Institut Informatique Claude Chappe

Licence L2 STS Mention SPI
Parcours Informatique
Unité 174EN007
Sécurité Informatique

Correction TP3
Implémentation du chiffre RC4

Ce TP porte sur l'implémentation et la mise en œuvre du système de chiffrement symétrique RC4 en langage C.

Questions 1), 2) et 3)

Voici les fichiers sources correspondants à la réponse des questions 1) , 2) et 3) du sujet.

Fichier *RC4.h*

```
#ifndef RC4_H
#define RC4_H

/****************** DATA TYPES ********************************/

typedef unsigned char BYTE; // 8-bits byte

typedef struct {
    BYTE s[256];
} RC4_CTX;

/****************** FUNCTION DECLARATIONS **********************/

void rc4_init(RC4_CTX * ctx, const BYTE key[], int keylen);
void rc4_crypt(RC4_CTX * ctx, const BYTE m[], int mlen, BYTE c[], int clen);

#endif
```

Fichier *test_RC4.c*

```
#include <stdio.h>
#include <string.h>
#include "RC4.h"


void print_hexa(BYTE c[], int len) {
    for (int i=0; i < len; i++)
        printf("%02X", c[i]);
    printf("\n");
}


void print_char(BYTE c[], int len) {
    for (int i=0; i < len; i++)
        printf("%c", c[i]);
    printf("\n");
}
```

```
int main() {
    BYTE key1[] = {"Key"};
    BYTE key2[] = {"Wiki"};
    BYTE key3[] = {"Secret"};

    BYTE text1[] = {"Plaintext"};
    BYTE text2[] = {"pedia"};
    BYTE text3[] = {"Attack at dawn"};
    BYTE text4[] = {"From: Bob"};
    BYTE text5[] = {"From: Eve"};
    BYTE p[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x19, 0x07};

    BYTE chiffre[200];
    BYTE message[200];

    RC4_CTX ctx;

    // Test1
    rc4_init(&ctx, key1, strlen((const char*)key1));
    rc4_crypt(&ctx, text1, strlen((const char*)text1), chiffre, 200);
    print_hexa(chiffre, strlen((const char*)text1));

    rc4_init(&ctx, key1, strlen((const char*)key1));
    rc4_crypt(&ctx, chiffre, strlen((const char*)text1), message, 200);
    print_char(message, strlen((const char*)text1));

    // Test2
    rc4_init(&ctx, key2, strlen((const char*)key2));
    rc4_crypt(&ctx, text2, strlen((const char*)text2), chiffre, 200);
    print_hexa(chiffre, strlen((const char*)text2));

    rc4_init(&ctx, key2, strlen((const char*)key2));
    rc4_crypt(&ctx, chiffre, strlen((const char*)text2), message, 200);
    print_char(message, strlen((const char*)text2));

    // Test3
    rc4_init(&ctx, key3, strlen((const char*)key3));
    rc4_crypt(&ctx, text3, strlen((const char*)text3), chiffre, 200);
    print_hexa(chiffre, strlen((const char*)text3));

    rc4_init(&ctx, key3, strlen((const char*)key3));
    rc4_crypt(&ctx, chiffre, strlen((const char*)text3), message, 200);
    print_char(message, strlen((const char*)text3));

    // Test4
    rc4_init(&ctx, key3, strlen((const char*)key3));
    rc4_crypt(&ctx, text4, strlen((const char*)text4), chiffre, 200);
    print_hexa(chiffre, strlen((const char*)text4));

    // Test5
    rc4_init(&ctx, key3, strlen((const char*)key3));
    rc4_crypt(&ctx, text5, strlen((const char*)text5), chiffre, 200);
    print_hexa(chiffre, strlen((const char*)text5));

    // Question 6
    rc4_init(&ctx, key3, strlen((const char*)key3));
    rc4_crypt(&ctx, text4, strlen((const char*)text4), chiffre, 200);
    for (int i = 0; i < strlen((const char*)text4); i++)
        chiffre[i] = chiffre[i] ^ p[i];
```

```
    rc4_init(&ctx, key3, strlen((const char*)key3));
    rc4_crypt(&ctx, chiffre, strlen((const char*)text4), message, 200);
    print_char(message, strlen((const char*)text4));

    return 0;
}
```

Fichier *RC4.c*

```
#include "RC4.h"

void rc4_init(RC4_CTX * ctx, const BYTE key[], int keylen) {
    BYTE t[256];

    /* Initialisation de s et t */
    for (int i = 0; i < 256; i++) {
        ctx->s[i] = i;
        t[i] = key[i % keylen];
    }

    /* Première permutation de s */
    int j = 0;
    BYTE tmp;

    for (int i = 0; i < 256; i++) {
        j = (j + ctx->s[i] + t[i]) % 256;
        tmp = ctx->s[i];
        ctx->s[i] = ctx->s[j];
        ctx->s[j] = tmp;
    }
}

void rc4_crypt(RC4_CTX * ctx, const BYTE m[], int mlen, BYTE c[], int clen) {
    int i, j, t;
    BYTE tmp;
    i = j = 0;
    for (int l = 0; l < mlen; l++) {
        i = (i + 1) % 256;
        j = (j + ctx->s[i]) % 256;
        tmp = ctx->s[i];
        ctx->s[i] = ctx->s[j];
        ctx->s[j] = tmp;
        t = (ctx->s[i] + ctx->s[j]) % 256;
        if (l < clen)
            c[l] = m[l] ^ ctx->s[t];
    }
}
```

Questions 4)

La fonction $D(k, c) = k \oplus c = m$ c'est à dire que c'est la même fonction que pour le chiffrement !

Il faut simplement ajouter la fonction *print_char* pour afficher les tableaux de BYTE sous forme de chaîne de caractères.

Questions 5)

m1 chiffré = 42 A6 04 68 06 88 39 36 23
m2 chiffré = 42 A6 04 68 06 88 3E 2F 24

Le chiffrement de *m1* est très proche du chiffrement de *m2*. Seule la fin des deux chiffrement sont différents. Le début du chiffrement est identique dans les deux cas car le début du message est identique dans les deux cas et également la clé utilisée est la même.

Questions 6)

Après déchiffrement de *D(k, c')* on obtient la chaine *"From: Eve" = text5*.

p = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x19, 0x07 } = "Bob" ⊕ "Eve"
car *"Bob" = { 0x42, 0x6F, 0x62 } et "Eve" = { 0x45, 0x76, 0x65 }*

$c' = E(k, m) \oplus p$
$= m \oplus k \oplus p$
$= \text{"From: Bob"} \oplus k \oplus \text{"Bob"} \oplus \text{"Eve"}$
$= \text{"From: Eve"} \oplus k$

$D(k, c') = k \oplus c'$
$= k \oplus \text{"From: Eve"} \oplus k$
$= \text{"From: Eve"}$

$D(k, c') = m \oplus p$
$= \text{"From: Bob"} \oplus \text{"Bob"} \oplus \text{"Eve"}$
$= \text{"From: Eve"}$