Cybersécurité hardware

Chiffrement AES et attaques par canaux auxiliaires

Quentin Daval

Table des matières

1 Introduction **3**

1.1 Encodage et chiffrement symétrique / asymétrique 3

2 Le chiffrement AES **5**

2.1 Fonctionnement général 5

2.1.1 Definition 5

2.1.2 La clef 6

2.2 Key schedule 7

2.2.1 Déroulement 7

2.2.2 RotWord 8

2.2.3 SubWord 8

2.2.4 Rcon 9

2.3 Les 10 rounds de chiffrement 10

2.3.1 SubWord 10

2.3.2 ShiftRows 10

2.3.3 MixColumns 11

2.3.4 AddRoundKey 13

2.3.5 Le $10^{ème}$ round 13

3 Les attaques par canaux auxiliaires **14**

3.1 Principes 14

3.2 Les attaques par analyse de consommation 15

3.2.1 DPA : Differential Power Analysis 15

3.3 Les attaques par timing 19

4 Application **20**

4.1 AES 20

4.2 Attaques side-channel 20

Chapitre 1

Introduction

1.1 Encodage et chiffrement symétrique / asymétrique

Vocabulaire (Encodage)

L'encodage est la transformation, par un protocole connu, de données en un format différent. Cela peut n'occasionner aucune perte de donnée dans le cas d'une transformation en **base 64** par exemple ou en provoquer dans le cas d'une compression comme le **JPEG** ou le **ZIP**.

Vocabulaire (Chiffrement)

Le chiffrement est la transformation de donnée ayant pour but la **dissimulation** de ces données aux personnes non autorisées.

Distinction (Chiffrement symétrique / asymétrique)

- Chiffrement **symétrique** : La clef de chiffrement est la même que la clef de déchiffrement. (Ex : DES, AES ...)
- Chiffrement **asymétrique** : Les clef de chiffrement et de déchiffrement sont différentes : la clef de chiffrement peut alors être publique. (Ex : RSA, DSA ...)

Vocabulaire (Hachage)

Le hachage est une transformation **irréversible** d'une données quelconque en un texte de taille fixe (Ex : signature en 32 caractères d'un logiciel pour un MD5).

Vocabulaire (décryptage)

Le décryptage est l'action effectuée par une personne cherchant à déchiffrer des données **sans avoir accès à la clef** de déchiffrement.

Chapitre 2

Le chiffrement AES

2.1 Fonctionnement général

2.1.1 Definition

Définition (Chiffrement AES)

Le chiffrement AES pour Advanced Encryption Standard est un standard de chiffrement établi par le NIST : National Institute of Standards and Technology (https ://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf). C'est un **chiffrement par bloc** qui est parmi les plus utilisés dans la monde grâce à sa robustesse et à sa **simplicité de calcul**.

Fonctionnement

Le chiffrement AES se déroule en **10 rounds** équivalents (avec une petite variation pour le dernier round) et a besoin d'une clef de chiffrement et d'un texte de **128 bits** (ou 16 caractère pour un text encodé en ASCII). Ce texte est alors transformé tout au long de l'algorithme qui renvoi 128 octets constituant le message chiffré.

Propriétés (Le state)

On appellera **state** l'état des 16 octets de données manipulées **à n'importe quel moment de l'algorithme**. Le chiffrement ne dépend en effet que de ce state, de l'étape du programme et de la clef de chiffrement.

Notation

On remarquera qu'un **élément du tableau ASCII**, un **nombre hexadécimal à deux chiffres** et un **entier entre 0 et 255** sont des représentations équivalentes pour une information stockée sur 8 bits.

2.1.2 La clef

Propriétés (Taille)

La clef de l'algorithme est généralement composée de **128 bits** mais il existe deux variations de l'AES qui utilisent des clefs de 192 et 256 bits.
La clef est représentée dans l'algorithme comme une série de 16 octets

```
static const unsigned char key[16] = {
0×00, 0×01, 0×02, 0×03, 0×04, 0×05, 0×06, 0×07, 0×08, 0×09, 0×0a, 0×0b,
0×0c, 0×0d, 0×0e, 0×0f };
```

Lors de chacun des rounds de l'AES, la clef de départ est modifiée par un algorithme nommé **key Schedule**.

