Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2021

Practice Homework on Bitwise Operators – Simple Data Encryption

Not to be submitted and not to be graded

Introduction

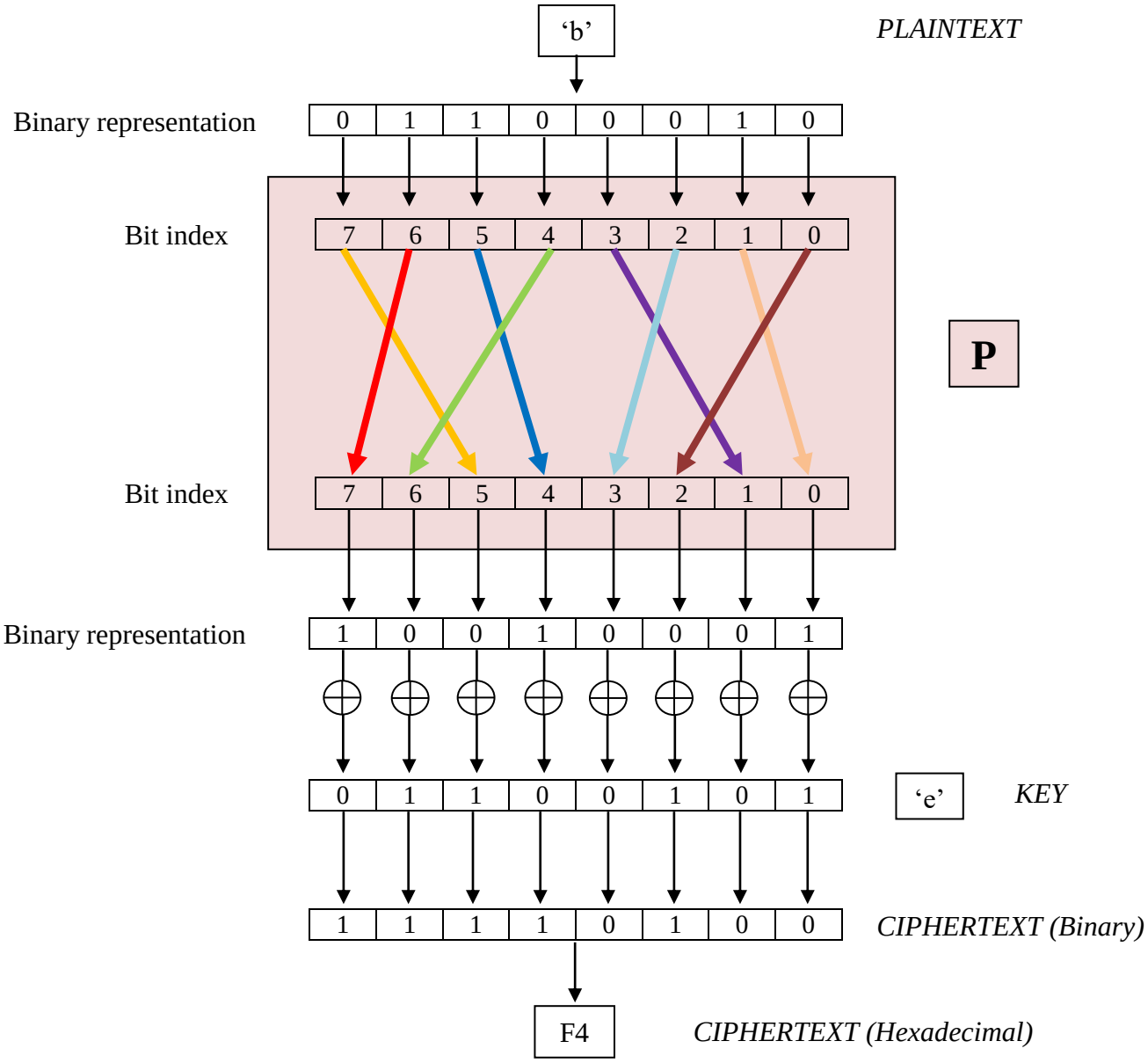
In this homework, you are asked to write a program that encrypts given data. Your program will read the input from the keyboard and apply encryption operation which will be explained in detail later. In order to further secure the encryption operation, we will use a *key* in the operation. The encrypted data will be displayed on the screen in hexadecimal format.

