DPA — Comment ca fonctionne ?

Benjamin Grolleau

ENSEIRB–MATMECA, Bordeaux INP, France

benjamin.grolleau@bordeaux-inp.fr

***Abstract*—Ce papier présente une étude pratique de l'attaque DPA (*Differential Power Analysis*) appliquée à une implémentation matérielle d'AES-128. À partir d'un ensemble de 600 traces de consommation et de plaintexts associés, nous illustrons pas à pas le processus permettant de retrouver un octet de la clé secrète. L'analyse repose sur un modèle de fuite simple basé sur le bit de poids faible de la sortie de la SBox. Nous détaillons la construction des hypothèses, la séparation des traces, le calcul du vecteur de différences et la sélection de la clé. Nous étudions également le taux de confiance et la convergence en fonction du nombre de traces. Les résultats montrent qu'une fuite exploitable permet de retrouver la clé correcte de manière fiable dès quelques centaines de traces.**

I. Introduction

Les attaques par analyse de consommation, telles que la DPA — *Differential Power Analysis* — constituent aujourd'hui l'une des menaces les plus efficaces contre les implémentations matérielles d'algorithmes cryptographiques tels qu'AES. En exploitant les variations de consommation électrique d'un composant dues au fonctionnement des circuits CMOS pendant l'exécution de l'algorithme, il est possible d'inférer des informations internes normalement secrètes, comme des bits de clé.

L'objectif de ce travail est d'illustrer concrètement le fonctionnement interne d'une attaque DPA appliquée au premier octet d'une clé AES, en montrant comment la corrélation entre les hypothèses de fuite et les mesures physiques permet d'isoler le bon candidat parmi les 256 valeurs possibles. À partir d'un jeu de 600 traces accompagnées de textes d'entrée connus, nous détaillons la méthodologie, les traitements effectués et l'interprétation des résultats qui permettent, pas à pas, de reconstituer un octet de la clé.

II. Données d'entrée

Pour réaliser l'attaque DPA, nous disposons de deux types de données d'entrée. La première est constituée d'un ensemble de 600 traces de consommation, chacune contenant 5000 échantillons représentant l'évolution de la consommation électrique au cours de l'exécution de l'algorithme AES. Chaque trace correspond donc à une exécution complète d'AES réalisée avec un plaintext différent, mais toujours avec la même clé secrète. Un exemple de plusieurs traces superposées est illustré en Figure 1.

En parallèle, nous possédons 600 textes d'entrée, chacun composé de 16 octets (correspondant à la taille du bloc AES-128 qui a été chiffré). Chaque plaintext est associé exactement

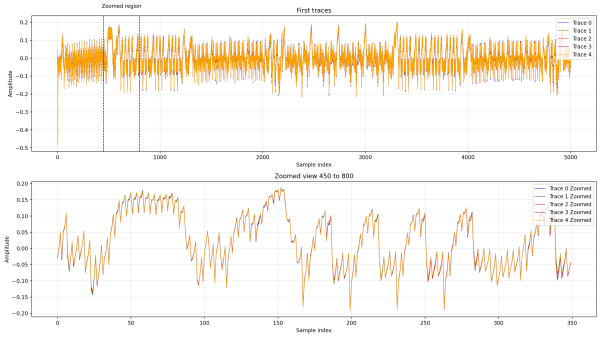


Figure 1. Exemple de plusieurs traces de consommation utilisées pour l'attaque.

à une trace, ce qui permet de relier une mesure physique à la donnée manipulée par l'algorithme.

Dans ce contexte, chaque trace peut être vue comme le résultat d'une fonction de fuite physique $\mathcal{L}$, appliquée à l'exécution de l'AES sur un texte d'entrée donné. On peut donc formaliser la relation entre plaintext, clé et trace de la manière suivante :

$$\text{trace}[i] = \mathcal{L}(\text{AES}(\text{secret\_key},\ \text{textin}[i])) \tag{1}$$

III. Méthodologie de l'attaque DPA

L'objectif de l'attaque DPA est d'identifier la valeur de chaque octet de la clé AES en exploitant la dépendance entre la consommation électronique du composant et certaines valeurs intermédiaires calculées pendant l'exécution de l'attaque. La méthode repose sur la formulation d'hypothèses de clé, puis sur l'analyse statistique des traces associées à ces hypothèses. Dans notre cas, nous étudierons simplement le *byte* d'index 0 car l'exécution sera la même pour les 15 autres.

