Examen – Attaques sur carte à puce

Durée : 1h

Alberto Battistello

a.battistello@oberthur.com

16 décembre 2015

Exercice 1. Attaques Physiques.

L'implémentation des cryptosystems modernes demande une grande attention. Nous avons vu en cours que les données sensibles manipulées par les algorithmes comme AES, DES, RSA etc... peuvent etre recuperer à partir d'attaques physiques sur les hardware et software sur lesquelles les algorithmes cryptographiques sont executées.

1. Donner quelques exemple d'attaque physique (passives et actives). [0.2 pt]
2. Expliquer la difference entre une attaque physique passive et active. [0.2 pt]
3. Expliquez la difference entre une attaque par side-channel simple et une attaque par side-channel differentiel. [0.2 pt]
4. Expliquez l'attaque sur la comparaison du PIN vu en cours (slide 20 du cours). Quel type de side-channel est utilisé pour mener l'attaque ? Suggerer une contremeasure. [0.4 pt]

Exercice 2. Analyse différentielle de consommation de courant sur l'AES.

L'analyse différentielle consiste à analyser la manipulation d'une variable intermédiaire dépendant à la fois du message et d'un petit nombre de bits de la clé secrète. Par exemple, nous avons implémenté en TD une DPA sur un octet de la sortie de la transformation *SubBytes* lors du premier tour de l'AES.

1. Quelles autres variables intermédiaires de l'AES peuvent être attaquées par analyse différentielle en faisant une hypothèse sur 8 bits de clé ? [0.5 pt]
2. Supposons qu'un attaquant souhaite mener une analyse différentielle ciblant un octet de la sortie de la transformation *MixColumns* lors du premier tour de l'AES. Sur combien de bits de clé devra-t-il faire son hypothèse afin de mener à bien son attaque ? [0.5 pt]

3. Lorsque nous observons les différentes courbes de corrélation obtenues suite à l'analyse différentielle d'un octet de la sortie de la transformation *SubBytes*, nous voyons apparaître plusieurs pics à plusieurs instants temporels différents, cf. Fig. 1. Expliquez ce phénomène. [0.5 pt]

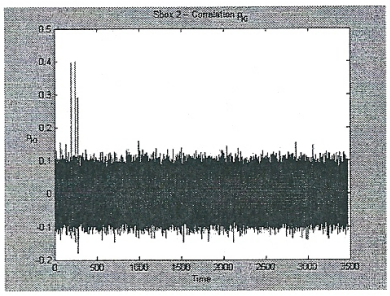

FIGURE 1 – Résultat d'une analyse CPA sur le second octet de la transformation *SubBytes* lors du premier tour de l'AES.

4. Un développeur choisie de vérifier l'intégrité de l'exécution d'un chiffrement AES en déchiffrant le chiffre obtenu et en comparant la sortie correspondante avec le texte clair d'origine. En supposant qu'un attaquant puisse faire une double faute, où chaque faute flip la valeur d'un bit dont la position peut être choisie, est-il possible d'obtenir des chiffrées erronées permettant de mener une attaque DFA tel que présentée en cours ? [0.5 pt]

Exercice 3. On a vu en cours le cryptosystème RSA-CRT. Ce système permets d'ameliorer les performances de l'algorithme RSA classique en utilisant les propriétés du théorème des restes chinois.

1. Rappeler le gain moyen en performances du RSA-CRT par rapport à un RSA classique et l'expliquer. [0.5 pt]
2. L'attaque "BELLCORE" que on a etudié en cours permet de retrouver l'un des facteur premier (p ou q) du modulus n du RSA à partir des parametres publiques (n, e), d'une signature valide S et d'une signature fauté $\tilde{S}$. Suggerer une adaptation de l'attaque dans le cas où l'attaquant ne connais pas la valeur de la signature correct S mais il connais le message m qui a été signé, ainsi que les parametres publiques (n, e), et la signature fauté $\tilde{S}$. [0.5 pt]
3. Suggérer au moins deux contremeasures qui permettent de contrecarrer l'attaque "BELLCORE" et qui ne necessitent pas de modifier l'algorithme d'exponentiation modulaire. [0.5 pt]

Exercice 4. Une implémentation du RSA sur une carte à puce utilise l'algorithme *Square-and-Multiply Always* pour effectuer l'exponentiation modulaire de la signature RSA afin de résister à l'analyse simple :

Algorithm 1: Square-and-Multiply Always RSA Signature

Input : Le message m, l'exposant privé $d = (d_{n-1}, \ldots, d_0)_2$ et le modulus N

Output: La signature $S = m^d \bmod N$

1 $T_0 \leftarrow 1$;
2 **for** $i \leftarrow n-1$ **to** 0 **do**
3 $\quad T_0 \leftarrow T_0^2 \bmod N$;
4 $\quad T_1 \leftarrow T_0 \cdot m \bmod N$;
5 $\quad T_0 \leftarrow T_{d_i}$;
6 **return** T_0;

1. Quel est le surcoût de cet algorithme en moyenne par rapport à un *Square-and-Multiply* classique si nous supposons qu'une élévation au carré prend le même temps qu'une multiplication ? `[0.5 pt]`
2. Comparer le surcoût de l'algorithme *Square-and-Multiply Always* et de l'algorithme *Montgomery Ladder* vu en cours. `[0.5 pt]`
3. Expliquer pourquoi l'algorithme *Square-and-Multiply Always* peut avantager un attaquant qui peut injecter des fautes en utiliseant un model de faute de type "saute d'instruction". Est-ce-que l'algo *Montgomery Ladder* a le meme type de probleme ? `[0.5 pt]`