Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming, Fall 2016-2017

Homework 6 – Bitwise operations

Due: 29/11/2016, 23:30

(Late submission penalty: -20%)

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.
You CANNOT HELP any friend while coding.
Plagiarism will not be tolerated!

Introduction: Cryptography heavily depends on secure data sharing (to-some-degree) and it is an important part of the computer science. In this homework, you are going to perform two basic cryptographic functions on messages; encryption and decryption. Most of the algorithms in Cryptography perform **encryption** on a message (making it a secure and unreadable ciphertext) and **decryption** on a ciphertext(converting the message back to human readable form). These algorithms usually employ a secret **key** string for these functions.

x	*y*	*x* XOR *y*
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

For this homework, the encryption operation works as follows: you are going to *take a message as an array of integer elements*, and (bitwise) **XOR** each element with the corresponding element of a key (you can find the **XOR** truth table above for your convenience); you will replicate the key whenever the message size is larger than the key in bytes.

Program flow

1. First, the program will request the number of integers in the message array and their values from the user. Each element of a message array is a 4-byte unsigned integer value. You do not need to check if the elements are valid unsigned integers. You can assume that the input will be correct.
2. The key's type and value will be requested later. The type of the key can be unsigned **int/short/char**. When the key type is not one of them the program reports that and exit. The size of the key type changes the encryption and decryption slightly.
3. If key is a **char**(which is 1 byte), then you have to replicate the key 4 times and generate an encrypted **int** by separately XORing each byte of the element with the key. In particular, the key's

type will be the template type parameter. That is instead of multiple functions with different parameter types, you need to implement templated encryption and decryption functions (in fact a single function, would be enough). You should write the ciphertext in (unsigned) integer format to the output screen.

4. In order to decrypt the ciphertext, you repeat the same operation with a slight difference: this time, the ciphertext will be the input and the message (it is also called **the plaintext**) will be the output. You need to write the plaintext to the output screen.

In C/C++, the size of a variable can be obtained by the **sizeof** function; for instance, **sizeof**(**char**) is 1 since char is a 1-byte variable and **sizeof**(**int**) returns 4 since **int** is a 4-byte variable in VS by definition For more information on **sizeof**, please visit http://en.cppreference.com/w/cpp/language/sizeof.

For demonstration purposes, you can find the encryption of first element of the 12with (unsigned) char key = 'e', short key = 2000, and int key = 40000in Figure 1, Figure 2, and Figure 3 respectively.

In the Figure 1 the key type is char that is represented in 1 byte and replicated 4 times because the message 12is 4 bytes. Thus we have to XOR each byte of the message with the key.

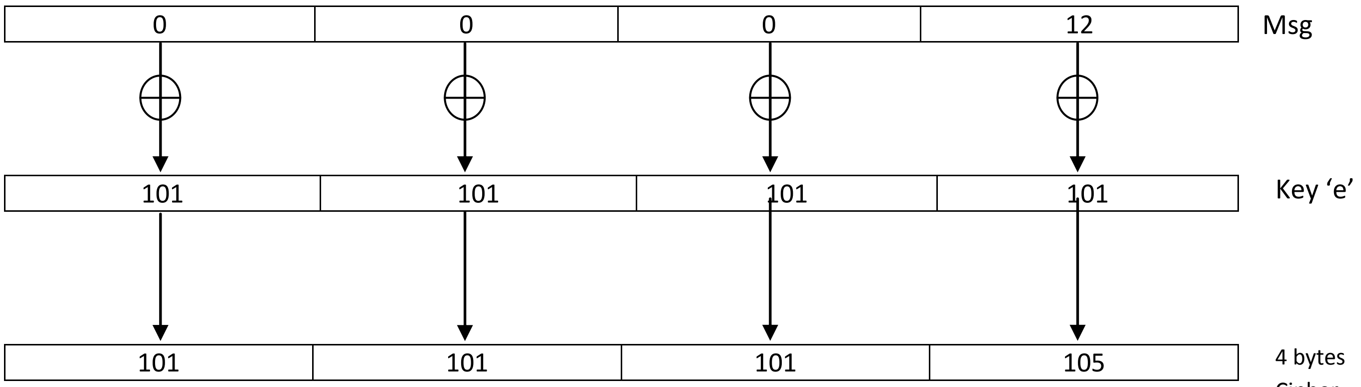

Figure 1

The 4 bytes cipher will be represent in output as an unsigned integer which is 1701143913.

In the Figure 2 the key type is unsigned short which is represented in 2 bytes and will be replicated 2 times since the message is integer type. Thus we have to COR each two bytes of the message with the key.

