SFA - Lab 1

Side channel attack on AES with chipwhisperer

Maxime Chantemargue, Ruben Pereira Lopes, Charles Matrand

HEIG VD

7 avril 2024

Table des matières

SFA - Lab 1

Links

- 5 - Real target
- Documentation chipwisperer

Measurement setup

Pour l'équipement utilisé, nous utilisons uniquement un Chipwhisperer pour générer les traces et ciphertexts.

Le code suivant doit être installé avant de lancer le programme.

```
pip install tqdm
pip install chipwhisperer
```

Code qui génère les ciphertexts et les traces

```
import secrets
from tqdm import tqdm

NUMBER_OF_PLAINTEXTS = 3000

# Prepare the matrix of 10 different plaintexts
list_plaintexts = [secrets.token_hex(16) for _ in range(
    NUMBER_OF_PLAINTEXTS)]
list_ciphertexts = []
list_traces = []

for p in tqdm(list_plaintexts):
    target_reset()
    scope.arm()
    target.simpleserial_write("p", p.encode())
    scope.capture()
    cipertext = target.simpleserial_read('r', 16)#For loop check
    list_ciphertexts.append(cipertext)
    list_traces.append(scope.get_last_trace())
```

Number of traces acquired

Nous avons commencé par utiliser 3000 traces pour en avoir assez pour le reste des tests et ne plus avoir besoin de faire d'autres captures chronophages. Les traces sont des ADC samples : la consommation électrique du Chipwhisperer lors du chiffrement AES-128. En effet, des Triggers ont déjà été mis en place.

The attack methodology

Nous commençons avec en notre possession :

- la liste de plaintexts
- la liste de ciphertexts correspondante
- la liste de traces correspondante

La trace représente l'entièrté d'AES (les 10 rounds). Nous ne sommes ici qu'intéressés par le 10ème round. Donc afin d'accélérer la recherche de la solution, nous avons seulement étudié la partie de la trace qui était sur ce 10ème round. Pour faire cela, nous avons étudié tout le range (0 à 8000) pour nos 3 milles traces et nous avons trouvé des meilleurs coefficients de corrélation en dehors du range 6000-7000. Nous avons donc pris cet intervalle.

Pour la méthodologie d'attaque :

1. Nous procédons byte par byte. Pour chaque byte, nous récupérons sa valeur dans tous les cyphertextes. Pour chacune de ces valeurs, on teste chacune des clés candidates (0 à 255) avec le but de récupérer le Hamming Weight pour chaque clé possible. Pour cela, il nous faut effectuer le xor entre la clé candidate et le ciphertext, récupérer la SBox inverse puis récupérer le Hamming Weight.
2. Après cela, nous avons itéré sur les valeurs de 6000 à 7000 ADC de notre trace. Pour chacune de ces valeurs, nous calculons le coefficient de corrélation de Pearson avec les 256 listes d'Hamming Weight.
3. Nous avons formé une matrice composée de tous nos coefficients de corrélation. Chaque ligne correspond à une valeur ADC de la trace et chaque colonne est une valeur de clé candidate. La matrice fait donc 1000 lignes et 256 colonnes.
4. Au final, après avoir récupéré cette matrice, nous sélectionnons le meilleur en prenant la plus grande valeur absolue des coefficients de corrélation. Nous récupérons l'indice de la colonne de ce meilleur coefficient et cet indice est égal à la valeur de la clé pour ce byte.
5. Nous répétons ces étapes pour chaque byte de la clé.

The recovered key

Une fois la clé K10 récupérée, nous appelons la fonction `get_master_key` de `sca_training` qui nous donne la clé :

`SCA{RealAES-128}`

The correlation coefficient of the best three candidates

Voici pour chaque byte, les coefficients de corrélation des trois meilleurs candidats, la valeur de la clé correspondante et à quelle valeur ADC de la trace ils ont été trouvés.

