Partie I. Tobor

10 points

Exercice 1, Questions de cours

1. Quelle est la différence entre « Cold Reset » et « Warm Reset » ? Et quelles sont les conséquences sur le contenu de mémoires ROM, RAM et EEPROM ?
2. En Java Card, apres `byte a=(byte)0xf0; short b=(short)a;` combien vaut b ?
3. La même carte, même application, même commande (et en considerant la même historique de la carte) peut donner les resultats differents. Par exemple la commande SELECT (supposons `00 a4 04 00 07 a0 00 00 00 45 10 14`) peut donner :
   - `61 43`
   - ou bien `6f 41 84 07 a0 00 00 00 45 10 14 a5 36 50 0a 4d 41 53 54 45 52 43 41 52 44 87 01 02 5f 2d 06 66 72 65 6e 65 73 9f 11 01 01 9f 12 0a 4d 41 53 54 45 52 43 41 52 44 bf 0c 0a df 60 02 0b 19 9f 4d 02 0b 19 90 00`

   Pourquoi ? Que faut-il faire dans le 1er cas ?

Exercice 2, Authentification

L'authentification est la procédure qui consiste, pour un système informatique, à vérifier l'identité d'une autre entité afin d'autoriser l'accès de cette entité à des ressources. L'authentification permet donc de valider l'authenticité de l'entité en question. Dans le cadre de cartes à puce la procédure d'authentification permet donc, par exemple de vérifier l'authenticité de la carte du point de vue du terminal ou l'inverse.

Plusieurs exemples peuvent se présenter :

1. Commande `INTERNAL AUTHENTICATE` permet au terminal de s'assurer que la carte est authentique. La carte chiffre les données aléatoires envoyée par le terminal et le terminal vérifie si le chiffrement est correct.
2. Commande `EXTERNAL AUTHENTICATE` qui permet à la carte de s'assurer que le terminal qui lui envoie des commandes est authentique. L'idée cette fois-ci est que c'est le terminal qui doit chiffrer les données aléatoires envoyées par la carte et puis la carte, à sont tour, doit les vérifier.
3. Commande `MUTUAL AUTHENTICATE` permet de prouver mutuellement à la carte et au terminal que l'"autre" entité connaît bien les même clefs de chiffrement. Elle regroupe les deux cas précédents.

Notez que, pour les besoins de cet exercice, les hypothèses et les suppositions sont simplifiées par rapport aux cas réels. Il n'est pas demandé de donner la réponse exacte (un tel nombre d'octets, un tel format), seulement des idées ou des principes.

Questions :

1. Dans le cas (1) un échange (commande APDU et puis sa réponse) est nécessaire. Quelles données sont envoyés et quelles données sont reçues ?
2. Dans les cas (2) deux échanges sont nécessaires (en général les commandes Get Challenge et External Authenticate). D'où vient cette asymétrie avec le cas (1) ?
3. Il faut combien d'echanges dans les cas (3) ? Comme c'est un peu peu le cas (1) + (2) il est claire que trois echanges suffisent. Peut on faire mieux (moins) ? Comment ?

Exercice 3, Analyse de log

Les lignes qui suivent correspondent aux 7 records consecutifs du fichier "Log de transactions" de ma carte bancaire EMV. On peut y trouver des transactions faites pendant mes deplacements des 3 jours entre France, Allemagne et Suisse. En Suisse dans certains distributeurs d'argent on peut choisir entre faire une transaction en francs suisses (CHF) ou bien en euros (€) pour éviter les frais de change de la banque émettrice de la carte (et dans ce cas là c'est la banque propriétaire du distributeur qui applique sa commission).

