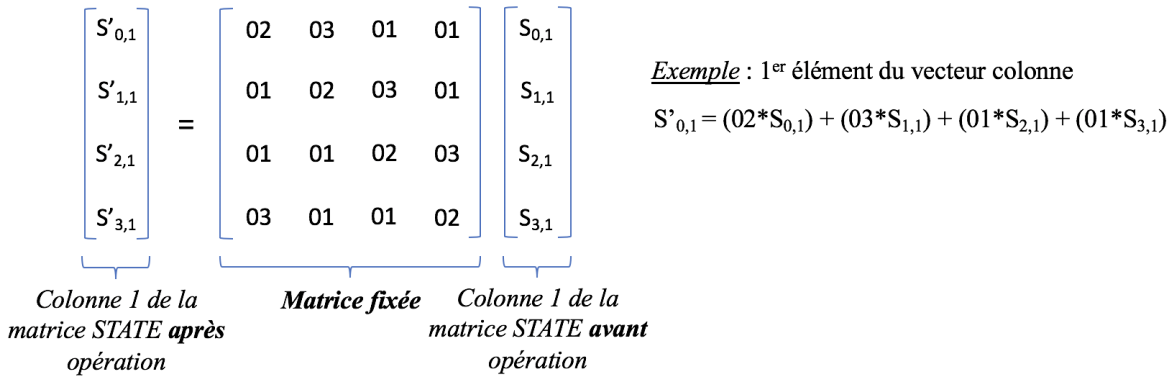

Figure 10 : Exemple de l'opération *MixColumns* exécutée sur la deuxième colonne de la matrice STATE.

partir de la clé de départ ; ceci pourra se généraliser pour les autres clés en remplaçant « première sous-clé » par « i-ième sous-clé » et « clé de départ » par « sous-clé précédente ».

3.3 Introduction aux *attaques par canal auxiliaire*

3.3.1 Types d'attaques

A la fin des années 1990, une nouvelle contrainte pour la conception de système informatique a vu le jour : la sécurité matérielle. Bien souvent, la sécurité d'un système informatique s'appuie plus sur les concepts software que hardware. Cependant, un nouveau mode d'attaque s'est développé. Il s'agit d'attaques physiques, c'est-à-dire d'attaques réalisées sur le circuit électronique lui-même. Deux grandes familles d'attaques sont recensées :

- ***Attaques actives*** : Une attaque est dite active lorsque les entrées et/ou l'environnement du device cryptographique sont manipulés par l'attaquant en vue de produire un comportement anormal du device. La clé secrète est révélée en exploitant les données issues de ce comportement anormal. Cela peut être une variation de la tension du device, une injection de glitch d'horloge, etc. On distingue deux types d'attaques actives :
  - Les attaques actives ***irréversibles*** qui conduisent à la destruction du device cryptographique. Ce type d'attaque est souvent réalisé pour connaître la conception physique d'un device. *Exemple* : Découpage laser d'un circuit intégré.
  - Les attaques actives ***pseudo-réversibles*** qui n'entrainent pas forcément la destruction du device cryptographique, mais qui sont souvent tout de même invasives puisqu'elles nécessitent la préparation du circuit (découpe partielle du boitier du circuit intégré par exemple). Un exemple typique de ce type d'attaque est ce qu'on appelle les *attaques en fautes*. Le principe est d'introduire volontairement des fautes dans le circuit (exemple : Injection de rayon laser, injection de glitch d'horloge, etc.). Les fautes ainsi créées peuvent entrainer le circuit dans des modes de fonctionnement conduisant à des erreurs. Ces erreurs peuvent ensuite être exploitées pour déterminer la clé.
- ***Attaques passives*** : Une attaque est dite passive lorsque l'attaquant exploite l'analyse, en fonctionnement normal, d'informations s'échappant d'un device cryptographique. Cela peut être l'analyse de la consommation de puissance, l'analyse temporelle, l'analyse par rayonnement électromagnétique, etc. C'est ce type d'attaque qui sera détaillé tout au long de ce stage et durant la réalisation de TFE.

La figure 11 ci-dessous résume les différents types d'attaques physiques possibles.

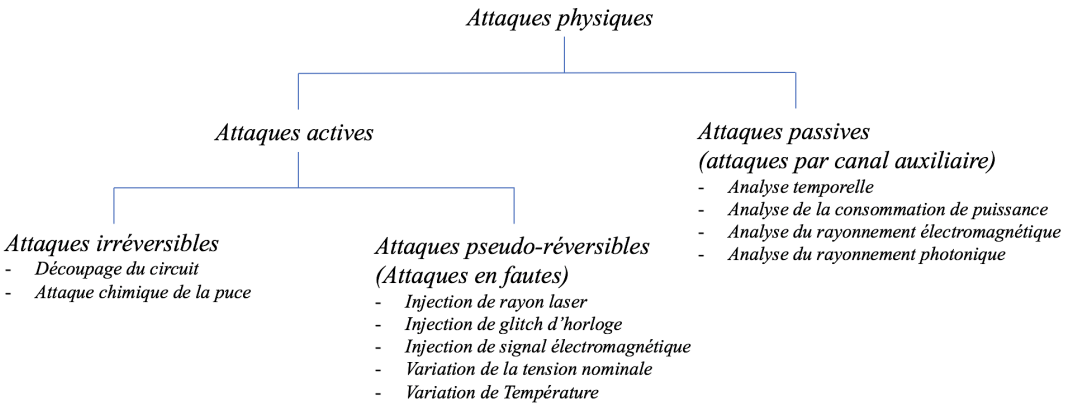

Figure 11 : Les 2 grandes familles d'attaques physiques possibles. La suite de ce rapport se concentre essentiellement sur les attaques physiques dites passives.

Les attaques passives sont globalement beaucoup plus simples à mettre en oeuvre que les attaques actives. Comme définit précédemment, ces attaques consistent à analyser des données issues de canaux auxiliaires au device cryptographique (lorsque ce dernier est en état de fonctionnement normal). Ces canaux auxiliaires sont des canaux présents physiquement sur le circuit attaqué et le long desquels de l'information s'échappe (sous différentes formes : rayonnement électromagnétique, rayonnement photonique, consommation de puissance, etc.). C'est là qu'intervient la notion de *side-channel attacks* ou en français *l'attaque par canal auxiliaire*. En effet, les fonctions cryptographiques, bien que pouvant être extrêmement robustes théoriquement (c'est-à-dire mathématiquement) sont très sensibles aux fuites d'informations. C'est-à-dire qu'une quantité très faible d'informations peut être exploitée pour casser un algorithme cryptographique très fort. C'est ce que les attaques par canaux auxiliaires exploitent.

La figure 12 ci-dessous présente les différentes façons possibles de réaliser des attaques passives.

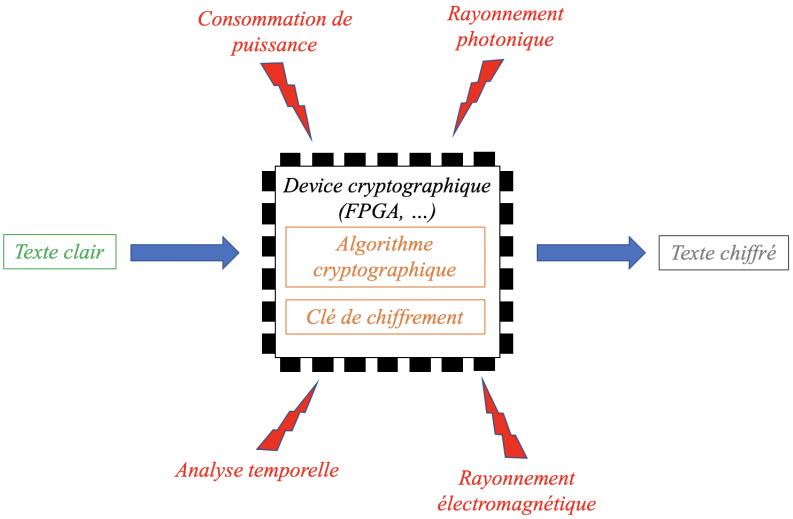

Figure 12 : Les différentes façons d'attaquer passivement un device cryptographique en vue de casser l'algorithme de chiffrement qui se trouve dessus.

Dans la suite de cet ouvrage, on se concentrera sur un type précis d'attaque passive : l'attaque sur **l'analyse de la consommation de puissance**. Les sections suivantes décrivent le principe de fonctionnement de ce type d'attaque.

3.3.2 Introduction aux attaques par analyse de la consommation de puissance

Nous allons donc étudier un cas précis d'attaque passive : l'attaque par **l'analyse de la consommation de puissance**. Comme son nom l'indique, ce type d'attaque analyse la consommation de puissance du device cryptographique attaqué pour retrouver des informations sensibles. En effet, la consommation de courant d'un circuit électronique dépend de deux facteurs :

— Les opérations qui sont exécutées.
— Les données qui sont manipulées.

Ainsi, en mesurant un certain nombre de fois la consommation de puissance d'un circuit, il est possible de retrouver certaines informations comme les opérations exécutées (afin d'identifier un algorithme par exemple), les informations secrètes complètes (clé de chiffrement) ou partielles (poids de Hamming d'une clé de chiffrement). Pour ce faire, un oscilloscope est utilisé afin de capturer et d'enregistrer des données, appelées ***traces***, mesurées à partir des canaux auxiliaires du circuit électronique (dans notre cas, un FPGA). Pour réaliser la mesure, une résistance sera placée en série avec un canal du device cryptographique afin que l'oscilloscope soit capable de capturer une différence de potentielle minime. En effet, étant donné que les courants qui circulent sur le device cryptographique sont de valeurs très faibles (µA), un amplificateur est utilisé afin d'amplifier la différence de potentielle. La figure 13 ci-dessous présente le principe de mesure à l'oscilloscope.

