Examen – Attaques sur carte à puce 2017-2018

Durée : 1h

6 points

Alberto Battistello

alberto.battistello@idemia.com

Exercice 1. Attaques par fautes.

Nous avons vu en cours comme un erreur pendant l'exécution d'un algorithme cryptographique comme AES, DES, RSA, peut permettre de retrouver des données sensibles. En particulier, nous avons vu la DFA sur l'AES. Un attaquant qui peut injecter un erreur sur un octet de l'avant dernier round peut espérer de retrouver la clef du dernier round K_{10}.

1. Quelle propriété de l'AES permet à l'attaquant de discriminer la bonne valeur de clef ? `[0.25 pt]`
2. Si la faute cible toujours le même octet de l'avant dernier round, combien d'octet de clef l'attaquant peut espérer de retrouver ? `[0.25 pt]`
3. Est il possible d'attaquer toute la clef si on change le round attaquée ? Comment ? `[0.5 pt]`
4. Décrire aux moins deux contremesures et les comparer. Est il possible d'appliquer les mêmes contremesures au DES ? `[0.5 pt]`

Exercice 2. Attaques par analyse de courant.

En cours nous avons implémenté la DPA sur l'AES. Nous allons étudier le même type d'attaque sur le DES.

1. Décrire aux moins deux fonctions de sélection pour une DPA sur le DES. Combien de bit de clef sont ciblé par chaque fonction de sélection ? `[0.5 pt]`
2. Est que utiliser la contremesure basé sur $DES(M, K) = \overline{DES(\overline{M}, \overline{K})}$ est suffisant pour se protéger de ces attaques ? Pourquoi ? Comment est elle utilisé ? `[0.5 pt]`
3. Nous avons vu en cours la contremesure appelé "masquage'. Cette contremesure consiste à XORer un octet X (la valeur sensible) et un octet d'aléa M (le masque), puis manipuler $X \oplus M$ et M, mais jamais X. Nous avons supposé que l'octet M soit choisis aléatoirement parmi 0 et 255.

   Qu'est-ce qui se passe si la distribution du masque est biaisé et il ne peut prendre que la moitié des valeurs (par exemple de 0 a 127) ? `[0.5 pt]`

Exercice 3. Algorithme RSA.

Algorithm 1: Montgomery Ladder

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

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

$R_0 \leftarrow 1$;

$R_1 \leftarrow m$;

for $i \leftarrow n-1$ **to** 0 **do**

  if $d_i == 0$ **then**

    $R_1 \leftarrow R_0 R_1 \bmod N\,; R_0 \leftarrow (R_0)^2 \bmod N$;

  if $d_i == 1$ **then**

    $R_0 \leftarrow R_0 R_1 \bmod N\,; R_1 \leftarrow (R_1)^2 \bmod N$;

return R_0;

Algorithm 2: Square-and-Multiply Always

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

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

$R_0 \leftarrow 1$;

for $i \leftarrow n-1$ **to** 0 **do**

  $R_0 \leftarrow R_0^2 \bmod N$;

  $R_1 \leftarrow R_0 \cdot m \bmod N$;

  $R_0 \leftarrow R_{d_i}$;

return R_0;

---

1. Expliquer pourquoi utiliser l'algorithme *Square-and-Multiply Always* (cf. Alg. 2) pour calculer une signature RSA peut avantager un attaquant qui peut injecter des fautes perturbant le calcul d'une multiplication. Est-ce-que l'algorithme *Montgomery Ladder* (cf. Alg. 1) a le même type de problème ? `[1 pt]`

   Nous avons vu en cours le cryptosystème CRT-RSA. Ce système permets d'améliorer les performances de l'algorithme RSA classique en utilisant les propriétés du théorème des restes chinois.

2. Rappeler le gain moyen en performances du CRT-RSA par rapport à un RSA classique et l'expliquer. `[1 pt]`
3. L'attaque "BELLCORE" que on a étudié en cours permet de retrouver l'un des facteur premier (p ou q) du module N du RSA à partir des paramètres publiques (n, e), d'une signature valide S et d'une signature fauté $\tilde{S}$. Suggérer 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 paramètres publiques (n, e), et la signature fauté $\tilde{S}$.`[1 pt]`