- `00 00 00 00 06 99 40 02 50 09 78 18 11 20 00 00 6f 90 40 03 22 00 00 01`
- `00 00 00 00 10 50 40 02 76 09 78 18 11 15 00 00 6a 65 70 03 02 00 00 00`
- `00 00 00 00 39 00 40 02 76 09 78 18 11 15 00 00 69 95 40 03 02 00 00 00`
- `00 00 00 00 19 99 40 07 56 07 56 18 11 14 00 00 68 67 70 03 04 00 00 00`
- `00 00 00 00 24 63 40 07 56 09 78 18 11 13 00 00 67 67 70 03 04 00 00 00`
- `00 00 00 00 10 00 40 02 50 09 78 18 11 13 00 00 66 90 40 03 22 00 00 01`
- `00 00 00 00 67 60 40 02 50 09 78 18 11 10 00 00 65 95 40 03 02 00 00 00`

La commande `GET DATA` avec l'argument 9F 4F ("Log Format") répond:

- `9f 4f 19 9f 02 06 9f 27 01 9f 1a 02 5f 2a 02 9a 03 9c 01 9f 36 02 9f 52 06 df 3e 01`

Notez que pour répondre correctement à 2 premieres questions vous n'avez pas besoin de connaître la signification exacte de chacun des tags utilisés. Un peu de réflexion et de bon sens sont suffisants.

Questions :

1. Combien j'ai payé en CHF ?
2. Combien j'ai depensé en Allemagne ?
3. A votre avis que signifie le tag `9f36` ? Et `df3e` ?,

Barème (sur 10 points):

- Exercice 1 : 2 pts
- Exercice 2 : 3 pts
- Exercice 3 : 5 pts

Examen – Attaques sur carte a puce 2018-2019

Durée : 1h

6 points

Alberto Battistello

alberto.battistello@idemia.com

Exercice 1. Attaques par fautes.

Nous avons vu en cours comme une 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 une erreur sur un octet de l'avant dernier round peut espérer de retrouver une partie de clef du dernier round K_{10}.

1. Combien de bit d'hypothèse doit faire l'attaquant si la faute à été injecté en entrée du MixColumns du 9eme tour ? Expliquer. [0.5 pt]
2. Est il possible d'attaquer toute la clef si on change le round attaquée ? Comment ? [0.5 pt]
3. Suggérer une contremesure et l'expliquer. Est il possible d'appliquer la même contremesure au DES ? [0.5 pt]
4. Expliquer comment la contremesure basé sur $DES(M, K) = \overline{DES(\overline{M}, \overline{K})}$ est utilisé pour se protéger des attaques par faute avec modèle d'erreur stuck-at-0 sur un bit. Est-ce que cette contremesure est suffisant pour se protéger des attaques DFA sur le DES ? Pourquoi ? [0.5 pt]

Exercice 2. Algorithme RSA.

Nous avons vu en cours l'algorithme RSA, en particulier l'algorithme *Square-and-Multiply* (cf. Alg. 1) permet de calculer une exponentiation modulaire.

1. Expliquer comment l'attaquant peut retrouver l'exposant privé d en analyse simple (SPA) sur une seule courbe de consommation correspondante à l'exécution de l'algorithme *Square-and-Multiply* (cf. Alg. 1). [1 pt]
2. Expliquer comment sécurisér l'algorithme *Square-and-Multiply* (cf. Alg. 1) pour contrer l'attaque précédent. [0.5 pt]
3. Est-il possible de monter une attaque DPA pour retrouver l'exposant privé d pendant l'exécution de l'algorithme que vous avez suggéré á l'étape 2 ? (en particulier, quelle taille peut faire l'hypothèse de clef ? Comment l'utiliser ?). [1 pt]

Algorithm 1: Square-and-Multiply

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

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

```
1 R_0 ← 1;
2 for i ← n − 1 to 0 do
3 |   if d_i == 0 then
4 |   |   R_0 ← R_0^2 mod N;
5 |   R_0 ← R_0 · m mod N;
6 return R_0;
```

4. Est-ce que le nouveaux algorithme que vous avez conçu à l'étape 2 est résistant aux attaques par faute simple ? Est-ce que un attaquant qui peut injecter une faute (sur une ou plusieurs exécutions) de type saut d'instruction pourrait retrouver l'exposant privé ? [0.5 pt]

Exercice 3. Algorithme CRT-RSA.

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

1. Rappeler le gain moyen en performances du CRT-RSA par rapport á un RSA classique et l'expliquer. [0.5 pt]
2. 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 á partir des paramétres publiques (n, e), d'une signature valide S et d'une signature fauté $\tilde{S}$. Rappelez l'attaque, puis suggérez une adaptation de l'attaque dans le cas oú 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 $\tilde{S}$. [0.5 pt]