Encryption Operation

It is trivial that when two or more people communicate through a channel, the underlying framework must protect the data that is being sent among communicating people. This process can be done by the usage of cryptographic algorithms. Cryptographic algorithms consist of two parts: *encryption* and *decryption*. Encryption is the process of transforming a clear message (called *plaintext*) into unintelligible form (called *ciphertext*). Decryption is the reverse operation that converts the ciphertext into the original plaintext. In your program, you will implement a simple *encryption* algorithm as explained in this section (decryption is not part of the homework; but if you want, you can implement it for crosscheck purposes).

To encrypt a given message, your program processes the message using its character (ASCII) codes. For each such character (represented as `unsigned char`), your program should permute the bits according to the transformation rule, and make a XOR operation with the corresponding character of the key, which is an input entered by the user.

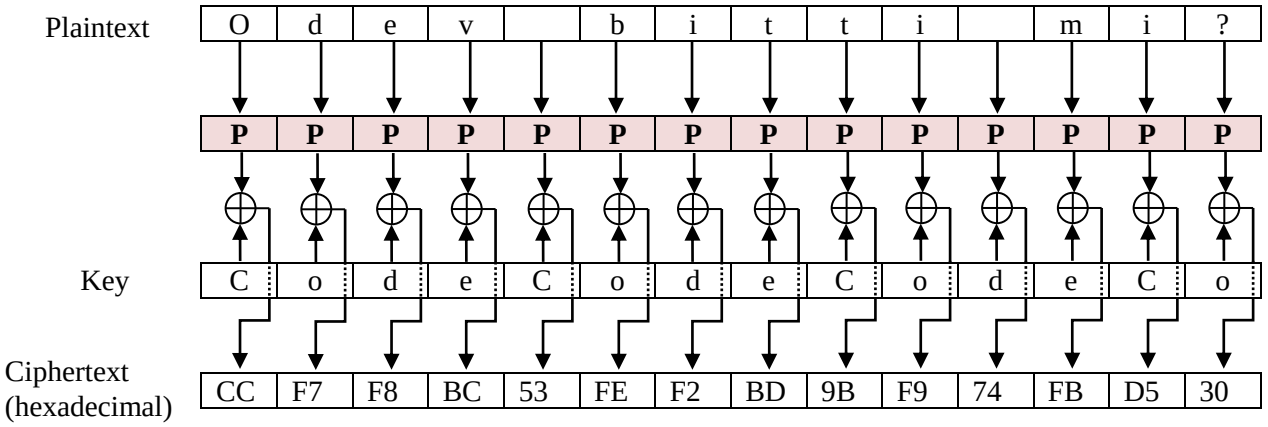
For illustration purposes, we will encrypt character '`b`'. Its ASCII code is 98 (decimal). The corresponding character of the key entered from the keyboard is, say '`e`' and its ASCII code is 101 (decimal). First of all, we permute the character to be encrypted (here 'b') according to the permutation function (denoted as **P**) below. After that, we make a bitwise XOR operation between the key (here 'e') and the permuted plaintext. Then, the result of the bitwise XOR operation generates the ciphertext character for 'b' as follows.



The illustration given above shows the encryption of character 'b' as an example. The permutation function **P** and the XOR operation must be applied to all of the characters of the input plaintext.

The encryption of the data, which contains several characters, is performed character by character. First, you permute the bits of the first character of the plaintext. Then, you make bitwise XOR operation between the permuted character of the plaintext and the first character of the key (given by the user) and end up with the corresponding ciphertext character. Then you continue with the next character of the plaintext and the next character of the key, and do the same operations. This process lasts until you run out of all the characters in the plaintext data. When you use all of the characters of the key, you will return to the beginning of the key

string and continue to perform encryption operation. In other words, the characters of the key will be used repeatedly. The following illustration demonstrates how the encryption operations are performed for a plaintext data that contains several characters. Assume that the key is: **`Code`** and the plaintext is: **`Odev bitti mi?`**



As you can see, encryption operation requires manipulating bits. Thus, you can use bitwise operators for your implementation.