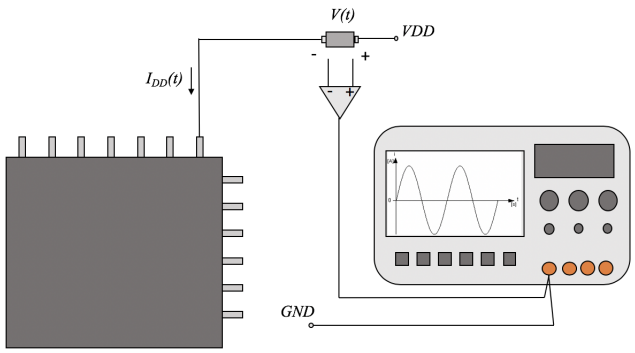

Figure 13 : Principe de mesure à l'oscilloscope.

Il existe différents types d'attaques par analyse de la consommation de puissance :
- Les attaques SPA (*Simple Power Analysis*)
- Les attaques DPA (*Differential Power Analysis*)
- Les attaques CPA (*Correlation Power Analysis*)

Nous allons en définir une plus précisément, c'est l'attaque CPA. C'est ce type d'attaque qui a été mis en place durant le stage afin de tenter de casser l'algorithme AES. Ainsi, en continuant l'introduction sur les attaques par analyse de la consommation de puissance, et en prenant le cas particulier d'une attaque dite CPA, nous pouvons dire que :
En supposant connu les messages clairs envoyés au device cryptographique et en supposant que ce dernier implémente l'algorithme AES, nous simulerons sur ordinateur le poids de Hamming de chaque donnée binaire obtenue en sortie de l'opération *SubBytes*. Ensuite, nous calculerons les différentes valeurs de coefficient de corrélation entre les traces de puissance capturées à l'oscilloscope et le poids de Hamming obtenu par simulation (sur ordinateur). Sur base de cette étude de la corrélation, nous serons (en principe) capable de déterminer la clé secrète, c'est-à-dire casser l'algorithme AES et ainsi exploiter les données confidentielles. La figure 14 présente vulgairement le principe général d'une attaque CPA.

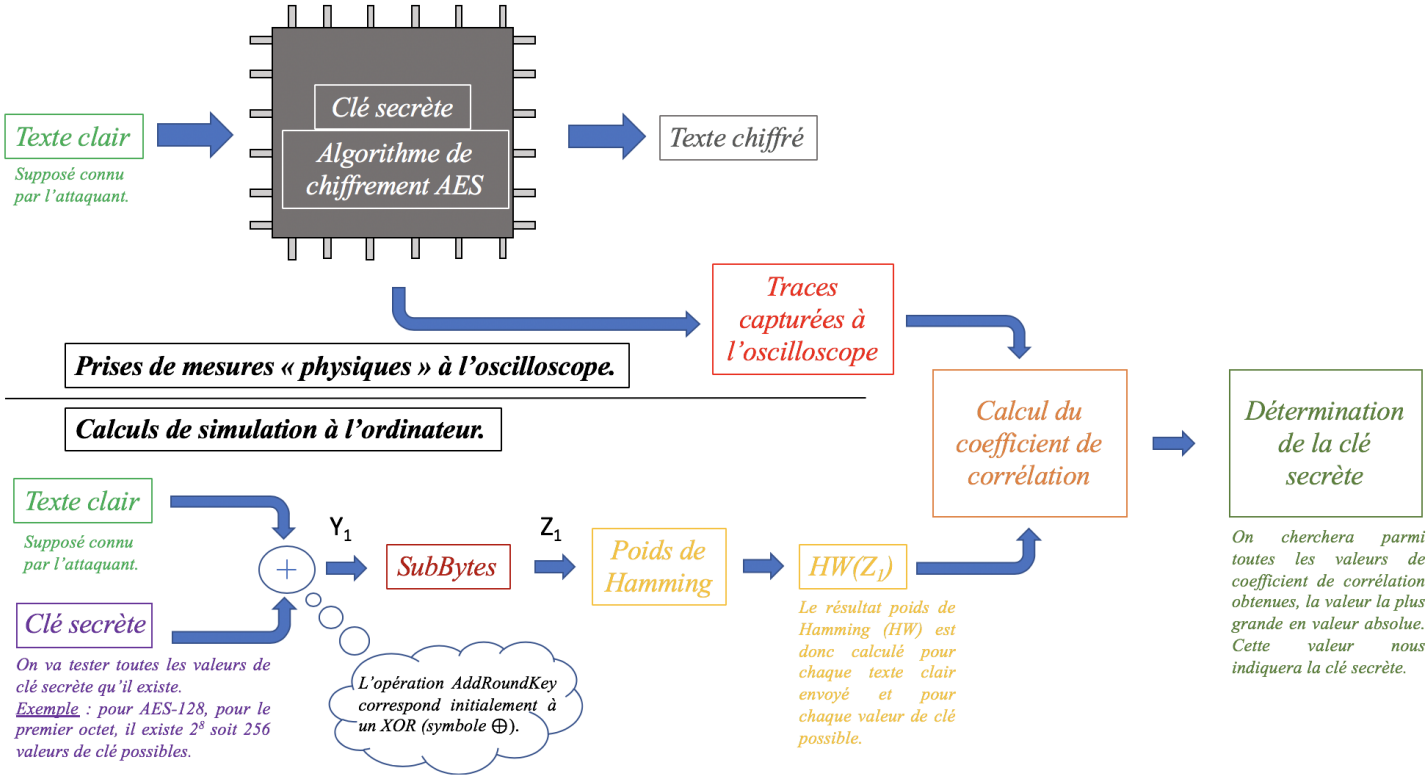

Figure 14 : Principe générale d'une attaque CPA.

Le principe d'une attaque par consommation de puissance, et plus particulièrement d'une attaque CPA, ayant été introduit de manière générale, nous allons maintenant revenir sur différentes notions citées ci-dessus afin de mieux les définir et ainsi mieux comprendre le raisonnement qui se cache derrière une attaque par calcul de corrélation (CPA).

3.4 Attaque CPA (*Correlation Power Analysis*)

3.4.1 Technologie CMOS

La ***technologie CMOS*** (pour *Complementary MOS*) est la technologie la plus répandue parmi toutes les technologies de semi-conducteurs. En effet, on la retrouve dans la majorité des systèmes informatiques modernes. En 2001, elle englobait 86% de la production mondiale des circuits intégrés. Pour cette raison, nous nous intéressons à leur conception afin de détecter des anomalies qui pourraient se révéler être utiles pour la cryptanalyse. Le nom de cette technologie vient du fait que toutes les fonctions logiques (portes OR, NAND, etc.) peuvent être réalisées moyennant l'utilisation d'une paire de transistors MOS complémentaires (N-MOS et P-MOS) associés symétriquement et fonctionnant en régime de commutation. Ainsi lorsqu'un des deux transistor MOS conduit, l'autre est par conséquent fermé. Grâce à ce principe, une porte logique CMOS ne consomme de l'énergie qu'au moment de la commutation. Cette caractéristique permet de distinguer le CMOS de toutes les autres technologies.

Pour expliquer le fonctionnement de cette technologie, on peut prendre un exemple simple : ***l'inverseur CMOS***. Un inverseur CMOS est simplement une fonction *NON*. Voici donc sa table de vérité :

Entrée	*Sortie*
0	*1*
1	*0*

Figure 15 : Table de vérité pour la fonction NON (inverseur CMOS).

La figure 16 ci-dessous présente le schéma de l'inverseur CMOS :

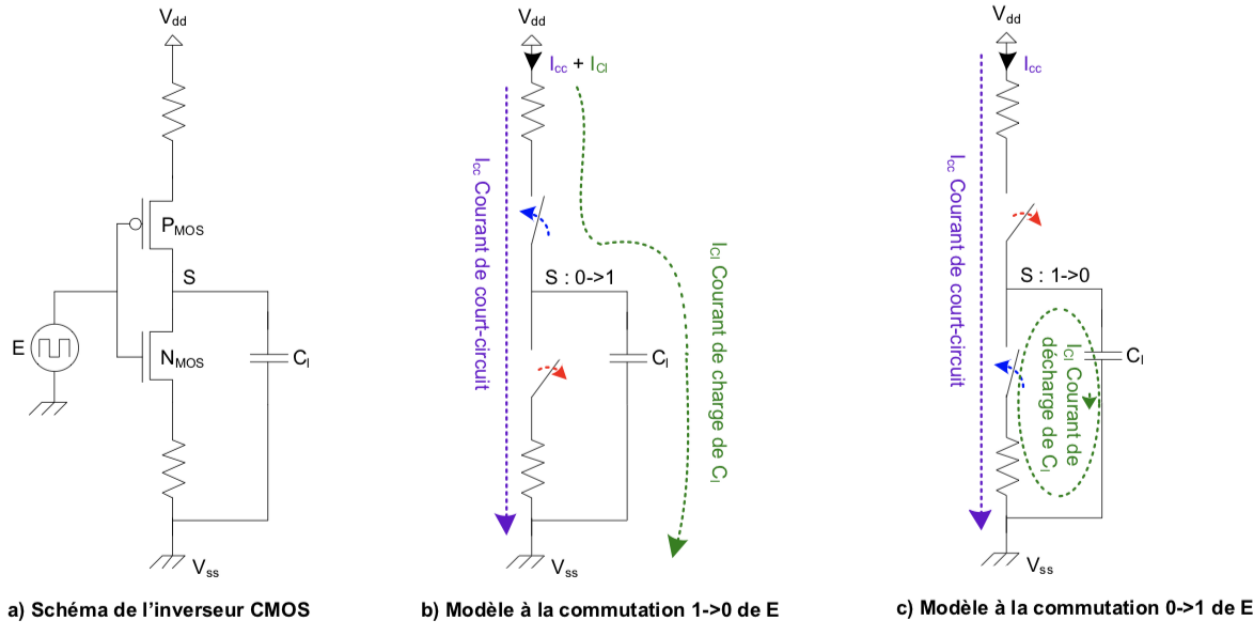

Figure 16 : Schéma de l'inverseur CMOS.