A. Valeur intermédiaire ciblée

La fuite exploitée est due à la première opération non linéaire d'AES :

$$v = \text{SBox}(\text{plaintext}[i] \oplus \text{key}[i]) \tag{2}$$

Comme nous cherchons à déterminer $\text{key}[0]$, nous utiliserons, pour chaque trace, l'octet d'entrée $\text{plaintext}[0]$. La valeur exacte manipulée par la SBox n'est pas observée directement, mais certaines de ses caractéristiques comme bit de poids faible et poids de Hamming influencent la consommation électrique.

Dans notre cas, la fuite modélisée est le bit de poids faible (LSB) de la sortie de la SBox :

$$\text{leak} = v \,\&\, 0x01 \tag{3}$$

B. Construction des hypothèses de fuite

Pour retrouver le byte de la clé, on teste les 256 possibilités présentes sur l'intervalle $guess \in [0x00; 0xff]$, pour chaque hypothèse *guess*, et pour les 600 traces disponibles, on calcule :

$$v_k = \text{SBox}(\text{plaintext}[k][0] \oplus guess) \tag{4}$$

puis le bit de fuite associé (bit de poids faible) :

$$mask[k] = v_k \,\&\, 0x01 \tag{5}$$

On obtient ainsi, pour une hypothèse donnée, un vecteur de 600 bits qui prédit, pour chaque trace, le comportement du circuit si la clé était guess.

C. Séparation en deux groupes de traces

La DPA repose sur l'observation que les consommations diffèrent selon que le bit de fuite vaut 0 ou 1. Nous regroupons les traces selon le masque :

- sel1 : traces pour lesquelles $mask[k] = 1$,
- sel0 : traces pour lesquelles $mask[k] = 0$.

Dans le cadre de données d'entrées idéalement aléatoires, la longueur de ces deux listes vaut environ la moitié du nombre de traces.

Chaque trace contenant 5000 échantillons, on peut ainsi comparer, pour chaque instant t, la consommation moyenne des deux groupes.

D. Calcul du vecteur de différences

Pour chaque hypothèse de clé, on calcule :

$$\text{diff}(t) = |\text{mean}(\text{sel1}[t]) - \text{mean}(\text{sel0}[t])| \tag{6}$$

Ce vecteur de 5000 valeurs mesure, pour chaque échantillon, à quel point la consommation discriminante est cohérente avec l'hypothèse de clé.

Si l'hypothèse est incorrecte, les deux groupes résultent d'un partitionnement aléatoire, sinon, le vecteur est quasi nul. Si l'hypothèse est correcte, un ou plusieurs instants présentent un écart significatif.

Dans notre cas, nous avons testé sur le *byte* d'index 0. Les courbes grises représentent les valeurs du vecteur de différences pour toutes les hypothèses de clé incorrectes, tandis que la courbe rouge correspond à l'hypothèse correcte. On observe que, pour la mauvaise clé, les écarts moyens restent faibles et bruités, alors que pour la bonne clé un pic net apparaît à un instant précis (Figures 2 et 3). Ce pic indique le moment exact où la consommation dépend de la valeur intermédiaire ciblée, et permet d'isoler la clé correcte.

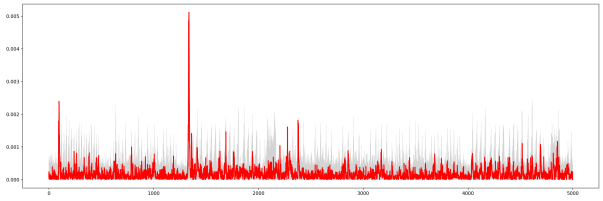


Figure 2. Vecteurs de différences pour les 256 hypothèses de clé (byte 0). La courbe rouge correspond à la bonne hypothèse.

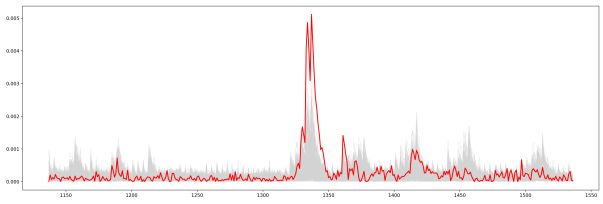


Figure 3. Zoom sur la zone de fuite illustrant le pic caractéristique de la bonne clé.

E. Définition du score DPA

Pour quantifier cette différence, on retient la valeur maximale du vecteur :

$$\text{score}(guess) = \max_t(\text{diff}(t)). \tag{7}$$

Ainsi, une hypothèse correcte produit généralement un score nettement plus élevé que les mauvaises.