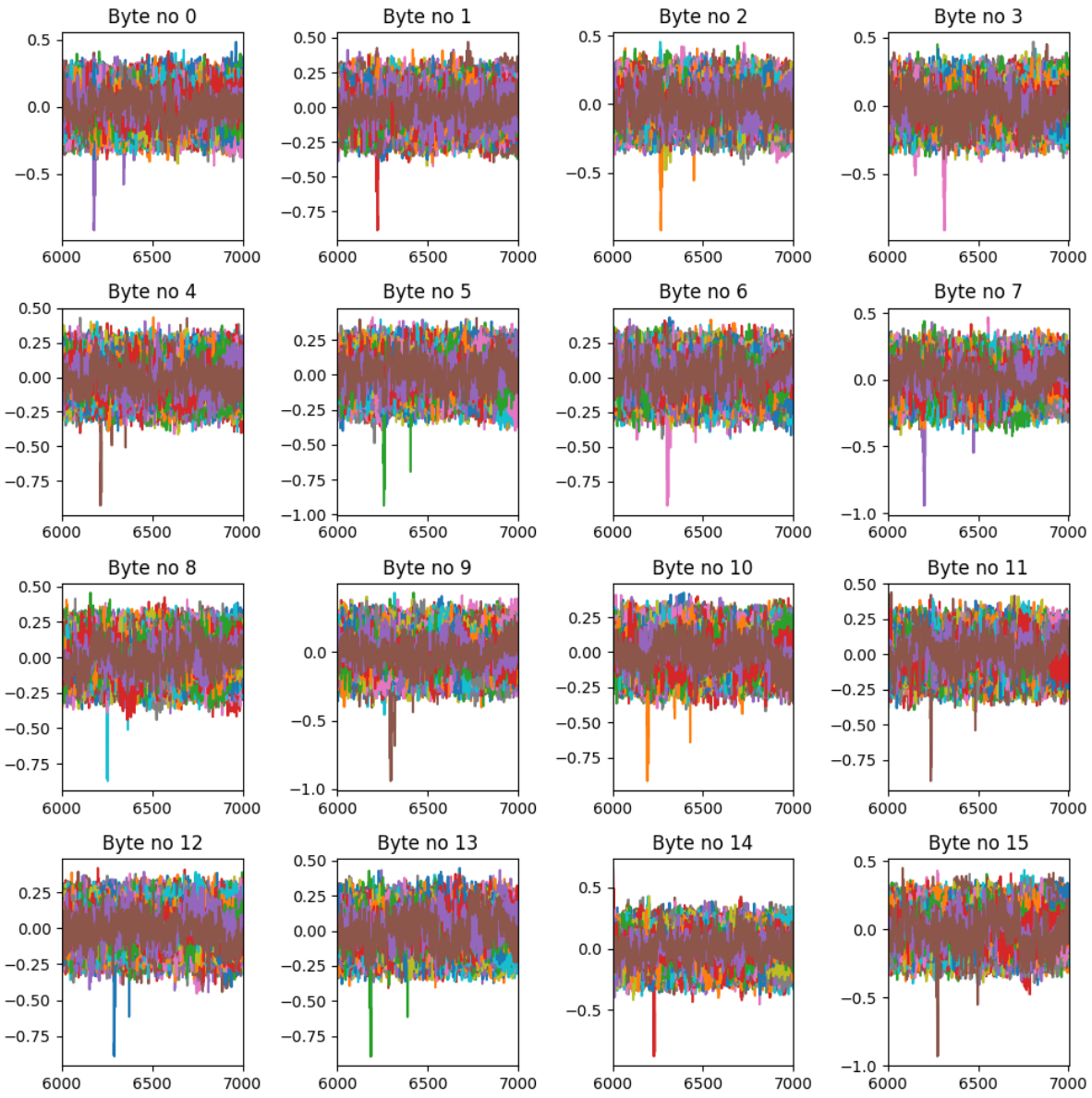
```
Byte  0
3 meilleures candidates (valeurs de clés) : [64 64 64]
Valeurs ADC correspondantes : [6178 6179 6177]
Coefficients de corrélation correspondants : [-0.86634475 -0.91593916
    -0.92121016]

Byte  1
3 meilleures candidates (valeurs de clés) : [103 103 103]
Valeurs ADC correspondantes : [6223 6224 6222]
Coefficients de corrélation correspondants : [-0.78629104 -0.87972562
    -0.88541649]

Byte  2
3 meilleures candidates (valeurs de clés) : [71 71 71]
Valeurs ADC correspondantes : [6268 6269 6267]
Coefficients de corrélation correspondants : [-0.78226344 -0.91785727
    -0.91875651]

Byte  3
3 meilleures candidates (valeurs de clés) : [136 136 136]
Valeurs ADC correspondantes : [6313 6312 6314]
Coefficients de corrélation correspondants : [-0.81269769 -0.91230804
    -0.91477081]

Byte  4
3 meilleures candidates (valeurs de clés) : [35 35 35]
Valeurs ADC correspondantes : [6218 6214 6213]
Coefficients de corrélation correspondants : [-0.83448318 -0.9217212
    -0.92857239]

Byte  5
3 meilleures candidates (valeurs de clés) : [202 202 202]
Valeurs ADC correspondantes : [6263 6259 6258]
Coefficients de corrélation correspondants : [-0.81684803 -0.9196291
    -0.93519903]

Byte  6
3 meilleures candidates (valeurs de clés) : [186 186 186]
Valeurs ADC correspondantes : [6308 6304 6303]
Coefficients de corrélation correspondants : [-0.86573776 -0.91191036
    -0.9249554 ]

Byte  7
3 meilleures candidates (valeurs de clés) : [54 54 54]
Valeurs ADC correspondantes : [6202 6203 6201]
Coefficients de corrélation correspondants : [-0.82375074 -0.93900086
    -0.94369259]