Si on applique à l'entrée un niveau haut, le transistor N est passant et le P est bloqué. On place ainsi la sortie au potentiel Vss (la masse), c'est-à-dire à l'état bas. Inversement, quand on met l'entrée à l'état bas, le transistor P est passant et le N est bloqué. La sortie est donc à l'état haut. On a donc bien réalisé une fonction inversion.

3.4.2 Puissance statique et dynamique

Il est évident que les circuits digitaux modernes consomment de la puissance lorsqu'ils exécutent des instructions (opérations) sur des données. Dans le domaine de la cryptanalyse, cette puissance va être mesurée et analysée afin de déterminer si le device cryptographique est attaquable ou non. Dans cette section, nous allons étudier plus particulièrement la consommation de puissance d'un type précis de circuit : les circuits utilisant la technologie CMOS (voir section 3.4.1 pour plus de précision). Cette technologie est très répandue et couvre la plupart des circuit digitaux modernes.

La consommation totale de puissance d'un circuit CMOS peut être obtenue en sommant les consommations de puissance respectives de chaque cellule logique du circuit CMOS. De cette façon, la consommation de puissance totale dépend essentiellement du nombre de cellules logiques dans le circuit CMOS.
En prenant comme exemple de cellule logique CMOS, l'inverseur CMOS (expliqué à la section 3.4.1), nous allons tenter de comprendre quand et pourquoi ces cellules CMOS dissipent de la puissance. Pour ce faire, il faut savoir que la consommation de puissance est essentiellement divisée en deux parties :

- La puissance statique (notée P_{stat}) : C'est la puissance qui est consommée lorsqu'il n'y a pas de commutation dans une cellule (c'est-à-dire dans l'inverseur). Autrement dit, c'est la puissance qui est consommée lorsque l'inverseur est en fonctionnement normal mais ne commute pas.
- La puissance dynamique (notée P_{dyn}) : C'est la puissance qui est consommée par une cellule si la sortie de cette cellule commute.

Ainsi, la puissance totale consommée par une cellule vaut la somme de ces deux composantes, soit :

$$P_{total} = P_{stat} + P_{dyn} \tag{1}$$

Mais de façon plus précise, que vaut la puissance statique ? De même, comment pourrait-on exprimer la puissance dynamique ?

Puissance statique :

Les cellules CMOS sont toujours construites de façon à ce que les deux transistors complémentaires ne soient jamais passants au même moment. En effet, si on reprend l'exemple de l'inverseur CMOS, lorsque on met le signal d'entrée à *GND* alors le transistor P1 est passant et N1 est bloqué. Par contre, lorsqu'on met le signal d'entrée à V_{DD}, le transistor P1 devient bloqué tandis que le transistor N1 devient passant. De cette façon, il n'y a pas de connexion directe entre V_{DD} et *GND*. Par conséquent, seul un petit pic de courant (notée I_{pic}) passe au travers de la cellule logique. La valeur de ce pic de courant est très très faible, de l'ordre du pA. Ainsi, la puissance statique peut se calculer de la façon suivante :

$$P_{stat} = I_{pic}.V_{DD} \tag{2}$$

Pour conclure, la consommation de puissance statique des circuits CMOS est typiquement très faible, nous pourrons donc la négliger (même si elle augmente significativement pour les circuits numériques modernes qui utilisent de très petites tailles de structure).

Puissance dynamique :

La consommation de puissance dynamique apparait typiquement lors d'une commutation des transistors. Une commutation est le passage d'un état haut à un état bas ou d'un état bas à un état haut. En réalité, il existe 4 transitions d'état possibles. Ces 4 possibilités sont reprises dans le tableau 17 ci-dessous. On constate que pour chaque transition possible, il y a présence de puissance statique. Cependant, il n'y présence de puissance dynamique que dans le cas d'une commutation, c'est-à-dire dans les deux transitions suivantes : 0-1 et 1-0.

La consommation de puissance totale dépend du type de cellule et de la technologie employée. Cependant, en général, on constate que :

- Lorsqu'il n'y a pas de commutation, la puissance totale reste plus ou moins identique.
- Lorsqu'il y a une commutation, la puissance totale augmente. Elle devient même supérieure à la puissance totale prise quand il n'y a pas de commutation.

Dans tous les cas, la consommation de puissance dynamique dépend des données qui sont manipulées. Ainsi, on peut conclure que la consommation de puissance dynamique des circuits CMOS constitue le facteur dominant dans la consommation de puissance totale.

Transitions	*Type de puissance consommée*
0 → 0	*Statique*
0 → 1	*Statique + Dynamique*
1 → 0	*Statique + Dynamique*
1 → 1	*Statique*

Figure 17 : Type de puissance consommée sur une cellule CMOS en fonction des 4 transitions d'état de sa sortie.

La consommation de puissance moyenne de chargement de la capacité ...

$$P_{chrg} = \frac{1}{T}\int_0^T p_{chrg}(t)dt = \alpha.f.C_L.V_{DD}^2 \tag{3}$$

La consommation de puissance moyenne qui est causée par les courants de court-circuit ...

$$P_{cc} = \frac{1}{T}\int_0^T p_{cc}(t)dt = \alpha.f.C_L.V_{DD}.I_{pic}.t_{cc} \tag{4}$$

Puissance : Comme vu à la section précédente (3.4.2), il faut analyser deux types de puissance. En ce qui concerne la puissance statique, celle-ci est négligeable lorsque l'entrée (E) est stable à l'état haut ou à l'état bas et en supposant que les fuites dans les transistors MOS sont maîtrisées. Par contre, en ce qui concerne la puissance dynamique consommée, celle-ci devient importante lors des commutations de l'entrée. Cette puissance dynamique dépend effectivement du type de commutation. Ainsi :

- Lors d'un passage de l'entrée (E) de l'état haut à l'état bas, l'alimentation fournit au travers du transistor P-MOS un courant de charge (I_{CI}) à la capacité de ligne (C_l) connectée à la sortie (S), plus un courant transitoire de court-circuit (I_{CC}) (lorsque les deux transistors sont simultanément en régime de fonctionnement linéaire).
- Lors d'un passage de l'entrée (E) de l'état bas à l'état haut, la capacité de ligne se décharge au travers du transistor N-MOS, l'alimentation fournit un courant transitoire de court circuit (I_{CC}).

3.4.3 Modèles de puissance

Distance de Hamming...

Poids de Hamming ...

3.4.4 L'attaque CPA en concret

La figure **??** ci-dessous présente le principe de fonctionnement d'une attaque CPA comme décrite ci-avant.

3.5 Simulation d'un tracé et d'une attaque par canal auxiliaire sur MATLAB

Afin de bien assimiler une attaque par canal auxiliaire, il m'a été demandé de tester, par des simulation sur le logiciel MATLAB, les notions théoriques développées précédemment. Deux exercices différents ont ainsi été réalisés.

1. **Simuler un point d'une trace et ensuite réaliser une attaque CPA sur ce point**. La première phase de la simulation a pour objectif de générer un point particulier d'une trace sur base de l'algorithme AES. La seconde phase de la simulation a pour but de réaliser une attaque par canal auxiliaire. Plus précisément, il s'agit d'une attaque CPA. Pour cette raison, seules les 2 premières étapes de l'algorithme AES sont nécessaires et seront donc simulées (*AddRoundKey*, *SubBytes*). À noter que, pour simplifier, cette attaque n'est réalisée que sur un seul byte de données et donc aussi un seul byte de clé.
2. **Réaliser une attaque CPA à partir de traces réelles**. Dans ce cas de figure, on connait 4 paramètres : les messages clairs envoyés (plaintexts), les traces capturées à l'oscilloscope, le nombre de traces ainsi que le nombre d'échantillons. Ainsi, sur base des traces qui nous sont fournies, l'objectif est de tenter de retrouver la clé secrète en réalisant une attaque CPA. La différence majeure avec l'exercice précédent est que l'on étudie une trace selon l'ensemble de points (les échantillons) qui la caractérise. Cet ensemble de traces étant par ailleurs fournit sur base de mesures réalisées à l'oscilloscope.

Exercice 1 : La figure 18 ci-dessous représente le schéma-block de la première phase de l'exercice 1, c'est-à-dire qu'il présente les différentes étapes à réaliser pour simuler un point d'une trace.

