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.

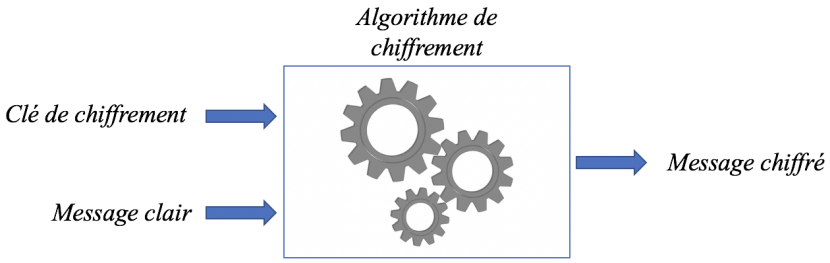

Figure 1 : L'algorithme de chiffrement, caractérisé par diverses opérations mathématiques, utilise une clé de chiffrement en entrée pour chiffrer un message clair. Cela produit un message chiffré, non compréhensible pour une personne ne connaissant pas la clé de chiffrement.

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 publique, 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.

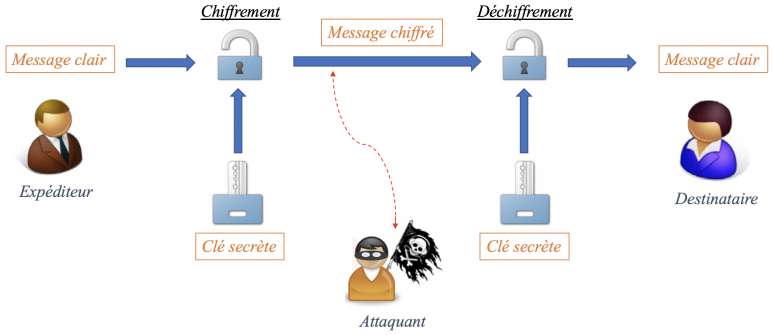

Figure 2 : Chiffrement symétrique : Une seule clé est utilisée pour chiffrer et déchiffrer les messages.

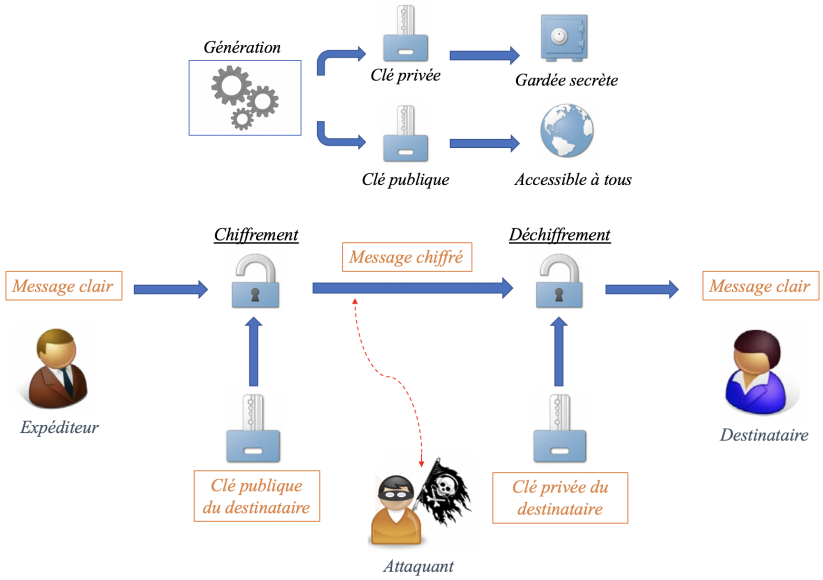

Figure 3 : Chiffrement asymétrique : Un clé publique est utilisée pour chiffrer le message et une clé privée est utilisée pour le déchiffrer.

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 longueurs : 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 avec les side-channel attacks, 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 soit 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) appelé STATE contient l'ensemble des résultats intermédiaires résultant des diverses opérations que subissent les données (depuis leur état initial).

La figure 4 présente 3 matrices : 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).