Program Flow and Some Hints

First, your program asks for the encryption key. The user enters the encryption key from the keyboard and presses the Enter button. After that, your program asks for the data to be encrypted (plaintext). The user enters the data to be encrypted from the keyboard and presses the Enter button. Then, your program makes the encryption operation as outlined above and display the result (ciphertext) in hexadecimal format. Afterwards, the program asks again for the plaintext and performs the encryption operation in the same way. This flow continues similarly, until the user presses Ctrl-Z.

As explained above, the key will be entered only <u>once</u> at the beginning, but several plaintext strings will be entered until the program terminates. Since the plaintext is the entire line with some blank characters in between, you have to use `getline(cin,`*plaintext*`)` where *plaintext* is a string variable. In order to stop when Ctrl-Z is entered, you can have the getline command as the loop condition. We also recommend you to read the key at the beginning using `getline(cin,`*key*`)`; if you read key using `cin>>`*key*, the remaining end of line character after the key causes problems while reading plaintexts.

Since you have to display the ciphertext in hexadecimal format, you can use the `hex` property of `cout` after typecasting the `unsigned char` to `int`. Assume the name of your `unsigned char` variable is `c`. Then, you can use the `hex` property in the following way.

```
cout << hex << (int) c;
```

You are not allowed to use `bitset` class of C++. Actually, we have not covered it in the course; however, if you somehow came across with it, do not use it in this homework.

Sample Runs

Some sample runs are given below, but these are not comprehensive. Therefore, you have to consider all cases, to get full mark. The inputs from the keyboard are written in ***boldface/italic***.

Sample Run 1

```
*** Welcome to the Simple Data Encryption Application ***
Please enter the encryption key: Code
Please enter the plaintext to be encrypted: Odev bitti mi?
Ciphertext: ccf7f8bc53fef2bd9bf974fbd530

Please enter the plaintext to be encrypted: CS204 is fun
Ciphertext: c6aa35351b7ff2b053f6b8fe

Please enter the plaintext to be encrypted: If you can't make it good, at least make it look good
Ciphertext:
c5f674b3dcb374f0d7f479bd53f1f0f2df7ff2bd53f2fbfadb7574f19b7ffef9d7ba
bc75ddfbf3f953f9bc75d9f0fbf253f2fbfadb

Please enter the plaintext to be encrypted: ^Z
```

Sample Run 2

```
*** Welcome to the Simple Data Encryption Application ***
Please enter the encryption key: 25April1915
Please enter the plaintext to be encrypted: Gallipoli
Ciphertext: bfa1dbeae4b9f3abaf

Please enter the plaintext to be encrypted: 25April1915
Ciphertext: 6369c5a0a3fff6656f6569

Please enter the plaintext to be encrypted: d
Ciphertext: aa

Please enter the plaintext to be encrypted: ^Z
```

Sample Run3

```
*** Welcome to the Simple Data Encryption Application ***
Please enter the encryption key: cs204
Please enter the plaintext to be encrypted: I wrote this code in 57 lines!
Ciphertext: e563efe1abbbef22e8a6f5a622a5abfbef22a6af732f6f20aef5e8aee520

Please enter the plaintext to be encrypted: You must be kidding!!
Ciphertext: a5ecee20aabfa6ea20a5ff63a5a6acfbe5a9ad2077

Please enter the plaintext to be encrypted: No really
Ciphertext: e8ec22e1a8f7e9a8e6

Please enter the plaintext to be encrypted: asdfggsdf
Ciphertext: f7a6aaa9a9fea6aaa9

Please enter the plaintext to be encrypted: ^Z
```

Good Luck!