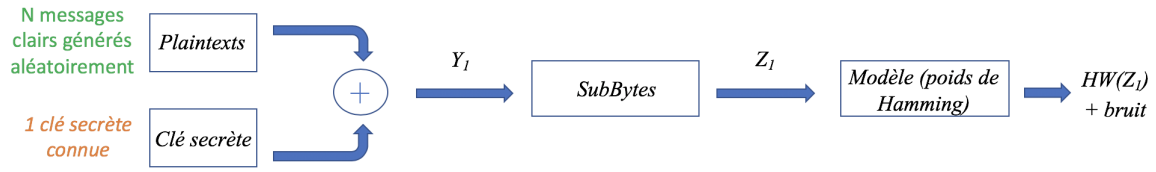

Figure 18 : Schéma-block permettant de comprendre la simulation d'un point d'une trace.

Sur le schéma-block, nous percevons deux entrées : la clé secrète (connue) utilisée pour le chiffrement ainsi que les N messages clairs devant être chiffrés. Nous réalisons ensuite les deux premières étapes de l'algorithme AES, à savoir les opérations *AddRoundKey* (correspondant à un *XOR*) et *SubBytes* respectivement. Le résultat obtenu à la sortie de l'opération *AddRoundKey* est noté Y_1 alors que le résultat obtenu à la sortie de l'opération *SubBytes* est noté Z_1. Ensuite, un modèle de puissance est utilisé afin d'imiter au mieux la consommation de puissance du circuit. Ce modèle de puissance est le *poids de Hamming*. Le but est donc de compter le nombre de bits à '1' pour chaque octet de données Z_1. Le résultat obtenu est noté $HW(Z_1)$. Enfin, une fois que le poids de Hamming a été calculé, on ajoute du bruit afin de rendre la simulation plus réelle. En effet, en réalité, lors d'une prise de mesure, un élément parasite vient toujours s'additionner au signal que l'on étudie, il s'agit de bruit électronique.

La figure 19 ci-dessous représente le schéma-block de la seconde phase de l'exercice 1. Il présente ainsi les différentes étapes à réaliser qui serviront *in fine* à réaliser l'attaque CPA.

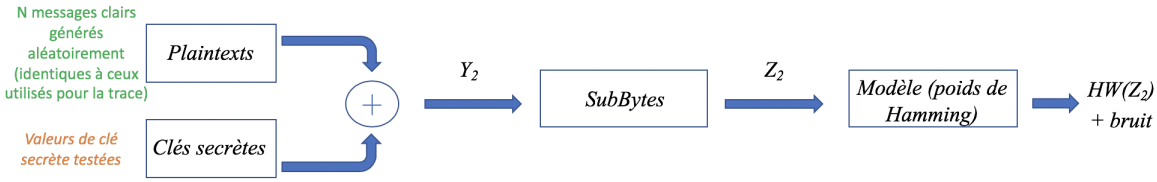

Figure 19 : Schéma-block permettant de comprendre la seconde phase de l'exercice 1.

La figure 19 est quasi identique à la figure 18. La seule différence concerne l'une des deux entrées. En effet, le but de l'attaque CPA est de retrouver la clé secrète utilisée dans la figure 18 pour chiffrer nos N messages. Ainsi, si on ne s'intéresse qu'à un seul octet de données et donc, par la même occasion, à un seul octet de clé, on va tester les 256 (2^8) valeurs de clé possibles. En notant respectivement Y_2 et Z_2 les résultats obtenus en sortie des opérations *AddRoundKey* et *SubBytes*, il ne nous restera plus qu'à calculer le poids de Hamming et à ajouter du bruit pour ensuite tenter de retrouver la clé secrète par calcul du coefficient de corrélation.

Concrètement, sur MATLAB, tous ces calculs vont être opérés sur des matrices. La figure 20 présente les différentes tailles de matrices utilisées pour réaliser la simulation.

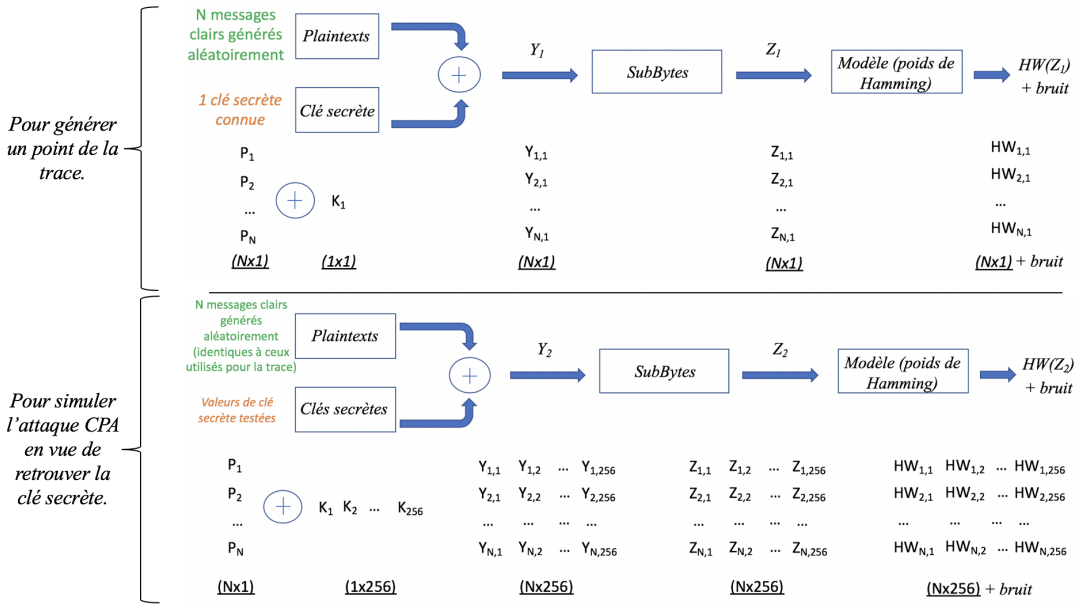

Figure 20 : Schéma-block permettant de visualiser la taille des différentes matrices employées pour la simulation.

Comme vu à la section 3.2, l'algorithme AES implémente deux matrices en entrée : la matrice STATE et la matrice clé. Ces deux matrices contiennent 16 éléments correspondants aux 16 octet de données (pour la matrice STATE) et aux 16 octets de clé (pour la matrice clé). Autrement dit, ces deux matrices sont de tailler 4x4. Pour le test, nous allons simplifier le procédé. En effet, en pratique les opérations s'exécutent octect par octet. Dans notre cas, on ne va s'intéresser qu'au premier octet de la matrice STATE et donc, par la même occasion, au premier octet de la matrice clé. ***Le but final de la simulation sera donc de retrouver le premier octet de la clé connaissant le premier octet des N messages clairs devant être chiffrés.*** Ainsi :

- La matrice STATE est représentée par une matrice de taille Nx1. N représente le nombre de messages clairs envoyés.
- La matrice clé, dans le cas de la génération d'un point de la trace, est représentée par une matrice de taille 1x1. Un seul octet de l'unique clé secrète est en effet utilisé.
- La matrice clé, dans le cas de la simulation par ordinateur, est représentée par une matrice de taille 1x256. En effet, le but étant de retrouver la valeur du premier octet de la clé secrète, il existe 2^8 soit 256 valeurs possibles.
- Y_1 et Y_2 sont des matrices de taille Nx1 etNx256 respectivement. En effet, pour Y_1, une seul clé est utilisée alors que pour Y_2, 256 valeurs de clé sont utilisées.
- Z_1 et Z_2 sont des matrices de taille Nx1 et Nx256 respectivement. En effet, l'opération *SubBytes* ne modifie pas la taille des données obtenues précédemment.
- Enfin, $HW(Z_1)$ et $HW(Z_2)$ sont des matrices de taille Nx1 et Nx256 pour les mêmes raisons que précédemment.

Pour rappel, le but final de la simulation est de retrouver le premier octet de la clé connaissant un point de la trace (obtenu par simulation) et connaissant les N messages clairs envoyés. Cela sera rendu possible en calculant le coefficient de corrélation pour chaque valeur de clé possible. Le figure 21 ci-dessous présente le calcul final qui permettra de retrouver le premier octet de la clé secrète.

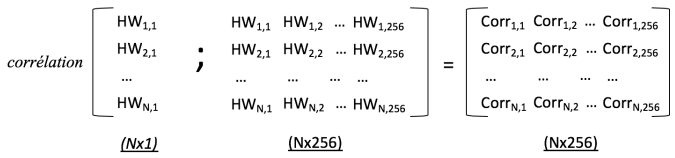

Figure 21 : Calcul du coefficient de corrélation entre un point d'une trace simulée et le poids de Hamming.

Les graphes 22, 23 et 24 ci-dessous présentent les résultats obtenus. Il s'agit de graphes indiquant les valeurs de coefficient de corrélation en fonction des 256 valeurs de clé testées. En toute logique, le coefficient de corrélation est maximum (en valeur absolue) pour la clé réellement utilisée pour le chiffrement des données. Cependant, on sait qu'en fonction du nombres de traces analysées, les valeurs du facteur de corrélation fluctuent dans un intervalle plus ou moins grand. Ainsi, au plus le nombre de traces sera grand, au plus l'intervalle sera petit et au plus ce sera facile de repérer la clé de chiffrement. En effet, si on observe les trois graphes ci-dessous, on peut remarquer que pour 10 traces, le coefficient de corrélation varie entre -0,83 et 0,8 ; pour 100 traces, le coefficient de corrélation varie entre -0,26 et 0,34 ; pour 1000 traces, le coefficient de corrélation varie entre -0,24 et 0,21. Pour chacun des trois graphes, on peut néanmoins distinguer la valeur de la clé secrète dont la valeur du coefficient de corrélation culmine à 0,98. La clé secrète utilisée vaut 200.