2.2 Key schedule

2.2.1 Déroulement

Fonctionnement

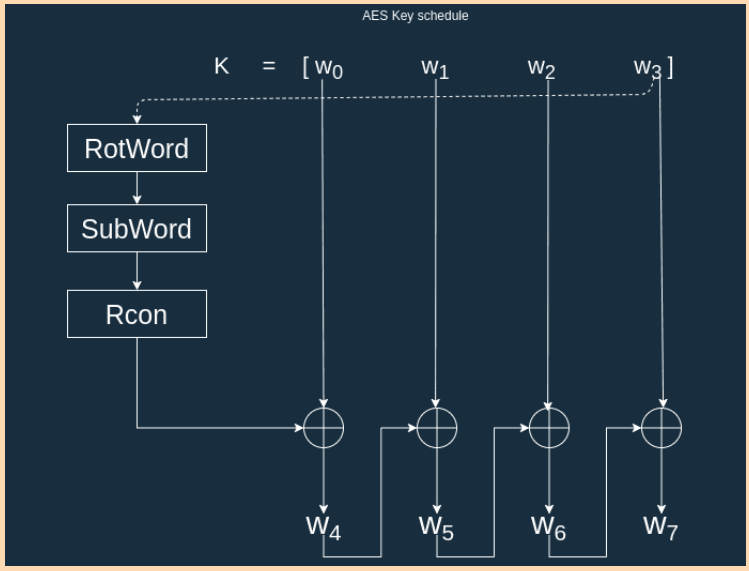
La première clef est XOR avec le message à chiffrer au tout début de l'algorithme et **le state sera XOR avec les clefs** calculées une fois par round du chiffrement. (Il y a donc au total 11 clefs et 11 XOR)

Optimisation

Il est possible de calculer les 10 clefs au début de l'algorithme mais il est moins couteux en mémoire de ne les calculer qu'au moment où elles sont utiles.

Fonctionnement

La clef est premièrement coupée en 4 parties de **4 octets** et les opérations se font sur la $4^{ème}$ **partie**. On applique alors les fonctions **RotWord**, **SubWord**, puis **Rcon** à cette dernière partie qui est ensuite XOR avec la première partie de la clef pour former La première partie de la clef suivante. Les parties suivantes sont alors obtenue par XOR successifs comme précisé dans le schéma ci-dessous :



2.2.2 RotWord

Définition (RotWord)

RotWord est un décalage vers la gauche des 4 octets de la partie de la clef.

$a_0, a_1, a_2, a_3 \Rightarrow a_1, a_2, a_3, a_0$ avec chaque a_i représentant un octet.

2.2.3 SubWord

Définition

SubWord est une simple **substitution** des 4 octets de la partie de la clef par 4 nouveaux octets donnés par le tableau de substitution **Sbox**.

$0\times00 \rightarrow 0\times63$ $\quad$ $0\times1f \rightarrow 0\times c0$ $\quad$ etc...

Exemple (Sbox)