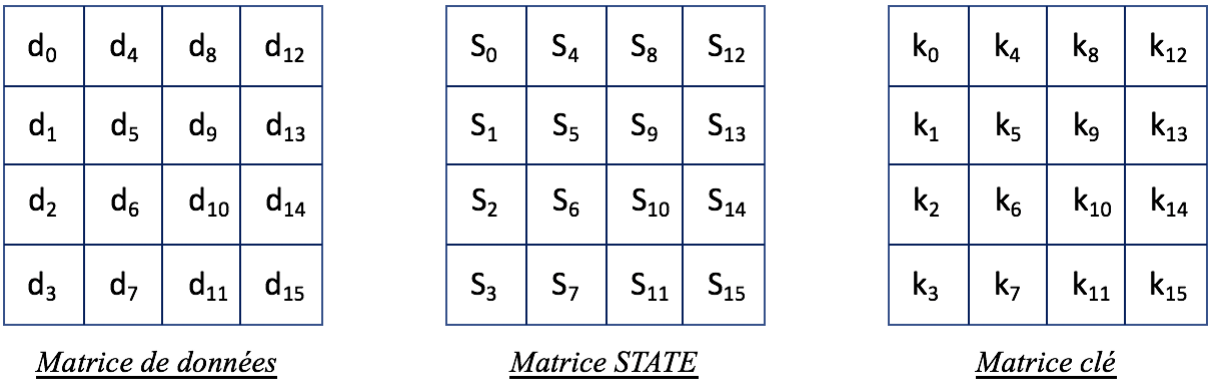

Figure 4 : Les 3 matrices utilisées par l'algorithme AES.

Remarque : En réalité, 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. En résumé, 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*, *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.

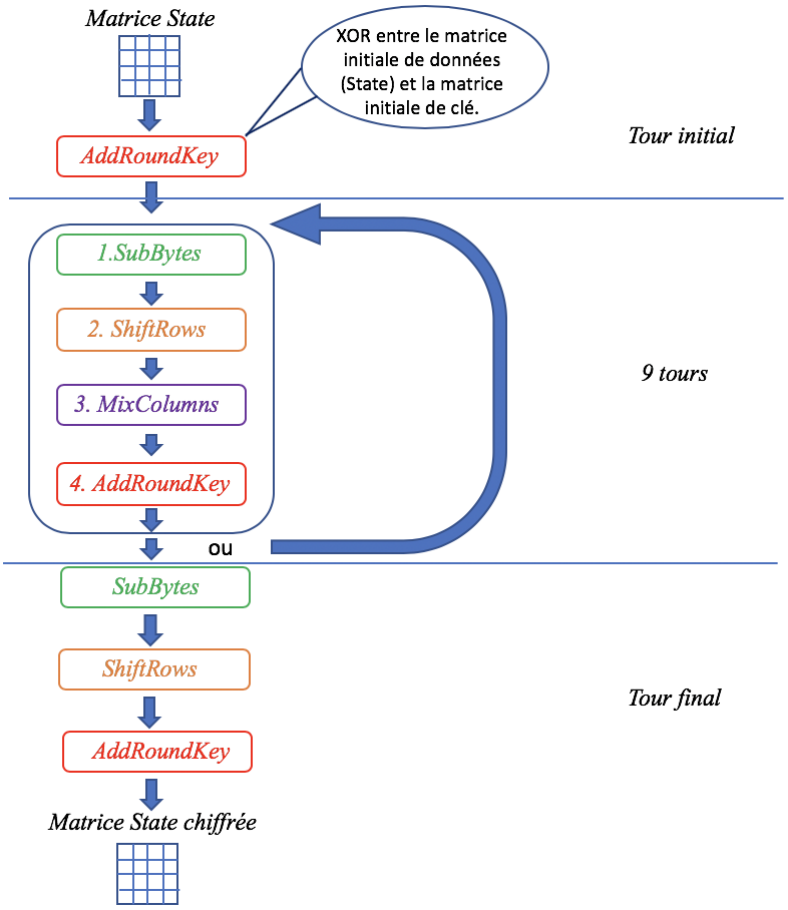

Figure 5 : Principe de fonctionnement de l'algorithme AES-128.

