SECAP – TP 01

Matériel : GNU Octave

Il vous est demandé un compte-rendu (en PDF) précisant les commandes et les résultats (capture de texte principalement).

1. Attaque DPA (Differential Power Analysis) :

On a réalisé des mesures de consommation électrique d'une carte à puce pendant un chiffrement AES. La clé utilisée est toujours la même `KEY=00112233445566778899aabbccddeeff`.

Pour cette première attaque, on va se focaliser sur le premier tour (round) AES.

a) Exécuter le script `ex1.m`. Combien il y a-t-il de mesures et combien d'échantillons a-t-on par mesure (voir variable `traces`) ?

b) Vérifier « à la main » avec OpenSSL ou un script Python/PHP/..., que pour une mesure donnée, on a bien la relation `AES128-ECB(KEY,PTI)=CTO`.
*NB : **PTI** (Plain Text Input) et **CTO** (Cipher Text Ouput)*

c) Afficher le graphique de la première mesure :

```
t1=traces(1,:);
plot(t1);
```

Puis déterminer approximativement la position du premier « round » AES et sa longueur (variables `offset` et `segmentLength`).

d) Paramétrer ces valeurs dans le script `ex1_dpa.m` puis exécuter celui-ci sachant qu'il implémente :
- le calcul de la sortie SubBytes du premier round AES dans la première boucle sur `k`.
- l'attaque différentielle dans la seconde boucle en testant le bit de poids faible de **`v`** (voir `bitget()`) et en calculant la différence absolue (variable **`diff`**) des moyennes des courbes « 0 » et des courbes « 1 ».

Vérifier que l'on retrouve quasiment tous les octets de la clé, mais que le calcul est assez long.

e) Comment afficher dans l'avant dernier `fprintf()`, la valeur recherchée de la clé à chaque octet (pour comparer à la valeur trouvée).

2. Attaque CPA (Correlation Power Analysis) :

Ici, plutôt que de tester le bit de poids faible des valeurs possibles (voir §1d), on va calculer la distance de HAMMING et faire une corrélation.

a) Vérifier que la variable **`HammingWeight`** issue du script `aes_utils` contient bien le nombre de 1 de l'écriture binaire du numéro de case (attention au décalage de numérotation des indices). Prendre une valeur au hasard sur [0,255], l'écrire en binaire (voir `dec2bin()`) et compter le nombre de 1, comparer ensuite à la case correspondante dans le tableau **`HammingWeight`**.

b) Créer alors le script `ex2_cpa.m` à partir du programme précédent et ajouter le calcul de la distance de Hamming à la fin de la première boucle après la SBOX.

c) Supprimer enfin toute la boucle différentielle et la remplacer par le calcul de la matrice corrélation (voir fonction `matCorr(x,y)`). Rechercher alors la position du maximum dans cette matrice.
Comparer alors le résultat à la technique DPA.

d) Utiliser les mesures du répertoire `DPA_Contest` pour déterminer la valeur inconnue de la clé AES.

e) Vérifier ensuite manuellement votre réponse avec un couple `PTI`, `CTO`.

3. Attaque CEMA (Correlation Electromagnetic Analysis) :

Dans cette partie, on utilise des mesures électromagnétiques (EM) réalisées avec une antenne en champ très proche (< 1 mm). On va maintenant se placer en fin du dernier tour AES (round 10), c'est pourquoi l'attaque ne permet pas de retrouver directement la clé AES comme précédemment, mais plutôt la sous clé K_{10} dérivée de la clé de départ.

a) Exécuter le script `ex3_cema.m`. Combien il y a-t-il de mesures (nombres de fichiers dans le répertoire CEMA) et combien d'échantillons a-t-on par mesure (voir variable `traces2`) ?

b) Déterminer la position et la longueur (variables `offset` et `segmentLength`) du round 10 en affichant la courbe d'un échantillon (`traces2`).

c) Vérifier les valeurs de `key`, `PTI2` et `CTO2` en hexadécimal (voir `dec2hex()`) à partir du fichier `CEMA-Traces.txt` ou bien de la variable `matrixFilename`, sachant qu'elles correspondent à la dernière valeur de i (soit `nbTraces`).

d) Vérifier ensuite les valeurs d'expansion de la clé AES contenues dans la variable `sub_keys` (Afficher les valeurs en hexadécimal : `dec2hex(sub_keys(:,:))`) en utilisant un site WEB de calcul en ligne (par exemple : https://www.cryptool.org/en/cto-highlights/aes).

e) Définir les variables suivantes entre les lignes 47 et 48 :

```
v1=zeros(nbBytes);
v2=zeros(nbBytes);
cbHamming2=zeros(nbTraces,nbKey,nbBytes);
```

f) Ajouter le calcul de Hamming dans la double boucle :

```
v1(b)=invSBOX(bitxor(CTO2(b),k-1)+1);
v2(b)=CTO2(shiftRow(b));
cbHamming2(i,k,b)=HammingWeight(bitxor(v1(b),v2(b))+1);
```

g) Dé-commenter la fin du fichier pour activer le calcul de corrélation et retrouver la valeur de la sous-clé K_{10}.

h) Comment varie le nombre d'octets correctement retrouvé en fonction du nombre de traces utilisées ?

i) Comment retrouver la clé AES initiale à partir de la clé K_{10} ? Donner le code source d'un programme personnel réalisant cela.