static const unsigned char sbox[256] = {
0×63, 0×7c, 0×77, 0×7b, 0×f2, 0×6b, 0×6f, 0×c5, 0×30, 0×01, 0×67, 0×2b, 0×fe, 0×d7, 0×ab, 0×76,
0×ca, 0×82, 0×c9, 0×7d, 0×fa, 0×59, 0×47, 0×f0, 0×ad, 0×d4, 0×a2, 0×af, 0×9c, 0×a4, 0×72, 0×c0,
0×b7, 0×fd, 0×93, 0×26, 0×36, 0×3f, 0×f7, 0×cc, 0×34, 0×a5, 0×e5, 0×f1, 0×71, 0×d8, 0×31, 0×15,
0×04, 0×c7, 0×23, 0×c3, 0×18, 0×96, 0×05, 0×9a, 0×07, 0×12, 0×80, 0×e2, 0×eb, 0×27, 0×b2, 0×75,
0×09, 0×83, 0×2c, 0×1a, 0×1b, 0×6e, 0×5a, 0×a0, 0×52, 0×3b, 0×d6, 0×b3, 0×29, 0×e3, 0×2f, 0×84,
0×53, 0×d1, 0×00, 0×ed, 0×20, 0×fc, 0×b1, 0×5b, 0×6a, 0×cb, 0×be, 0×39, 0×4a, 0×4c, 0×58, 0×cf,
0×d0, 0×ef, 0×aa, 0×fb, 0×43, 0×4d, 0×33, 0×85, 0×45, 0×f9, 0×02, 0×7f, 0×50, 0×3c, 0×9f, 0×a8,
0×51, 0×a3, 0×40, 0×8f, 0×92, 0×9d, 0×38, 0×f5, 0×bc, 0×b6, 0×da, 0×21, 0×10, 0×ff, 0×f3, 0×d2,
0×cd, 0×0c, 0×13, 0×ec, 0×5f, 0×97, 0×44, 0×17, 0×c4, 0×a7, 0×7e, 0×3d, 0×64, 0×5d, 0×19, 0×73,
0×60, 0×81, 0×4f, 0×dc, 0×22, 0×2a, 0×90, 0×88, 0×46, 0×ee, 0×b8, 0×14, 0×de, 0×5e, 0×0b, 0×db,
0×e0, 0×32, 0×3a, 0×0a, 0×49, 0×06, 0×24, 0×5c, 0×c2, 0×d3, 0×ac, 0×62, 0×91, 0×95, 0×e4, 0×79,
0×e7, 0×c8, 0×37, 0×6d, 0×8d, 0×d5, 0×4e, 0×a9, 0×6c, 0×56, 0×f4, 0×ea, 0×65, 0×7a, 0×ae, 0×08,
0×ba, 0×78, 0×25, 0×2e, 0×1c, 0×a6, 0×b4, 0×c6, 0×e8, 0×dd, 0×74, 0×1f, 0×4b, 0×bd, 0×8b, 0×8a,
0×70, 0×3e, 0×b5, 0×66, 0×48, 0×03, 0×f6, 0×0e, 0×61, 0×35, 0×57, 0×b9, 0×86, 0×c1, 0×1d, 0×9e,
0×e1, 0×f8, 0×98, 0×11, 0×69, 0×d9, 0×8e, 0×94, 0×9b, 0×1e, 0×87, 0×e9, 0×ce, 0×55, 0×28, 0×df,
0×8c, 0×a1, 0×89, 0×0d, 0×bf, 0×e6, 0×42, 0×68, 0×41, 0×99, 0×2d, 0×0f, 0×b0, 0×54, 0×bb, 0×16
};

2.2.4 Rcon

Définition

Rcon modifie le première octet de la partie de la clef : cet octet est XOR avec une case de **rcon**. La case de **rcon** dépend de la clef créée : rcon[1] pour la première clef calculée, rcon[10] pour la dernière.
(Oui rcon[0] ne sert à rien)

Exemple (rcon)

Ce tableau correspond à ...

static const unsigned char rcon[11] = {
0×8d, 0×01, 0×02, 0×04, 0×08, 0×10, 0×20, 0×40, 0×80, 0×1b, 0×36 };

2.3 Les 10 rounds de chiffrement

2.3.1 SubWord

Définition

SubWord est la généralisation à 16 octets de la fonction appliquée à une partie de la clef dans le key schedule : c'est la substitution des 16 octets par d'autres octets donnés par le tableau **Sbox**.