Fonctionnement :

Initialement, 2 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 :

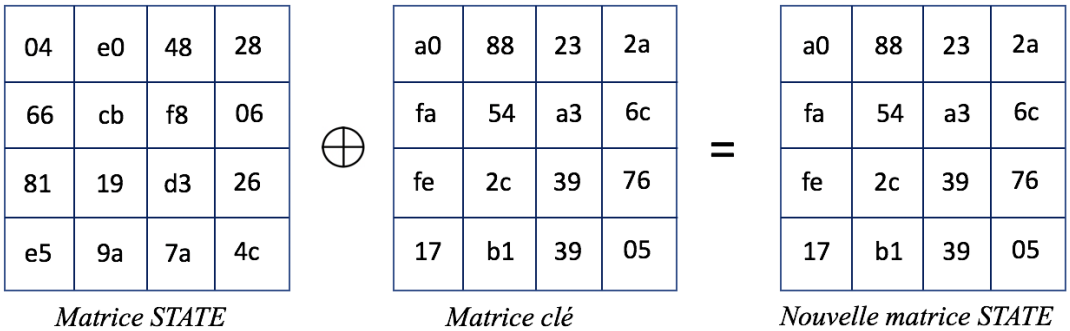

Figure 6 : Opération XOR entre la matrice STATE et la matrice clé.

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* :

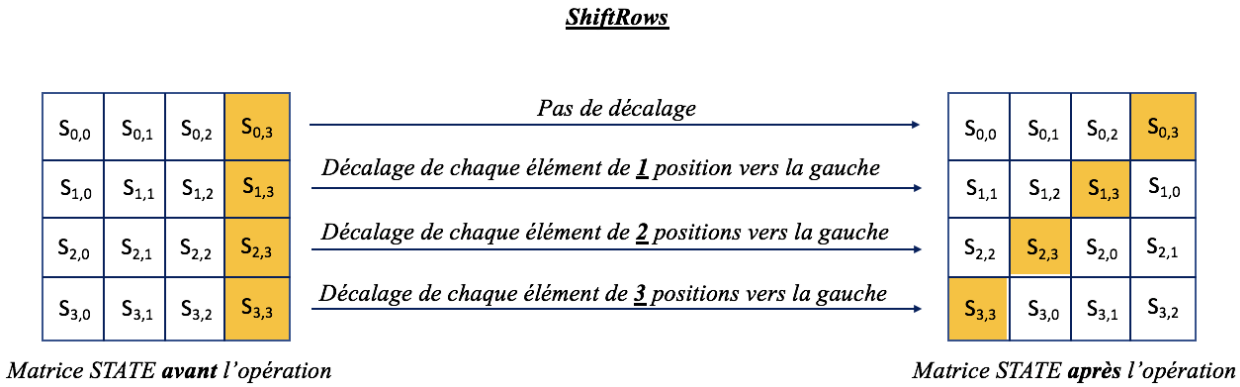

Figure 7 : Opération *subBytes* exécutée sur la matrice STATE.

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* :