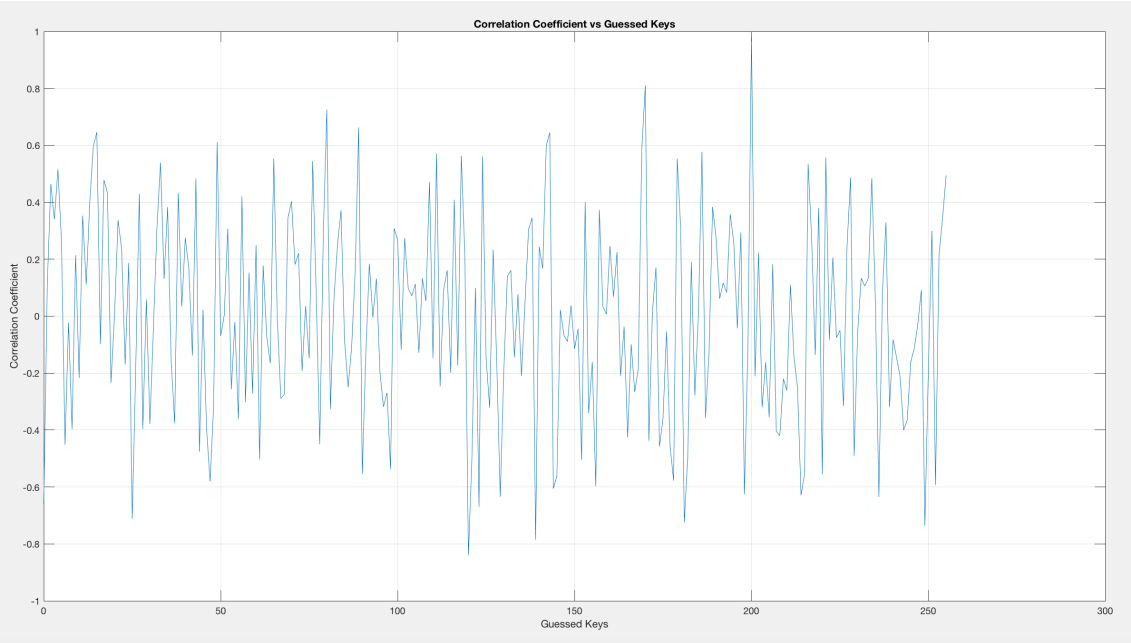

Figure 22 : Coefficient de corrélation en fonction de la valeur de la clé lorsqu'on analyse 10 traces.

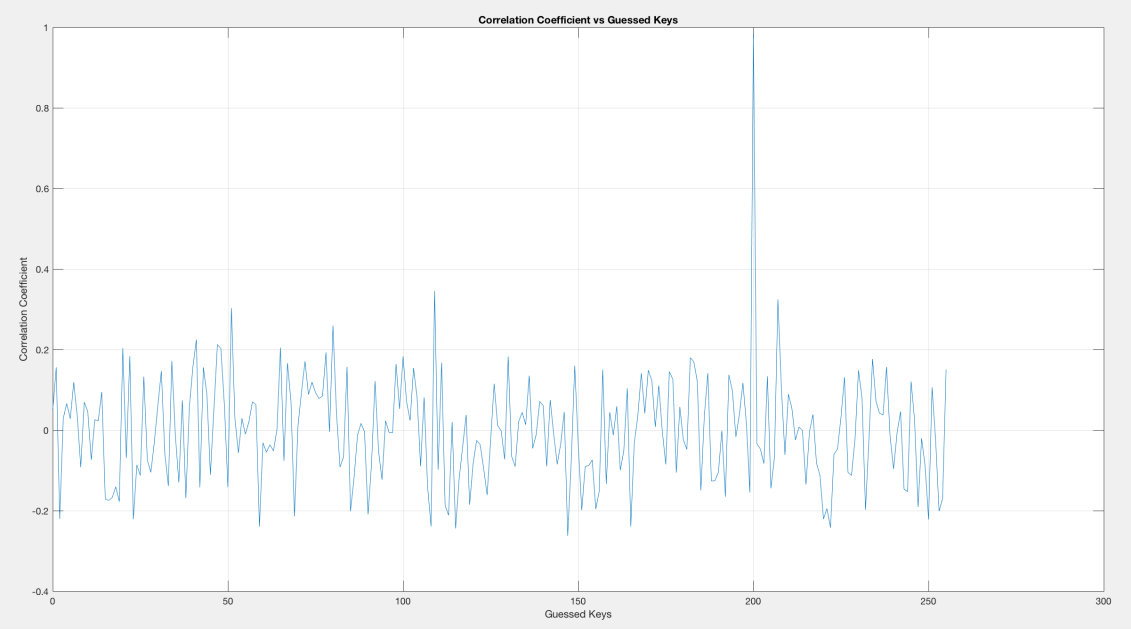

Figure 23 : Coefficient de corrélation en fonction de la valeur de la clé lorsqu'on analyse 100 traces.

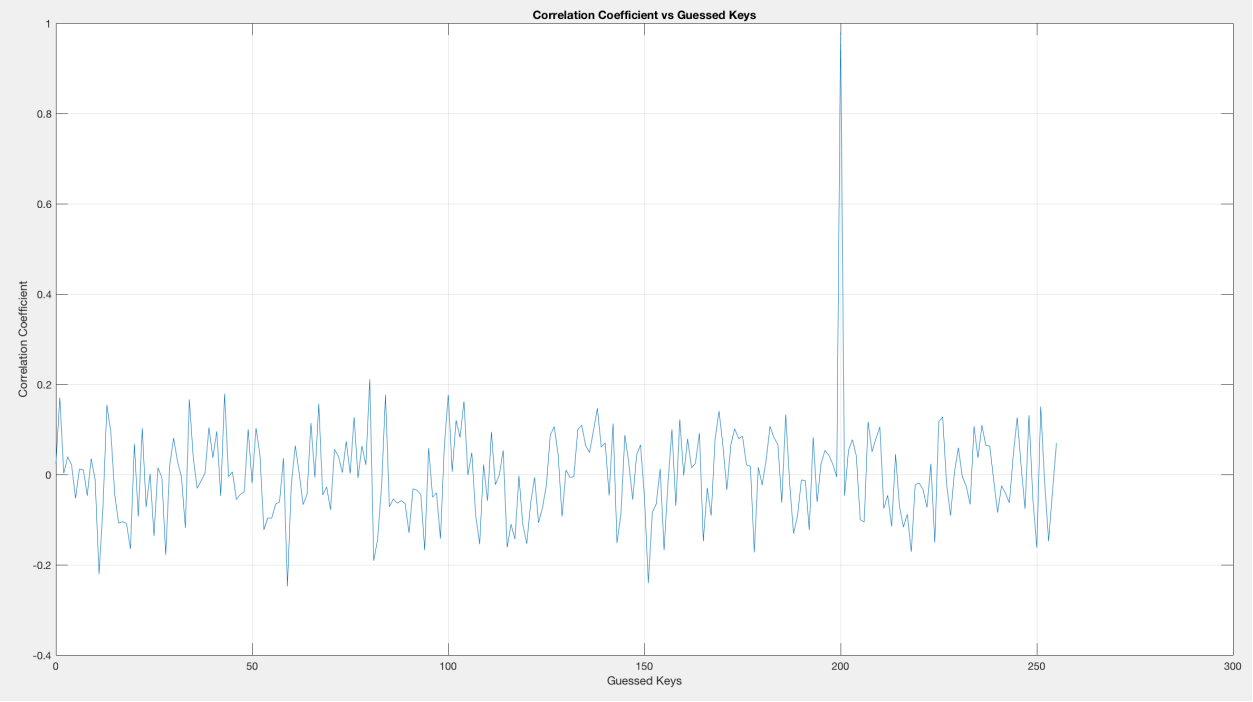

Figure 24 : Coefficient de corrélation en fonction de la valeur de la clé lorsqu'on analyse 1000 traces.

Exercice 2 : La figure 25 ci-dessous représente le schéma-block de l'exercice 2. Il confronte ainsi les mesures obtenues à l'oscilloscope avec le poids de Hamming calculé à partir des messages clairs. Pour l'exemple ci-dessous, on opère toujours sur le premier octet de données et le premier octet de la clé.

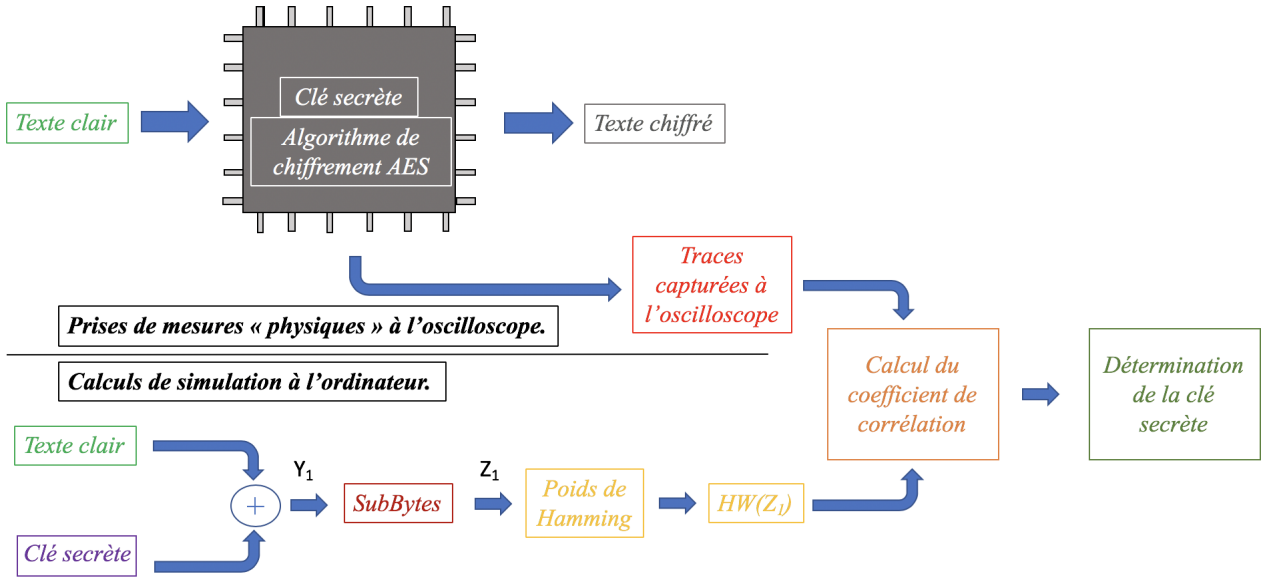

Figure 25 : Schéma-block permettant de comprendre le principe de fonctionnement du deuxième exercice.