2.3.2 ShiftRows

Notation (Simplification)

Lors de cette étape, il est plus facile de représenter le state comme un tableau de 4×4 caractères :

$$a_0, a_1, a_2, a_3, \ldots, a_{12}, a_{13}, a_{14}, a_{15}, a_{16} \quad \Rightarrow \quad \begin{matrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{matrix}$$

Définition (ShiftRows)

ShiftRows consiste alors à **décaler les lignes**. On ne touche pas à la première ligne, on décale de 1 la seconde ligne vers la gauche, de 2 la troisième et de 3 la dernière :

$$\begin{matrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{matrix} \Rightarrow \begin{matrix} a_0 & a_4 & a_8 & a_{12} \\ a_5 & a_9 & a_{13} & a_1 \\ a_{10} & a_{14} & a_2 & a_6 \\ a_{15} & a_3 & a_7 & a_{11} \end{matrix}$$

2.3.3 MixColumns

Multiplication de Galois

Définition

La multiplication de Galois est de manière générale une multiplication dans un corps fini : cela permet de définir des opérations dans un ensemble fini d'éléments sans que les calculs donnent des éléments qui ne sont plus dans l'ensemble de départ.
Cela permet donc dans le cadre de l'AES d'avoir une multiplication pour les nombres entre 0 et 255 qui ne donne que d'autres nombres entre 0 et 255.

Définition (Multiplication de Galois pour l'AES)

Je vous laisse ci-dessous une façon d'implémenter la multiplication de Galois en C mais je n'irais pas plus loin dans sa description :

```
unsigned char multGalois(unsigned char a, unsigned char b){
   unsigned char p=0;
   for (int i=0; i<8; i++){
      if ((b & 0x01) == 0x01){
         p^=a;
      }
      if ((a & 0x80) == 0x80){
         a<<=1;
         a^=0x1b;
      }else{
         a<<=1;
      }
      b>>=1;
   }
   return p;
}
```

MixColumns

Définition (MixColumns)

En utilisant la notation matricielle du state donné dans la partie précédente MixColumns peut être vu comme la multiplication matricielle entre le state et la matrice **mix** dans l'algèbre de Galois. C'est à dire en utilisant la multiplication de Galois à la place des multiplications et des XOR à la place des additions.

(mix)

$$\begin{matrix} 0\times02 & 0\times03 & 0\times01 & 0\times01 \\ 0\times01 & 0\times02 & 0\times03 & 0\times01 \\ 0\times01 & 0\times01 & 0\times02 & 0\times03 \\ 0\times03 & 0\times01 & 0\times01 & 0\times02 \end{matrix}$$

2.3.4 AddRoundKey

Définition

Lors de l'appel de **AddRoundKey** dans le round i, le state est XOR avec la $i^{ème}$ clef.

2.3.5 Le $10^{ème}$ round

Propriétés

Lors du $10^{ème}$ round, on appelle les fonctions **SubWord**, **ShiftRows** et **AddRoundKey** mais pas la fonction **MixColumns**

Chapitre 3

Les attaques par canaux auxiliaires

3.1 Principes

Distinction (Cryptanalyse / attaques par canaux auxiliaires)

Les **attaques par canaux auxiliaires** se distinguent de la cryptanalyse par les **informations** à la disposition de l'attaquant. En effet, en cryptanalyse, les informations connues sont des combinaisons du texte en clair et du texte chiffré là où, lors des attaques par canaux auxiliaires, l'attaquant peut aussi obtenir des informations sur le déroulé du chiffrement : Il attaque alors l'**implémentation physique** de l'algorithme.