F. Sélection de la clé

En répétant les étapes précédentes pour les 256 hypothèses possibles, on obtient 256 scores. L'hypothèse présentant le score maximal est alors sélectionnée :

$$\text{key}[0] = \underset{\text{guess} \in [0x00; 0xff]}{\arg\max} \text{score}(\text{guess}) \tag{8}$$

comme on peut le voir dans la figure 4.

Ce processus est ensuite répété pour chaque octet de la clé AES.

IV. Taux de confiance des résultats

En fonction des valeurs du score, nous pouvons déterminer un taux de confiance associé au résultat obtenu. Comme nous pouvons le voir dans la figure 4, le pic correspondant à l'hypothèse `0x2b` est nettement visible. Cependant, un

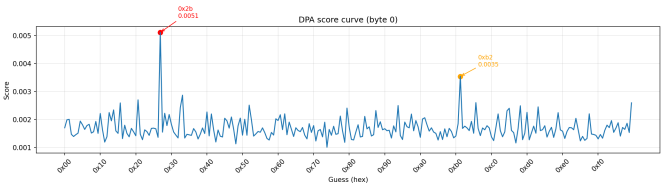


Figure 4. Score de chaque *guess*.

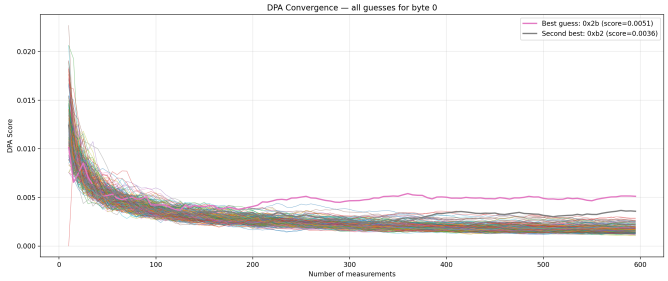


Figure 5. Convergence des *guess* en fonction du nombre de traces pour le *byte* 0.

second résultat — `0xb2` — ressort également, bien que plus faiblement.

Afin de quantifier cette différence, nous définissons le taux de confiance comme le contraste entre le meilleur score et le second meilleur score :

$$\text{contrast} = \frac{\text{best\_score} - \text{second\_score}}{\max(\text{second\_score},\ 10^{-15})} \tag{9}$$

$$\text{confidence} = \min(\max(\text{contrast},\ 0),\ 1) \tag{10}$$

V. Taux de convergence

Pour évaluer la robustesse et l'efficacité de l'attaque DPA, nous étudions son taux de convergence pour un byte de la clé, c'est-à-dire la vitesse à laquelle la bonne hypothèse de clé émerge lorsque l'on augmente le nombre de traces utilisées. Nous définissons le nombre de traces utilisées par l'équation suivante :

$$N = \{k = 10 | k = 1; 2; ...; 60\} \tag{11}$$

Pour un nombre donné de traces N, le score pour une hypothèse g est défini comme

$$\text{score}_N(g) = \max_t |\text{mean}(\text{sel1}_N[t]) - \text{mean}(\text{sel0}_N[t])|. \tag{12}$$

En suivant l'évolution de $\text{score}_N(key[0])$ et en le comparant aux autres scores, on observe :

- Lorsque N est faible, la moyenne est bruitée et la clé n'est pas visible,
- au-delà de ce seuil — dans notre cas, vers les 200 traces, un pic commence à émerger,
- plus N augmente, plus le pic devient stable et plus l'écart avec les mauvaises clés se creuse,
- autour de 300–500 traces, la convergence est nette et la bonne clé domine très clairement.

Une courbe typique de convergence est illustrée figure 5, où l'on observe que la clé correcte devient systématiquement dominante après un certain nombre de traces.

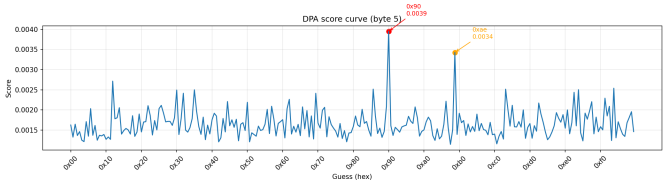


Figure 6. Courbe des scores DPA pour le byte 5 : `0x90` est sélectionné, mais `0xae` apparaît en second.