3.6 Notions de statistiques et de t-test

Les notions de statistiques et de t-test présentent un intérêt certain en ce qui concerne la compréhension et l'analyse dans les side-channel attacks. Pour rappel, les traces de puissance enregistrées par un oscilloscope représentent des vecteurs de valeurs en tension. Ces valeurs de tension mesurées sont proportionnelles à la consommation de puissance d'un device cryptographique. En effet, l'oscilloscope sera toujours connecté en un endroit approprié de mesure sur le circuit du device cryptographique. En fonction du paramètrage de l'oscilloscope, on sera en mesure de déterminer la longueur des traces de puissance ainsi que le nombre de points enregistrés par seconde : c'est là que vont intervenir les notions de modèles statistiques. Ces modèles statistiques décrivent les traces de puissance enregistrées à l'oscilloscope de manière à comprendre et analyser, de façon simple, les attaques réalisées.

3.6.1 Composition des traces de puissance

Les attaques basées sur l'analyse de la consommation de puissance exploitent le fait que la consommation de puissance d'un device cryptographique dépend des **opérations qu'il exécute** et des **données qu'il manipule**. Ces deux informations vont ainsi permettre de définir différentes propriétés intéressantes. Pour chaque point analysé dans une trace de puissance, on notera :

— P_{op} **la composante dépendante de l'opération exécutée** ;
— P_{data} **la composante dépendante de la donnée manipulée**.

De plus, une troisième composante doit également être prise en compte. Cette composante fait référence au **bruit électrique** (aussi appelé bruit de fond) et sera notée P_{noise}. En effet, un signal est toujours affecté de petites fluctuations plus ou moins importantes. Ces fluctuations, dont les origines peuvent être diverses, sont appelées bruit électrique (ou simplement bruit). Le bruit est considéré comme un élément parasite aléatoire, c'est-à-dire qu'on ne sait pas le déterminer à l'avance. Au plus cette composante sera élevée et au plus l'analyse de la consommation de puissance sera difficile.

Ainsi, chaque point d'une trace de puissance peut être modélisé comme la somme des 3 composantes définies ci-dessus, soit : $P_{total} = P_{op} + P_{data} + P_{noise}$.

3.6.2 Caractéristiques d'un seul point de la trace : Distribution normale

Pour simplifier, on commence par définir les différentes composantes des traces de puissance en observant uniquement un seul point de la trace, c'est-à-dire en observant la consommation de puissance d'un device cryptographique à un instant fixé dans le temps. Le but étant de déterminer, à cet instant donné, la distribution de probabilité de P_{noise}, P_{data} et P_{op}. En théorie, ces 3 composantes suivent une **distribution normale** (aussi appelée loi normale).

En théorie des probabilités et en statistique, la **loi normale** est une loi de probabilité utilisée pour modéliser des phénomènes naturels issus de plusieurs événements aléatoires. Une densité de probabilité est une fonction qui permet de représenter une loi de probabilité : on parle aussi de fonction de densité. La fonction de densité caractérisant la loi normale dépend des deux paramètres que sont la moyenne (symbole μ) et l'écart-type (symbole σ). L'équation 5 présente son expression mathématique :

$$f(x) = \frac{1}{(\sqrt{2.\pi}.\sigma))}.e^{-\frac{1}{2}.(\frac{x-\mu}{\sigma})^2} \tag{5}$$

Le graphe 26 ci-dessous présente la courbe de la densité de probabilité d'une distribution normale. Cette courbe est généralement appelée **courbe de Gauss** (ou courbe en cloche). Dans le cas particulier où la loi normale est de moyenne nulle et d'écart type unitaire, on parle de ***loi normale centrée réduite***.

Ainsi, chacune des 3 composantes P_{noise}, P_{data} et P_{op} suivent une loi normale de moyenne μ et d'écart-type σ, notée : N(μ, σ). Plus précisément :

— La distribution du bruit électronique P_{noise} suit une distribution normale : N(0, σ).
— La distribution de P_{data} suit une distribution normale N(μ, σ) si les données manipulées sont distribuées uniformément.
— La distribution de P_{op} suit une distribution normale : N(μ, σ).

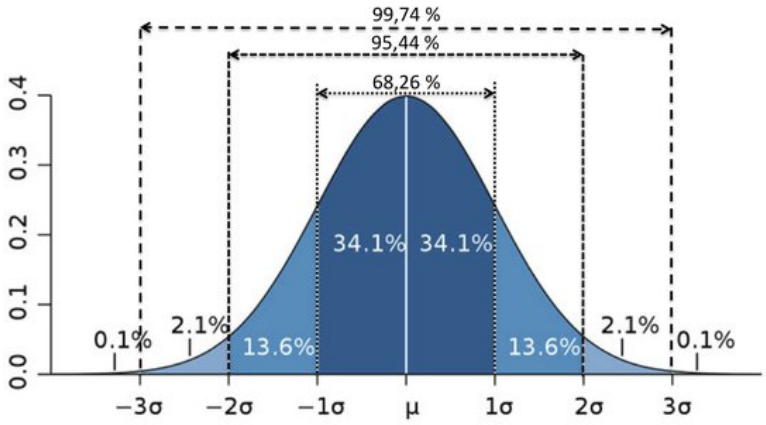

Figure 26 : Courbe de Gauss : fonction de densité d'une distribution normale.

Exemple : un exemple est réalisé pour démontrer que la distribution du bruit électronique suit bien une loi normale N(0, σ).

3.6.3 Caractéristiques de plusieurs points de la trace : Corrélation

Dans le section précédente 3.6.2, nous avons étudié les caractéristiques d'un seul point de la trace de puissance. On a alors du définir les notions de distribution normale. Dans cette section, on va définir des méthodes statistiques pour décrire les relations entre plusieurs points issus d'une trace de puissance.

En théorie, il existe une relation entre les différents points d'une trace de puissance : il s'agit d'une relation dite **linéaire**. Une relation linéaire est ... En pratique, cette relation peut être mesurée en calculant les coefficients de corrélation. Les coefficients de corrélation ont une valeurs comprises dans l'intervalle -1 à 1.

3.6.4 Définition Test-T

Le test-t de Student est un test statistique permettant de ***comparer les moyennes de deux groupes d'échantillons***. L'objectif est donc de savoir si les moyennes des deux groupes sont significativement différentes d'un point de vue statistique. Ce test de Student est dit paramétrique car, comme le montre la formule 6 (ci-dessous), celle-ci dépend de la moyenne et de l'écart-type des observations à comparer.

Il existe plusieurs variantes du test-t de Student :
- Le test-t de Student pour échantillon unique.
- Le test-t de Student comparant deux groupes d'échantillons indépendants (on parle de test de Student non apparié).
- Le test-t de Student comparant deux groupes d'échantillons dépendants (on parle de test de Student apparié).

Dans notre étude, on ne développera que le cas du test-t de Student comparant **deux groupes d'échantillons indépendants** (non appariés). Ce cas n'est utilisé que lorsque les deux groupes d'échantillons à comparer n'ont aucun lien. Autrement dit, il s'agit de comparer deux moyennes observées.

Pour appliquer un test-t de student non-apparié, il faut s'assurer de deux conditions :

1. La distribution des échantillons étudiés suit une distribution normale.
2. Les données des échantillons doivent être indépendantes. Autrement dit, les données d'un sujet ne sont pas censées avoir influencé les mesures faites sur un autre sujet.

Exemple pratique : Afin de bien comprendre la notion de test-t de Student non-apparié, voici un exemple concret. Nous avons un groupe de 100 individus (50 femmes et 50 hommes) pris au hasard au sein de la population. On se pose la question de savoir si le poids moyen des femmes est significativement différent de celui des hommes ?
Dans cet exemple on parle de test-t de Student non apparié car les deux groupes à comparer (hommes et femmes) n'ont aucun lien. Il s'agit donc de calculer le poids moyen des femmes et de celui des hommes et d'évaluer si la différence des ces moyennes est significative d'un point de vue statistique.

En pratique, on ne va pas utiliser le test-t de Student non-apparié mais une variante : le test-t de Welch. La raison est simple : le test-t de student non-apparié ne peut être utilisé que dans le cas où *les deux échantillons étudiés A et B suivent des lois normales de variances identiques*. Or, dans notre cas, pour l'analyse des traces de puissance, nos deux échantillons seront indépendants et ils n'auront donc pas la même variance. Pour ce faire, une variante du test-t de student non apparié existe, il s'agit du test-t de Welch.