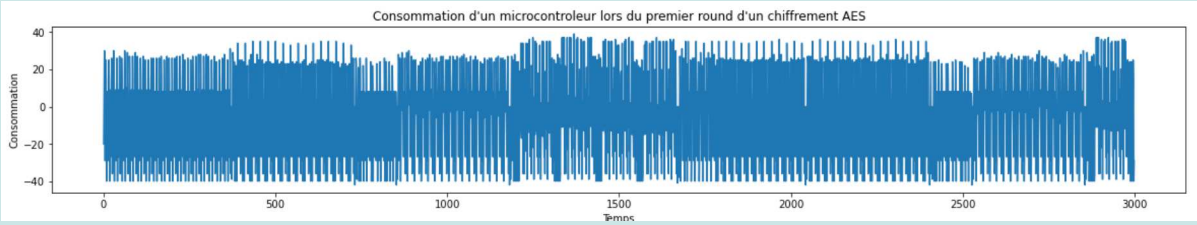
Propriétés (Les fuites d'information)

Différents éléments d'un circuit vont pouvoir être utilisés pour récupérer des informations utiles au décryptage. La suite portera principalement sur l'étude du temps d'exécution et de la consommation électrique des composants mais on trouvera aussi d'autres sources plus ou moins utilisables de fuites :

- le rayonnement électromagnétique
- La température des composants
- Le bruit des composants

Vocabulaire (Trace)

On appelle une trace l'ensemble des points de mesures récoltés pendant l'exécution d'un algorithme de chiffrement. On y associe souvent le texte en clair utilisé et/ou le texte chiffré obtenu. C'est cette donnée qui sera à la base de toutes les analyses de l'algorithme.



3.2 Les attaques par analyse de consommation

3.2.1 DPA : Differential Power Analysis

Introduction

Fonctionnement (DPA)

Comme beaucoup d'attaques side-channel, l'attaque DPA est une attaque **statistique** qui tire profit des très petites différences de consommation du circuit chiffrant le texte pour en déduire la clef octet par octet. Il est alors infiniment moins long de retrouver la clef puisqu'il est possible d'attaquer celle-ci octet par octet, voir bit par bit.

Vocabulaire (Poids de Hamming)

Le poids de hamming d'une valeur correspond au nombre de 1 dans son écriture binaire.

$$\mathcal{H}(25_d) = \mathcal{H}(11001_2) = 3$$

Vocabulaire (Distance de Hamming)

La distance de Hamming entre deux valeurs binaires de même taille correspond au nombre de positions où les deux représentations diffèrent.

$$\mathcal{D}_H(25_d, 14_d) = \mathcal{D}_H(11001_2, 01110_2) = 4$$

Propriétés (Consommation électrique)

On considère que la consommation des composants d'un circuit dépend du poids de Hamming des données manipulés ou de la distance de Hamming entre deux calculs.

