Principes et Algorithmes de Cryptographie

Examen du 10 juin 2004

Durée 3 heures - documents autorisés
Toute réponse devra être justifiée

Exercice 1 : [40mn] *Un générateur de bits pseudo-aléatoires*

On considère un générateur de bits obtenu à partir d'un automate dont les états sont des entiers codés sur n bits. La fonction de transition de cet automate utilise des opérations arithmétiques (addition et élévation au carré modulo 2^n) et une opération logique (le ou bit à bit, noté $\vee$). Plus précisément, si e_i désigne l'état de l'automate à l'instant i, à l'instant suivant l'état est donné par

$$e_{i+1} = (e_i + (e_i^2 \vee 5)) \pmod{2^n}$$

On considère dans cet exercice que les entiers sont codés sur $n = 3$ bits, et que l'état initial est $e_0 = 0$.

Q 1 . Calculez les suites de bits produites par cet automate en supposant qu'à chaque état le bit produit est

1. le bit de poids faible de l'état,
2. le bit intermédiaire,
3. le bit de poids fort.

Dans chacun des cas indiquez la période de la suite produite ?

Q 2 . Quelle est la complexité linéaire de la suite produite dans le cas du bit de poids fort ?

Exercice 2 : [30mn] *Fonctions de hachage*

Soit $m = m_1 m_2 \dots m_n$ une chaîne de bits dans laquelle pour chaque $i = 1 \dots n$, m_i est un bloc de 128 bits. On définit une fonction de hachage H qui opère sur les mots binaires de cette forme en posant

- h_0 est un bloc de 128 bits tous nuls,
- pour chaque $i = 1 \dots n$, $h_i = AES_{m_i}(h_{i-1})$, où $AES_K(m)$ est le résultat du chiffrement du bloc m avec la clé K,
- $H(m) = h_n$.

Q 1 . Montrez comment on peut trouver des collisions pour H en appliquant approximativement $c.2^{64}$ fois l'AES où c est une constante.

Q 2 . Étant donnée une chaîne m, montrez comment trouver une chaîne différente m' telle que $H(m) = H(m')$, en appliquant approximativement 2^{64} fois l'AES. (indication : inspirez-vous de l'attaque sur le double DES)

Exercice 3 : [1h] *Partage de secret*

Nous allons construire dans cet exercice un schéma de partage de secret vectoriel, reposant sur des structures d'accès monotones.

Présentation du schéma Soit $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ un corps fini à p éléments, p étant un nombre premier. Soit $\mathcal{P}$ l'ensemble des personnes devant recevoir une part, $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$. Soit Γ la structure d'accès du schéma.

On supposera[1] que l'on dispose d'une fonction $\phi : \mathcal{P} \to \mathbb{F}_p^d$, $d \geq 2$ telle que

$$B \in \Gamma \iff (1, 0, \dots, 0) \in Vect\,(\phi(P_i) : P_i \in B)\,, \tag{1}$$

autrement dit : $(1, 0, \dots, 0)$ est une combinaison linéaire (à coefficients dans $\mathbb{F}_p$) des vecteurs de l'ensemble $\{\phi(P_i) : P_i \in B\}$ si et seulement si B est un sous-ensemble autorisé à accéder au secret.

Soit K le secret à partager. Pour tout élément $a = (K, a_2, \dots, a_d) \in \mathbb{F}_p^d$, on peut alors définir une distribution de parts pour chaque $x \in \mathcal{P}$: $f_a(x) = a \cdot \phi(x)$, où $\cdot$ désigne le produit scalaire.

[1] Une telle fonction existe toujours.

Exemple Soient $p = 19$, $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$, $d = 3$, $\Gamma_0 = \{\{P_1, P_2, P_3\}, \{P_1, P_4\}\}$, et

$$\phi(P_1) = (0, 1, 0),\ \phi(P_2) = (1, 0, 1),\ \phi(P_3) = (0, 1, -1),\ \phi(P_4) = (1, 1, 0).$$

On souhaite partager le secret $K = 17$ en quatre parts. On choisit alors un élément a de $\mathbb{F}_{19}^3$ dont la première composante est 17, par exemple, $a = (17, 5, 9)$. On calcule alors les parts, dans $\mathbb{F}_{19}$:

$$\begin{array}{rcl} P_1 & : & s_1 = (17, 5, 9) \cdot (0, 1, 0) = 5 \\ P_2 & : & s_2 = (17, 5, 9) \cdot (1, 0, 1) = 7 \\ P_3 & : & s_3 = (17, 5, 9) \cdot (0, 1, -1) = 15 \\ P_4 & : & s_4 = (17, 5, 9) \cdot (1, 1, 0) = 3 \end{array}$$