Le *test-t de Welch* est une adaptation du test-t de Student non apparié permettant de comparer les moyennes de deux groupes d'échantillons indépendants lorsque leurs variances sont différentes.

Ce test permet de calculer la statistique de test t par la formule (6) suivante :

$$t = \frac{\mu_A - \mu_B}{\sqrt{\frac{\sigma_A^2}{n_A} + \frac{\sigma_B^2}{n_B}}} \tag{6}$$

Où :
- A et B sont les deux échantillons différents à comparer.
- μ_A et μ_B représentent la moyenne de l'échantillon A et celle de l'échantillon B, respectivement.
- n_A et n_B représentent la taille de l'échantillon A et celle de l'échantillon B, respectivement.
- σ_A et σ_B représentent l'écart-type de l'échantillon A et celui de l'échantillon B, respectivement.

Une notion importante à définir pour réaliser un test-t de student de manière général (et donc pour un test-t de Welch dans un cas précis) est la notion de degré de liberté.

En statistique, le *degré de liberté* désigne le nombre de valeurs aléatoires qui ne peuvent être déterminées ou fixées par une équation.

Le calcul du degré de liberté (*ddl*) du test-t de Welch se fait selon la formule (7) suivante :

$$ddl = \frac{(\frac{\sigma_A^2}{n_A} + \frac{\sigma_B^2}{n_B})^2}{\frac{\sigma_A^4}{n_A^2.(n_A-1)} + \frac{\sigma_B^4}{n_B^2.(n_B-1)}} \tag{7}$$

Exemple pratique : Afin de bien comprendre la notion de degré de liberté, voici un exemple concret. Si l'on cherche deux nombres dont la somme est 30, aucun des deux nombres ne peut être directement déterminé par la simple équation X + Y = 30. En effet, X peut être choisi arbitrairement, mais dans ce cas, il n'y a plus de choix pour Y. Ainsi, si on choisit 20 comme valeur pour X, Y vaut obligatoirement 10. Il y a donc deux variables aléatoires (X,Y), mais un seul degré de liberté.

L'objectif final d'un test statistique consiste à rejeter ou non une certaine hypothèse de départ, appelée ***hypothèse nulle***. Dans le cas où cette hypothèse nulle est rejetée, une ***hypothèse alternative*** doit être définie. Ainsi, on définit ces deux hypothèse contradictoires de la façon suivante :

- ***L'hypothèse nulle (notée H_0)*** : C'est l'hypothèse selon laquelle les paramètres de deux échantillons étudiés sont égaux. Dans le cas précis du test-t de Welch, ce paramètre est la **moyenne**. Ainsi, l'hypothèse nulle du test-t de Welch s'énonce : *La moyenne de l'échantillon A est égale à la moyenne de l'échantillon B, soit* $\mu_A = \mu_B$.
- ***L'hypothèse alternative (notée H_1)*** : C'est l'hypothèse selon laquelle les paramètres de deux échantillons étudiés sont inégaux ou sont inférieurs ou supérieurs l'un par rapport à l'autre. Dans ce cas, on distingue deux types de test :
  - ***Test bilatéral*** : Un test bilatéral est associé à une hypothèse alternative selon laquelle le signe de la différence potentielle entre les deux paramètres comparés est inconnu. Par exemple, si nous comparons les moyennes de deux échantillons A et B. Avant de mettre en place l'expérimentation et de lancer le test, nous ne savons pas avec certitude si A serait inférieure à B ou le contraire dans la situation où une différence entre A et B serait mise en relief par le test. Ceci nous conduit à opter pour un test bilatéral, associé à l'hypothèse alternative suivante : $\mu_A \neq \mu_B$.
  - ***Test unilatéral*** : Un test unilatéral est associé à une hypothèse alternative selon laquelle le signe de la différence potentielle entre les deux paramètres comparés est connu avant le lancement de l'expérimentation et du test. En reprenant l'exemple sur la comparaison des moyennes, l'hypothèse alternative associée à un test unilatéral peut être écrite de la sorte : $\mu_A < \mu_B$ ou $\mu_A > \mu_B$, en fonction du signe attendu de la différence.

Pour résumer ce qui vient d'être dit, lorsqu'on réalisera un test-t de Welch, on définira deux hypothèses :

$H_0 : \mu_A = \mu_B$

$H_1 : \mu_A \neq \mu_B$ (test bilatéral)

La figure 27 présente respectivement le principe de test unilatéral gauche, bilatéral et unilatéral droit.

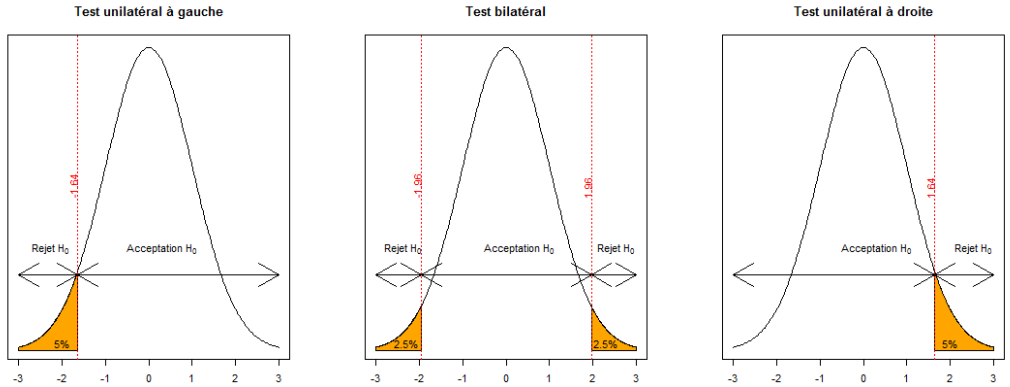

Figure 27 : Représentation graphique des principes de test unilatéral et bilatéral.

Aucun test statistique n'est fiable à 100%. Les tests sont basés sur des probabilités. Par conséquent, il existe toujours un risque de tirer une mauvaise conclusion. En statistiques, lorsqu'on effectue un test d'hypothèse, on définit deux types d'erreurs :

— ***L'erreur de type 1 (notée α)*** : Une erreur de type 1 est commise lorsque l'hypothèse nulle est vérifiée mais qu'on la rejette : on se trompe dans la conclusion du test. La probabilité de commettre ce type d'erreur est représentée par le symbole α, qui désigne le *seuil de signification* définit pour le test d'hypothèse.
Exemple pratique : Un niveau α de 0,05 indique que nous sommes disposés à avoir 5% de chances de rejeter l'hypothèse nulle à tort. Pour réduire ce risque, nous devons réduire la valeur de α.

— ***L'erreur de type 2 (notée β)*** : Une erreur de type 2 est commise lorsque l'hypothèse nulle est fausse mais que nous ne la rejetons pas. La probabilité de commettre ce type d'erreur est représenté par le symbole β. Nous ne nous intéresserons pas à ce type d'erreur dans la suite de ce rapport.

La tableau 28 résume les deux types d'erreurs possibles.

	H_0 *vraie*	H_0 *fausse*
Accepter H_0	$1-\alpha$	β
Rejeter H_0	α	$1-\beta$

Figure 28 : Tableau d'erreurs : erreur de type 1 (α) et erreur de type 2 (β).

Le ***seuil de signification***, noté α, représente donc une valeur, comprise entre 0 et 1 (ou exprimée en pour-cent), qui va être comparée à la valeur-p (définie ci-après) en vue de tirer des conclusions sur un test réalisé. Plus précisément, ce seuil de signification va indiquer le pourcentage de probabilité de rejeter l'hypothèse nulle à tort. Autrement dit, il va indiquer le pourcentage de probabilité de s'être trompé dans la conclusion d'un test. La valeur de ce seuil de signification vaut en général 5%, 2% ou encore 1%.

La ***valeur-p*** (*p-value* en anglais, notée p) est une nombre utilisé en statistique et qui doit être défini afin de pouvoir tirer des conclusion sur l'analyse du test réalisé. Ce nombre représente la probabilité de faire une erreur de type 1, c'est-à-dire de rejeter H_0 alors qu'elle est vraie. En statistique, la procédure généralement employée consiste à comparer la valeur-p à un seuil préalablement défini (le seuil de signification). Si :

— $p < \alpha$, on rejette l'hypothèse nulle en faveur de l'hypothèse alternative, et le résultat du test est déclaré statistiquement *significatif*.

— $p \geq \alpha$, on ne rejette pas l'hypothèse nulle, et on ne peut rien conclure quant aux hypothèses formulées. Le résultat du test est déclaré statistiquement *non-significatif*.

Pour résumer ce qui vient d'être dit, on doit comparer le seuil de signification (α) avec la valeur-p (p) :

α représente un seuil défini par nous-même avant l'exécution du test.

p représente une valeur calculée, obtenue après l'exécution du test.

Si $p < \alpha$, on rejette H_0 en faveur de H_1, et le résultat du test est déclaré statistiquement *significatif*.

Si $p \geq \alpha$, on ne rejette pas H_0. Le résultat du test est déclaré statistiquement *non-significatif*.

Exemple pratique : On souhaite savoir si, dans une certaine population, les femmes fument autant que les hommes. On décide de faire un test avec un seuil de signification (α) de 5% (risque d'erreur accepté). Dans ce test, l'hypothèse nulle est : *il n'y a pas de différence entre hommes et femmes dans la population.* Le premier objectif de ce test est de savoir s'il est raisonnable de rejeter cette hypothèse nulle. Si on note :

— f1 et f2, le nombre de fumeurs observés respectivement chez les hommes et les femmes dans l'échantillon.