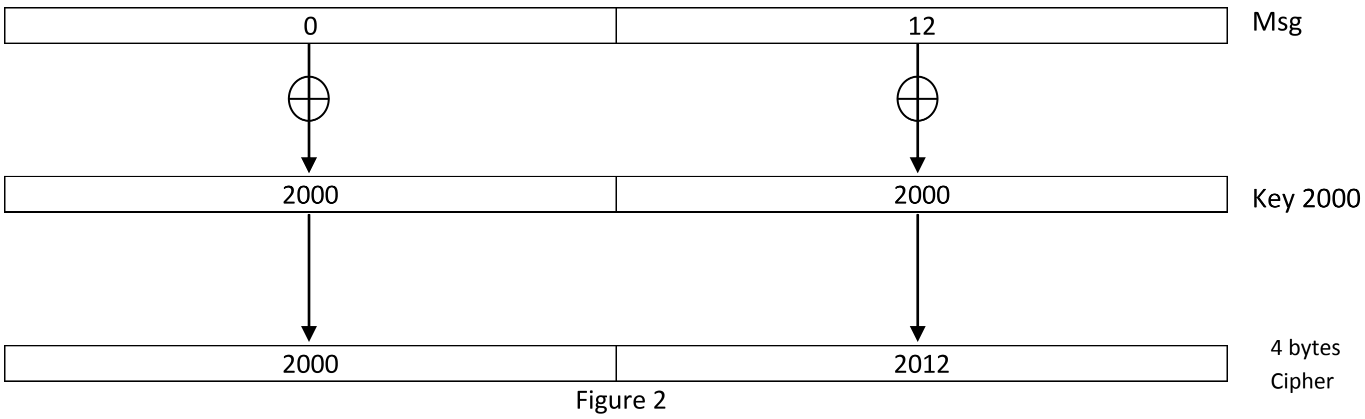

Figure 2

The 4 byte cipher will be represent in output as an integer which is 131074012.

In the Figure 3 the key type is integer type which is represented in 1 byte and will be xored once with the message because the message and key are both 4 bytes.

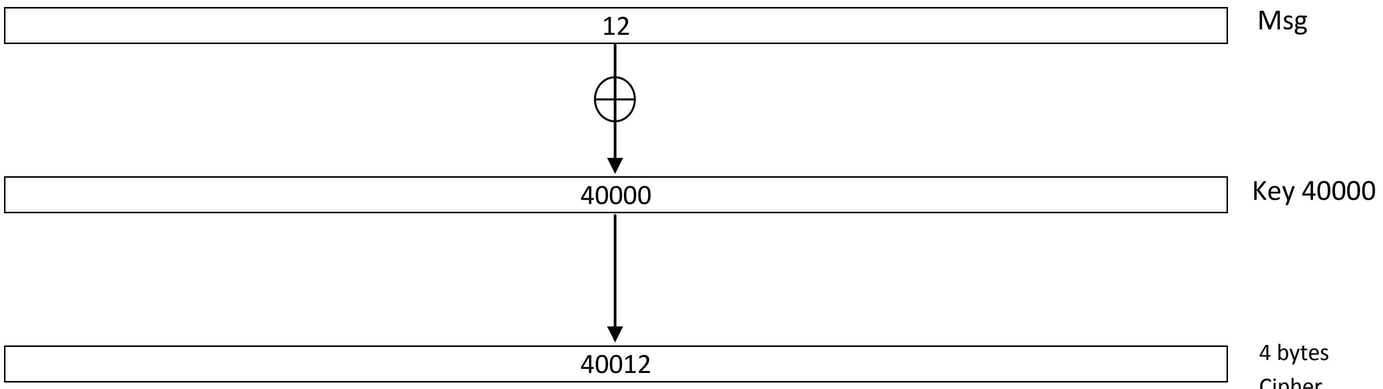

Figure 3

4 bytes cipher will be represent in output as an integer which is 40012.

The decryption process is exactly the same only the place of cipher and message are changed.

Sample runs

Below, we provide some sample runs of the program that you will develop. The bold phrases are inputs taken from the user. You should follow the input order in these examples.

```
Just for FYI --------------------------
size of unsigned char is 1
size of unsigned short is 2
size of unsigned int is 4
---------------------------------------


Size of the message array is 6
Enter the message array integers one after another by pressing enter after each number:
12
13
45
46
23
76
Type of key is unsigned char
The value of the key is e
Ciphertext is: 1701143913 1701143912 1701143880 1701143883 1701143922 1701143849
Plaintext is: 12 13 45 46 23 76
Press any key to continue . . .
```

```
Just for FYI --------------------------
size of unsigned char is 1
size of unsigned short is 2
size of unsigned int is 4
---------------------------------------


Size of the message array is 6
Enter the message array integers one after another by pressing enter after each nu
12
13
45
46
23
76
Type of key is unsigned short
Enter the value of the key
2000
Ciphertext is: 131074012 131074013 131074045 131074046 131073991 131073948
Plaintext is: 12 13 45 46 23 76
Press any key to continue . . .
```

```
Just for FYI --------------------------
size of unsigned char is 1
size of unsigned short is 2
size of unsigned int is 4
---------------------------------------


Size of the message array is 6
Enter the message array integers one after another by pressing enter after each numb
12
13
45
46
23
76
Type of key is unsigned int
Enter the value of the key
40000
Ciphertext is: 40012 40013 40045 40046 40023 39948
Plaintext is: 12 13 45 46 23 76
Press any key to continue . . .
```

Some Important Rules:

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**.

What and where to submit (PLEASE READ, IMPORTANT): You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.
Name your cpp file that contains your program as follows:

"SUCourseUserName_YourLastname_YourName_HWnumber.cpp"

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw2.cpp

Do not add any other character or phrase to the file name. Make sure that this file is the latest version of your homework program. Compress this cpp file using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct file. The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are**NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
CS204 Team (Leyli Javid Khayati, Kamer Kaya)