Byte  8
3 meilleures candidates (valeurs de clés) : [149 149 149]
Valeurs ADC correspondantes : [6252 6250 6254]
Coefficients de corrélation correspondants : [-0.78071562 -0.85918488
    -0.87220625]

Byte  9
3 meilleures candidates (valeurs de clés) : [215 215 215]
Valeurs ADC correspondantes : [6297 6299 6295]
```

```
Coefficients de corrélation correspondants : [-0.8833152  -0.93539851
    -0.94006632]

Byte  10
3 meilleures candidates (valeurs de clés) : [201 201 201]
Valeurs ADC correspondantes : [6197 6192 6193]
Coefficients de corrélation correspondants : [-0.79689697 -0.91417319
    -0.91748702]

Byte  11
3 meilleures candidates (valeurs de clés) : [235 235 235]
Valeurs ADC correspondantes : [6242 6238 6237]
Coefficients de corrélation correspondants : [-0.79879351 -0.89764047
    -0.90039474]

Byte  12
3 meilleures candidates (valeurs de clés) : [130 130 130]
Valeurs ADC correspondantes : [6289 6288 6290]
Coefficients de corrélation correspondants : [-0.79748323 -0.88099475
    -0.88945887]

Byte  13
3 meilleures candidates (valeurs de clés) : [212 212 212]
Valeurs ADC correspondantes : [6186 6188 6184]
Coefficients de corrélation correspondants : [-0.80404793 -0.8925606
    -0.89282894]

Byte  14
3 meilleures candidates (valeurs de clés) : [153 153 153]
Valeurs ADC correspondantes : [6231 6233 6229]
Coefficients de corrélation correspondants : [-0.76547219 -0.84637644
    -0.88067408]

Byte  15
3 meilleures candidates (valeurs de clés) : [255 255 255]
Valeurs ADC correspondantes : [6276 6278 6274]
Coefficients de corrélation correspondants : [-0.84333545 -0.92317991
    -0.93121305]
```