— δ = f1—f2, la différence entre les valeurs observées chez les hommes et les femmes dans l'échantillon.

— α, le seuil de signification, c'est-à-dire le risque d'erreur accepté, fixé à l'avance (ici $\alpha = 5\%$).

— p, la probabilité, si l'hypothèse nulle était vraie (c'est-à-dire en l'absence de différence entre hommes et femmes dans la population) que le hasard seul explique une différence au moins aussi grande que la différence δ effectivement observée (des formules permettent de calculer cette probabilité).

La règle de décision est la suivante :

- si p $\geq \alpha$: on admet que la différence observée peut provenir des fluctuations d'échantillonnage. On ne peut pas rejeter l'hypothèse nulle (absence de différence dans la population). On conclut donc que la différence n'est pas significative au seuil de 5%.
- si p $< \alpha$: on n'admet plus que la différence observée peut être due au hasard. On rejette donc l'hypothèse nulle. On conclut que la différence est significative à 5%.

Ainsi, pour en revenir au test-t de Welch, après calcul de la valeur de t et de la valeur du *degré de liberté*, on va pouvoir conclure sur le test : *l'hypothèse nulle doit-elle être acceptée ou rejetée ? Autrement dit, la différence des moyennes de chacun des deux échantillons est-elle significative ou non ?*

Pour répondre à cette question, il faut réaliser un test bilatéral. Connaissant le degré de liberté (ddl) et la valeur de la statistique de test t, on peut déterminer la valeur-p à partir d'un tableau donné (voir annexe 29) en vue de tirer une conclusion sur le test. En prenant un seuil de signification (α) de 0,05, on conclut :

Si $p < \alpha$, on rejette H_0 en faveur de H_1, et le résultat du test est déclaré statistiquement *significatif.*

Si $p \geq \alpha$, on ne rejette pas H_0. Le résultat du test est déclaré statistiquement *non-significatif.*

3.7 Simulation d'un t-test sur MATLAB

4 Conclusion

Crédits

- Tableau 29 provenant de :
  Le blog officiel de lemakistatheux sur le test de Welch : https://lemakistatheux.files.wordpress.com/2013/05/add5.png
- Figure ?? provenant de :
  Par Yves-Laurent (Travail personnel) [GFDL (http://www.gnu.org/copyleft/fdl.html), CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)], de Wikimedia Commons
- Figure ?? provenant de :
  Par Inductiveload (Travail personnel) [Public domain], de Wikimedia Commons
- Figure ?? provenant de :
  Par Toriicelli (Travail personnel) [Public domain], de Wikimedia Commons
- Figure ?? provenant de :
  Par Daniel Braun [GFDL (http://www.gnu.org/copyleft/fdl.html), CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/) ou CC BY 2.5 (https://creativecommons.org/licenses/by/2.5)], de Wikimedia Commons

Références

- https://www.emse.fr/~dutertre/documents/synth_AES128.pdf
- https://www.j3ea.org/articles/j3ea/pdf/2012/01/j3ea12004.pdf
- Attaques par canaux auxiliaires : nouvelles attaques, contre-mesures et mises en oeuvre (thèse)
- From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces
- https://support.minitab.com/fr-fr/minitab/18/help-and-how-to/statistics/basic-statistics/supporting-topics/basics/type-i-and-type-ii-error/
- https://www.j3ea.org/articles/j3ea/pdf/2012/01/j3ea12004.pdf
- https://www.emse.fr/~dutertre/documents/synth_AES128.pdf
- http://www.sthda.com/french/wiki/test-t-de-welch
- https://hal.archives-ouvertes.fr/hal-00753215/document
  A voir :
- https://tel.archives-ouvertes.fr/tel-00937136/document
- http://pagesped.cahuntsic.ca/sc_sociales/psy/methosite/consignes/testt.htm
- http://www.pifo.uvsq.fr/epideao/esp/chap_2/hypothse_nulle.html
- http://www.ta-formation.com/acrobat-cours/bruit.pdf
- http://www.sthda.com/french/wiki/table-de-student-ou-table-t
- http://www.sthda.com/french/wiki/test-t-de-welch
- https://icwww.epfl.ch/~chappeli/it/courseFR/I2subsec_Hdist.php
- http://www.montefiore.ulg.ac.be/~dumont/pdf/crypto09-10.pdf
- https://perso.uclouvain.be/fstandae/PUBLIS/196.pdf
- https://help.xlstat.com/customer/fr/portal/articles/2062454-quelle-est-la-différence-entre-un-test-bilatéral-et-un-test-unila
  b_id=9283
- http://homepages.ulb.ac.be/~cverhoev/BMOL-G4400/chapitre1.pdf
-

A Table de Welch (ou table t)

α (bilatéral)	50 %	60 %	70 %	80 %	90 %	95 %	98 %	99 %	99,5 %	99,8 %	99,9 %
$1-\gamma$ (unilatéral)	75 %	80 %	85 %	90 %	95 %	97,5 %	99 %	99,5 %	99,75 %	99,9 %	99,95 %
k											
1	1,000	1,376	1,963	3,078	6,314	12,71	31,82	63,66	127,3	318,3	636,6
2	0,816	1,061	1,386	1,886	2,920	4,303	6,965	9,925	14,09	22,33	31,60
3	0,765	0,978	1,250	1,638	2,353	3,182	4,541	5,841	7,453	10,21	12,92
4	0,741	0,941	1,190	1,533	2,132	2,776	3,747	4,604	5,598	7,173	8,610
5	0,727	0,920	1,156	1,476	2,015	2,571	3,365	4,032	4,773	5,893	6,869
6	0,718	0,906	1,134	1,440	1,943	2,447	3,143	3,707	4,317	5,208	5,959
7	0,711	0,896	1,119	1,415	1,895	2,365	2,998	3,499	4,029	4,785	5,408
8	0,706	0,889	1,108	1,397	1,860	2,306	2,896	3,355	3,833	4,501	5,041
9	0,703	0,883	1,100	1,383	1,833	2,262	2,821	3,250	3,690	4,297	4,781
10	0,700	0,879	1,093	1,372	1,812	2,228	2,764	3,169	3,581	4,144	4,587
11	0,697	0,876	1,088	1,363	1,796	2,201	2,718	3,106	3,497	4,025	4,437
12	0,695	0,873	1,083	1,356	1,782	2,179	2,681	3,055	3,428	3,930	4,318
13	0,694	0,870	1,079	1,350	1,771	2,160	2,650	3,012	3,372	3,852	4,221
14	0,692	0,868	1,076	1,345	1,761	2,145	2,624	2,977	3,326	3,787	4,140
15	0,691	0,866	1,074	1,341	1,753	2,131	2,602	2,947	3,286	3,733	4,073
16	0,690	0,865	1,071	1,337	1,746	2,120	2,583	2,921	3,252	3,686	4,015
17	0,689	0,863	1,069	1,333	1,740	2,110	2,567	2,898	3,222	3,646	3,965
18	0,688	0,862	1,067	1,330	1,734	2,101	2,552	2,878	3,197	3,610	3,922
19	0,688	0,861	1,066	1,328	1,729	2,093	2,539	2,861	3,174	3,579	3,883
20	0,687	0,860	1,064	1,325	1,725	2,086	2,528	2,845	3,153	3,552	3,850
21	0,686	0,859	1,063	1,323	1,721	2,080	2,518	2,831	3,135	3,527	3,819
22	0,686	0,858	1,061	1,321	1,717	2,074	2,508	2,819	3,119	3,505	3,792
23	0,685	0,858	1,060	1,319	1,714	2,069	2,500	2,807	3,104	3,485	3,767
24	0,685	0,857	1,059	1,318	1,711	2,064	2,492	2,797	3,091	3,467	3,745
25	0,684	0,856	1,058	1,316	1,708	2,060	2,485	2,787	3,078	3,450	3,725
26	0,684	0,856	1,058	1,315	1,706	2,056	2,479	2,779	3,067	3,435	3,707
27	0,684	0,855	1,057	1,314	1,703	2,052	2,473	2,771	3,057	3,421	3,690
28	0,683	0,855	1,056	1,313	1,701	2,048	2,467	2,763	3,047	3,408	3,674
29	0,683	0,854	1,055	1,311	1,699	2,045	2,462	2,756	3,038	3,396	3,659
30	0,683	0,854	1,055	1,310	1,697	2,042	2,457	2,750	3,030	3,385	3,646
40	0,681	0,851	1,050	1,303	1,684	2,021	2,423	2,704	2,971	3,307	3,551
50	0,679	0,849	1,047	1,299	1,676	2,009	2,403	2,678	2,937	3,261	3,496
60	0,679	0,848	1,045	1,296	1,671	2,000	2,390	2,660	2,915	3,232	3,460
80	0,678	0,846	1,043	1,292	1,664	1,990	2,374	2,639	2,887	3,195	3,416
100	0,677	0,845	1,042	1,290	1,660	1,984	2,364	2,626	2,871	3,174	3,390
120	0,677	0,845	1,041	1,289	1,658	1,980	2,358	2,617	2,860	3,160	3,373
∞	0,674	0,842	1,036	1,282	1,645	1,960	2,326	2,576	2,807	3,090	3,291

Figure 29 : La table de Welch donne la probabilité α pour que t égale ou dépasse, en valeur absolue, une valeur donnée, en fonction du nombre de degrés de liberté (d.d.l.)..