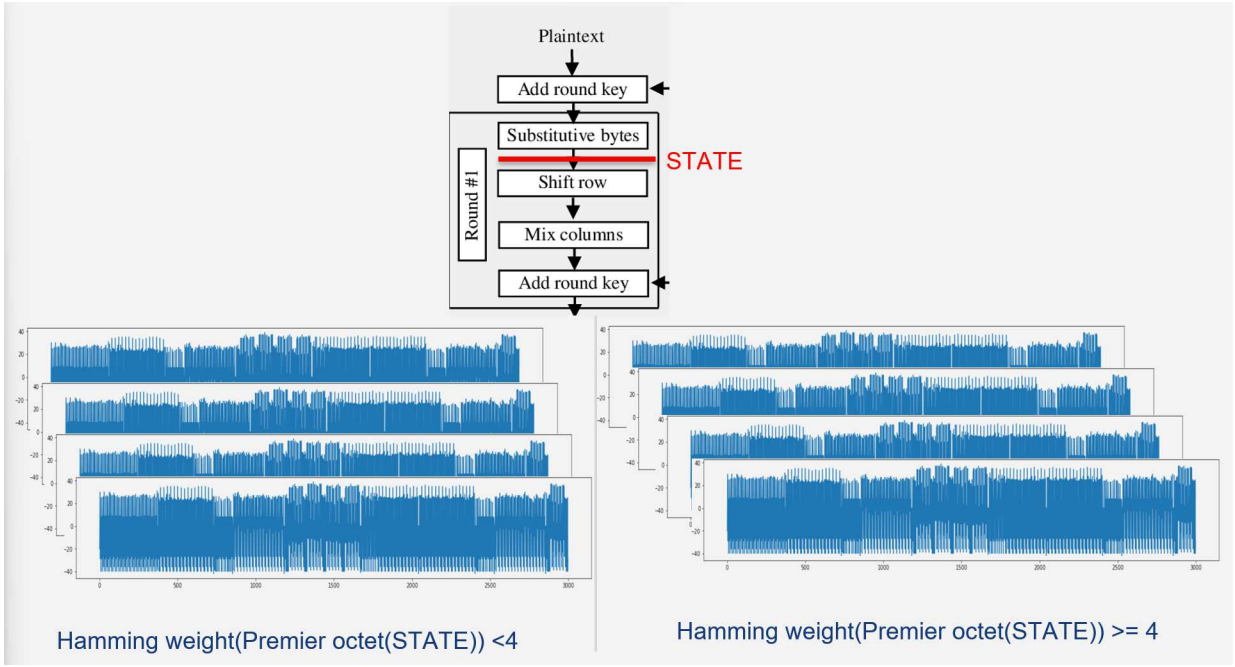
L'Attaque de l'AES

Fonctionnement

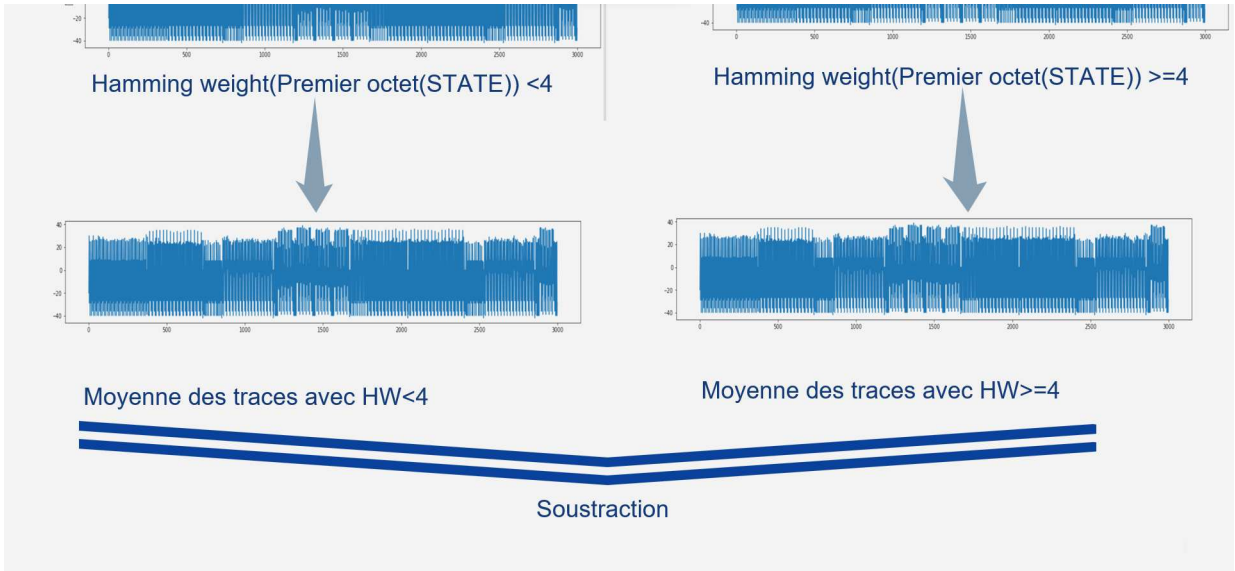
On se place au tout début de l'algorithme AES, après les deux premières opérations. L'état d'un octet du state ne **dépend alors que de l'octet du plaintext et de l'octet de la clef à la même position** : Il devient alors possible de trouver les octets de la clef un à un. On passe ainsi de 2^{128} clefs à tester à $16 \times 256 = 2^{12}$ clefs rendant ainsi possible la récupération de cette clef.
La suite des explication se concentre alors sur la recherche du **premier octet** de la clef puisque les autres se trouvent de manière identique.

