Master Cryptis
UE Cartes à puce

2012-2013

Examen

Janvier 2013

Tout document non électronique autorisé

Partie 1 : Sécurité Java Card (10 points)

1) Imaginez un mécanisme pour se prémunir efficacement contre une modification de l'adresse de retour d'une fonction.

2) Comment puis je efficacement et sans gestion de la part du développeur d'application protéger en intégrité et en confidentialité mes variables d'instance si ces dernières contiennent un short, deux octets et un booléen ?

3) A quelles attaques l'algorithme suivant de vérification de PIN est-il sensible et pourquoi ?

```
boolean test = true;
for ( i = 0 ; i <= 7; i++)
      test = test && ( pinCarte [ i ] == pinPresente [ i ] );
if (test)
      essai = (byte) (4 – 1);
else
      essai --;
return test;
```

4) Soit le code Java Card suivant pour initialiser et utiliser une clé RSA **Key** instanciée dans le constructeur et ce pour une signature, que manque-t-il (on suppose que toutes les variables ont été correctement déclarées) ? Est-il possible d'améliorer la sécurité de cet algorithme ?

```
public void process(APDU apdu) {

if (selectingApplet()) {return;}
byte[] buf = apdu.getBuffer();
short len = (short)((short) buf[(short)ISO7816.OFFSET_LC] & (short)0x00FF);
short numBytes = apdu.setIncomingAndReceive();

while (numBytes < len)
       numBytes += apdu.receiveBytes((short) ((short)ISO7816.OFFSET_CDATA
+numBytes));
   switch (buf[ISO7816.OFFSET_INS]) {
        case (byte) 0xB0: break;
        case (byte) 0x10: Key.setExponent(buf, ISO7816.OFFSET_CDATA, len);
                          break;
        case (byte) 0x20: Key.setModulus(buf, ISO7816.OFFSET_CDATA, len);
                          break;
        case (byte) 0x40: Sign.update(buf, ISO7816.OFFSET_CDATA, len);
                          break;
        case (byte) 0x30: verifies = Sign.verify (miss,
ISO7816.OFFSET_CDATA, len, buf, ISO7816.OFFSET_CDATA, len);
        default: ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }
}
```

Master Cryptis
UE Cartes à puce

2012-2013

Partie 2 : Attaques matérielles (10 points)

5) Attaque par collision sur le COMP128

Dans l'attaque par collision classique (dite « BGW ») sur l'algorithme COMP128, décrivez comment l'attaquant choisit les valeurs des RAND qu'il envoie à la carte lorsque qu'il souhaite retrouver les octets de clé K[2] et K[10].

6) Analyse différentielle du courant sur l'AES

Voici le pseudo-code simplifié d'une implémentation de l'AES :

```
Input : message M de 16 octets, clé K de 16 octets
Output : chiffré de 16 octets

1. (K_0, K_1, …, K_{10}) = KeySchedule(K)
2. S = M
3. S = AddRoundKey(S, K_0)
4. Si (rand()%2) == 0 alors  // rand()%2 renvoie 0 ou 1 avec proba ½ chacun
      4.1 attendre 10 micro-secondes
5. Pour i de 1 à 9
      5.1 S = SubBytes(S)
      5.2 S = ShiftRows(S)
      5.3 S = MixColumns(S)
      5.4 S = AddRoundKey(S, K_i)
6. S = SubBytes(S)
7. S = ShiftRows(S)
8. S = AddRoundKey(S, K_{10})
9. Retourner S
```

Expliquez l'effet qu'aura l'instruction 4. lorsqu'on réalise une DPA sur les sorties des S-Box du premier tour.

7) 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.

8) 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.

c) L'algorithme appelé « Montgomery ladder » fonctionne ainsi :

```
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
```

Cet algorithme est-il régulier ?

Donnez une propriété reliant R_0, R_1, a et n qui est toujours vérifiée à la fin de chaque tour de boucle.

Expliquez comment tirer partie de cette propriété pour contrer certains types d'attaques physiques.