Lab1: Decrypt Stream Codes
ID:13354222

by Shuang Liu
Sun Yat-sen University

March 12, 2016

1 The deciphered Result

The output of the program is indicated in the Fig.1. Scrutinizing the results, we can see that the intermediate processes of the decryption are demonstrated by corresponding lines, such as ”Initialization done”, ”Decryption done” and so on.The result showed in the last line of the output in Fig.1 is the result decrypted by the program using the downside of using the same key multiple times.Moreover, owing to the reason that the program does not decrypt the key at all, instead, it uses the result of exclusive or operation results among any two given cipher texts to decrypt the target cipher text, all the decrypted given cipher texts are not included in the output. Besides, as

```
Intializing the given ciphertexts and the target texts...
Initialization done.
Now preparing to decrypt...
Decryption done.
Now preparing to decipher the tagretCT...
Deciphering the targetCT done.
The target plain text is as followed:

Comfort is the*ene*y of a**ievemen**
```

Figure 1: Experimental Result without modification

illustrated in Fig.2, individuals are allowed to modify the result by inputting the position they are inclined to change and the value they intend to replace with, predicting the value or texts which are not deciphered by human brains. Obviously, we can alter the result multiple times until we are satisfied with the final result.Eventually, what the final result in Fig.2 demonstrated that through the cooperation of the program of decryption

and the human brain, we are able to obtain nearly a complete deciphered plain text, which substantiated that the usage of a same key in multiple encryptions is rather hazardous.Therefore, at the time individuals use the stream cipher to encrypt plain texts more than one, the key must be altered after it is used to encrypt a plain text every time.

```
The target plain text is as followed:
Comfort is the*enemy of achievemen**
Do you want to modify the target plain text manually?(Y/N)
Y
Please input the position you want to change and the value you want to replace with(start from 0)
34 t

The target plain text is as followed:
Comfort is the*enemy of achievement*
Do you want to modify the target plain text manually?(Y/N)
N
The process is done.

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Figure 2: Experimental Result with modification

2 The method of decryption

This section firstly describes about the principle behind the program designed and the method adopted to decrypt the target plain text using the properties of exclusive or operation between two values, without the necessity to decrypt the actual key used in the encryption process.And then, it describes the implementation of the method using C++ language with a few appropriate comments.

2.1 Principle

There are two fundamental properties the program uses in order to decrypt the target plain text.

Exclusive Operation: As showed in the formula below, if A and B do XOR operation the same value, and the corresponding two results do XOR operation with each other, the final value is the same as just A and B do XOR operation with each other.

$$(AxorC)xor(BxorC) = AxorB$$

Amazing Space: It is known to individuals that the binary value of ASCII of a space is 00100000, and the difference between the binary values of a uppercase letter and the corresponding lowercase letter lies on the third most significant bit, which, to be more specific, is that the third most significant bit in the binary form of

all lowercase letters is always 1, and the corresponding significant bit in the binary form of all uppercase letters is always 0.Besides, according to the property of XOR operation showed below, we know that any value does XOR operation with 0 gets itself, and any value does XOR operation with 1 gets its inverse value in binary form.

$$1xor0 = 1$$

$$1xor1 = 0$$

$$0xor1 = 1$$

$$0xor0 = 0$$

Surprisingly, what is obvious is that the only non-zero value of the space in the binary form is the third most significant bit, which indicates that when a uppercase letter does XOR operation with a space, a corresponding lowercase letter will be generated and vice versa, for the third most significant value of the letter will turn into its inverse value in binary form.The formula showed below illustrated this, replacing the space with a notation '_'

$$axor\_ = A$$

$$Axor\_ = a$$

2.2 Method

From the two principles listed above, we can be aware that the two principles are vital elements of the decryption method.
Generally, the program does XOR operation of one of the given cipher text, say A, with all other ones, obtaining all the XOR operation results.By the amazing space property, we know that when a space does a XOR operation with a letter, the consequent result is also a letter.Hence, what we need to do is to count the number of letters in the same position of all the XOR operation results.If the number is larger than a threshold set up before, say 6, it means that the corresponding position in A is a space.
Now, we can obtain a result of the XOR operation between the give cipher text A, and the target cipher text C.By the property that a uppercase letter doing XOR operation with a space will obtain a lowercase one and vice versa, we can predict the actual letter in the plain text if the value in the position of the XOR operation result is a letter, the value in which of the given cipher text A is a space.For example, if the third value, or the third character, in the XOR operation result is H, and the third value in the given cipher text A, is a space, then the third value or character in the target plain text is h, obviously.This process is illustrated below.

$$\textit{target CT } xor \textit{ given CT} = \textit{target PT } xor \textit{ given PT}$$

if the target PT is y, the given PT is a space_ , then

$$\textit{target PT } xor \textit{ given PT } = Y$$

2.3 Implementation

The whole process can be divided into a few steps.
Step1:Initialization
As the program needs no outer files or inputs, the given CTs and the target CT are initialized in the function initInput().Moreover, in the function initInput(), we add and initialize a variable named targetPT using notation '*' to indicate all the characters in targetPT is unknown.If a predication of some character in the targetPT is achieved, the '*' will be replaced with the predicated character.The process is illustrated in Fig.3 partially.

