Composition de Cryptographie - 2009

Il est recommandé aux élèves de bien choisir l'ordre des questions selon leurs compétences et rapidités.

Si l'élève n'arrive pas à faire une démonstration, il peut considérer que le résultat de la démonstration est admis sur le reste l'exercice.

Le support de cours et les calculatrices sont permis.

1. DES et carte à puce

Les cartes D1 et D2 Philips fournissent le DES en chiffrement et déchiffrement. Or la mémoire de programme de ces cartes étant limité, toute économie en code exécutable dans la carte doit être prise en compte.
Le bloc de données à chiffrer ou déchiffrer est présenté à la carte, et la requête d'exécution du DES est appelée.

a) Quel est le rôle des permutations initiales et finales du DES ?
b) Comment le DES au niveau des cartes D1 et D2 peut être implémenté pour économiser de l'espace mémoire programme ?
c) Dans le dernier cas que faut-il présenter aux cartes comme bloc pour chiffrer ?
d) Est-ce que le point c (précédent) est suffisant et pourquoi ?

2. Padding Chiffrement AES

Dans le standard EBICS il est précisé que les deux paddings suivants sont supportés

- The ANSIX923 padding string consists of a sequence of bytes filled with zeros before the length byte.
- The ISO10126 padding string consists of random data before the length byte.

Or le protocole Ebics ne précise pas dans les données protocolaires échangées le type du padding et les deux modes doivent être supportés.

a) Quel est le mode de padding le plus faible parmi les deux précités ?
b) Que préconiser vous côté entité envoyant les données chiffrées d'utiliser comme mode de padding ?
c) Que préconisez-vous côté récepteur de données chiffrées de faire?
d) Indépendamment d'Ebics et de point de vue général si on s'attend à un padding avec des 00 et qu'on reçoit un padding avec des FF, donner deux sources possibles de ce dysfonctionnement.

3. Modes RSA

a) Rappeler le mode d'attaque contre les petits exposants publics en RSA ?
b) Comment le mode PKCS1 permet de se protéger contre cette attaque ?

4. Codage DER du MD5

A partir de la déclaration ASN1 d'un algorithme

```
AlgorithmIdentifier  ::= SEQUENCE {
   algorithm      id, //OID
   parameters   Type OPTIONAL
}
```

Implémenter le MD5 en DER (valeurs différentes dans support du cours).

5. Analyse DER d'une clé publique RSA

A partir de l'implémentation DER d'une clé publique :

```
30 82 01 09 02 82 01 00 00 f5 79 80 ee f7 26 06 42 79 d3 ef e0 8b 6a d0 0e d6 7c 8b 38
be 9b 36 08 91 8f 70 f1 63 42 75 d1 ed d7 ff b9 ba d3 25 0d 2d c8 1b 3e 70 2b 21 bc e3
83 d0 79 17 3e 96 c5 0d 83 f1 fb 85 35 10 59 36 da 24 f3 c6 70 4c 15 be e6 74 94 14 19
dc 33 b6 3d f6 65 af 3f 4c 8c df be 03 24 97 5c 16 57 9e 2f b0 95 1f 7c 59 e2 86 7e 72
c9 10 68 65 ed bc 9b c1 55 a0 9f 60 24 5d 58 09 c2 3b ed 4e 06 17 5b b0 5b 2e ee 48 1a
82 85 5e ce e3 dc fa a0 b5 a1 12 1c 96 05 0f 52 b7 aa bd e1 b4 bc fb 54 58 fc 88 2b f7
5d 5e d1 83 98 95 9e a6 fd 88 80 0f 92 4d 2d 90 f7 b9 3a 78 dd d5 df 83 0d a8 c4 05 94
a7 96 c9 1a c6 14 23 bf 5e 4e fd 59 e7 8f 26 c5 c8 bb 23 ab a9 4f c0 25 fa c8 b8 06 44
99 41 63 05 c6 a3 8c 39 00 66 36 42 6e ef 94 12 f7 57 64 3d 7d ba 36 2a 01 0e a1 a4 25
b9 aa 01 02 03 01 00 01
```

a) Analyser les divers champs de cette implémentation. (combien, le type, la taille, …)
b) Que vaut le modulo ? (définir les deux octets de début et les deux octets de fin)
c) Quel vaut l'exposant ?
d) Quelle est la taille réelle en bits de cette clé ?
e) A partir du point d conclure sur une méthode pratique de calculer la taille réelle de la clé à partir de l'implémentation DER à partir d'une taille de 2048 bits.

6. Analyse de Code

Le code suivant est utilisé pour générer des valeurs aléatoires

```
int fRandom = 0 ;
char X0[??] ;

void GenerateRandom(int lgRandom /* lgRandom est fixe et vaut 40 */,  char *Random)
{
char xdatetimez[65];
int lgHash;
char Hash[256];

……. . . . .


// obtention du temps  sous la forme YYYY-MM-DD-HH-MM-SS-Milliseconds
       os_datetimez(xdatetimez);

// Hasher le temps en utilisant le SHA256
// Sortie est Hash
       cr = HashData(sha256, strlen(tobehashed), xdatetimez, &lgHash, Hash);
       if(fRandom==0)    // Opération XX
       {
              memcpy(X0, Hash, lgHash);
              fRandom=1;
       }
       else
              for(i = 0; i<lgHash; i++)
                           X0[i]^=xdatetimez[i];
       cr = HashData(sha256, lgHash, X0, &lgHash, Hash); // Rehasher
       memcpy(X0, Hash, lgHash);
       for(i = 0; i<lgRandom; i+=2) // Transformer
                    sprintf(Random+i,"%2.2X", X0[i/2]);
}
```

a) Que doit être la taille minimale de X0? Et pourquoi ?
b) Quelle est la taille utilisée de X0 ?
c) A quoi sert fRandom notée Opération XX en commentaire ?
d) Pourquoi fRandom et X0 sont déclarés globalement ?
e) Pourquoi ce code ne génère pas de collisions sur un poste client qui ne fait que des transferts vers des serveurs en série (pas de plusieurs transferts parallèles).
f) Pourquoi ce code peut générer des collisions sur un poste serveur multi-threads qui peut recevoir des transferts en parallèle ?
g) Que peut-on ajouter au code précédent pour se protéger contre les collisions sur le poste serveur ?