Remarque : les coefficients de corrélation affichés sont affichés dans l'ordre inverse (donc du 3e au 1e depuis la gauche).

Correlation graphs

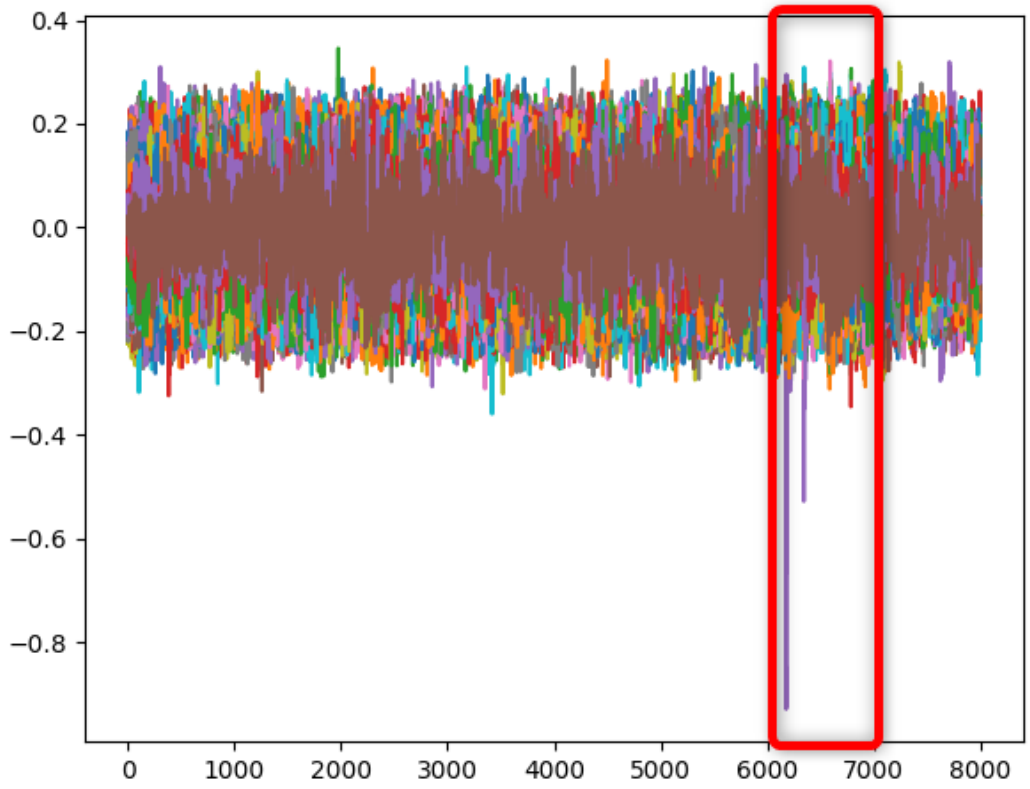
Voici les graphes de corrélation pour chacun des bytes de la k10. L'axe des x donne les valeurs ADC (de consommation électrique) de la trace. L'axe des y donne le coefficient de corrélation. Les différentes couleurs représentent les différents valeurs candidates de clé. On peut observer que le meilleur coefficient se distingue d'assez loin des autres. On en conclut que notre résultat est assez robuste.



A trace with a highlight of the attacked region

Dans le graphe ci-dessous, nous avons remarqué que pour les 16 graphes, le même intervalle de mesure avait des coefficients de corrélation élevés dans la zone se situant entre 6000 à 7000 comme expliqué précédemment.

Comme nous l'avons vu dans les précédents exercices, les traces contenaient des informations sur le dernier round. Dans le cas du laboratoire, nous remarquons que la zone cible contient les meilleurs coefficients de corrélation donc très probablement parce que les ADC correspondent à ceux du dernier round (donc corrélation plus proche entre les Hamming Weight et les traces).



Bonus points if you can show the minimum number of traces needed to perform the attack (rounded to 100 traces)

Dans notre code, nous reprenons uniquement 100 traces et 100 ciphertexts correspondants. Entre 6000 et 7000 ADC provenant de la trace, nous arrivons à récupérer la clé.

```
sample_size = 100

# cree les matrices de ciphertexts et matrices de traces
ciphertexts_matrix = np.array(list_ciphertexts[:sample_size])
traces_matrix = np.array(list_traces[:sample_size])
```

Nous n'avons pas spécialement de preuves visuelles mais vous pouvez vous référer à notre notebook et voir les output générées, dont la dernière (qui affiche la clé) prouve bien son fonctionnement.