Challenge Side-Channels
ISEN Toulon – M1 Cybersécurité
Julien FRANCQ
8 heures

Objectif

L'objectif de ce challenge est de retrouver la clé cryptographique d'une implémentation matérielle d'un AES 128 bits sur un FPGA Spartan 6.

Vous disposerez de relevés de rayonnement électromagnétique (EM) des chiffrements réalisés par cette implémentation (2000 courbes au maximum), ainsi que les couples textes clairs/textes chiffrés correspondants à ces différents chiffrements.

Nous sommes dans un cas simplifié où les 16 SBoxes de l'AES sont traitées séquentiellement afin de pouvoir facilement traiter séparément les différentes SBoxes [1].

Nous vous proposons de suivre la méthodologie de la « *Correlation Side-Channel Analysis* ».

Livrable attendu

Chaque binôme devra fournir à l'issu du challenge la clé de ronde retrouvée, et son code Matlab.
Un classement sera réalisé en fonction de la rapidité à laquelle chaque binôme retrouve une clé de ronde. Donc, dès qu'un binôme croit retrouver la bonne valeur de clé de ronde, il doit me la communiquer immédiatement.

1. Etape 1 : comprendre les données

La première étape pour n'importe quel problème de science des données est d'explorer les données à notre disposition [2].
Ouvrez le fichier « CEMA.m » et chargez son contenu. Il contient 2000 courbes de 10000 points chacune. Tracez quelques courbes (le nombre de courbes sera nommé « NbTraces ») et trouvez le début et la fin du chiffrement. En déduire l'intervalle de temps sur lequel vous allez lancer votre attaque. Il sera réglé à l'aide de deux paramètres « xMin » et « xMax ».

2. Etape 2 : lister les points d'attaque possibles

En utilisant les textes clairs et chiffrés fournis, listez les points d'attaque possibles, et le contenu des matrices V correspondantes. Retenez une hypothèse et tentez la corrélation avec les relevés EM dans les étapes suivantes. Si aucune corrélation n'apparaît, il faudra donc tester une autre hypothèse, et retenter une nouvelle corrélation.

3. Etape 3 : calcul de la matrice V (valeurs intermédiaires hypothétiques)

Chargez les textes clairs (PTI.mat) et chiffrés (CTO.mat). Ceux-ci sont découpés en textes de 16 octets.
Les différents octets des textes : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 peuvent être aussi lus sous forme matricielle :

```
1  5 9   13
2  6 10 14
3  7 11 15
4  8 12 16
```

En partant de votre point d'attaque choisi, calculez la matrice V correspondante contenant les valeurs prédites pour chacune des hypothèses de clé et chaque octet de clé. Pour vous faire gagner du temps, vous pourrez utiliser la SBox de l'AES fournie en notation décimale sous forme de vecteur [3].

4. Etape 4 : calcul de la matrice H (consommation électrique/rayonnement EM hypothétique)
Calculez la matrice H correspondante, qui utilise soit le poids soit la distance de Hamming.

5. Etape 5 : corrélation
Calculer la corrélation entre la matrice H et la matrice Traces. Plusieurs raisons peuvent expliquer une corrélation faible voire inexistante : pas assez de courbes utilisées, trop de bruit, contre-mesures implémentées [4], mauvaise hypothèse de consommation, etc. Après analyse, les hypothèses de l'attaque peuvent être changées, en retournant à l'étape 2.

6. Retrouver la clé de chiffrement
Suivant l'endroit de l'attaque, vous allez récupérer une certaine clé de ronde. Quelles étapes supplémentaires faudrait-il réaliser afin de retrouver la clé maître ?

7. Questions bonus

- Ajoutez une barre de progression du temps d'exécution de l'attaque.
- Quel est le nombre de courbes minimal qui vous fournit la bonne clé de ronde ?
- Retrouvez l'intervalle de temps complet sur lequel se calcule l'intégralité des S-Boxes.
- Mesurez le temps de calcul total nécessaire pour réaliser complètement votre attaque, et tentez de le diminuer.
- Nous vous fournissons une implémentation du Key Schedule de l'AES. Le cas échéant, vous pouvez le réutiliser afin de retrouver la clé de chiffrement maître K0.
- Tester d'autres distingueurs statistiques (Différences de moyennes, MIA) et comparez leurs performances avec le coefficient de Pearson.

Quelques commandes Matlab qui peuvent être utiles :

- zeros(x,y) : matrice remplie de zéros de x lignes et y colonnes.
- disp('xxx') : affiche le message « xxx » dans l'invite de commandes (pratique pour savoir où en est l'exécution du code).
- tic/toc : mesure et affiche le temps des opérations effectuées entre « tic » et « toc » (pratique pour connaître les opérations qui prennent le plus de temps).
- load('xxx.mat') : charge le fichier Matlab « xxx ».
- plot(M) : permet de tracer les colonnes de M.
- bitxor(X,Y) : xor bit-à-bit entre X et Y.
- dec2bin : traduction d'un vecteur en notation décimale vers la notation binaire.
- sum(X) : retourne la somme des éléments de X.
- corr(X,Y) : retourne une matrice de dimension p1 x p2 contenant les coefficients de corrélation des matrices X et Y de dimension n x p1 et n x p2.
- abs : abs(X) retourne la valeur absolue de X.
- max : [Y,I] = max(X) retourne les maximums de X et leur indice.
- Hold on/Hold off : maintien des courbes sur le même graphe.

Notes :

[1] Une implémentation matérielle calquée sur les spécifications de l'AES (« *straightforward implementation* ») aurait normalement conduit à implémenter les 16 SBoxes en parallèle, pour simplifier la machine d'états globale du circuit. Nous avons décidé pour simplifier ce challenge de calculer les 16 SBoxes en série afin d'éviter que pour chaque SBox attaquée, les 15 autres apportent du « *switching noise* » additionnel qui complexifierait l'attaque.

[2] Un *Data Scientist* doit souvent affronter des données dupliquées et/ou incomplètes (c'est-à-dire que certaines valeurs de colonnes sont absentes).

[3] Suivant le point d'attaque choisi, la SBox pourrait suffire. Sinon, vous serez peut-être amené à recoder d'autres fonctions de l'AES.

[4] Pour faciliter l'attaque, aucune contre-mesure n'est implémentée.