Q 1 . Vérifiez que, dans cet exemple, la propriété (1) est bien satisfaite.
Q 2 . Montrez comment, dans l'exemple, les ensembles de participants $\{P_1, P_2, P_3\}$ et $\{P_1, P_4\}$ opèrent pour retrouver le secret.
Q 3 . Les valeurs de ϕ peuvent-elles être rendues publiques ?
Q 4 . Nous avons vu en cours un autre schéma de partage de secret utilisant des structures d'accès, et reposant sur les circuits monotones. Lequel est le plus pratique du point de vue des participants ?
Q 5 . Montrez que le schéma de partage de secret à seuil de Shamir est un cas particulier de celui présenté dans cet exercice.

Exercice 4 : [30mn] *Générateur de clés RSA*

Voici un générateur de clés RSA écrit en Java

```
import java.io.*;
import java.math.BigInteger;
import java.util.Random;

class genRSA {

  public static void main(String arg[]) {
    BigInteger P,P1,Q,Q1,N,PHI,E,D;
    Random alea = new Random();

    E = BigInteger.valueOf(65537);

    do {
      P = new BigInteger(512,20,alea);
      P1 = P.subtract(BigInteger.ONE);
    } while (P1.gcd(E).equals(BigInteger.ONE) == false);

    do {
      Q = new BigInteger(512,20,alea);
      Q1 = Q.subtract(BigInteger.ONE);
    } while (Q1.gcd(E).equals(BigInteger.ONE) == false);

    N = P.multiply(Q);
    PHI = P1.multiply(Q1);
    D = E.modInverse(PHI);

    System.out.println("n = " + N.toString(16));
    System.out.println("e = " + E.toString(16));
    System.out.println("d = " + D.toString(16));
  }
}
```

dont voici une trace d'exécution

```
$ java genRSA
n = b3414836c17988f4399739494cbc39ffd5727dd7b16a065ddb84afa749e080d616b0a10ede6
19b8e698fb5d2848632033d2c94dde6500ccb543c6a50ba65269c6320a8db75da4aee8f09f49072
d413fb7d347c05ef6c5a427d4366f46d6b6f4f20bce39dc3f89b9bec805bb7251f2ddc994ce88af
4e646760b7802be3f049a81
e = 10001
d = 6d02a356e13bf6c4870d6702238f483a43e4f790a74cd2085c0a0a0453121b6796aedd933c3
be1acae977dbc4369949a91a63df5e02d084ad2b456f7371372719d443012b6d18f1376d4a1a8ab
5af4e3f01cd22e3561f0f353b711a981d35ab45e900111676b01d5e1e20f7577033e4b25103264f
a6f5c8f1b485055e8a86f7d

$
```

Voici un extrait de la spécifiaction de la classe `Random`

```
public class Random
extends Object
implements Serializable

An instance of this class is used to generate a stream of pseudorandom numbers.
The class uses a 48-bit seed, which is modified using a linear congruential formula.
(See Donald Knuth, The Art of Computer Programming, Volume 2, Section 3.2.1.)

[...]

Random

public Random()

    Creates a new random number generator. Its seed is initialized to a value based
    on the current time:

 public Random() { this(System.currentTimeMillis()); }

    Two Random objects created within the same millisecond will have the same sequence
    of random numbers.
```

et un extrait de celle de `BigInteger`

```
BigInteger

public BigInteger(int bitLength,
                  int certainty,
                  Random rnd)

    Constructs a randomly generated positive BigInteger that is probably prime,
    with the specified bitLength.

    It is recommended that the probablePrime method be used in preference to this
    constructor unless there is a compelling need to specify a certainty.

Parameters:
    bitLength - bitLength of the returned BigInteger.
    certainty - a measure of the uncertainty that the caller is willing to tolerate.
                The probability that the new BigInteger represents a prime number
                will exceed (1 - 1/2certainty). The execution time of this constructor
                is proportional to the value of this parameter.
    rnd - source of random bits used to select candidates to be tested for primality.
```

Q 1 . D'après-vous que représente la variable `E` ? Pourquoi choisir cette valeur ?
Q 2 . Que représentent les variables `P` et `Q` ? Quel est le rôle des boucles `do ... while (...)` qui les définissent ?
Q 3 . Quelle est la taille des clés RSA générées ?
Q 4 . Indiquez pourquoi ce générateur n'offre pas une sécurité suffisante en proposant une méthode d'attaque dont vous évaluerez la complexité.