Heidi: Cyberspace Consultant

Robert R. Enderlein

Medium Task

Thanks to the resounding success of her messages-through-Ethernet-cable business, Heidi has made a name for herself in the cyberspace information-exchange market, and is ready to open her very own independent consulting firm.

Her first client provides Heidi with the encryption algorithm reproduced below. It is a block cipher with a block size of 16 bits that uses two keys of 16 bits each. The client wants her to implement the corresponding decryption function. Heidi already started implementing it in her own way, but thinks that you can come up with something a lot faster.

The encryption function of Heidi's client has the following signature:

```
int encrypt(int plaintext, int key1, int key2);
```

Given a 16-bit plaintext, and the two 16-bits encryption keys, it returns the 16-bit ciphertext. The source code of that function is provided below.

In this task, you must implement the corresponding decryption function with the following signature:

```
int decrypt(int ciphertext, int key1, int key2);
```

It receives as input the 16-bit ciphertext, and the two keys of 16 bits each. The function should return the 16-bit plaintext.

(For Java users: please look into the provided zip file for the source code in your language.)

Tâche Moyenne

Grâce au franc succès de son système de messages-à-travers-câbles-Ethernet, Heidi s'est faite un nom dans le cyberespace sur le marché de l'échange d'information et elle est maintenant prête à ouvrir son propre cabinet en tant que consultante.

Son premier client fournit à Heidi l'algorithme de chiffrage fourni ci-dessous. C'est un chiffrement par blocs avec des blocs de 16 bits qui utilise deux clefs de 16 bits chacune. Le client voudrait que Heidi implémente la fonction de déchiffrement correspondante. Heidi a déjà commencé à l'implémenter à sa façon, mais elle pense que vous pouvez trouver quelque chose de bien plus rapide.

La fonction de chiffrement du client de Heidi a la signature suivante:

```
int encrypt(int texte, int clef1, int clef2);
```

Étant donné le texte de 16 bits, et deux clefs de chiffrement à 16 bits, elle retourne le texte chiffré de 16 bits. Le code source de la fonction est donné ci-dessous.

Dans cette tâche, vous devez implémenter la fonction de déchiffrement correspondante, avec la signature suivante:

```
int decrypt(int texteChiffre, int clef1, int clef2);
```

Elle reçoit comme entrée un texte chiffré de 16 bits et les deux clefs de 16 bits chacune.

(Pour les utilisateurs Java: veuillez regarder dans le fichier zip fourni pour trouver le code source dans votre langage.)

Mittlere Aufgabe

Dank dem durchschlagenden Erfolg ihres Nachrichten-durch-Ethernet-Kabel Geschäfts hat Heidi einen guten Ruf erlangt im Cyberspace *Information-Exchange* Markt. Daher eröffnet sie ihr eigenes Beratungsunternehmen.

Heidi's erster Kunde gibt ihr einen Verschlüsslungsalgorithmus. Der Quellcode dieses Algorithmus ist weiter unten abgedruckt. Es handelt sich um einen Block-Cipher Algorithmus mit einer Blockgrösse von 16-bit. Zur Verschlüsslung werden zwei 16-bit Schlüssel eingesetzt. Heidi soll nun eine Entschlüsselungsfunktion schreiben für den Kunden. Nun hat sie schon mit der Implementierung einer eigenen Lösung angefangen. Sie kommt aber zum Schluss, dass ihr in der Lage seid, eine viel schnellere Lösung zu implementieren.

Die Verschlüsslungsfunktion des Kunden hat die folgende Signatur:

```
int encrypt(int plaintext, int key1, int key2);
```

Die Funktion nimmt ein 16-bit Plaintext und zwei 16-bit Schlüssel und generiert daraus einen 16-bit Ciphertext. Der Quellcode dieser Funktion ist weiter unten abgedruckt.

Die Aufgabe besteht nun daraus die entsprechende Entschlüsslungsfunktion zu schreiben. Diese Funktion hat die folgende Signatur:

```
int decrypt(int ciphertext, int key1, int key2);
```

Die Funktion nimmt den 16-bit Ciphertext und die zwei 16-bit Schlüssel. Daraus sollte die Funktion den originalen Plaintext berechnen.

(Für Java-User: Im zur Verfügung gestellten ZIP-File befindet sich der Sourcecode in Java.)

Source code / Code source / Quellcode

```
static const int MAX_BYTE = 1<<8;
static const int MAX_WORD = 1<<16;
static const int SBOX1_PRIME = 65537;
static const int SBOX2_PRIME = 257;
static const int NUMBER_OF_SBOXES = 3;
static const int KEY_ROUNDS = 16;

static void assertByte(int input) {
  if(input < 0 || input >= MAX_BYTE) { exit(1); }
}

static void assertWord(int input) {
  if(input < 0 || input >= MAX_WORD) { exit(1); }
}

static int sBox0(int input, int key) {
  assertByte(input); assertByte(key);
  return input ^ key;
}
static int sBox1(int input, int key) {
  assertByte(input); assertByte(key);
  long long int e = input + 256*key + 1;
  e = ((e * e) % SBOX1_PRIME) - 1;
  return (e % MAX_BYTE) ^ ((e / MAX_BYTE) % MAX_BYTE);
}
static int sBox2(int input, int key) {
  assertByte(input); assertByte(key);
  return (((input + 1) * (key + 1)) % SBOX2_PRIME) % MAX_BYTE;
}
int sBox(int input, int key, int sBox) {
  assertByte(input); assertByte(key);
  switch(sBox) {
    case 0: return sBox0(input, key);
    case 1: return sBox1(input, key);
    case 2: return sBox2(input, key);
    default: exit(1);
  }
}

int keySchedule(int key, int schedule) {
  assertWord(key);
  long long int extendedKey;
  extendedKey = key | (key % MAX_BYTE)<<16LL;
  return (extendedKey>>schedule) % MAX_BYTE;
}

int feistel(int plaintext, int key) {
  assertWord(plaintext); assertWord(key);
  int i;
  for(i=0; i < NUMBER_OF_SBOXES * KEY_ROUNDS; ++i) {
    int keyI, boxI, roundKey, left, right, newLeft, newRight;
    keyI = i / NUMBER_OF_SBOXES;
    boxI = i % NUMBER_OF_SBOXES;
    roundKey = keySchedule(key, keyI);
    left = plaintext % MAX_BYTE;
    right = (plaintext / MAX_BYTE) % MAX_BYTE;
    newLeft = right;
    newRight = left ^ sBox(right, roundKey, boxI);
    plaintext = newLeft + newRight * MAX_BYTE;
  }
  return plaintext;
}

int encrypt(int plaintext, int key1, int key2) {
  assertWord(plaintext); assertWord(key1); assertWord(key2);
  return feistel(feistel(plaintext, key1), key2);
}
```