```
CT[9] =
"466d06ece998b7a2fb1d464fed2ced7641ddaa3cc31c9941cf110abbf409ed39598005b339
f537530541ab0f9f3cd04ff50d66f1d559ba520e89a2cb2a83";
targetCT =
"255603efa6a9acecf107155ea506b57540caa3018a108b08c11c03a0e85fa0660cde47b4";
targetPT =
"************************************";
cout << "Initialization done." << endl;
```

Figure 3: The partial content in function initInput()

Step2:XOR Operation
Owing to the reason there is only XOR operation of values in decimal form in C++, we need to transform the string value in hexadecimal form before using the XOR operation, by using the function of *strtol()* in C++. After obtaining the result of XOR operation, we need to assign the value to a string in hexadecimal form, by using ostringstream library in C++.The process is illustrated in Fig.4.

```
for(int i = 0;i < cnt;i=i+2){
    a_lett[0] = a[i];a_lett[1] = a[i + 1];
    b_lett[0] = b[i];b_lett[1] = b[i + 1];
    //Turn string into a decimal number
    a_value = strtol(a_lett, NULL, 16);
    b_value = strtol(b_lett, NULL, 16);
    xor_value = a_value ^ b_value;
    //Consider the case of 0x0_
    if(xor_value < 16){
        oss << 0 << std::hex << xor_value;
    }
    else{
        oss << std::hex << xor_value;
    }
}
result = oss.str();
```

Figure 4: The content in function xoOper()

Step3:Get the spaces in given PTs using given CTs

Through the method described above, we need to use a loop traverse all the given CTs to get the XOR operation result between one of the given CTs and all other ones.After that, we need to count the number of letters in all the positions of the results.In the case where the number is larger than 6, we predict that the corresponding position in the given CT is a space and use it to obtain the character in the target PT directly only if the character in the corresponding position of the target PT is a letter.The process is illustrated in Fig.5.

```
int letCnt = 0;
for(int ind = 0;ind < 9;ind++){
    char let[2];
    let[0] = xoResult[ind][k];
    let[1] = xoResult[ind][k+1];
    long lttr = strtol(let, NULL, 16);
    //judge if the value is a letter
    if((lttr <= 122 && lttr >= 97) || (lttr <= 90 && lttr >= 65)){
        letCnt++;
    }
}
if(letCnt > 6){//k-th value of CT[i] is a space
    char letter[2];
    letter[0] = targetXOTempResult[k];
    letter[1] = targetXOTempResult[k + 1];
    int lttr = strtol(letter, NULL, 16);
    //interchange bewtween the uppercase number and the lowercase number
    if((lttr <= 122 && lttr >= 97) || (lttr <= 90 && lttr >= 65)){
        targetPT[k/2] = (const char)lttr ^ 32;
    }
}
```

Figure 5: The partial content in function decrypt()

Step4:Get the spaces in target PT using target CT
The process of finding the space in the target PT is considerably similar to Step3.The difference lies on that in Step4 we need consider the circumstance of conflict.Specifically, if one position in the target PT has already been predicted and the new predicated value is not equal to the old one, then there exists a confliction, represented by a notation '?', indicating that there is something wrong about the prediction.The process is illustrated in Fig.6.
Step5:Modify of result by users
After the result predicted by the program is showed in the screen, users are allowed to modify the result multiple times as they like when they think the result is not precise enough or if they are able to predict the yet unknown characters by their brains.The process illustrated in Fig.7.

```
int letCnt = 0;
for(int ind = 0;ind < 10;ind++){
    char let[2];
    let[0] = targetXOResult[ind][i];
    let[1] = targetXOResult[ind][i+1];
    long lttr = strtol(let, NULL, 16);
    //judge if the value is a letter
    if((lttr <= 122 && lttr >= 97) || (lttr <= 90 && lttr >= 65)){
        letCnt++;
    }
}
if(letCnt > 6){//i-th value of targetPT is a space
    if(targetPT[i/2] == '*'){
        targetPT[i/2] = ' ';
    }
    else{
        //There is some prediction wrong
        targetPT[i/2] = '?';
    }
}
```

Figure 6: The partial content in function decrypt()

```
while(1){
    int pos;
    char charValue;
    cin >> pos;
    cin >> charValue;
    if(pos >= targetPT.length() || pos < 0){
        cout << "The position is invalid, please input again." << endl;
        continue;
    }
    else{
        targetPT[pos] = charValue;
        break;
    }
}
```

Figure 7: The partial content in function printModificationHint()