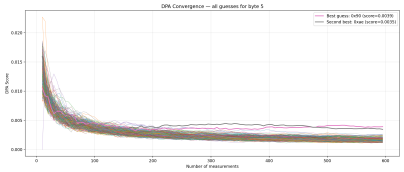


Figure 7. Convergence du byte 5 : la clé correcte `0xae` apparaît, mais ne dépasse pas `0x90`.

A. Erreurs de guess

Il peut néanmoins arriver que l'attaque ne sélectionne pas la bonne clé. Dans l'exemple illustré figures 6 et 7, la procédure de DPA identifie `0x90` comme meilleur candidat pour le byte considéré, alors que la vraie valeur est `0xae`.

En observant la courbe des scores (figure 6), on remarque toutefois que la bonne clé apparaît en deuxième position, avec un score très proche du maximum. Cette proximité indique que, même si le programme s'est trompé, il reste « hésitant » entre deux valeurs plausibles.

La convergence (figure 7) confirme cette observation : la clé correcte émerge visiblement parmi les plus fortes hypothèses, mais sans dominer suffisamment pour surpasser `0x90`.

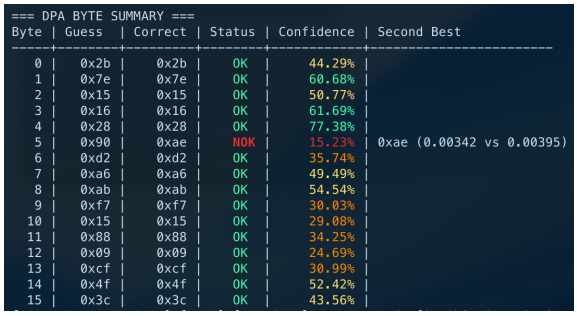
Dans ce cas particulier, le contraste entre le meilleur et le second meilleur score est faible, ce qui se traduit par un taux de confiance d'environ 15 %. L'erreur n'est donc pas due à une divergence totale, mais à une insuffisante séparation statistique entre les deux meilleures hypothèses.

VI. Conclusion

Dans cet article, nous avons présenté une implémentation complète et pédagogique de l'attaque DPA appliquée au premier octet d'une clé AES-128. En exploitant la corrélation entre un modèle de fuite simple (bit de poids faible de la sortie de SBox) et 600 traces de consommation, nous avons pu reconstruire efficacement la valeur de la clé.

Les résultats montrent qu'un pic de différence clairement identifiable apparaît pour la bonne hypothèse, tandis que les hypothèses incorrectes restent proches du bruit. L'étude du taux de confiance confirme que le contraste entre les deux meilleures valeurs permet d'évaluer la fiabilité du résultat. L'analyse de convergence montre également que, dès 200 à 300 traces, la clé correcte commence à dominer, et qu'elle devient nettement isolée au-delà de 300 à 600 traces.

Enfin, nous illustrons ci-dessous la sortie finale du programme, qui confirme expérimentalement la récupération correcte du byte ciblé :



```
=== DPA BYTE SUMMARY ===
Byte | Guess  | Correct | Status | Confidence | Second Best
-----+--------+---------+--------+------------+------------------------
   0 |   0x2b |    0x2b |    OK  |     44.29% |
   1 |   0x7e |    0x7e |    OK  |     60.68% |
   2 |   0x15 |    0x15 |    OK  |     50.77% |
   3 |   0x16 |    0x16 |    OK  |     61.69% |
   4 |   0x28 |    0x28 |    OK  |     77.38% |
   5 |   0x90 |    0xae |   NOK  |     15.23% | 0xae (0.00342 vs 0.00395)
   6 |   0xd2 |    0xd2 |    OK  |     35.74% |
   7 |   0xa6 |    0xa6 |    OK  |     49.49% |
   8 |   0xab |    0xab |    OK  |     54.54% |
   9 |   0xf7 |    0xf7 |    OK  |     30.03% |
  10 |   0x15 |    0x15 |    OK  |     29.08% |
  11 |   0x88 |    0x88 |    OK  |     34.25% |
  12 |   0x09 |    0x09 |    OK  |     24.69% |
  13 |   0xcf |    0xcf |    OK  |     30.99% |
  14 |   0x4f |    0x4f |    OK  |     52.42% |
  15 |   0x3c |    0x3c |    OK  |     43.56% |
```

Figure 8. Sortie finale du programme DPA montrant l'identification correcte de la clé.

Ce travail pourrait être et sera prolongé en testant d'autres modèles de fuite (poids de Hamming, distance de Hamming), en appliquant la CPA (*Correlation Power Analysis*) ou en étudiant l'effet de contremesures matérielles et logicielles.