ENSTA Bretagne

ENSI 2 - UV 4.1
Sécurité matérielle
F. Le Roy

Projet : attaque EMA AES-128

Objectif du TE

L'objectif de ce projet est de trouver la clé secrète utilisée pour un chiffrement AES implémenté en matériel sur une carte FPGA. La maquette utilisée pour obtenir la campagne de mesures est très semblable à celle étudiée dans les TE3 et TE4. La seule différence avec ces deux maquettes est que la campagne de mesures nous a donné des fuites électromagnétiques et non des fuites de courant.

---

1 Modèle de consommation

Afin de mener à bien l'attaque, il vous est proposé d'utiliser un modèle de consommation du type "poids de Hamming". Vous pourrez vous inspirer du TE5 et de son corrigé disponible sous moodle afin de vous remémorer les attaques d'ordre un et deux.

2 Information sur la campagne de mesures

La campagne de mesures est disponible sous la forme d'une archive `SECU8917.zip` située sous moodle. Vous trouverez dans cette archive une campagne de 20000 mesures de consommation correspondant au chiffrement de 20000 textes clairs en 20000 textes chiffrés. Chaque mesure de fuite électromagnétique est enregistrée dans un fichier au format csv qui est lisible par un simple logiciel tableur. Chacune d'entre elles comprend 20000 points de mesure enregistrés sous la forme de 4000 nombres flottants simple précision séparés par une virgule.

Les noms utilisés pour chacun de ces 4000 fichiers permettent de connaître la clé de chiffrement `key`, le message clair `pli` et le message chiffré `cto` associés à la fuite.

Attention à la manipulation de cette archive, elle est assez volumineuse et une bonne solution consiste à la positionner sur votre machine en local. Vous pouvez par exemple la déarchiver dans le répertoire `/tmp` de votre machine exécutant une distribution linux.

3 Cahier des charges

Afin de faciliter la mise en place de votre attaque, il vous est conseillé de suivre les étapes suivantes en vous inspirant des commandes proposées dans l'annexe A :

1. Tracer une courbe de consommation afin de déterminer le début et la fin du chiffrement.
2. Tracer la moyenne des courbes de consommation afin de repérer le début et la fin du dernier round.
3. Vous démontrerez que le modèle "poids de Hamming" peut également être appliqué au dernier round, vous pouvez par exemple vous aider du schéma de l'AES-128 proposé dans les TE3-4.
4. Donner la représentation 4x4 décimale de la clé attendue par l'algorithme d'inversion.
5. Sélectionner la partie de la courbe correspondant au dernier round afin d'optimiser la vitesse de l'attaque.
6. A partir des hypothèses de clé, calculer les valeurs intermédiaires correspondant au point d'attaque.

7. Pour chacune des valeurs intermédiaires, utiliser un modèle de consommation afin de déterminer une hypothèse de consommation à partir du message chiffré et non du message clair comme étudié dans le TE5. Vous utiliserez des matrices utilisant une troisième dimension afin de découper vos hypothèses à l'aide de sous-clé de 8 bits, expliquer pourquoi ce choix est valide.
8. Utiliser les modèles statistiques d'ordre un et deux pour trouver l'hypothèse correspondant le mieux à aux consommations mesurées.
9. Etudier la métrique "Guessing Entropy" et proposer une implémentation capable d'optimiser le nombre de traces à étudier pour un test statistique à l'ordre deux.

4 Modalités de l'évaluation

Le BE noté se déroulant sur deux séances, il n'est pas nécessaire de penser à développer une IHM (ceci n'apporterai pas de valeur ajoutée sachant que pour des raisons de rapidité, ce type d'application est souvent développé en C). Vous nommerez votre archive `EMA_NOM1_NOM2.zip`.

L'application devra fonctionner au minimum dans un environnement Linux. Le langage de développement n'est pas imposé mais Matlab est bon candidat d'autant que vous disposez sous moodle d'un corrigé en Matlab du TE5 (même si la rapidité n'est pas forcément le point fort de Matlab).

J'évaluerais en plus du respect des consignes, la maîtrise des concepts de la programmation structurée, ainsi que la qualité de vos développements et de vos programmes :

1. la conception, l'analyse, le choix du découpage en fonctionnalités, et les explications de vos algorithmes, les commentaires, la lisibilité du code, l'indentation,
2. le compte-rendu final.

5 Généralités sur l'écriture du rapport

1. Réécrire le sujet ne sert à RIEN.
2. Ce n'est pas la peine de mettre les codes en annexe (mettez les simplement dans l'archive) mais pensez à décrire la fonction de chacun des fichiers de votre archive.
3. Le compte-rendu est en pdf, taille 11 pt, interligne simple, sans fioriture (pas de titre en couleur), maximum 8 pages format A4.
4. La grammaire et l'orthographe devront être corrects.
5. Vous identifierez clairement les points du cahier des charges qui ont été clairement traités et ceux qui manquent.
6. Les difficultés que vous avez rencontrées seront décrites.
7. Les algorithmes pourront par exemple être décrits selon le modèle proche de celui utilisé dans l'annexe A (en ajoutant des dessins si nécessaire). Le pseudo code est en général inutile pour des algorithmes de base.

A Quelques fonctions Matlab utiles

- `MyFolderInfo=dir(folderSrc)` : permet créer une liste de structures qui caractérise le contenu d'un répertoire. Le chemin du répertoire est exprimé sous la forme d'une chaîne de caractères contenu par la variable `folderSrc`.

  La fonction renvoit une liste `MyFolderInfos` de structures dont chacune est composée de cinq champs : `name`, `folder`, `date`, `bytes`, `isdir` et `datenum` qui caractérise chaque élement du répertoire.
- `findstr('trace',MyFolderInfo(i).name)` : permet de rechercher la chaîne de caractères `'trace'` dans le champs `name` de l'élement `i` de la structure `MyfolderInfo`.
- `vectorTraceCsv = csvread(fullfile(folderSrc,tracename))` : lecture d'un fichier cvs dont le nom est une chaîne de caractères contenu par la variable `tracename` et dont le répertoire est la chaîne ce caractères contenu par la variable `folderSrc`.
- `[header,tail] = strtok(tracename,'=')` : recherche le caractère de délimitation `'='` dans la chaîne `tracename` et renvoit les deux sous chaînes avant (`header`) et après (`tail`) le caractère recherché `'='`.