Projet 2 de cryptographie

March 12, 2020

La présentation de votre document, la qualité de votre code seront prises en compte

1 Parie I : Modes de Chiffrement

Faire un résumé des modes d'opération pour le chiffrement sysmétrique. Il faudra

- discuter de l'aspect sécurité
- faire des diagrammes
- faire des compaisons

2 Partie II : Implémentation de DES et AES

Pour chacun des schémas de chiffrement symétriques `DES` et `AES`, implémenter en Python

1. L'algorithme de génération de clefs `genkey_system` qui prend en entrée si necessaire la taille de la clef et retourne une clef.
2. L'algorithme de génération de sous-clefs `genkey_subkeys_system` qui prend en entrée une clef `K` et retourne des sous- clefs K_i.
3. L'algorithme de chiffrement `cipher_system` qui prend en entrée un message `M` (64 bits pour le DES, 128, 192, 256 bits pour l'AES) et une clef `K` et retourne son chiffré `C`. Vous pourriez faire appel à l'algorithme `genkey_subkeys_system`.
4. L'algorithme de déchiffrement `decipher_system` qui prend en entrée un ciphré `C` (64 bits pour le DES, 128, 192, 256 bits pour l'AES) et une clef `K` et retourne son déchiffré `M`. Vous pourriez faire appel à l'algorithme `genkey_subkeys_system`.

3 Partie III : Modes d'opérations et DES, AES

Faire une implémentation plus générale du `DES` et de l'`AES` en tenant compte des modes de chiffrement. Le message en entrée maintenant est de taille quelconque. On fera du Padding si nécessaire.

Reprener donc les question `3.` et `4.` de la `partie II`.

4 Partie IV : Implementation d'algorithmes sur $\mathbb{Z}$ et $\mathbb{Z}/n\mathbb{Z}$

Implémenter en python

1. l'agorithme d'Euclide pour le calcul du PGCD,
2. l'agorithme d'Euclide étendu
3. l'algorithme de calcul d'inverse modulo
4. l'algorithme de puissance (modulaire) rapide

5 Annexes

ECB (Electronic Code Book Mode)

Encryption:
Obtain ciphertext $C_1, \ldots, C_t$ as $C_i = E_K(M_i), i = 1, \ldots, t$

Decryption:
Obtain the plaintext from $C_1, \ldots, C_t$ as $M_i = E_K^{-1}(C_i), i = 1, \ldots, t$

CBC (Cipher Blockchaining mode)

CBC uses a (non-secret) initialization vector (IV) of n bits.

Encryption:
Obtain ciphertext $C_1, \ldots, C_t$ as $C_i = E_K(M_i \oplus C_{i-1}), i = 1, \ldots, t; C_0 = IV$

Decryption:
Obtain message from $C_1, \ldots, C_t$ as $M_i = E_K^{-1}(C_i) \oplus C_{i-1}, i = 1, \ldots, t; C_0 = IV$

CFB (Cipher Feedback Mode)

Also CFB uses a non-secret IV of n bits.

Encryption:
Obtain ciphertext $C_1, \ldots, C_t$ as $C_i = E_K(C_{i-1}) \oplus M_i, i = 1, \ldots, t; C_0 = IV$

Decryption:
Obtain plaintext from $C_1, \ldots, C_t$ as $M_i = E_K(C_{i-1}) \oplus C_i, i = 1, \ldots, t; C_0 = IV$

OFB (Output Feedback Mode)

OFB also uses a non-secret IV of n bits.

Encryption:
Generate $O_1, \ldots, O_t$ as $O_i = E_K(O_{i-1}), i = 1, \ldots, t, O_0 = IV$
Obtain $C_i = M_i \oplus O_i, i = 1, \ldots, t$

Decrytion:
Generate key stream $O_1, \ldots, O_t$ as in encryption
Obtain $M_i = C_i \oplus O_i, i = 1, \ldots, t$

CTR (Counter Mode)

The CTR mode uses a nonce N of l bits, $l < n$
CTR uses a counter start value C of $n - l$ bits.

Encryption:
Generate a Keystream $O_i, \ldots, O_t$ as $O_i = E_K(N|((C + i) \mod 2^{n-l}))$ Compute $C_i = M_i \oplus O_i, i = 1, \ldots, t$

Decryption:
Obtain the message as $M_i = C_i \oplus O_i, i = 1, \ldots, t$

Algorithm 1: Algorithme d'Euclide pour le calcul du PGCD

Input: Deux entiers a et b avec $a \geq b$

Output: Le pgcd de a et b i.e $\mathrm{pgcd}(a, b)$

```
1 while b ≠ 0 do
2     r = a mod b;
3     a = b;
4     b = r;
5 return a
```

Algorithm 2: Euclide-etendu(a, b)

Input: Deux entiers a et b avec $a \geq b$

Output: Trois entiers (u, v, d) tels que $au + bv = d$ où d est le pgcd de a et b

```
1  if b = 0 then
2      d = a; u = 1; v = 0;
3      return u, v, d
4  u_2 = 1; u_1 = 0; v_2 = 0; v_1 = 1;
5  while b > 0 do
6      q = ⌊a/b⌋ ;                    /* la partie entière de a/b */
7      r = a − qb;
8      u = u_2 − qu_1;
9      v = v_2 − qv_1;
10     a = b ; b = r ; u_2 = u_1 ; u_1 = u ; v_2 = v_1 ; v_1 = v ;
11 return u, v, d
```

Algorithm 3: InverseEntier(a, n)

Input: Deux entiers a et n avec $\mathrm{pgcd}(a, n) = 1$

Output: L'inverse de a modulo n

```
1 u, v, d =Euclide-etendu(a, n) /* au + nv = d = 1          */
2 return u mod n
```

Algorithm 5: Algorithme de calcul d'une puissance modulaire

Input: Trois entiers a, m et n ave $0 \leq m < n$

Output: $b = a^m \mod n$

```
1  Déterminer la représentation binaire de m i. e m = Σ_{i=0}^{k} m_i 2^i;
   /* m = (m_k m_{k-1} ... m_3 m_2 m_1 m_0)_2                          */
2  b = 1;
3  if m = 0 then
4      return b
5  A = a;
6  if m_0 = 1 then
7      b = a;
8  for i allant de 1 à k do
9      A = A^2 mod n;
10     if m_i = 1 then
11         b = A × b;
12 return b
```