On commence donc par faire une **hypothèse** sur le premier octet de la clef (càd à lui associer une valeur, ce que l'on fera 256 fois pour toutes les valeurs possibles) puis on **sépare un dataset** contenant des traces liées au plaintext utilisé par la trace.
Les deux subsets peuvent être créés selon **différentes démarcations**, on peut choisir par exemple la **valeur de l'un des bits du premier octet** du state (après les deux premières opérations) mais il est plus efficace de prendre le **poids de Hamming de l'octet** à cette étape ou encore la **distance de Hamming entre l'octet à cette étape et à l'étape précédente**.

On choisi ici le **poids de Hamming** et séparant les deux datasets avec **HammingWeigth<4** et **HammingWeigth≥4** pour avoir deux subsets de tailles équivalentes :

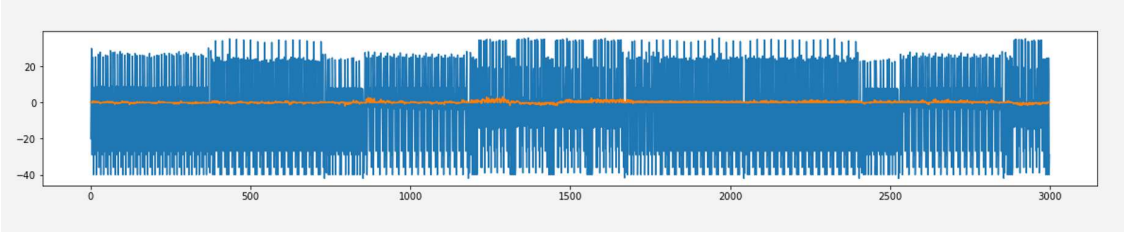


On fait ensuite la **moyenne point par point des traces** de chacun des datasets avant de prendre la **différence** les deux moyennes obtenues :



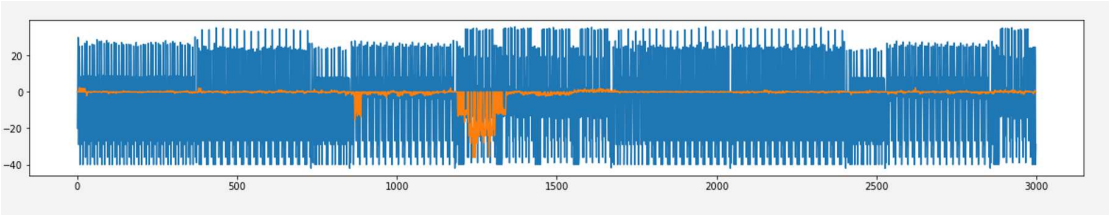
Distinction (Cas 1 : l'hypothèse sur la clef est fausse)

Dans 255 cas sur 256, l'hypothèse sur l'octet de la **clef sera fausse**, tout c'est alors passé comme si les deux subsets avaient été **choisis au hasard**, leur différence s'approche donc de 0 :



Distinction (Cas 2 : l'hypothèse sur la clef est bonne)

Dans le cas où la **clef est bonne**, la différence de consommation des composants va être **amplifiée** par le grand nombre de traces et la **corrélation** entre les éléments d'un même subset fait apparaitre une petite **différence entre les deux traces moyennes** au endroits où sont utilisés les octets manipulés (Après les deux premières opérations pour nous).



Principe

Il ne reste plus qu'à réitérer ces hypothèses pour les **15 autres octets** de la clef pour retrouver toute la clef !

3.3 Les attaques par timing

Propriétés (Temps de calcul)

Certaines opération prennent plus de temps que d'autres : dans le cas de l'algorithme DES, la différence de temps de calcul entre les multiplications et les additions permettent de retrouver la clef en ayant simplement une seule opération.

Chapitre 4

Application

4.1 AES

https ://cryptohack.org/challenges/aes/

4.2 Attaques side-channel

https ://www.newae.com/

https ://github.com/newaetech/