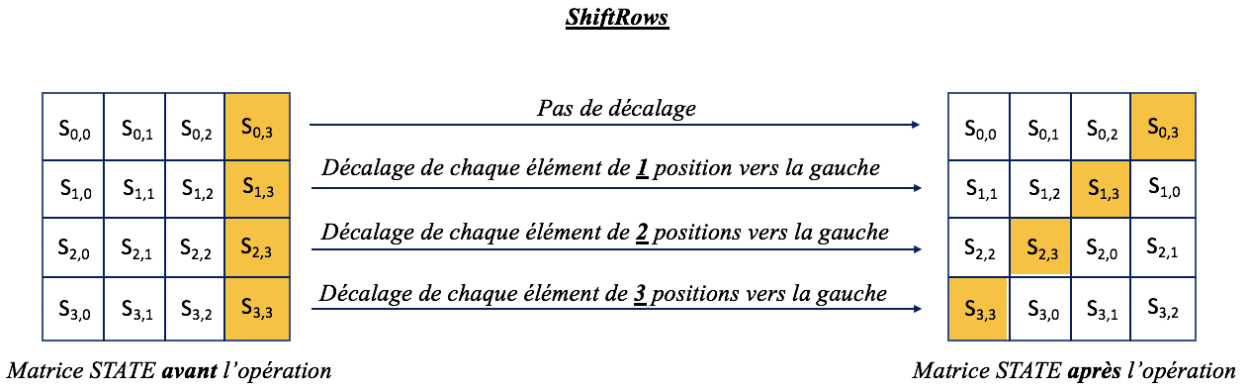

Figure 8 : Opération *ShiftRows* exécutée sur la matrice STATE.

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

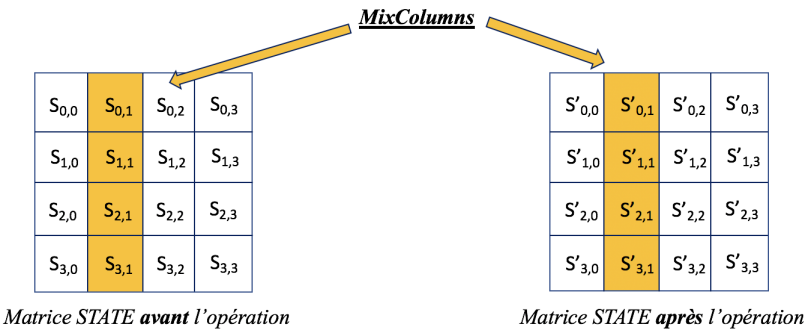

Figure 9 : Opération *MixColumns* exécutée sur la matrice STATE.

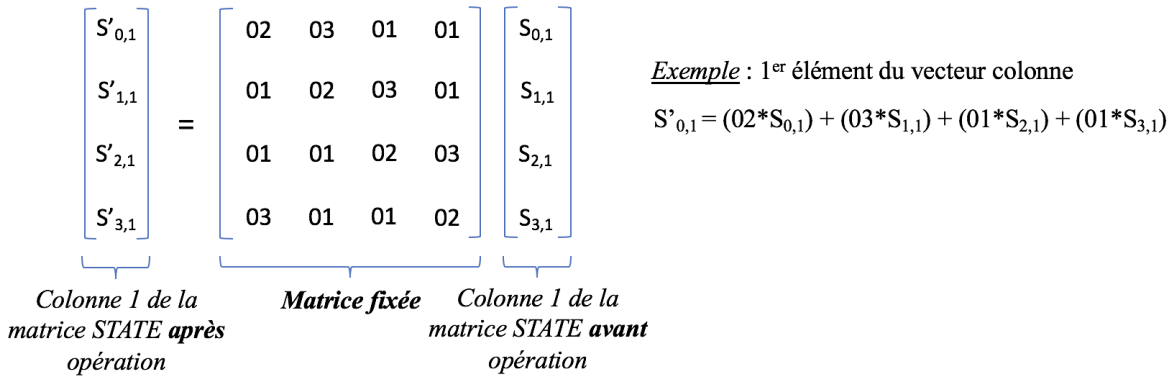

Figure 10 : Exemple de l'opération *MixColumns* exécutée sur la deuxième colonne de la matrice STATE.

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 Notions de *Side-Channel Attacks*

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.

