Examen de Cartes à Puce 2

Février 2014

<u>Aucun document autorisé</u>

1) Analyse différentielle de fautes sur le DES

L'analyse différentielle de fautes sur le DES permet de retrouver la clé de tour K16 en analysant l'effet de fautes introduites durant l'exécution du 15ème tour.

Selon vous, est-il possible d'adapter cette attaque à un Triple DES ? Si oui, expliquez comment vous mèneriez cette attaque.

2) Analyse différentielle de courant sur le DES

Dans une DPA sur le DES, combien l'attaquant doit-il générer de courbes de DPA pour retrouver les valeurs de toutes les sous-clés du premier tour ? Justifiez votre réponse.

3) Modèles de consommation

On s'intéresse à la consommation de courant relevée à un instant où est manipulé un octet lors de l'exécution d'une implémentation « software » d'un chiffrement par bloc. Considérons deux exécutions différentes pour lesquelles les valeurs de cet octet sont respectivement 208 et 55.

- Dans le modèle dit « en poids de Hamming », ces deux consommations sont-elles égales, ou sinon laquelle est la plus grande ?

- Est-il possible que ces deux consommations soient égales dans le modèle dit « en distance de Hamming » ? Justifier votre réponse.

4) Algorithmes d'exponentiation modulaire

a) Expliquez à quoi sert une fonction d'exponentiation modulaire dans une carte à puce.

b) Parmi les différents algorithmes d'exponentiation modulaire, certains sont dits « réguliers ».

- Expliquez ce que signifie ce terme, et dites pourquoi il est intéressant de choisir un algorithme régulier.

- Quel est selon vous l'algorithme régulier le plus économe en nombre de multiplications modulaires (carrés et non carrés inclus) ?

c) Un développeur de RSA embarqué sur carte à puce sait très bien que l'algorithme « Square and Multiply » classique est vulnérable à une analyse simple du courant qui permet de retrouver successivement les différents bits de l'exposant secret *d* manipulé. Souhaitant malgré tout utiliser cet algorithme d'exponentiation pour des raisons de performance, il décide de protéger son implémentation en utilisant la panoplie complète des masquages (masquages d'exposant, de message et de module).

- Expliquez en quoi consistent ces différents masquages.

- Donner votre avis argumenté sur la pertinence de cette contre-mesure dans ce cas précis.

d) Écrivez le pseudo-code de l'algorithme d'exponentiation modulaire appelé « Square & Multiply Always de gauche à droite ».

Expliquez en quoi consiste l'attaque dite « Safe Error » sur cet algorithme.

e) Écrivez le pseudo-code d'une version incluant la détection de faute de l'algorithme « Square & Multiply Always de droite à gauche ».

f) Quel est le nom de l'algorithme d'exponentiation ci-dessous ?

Quelle relation vérifient les valeurs des registres R_0 et R_1 à chaque itération ?

```
Input : a, b et n des entiers,  ( b = (b_{k-1}...b_0) en base 2 )
Output : a^b modulo n

1.  R_0 = 1
2.  R_1 = a
3.  Pour i de k-1 à 0
      3.1.  R_{1-b_i} = R_0 x R_1 modulo n
      3.2.  R_{b_i} = R_{b_i} x R_{b_i} modulo n
4.  Retourner R_0
```