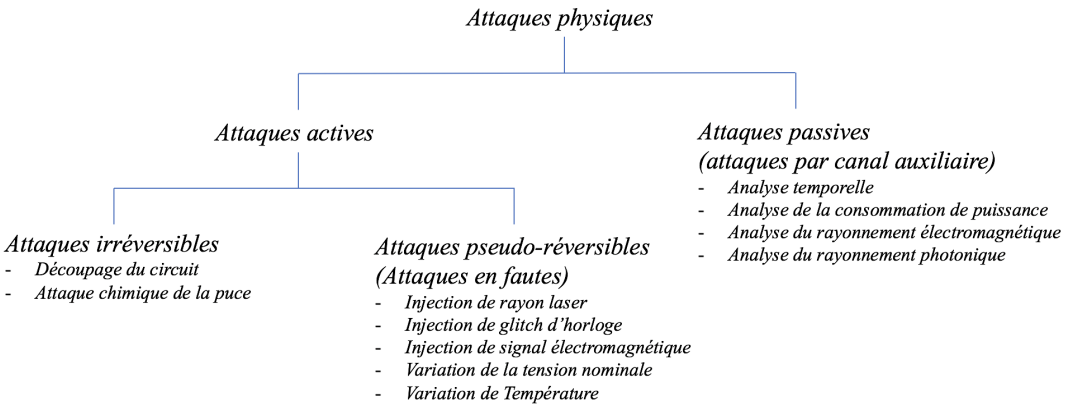

Figure 11 : Les 2 grandes familles d'attaques physiques possibles. La suite de ce rapport se concentre essentiellement sur les attaques physiques dites passives.

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. 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.

Dans notre cas, l'attaque portera sur l'analyse de la consommation de puissance. Autrement dit, un

oscilloscope sera utilisé pour capturer et enregistrer des données, appelées ***traces***, provenant des canaux auxiliaires. Sur base de ces traces, divers procédés seront mis en oeuvre afin de casser l'algorithme et ainsi exploiter les données confidentielles.

Nous allons définir deux types d'attaques :
— Les attaque (*Differential Power Analysis*)
— Les attaque CPA (*Correlation Power Analysis*)

Attaque DPA :

Attaque CPA :

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

Afin de bien assimiler les notions de *side-channel attacks*, il m'a été demandé de réaliser une simulation sur MATLAB. La première phase de la simulation génère des traces sur base de l'algorithme AES. La seconde phase de la simulation a pour but de réaliser une attaque par canal auxiliaire. Plus précisément, cette attaque est réalisée sur un seul byte de données et se fait en sortie de la SBox. Ce type d'attaque est appelé CPA. Pour cette raison, seules les 2 premières étapes de l'algorithme AES sont nécessaires et seront donc simulées (*AddRoundKey*, *SubBytes*).

Les graphes 9 ci-dessous présentent les valeurs de coefficient de corrélation en fonction des 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, en fonction du nombres de traces prises, les valeurs du facteur de corrélation fluctuent beaucoup entre -1 et 1. Ainsi, au plus le nombre de traces est grand, au plus on peut facilement repérer la clé de chiffrement. En effet, si on observe les 3 graphes ci-dessous, on peut remarque que 10 traces, le coefficient de corrélation varie entre -0,65 et 0,65 ; pour 100 traces, le coefficient de corrélation varie entre -0,4 et 0,4 ; pour 1000 traces, le coefficient de corrélation varie entre -0,2 et 0,2. Il est, en effet, beaucoup plus facile visuellement de retrouver la clé lorsque le nombre de traces est élevé.

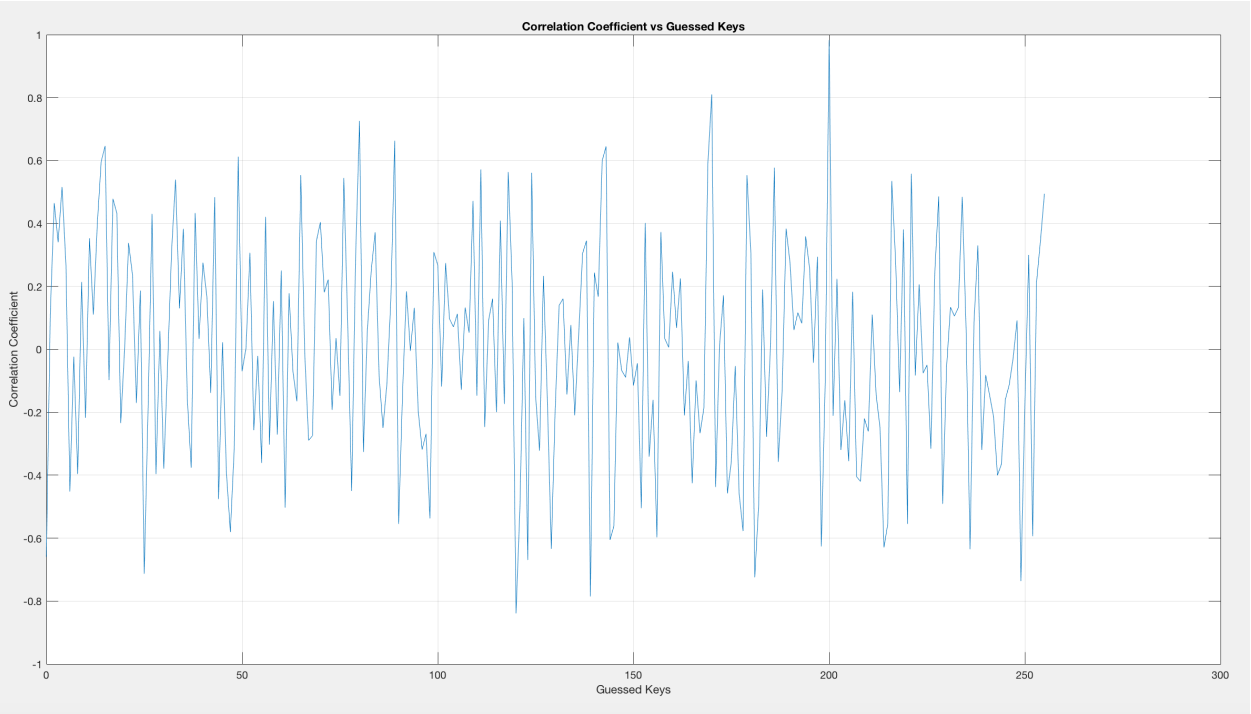

Figure 12 : Pour 10 traces.

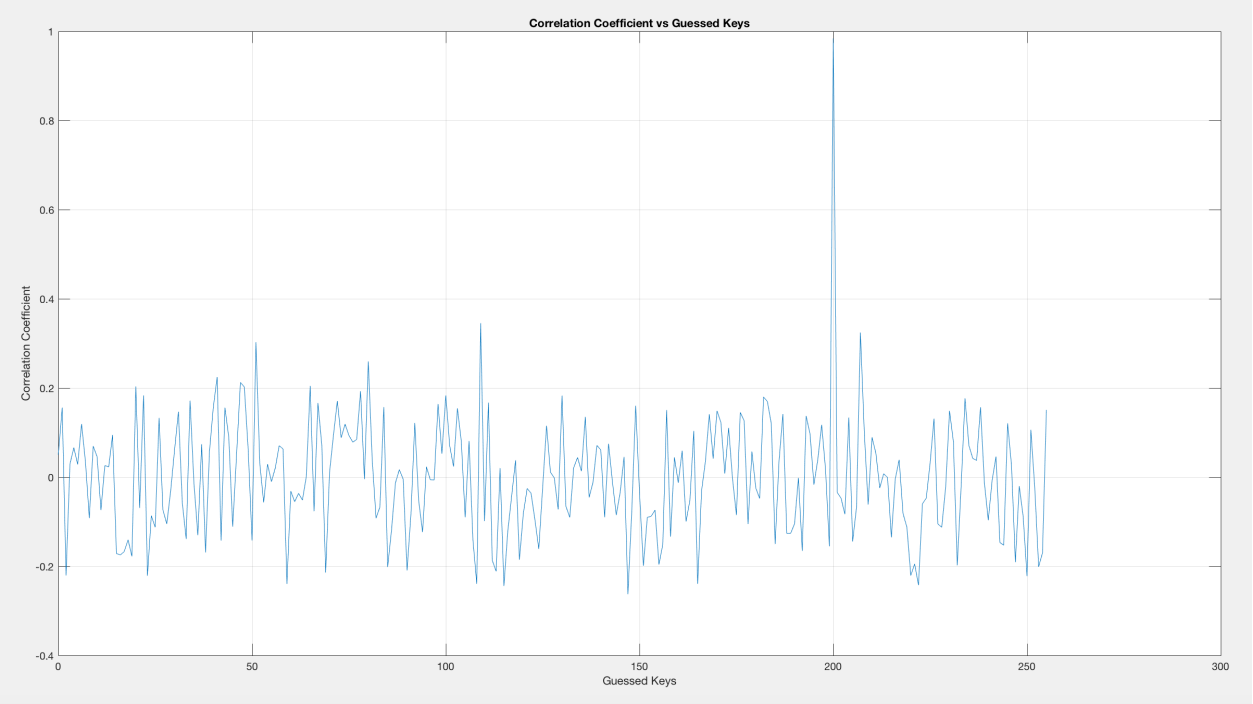

Figure 13 : Pour 100 traces.

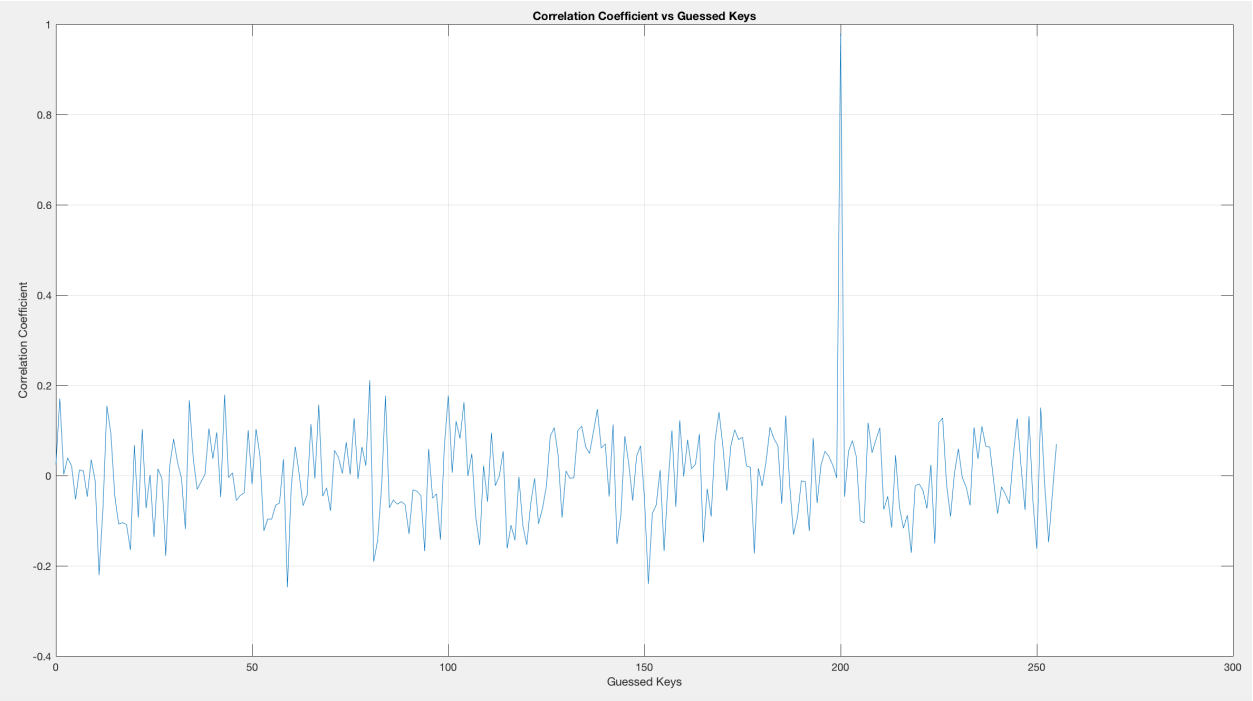

Figure 14 : Pour 1000 traces.

3.5 Notions de statistiques et de t-test

Être capable d'analyser les traces de puissance d'un point de vue statistique est une étapes primordiale dans les side-channel attacks.

3.6 Simulation d'un t-test sur MATLAB

4 Conclusion

Crédits

— Figure **??** provenant de :
Le blog officiel de Texas Instrument : https://e2e.ti.com/blogs_/archives/b/precisionhub/archive/2015/01/21/delta-sigma-adc-basics-understanding-the-delta-sigma-modulator
— 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
—
—