Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2016

Homework 8 – Encryption/Decryption Tool with GUI

Due: 13/05/2016, Friday, 21:00

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 can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

Introduction

In this homework, you are asked to implement a simple encryption tool with Graphical User Interface (GUI). We will explain the details of the GUI design and how the encryption/decryption operations work in the following sections.

Encryption/Decryption Operation

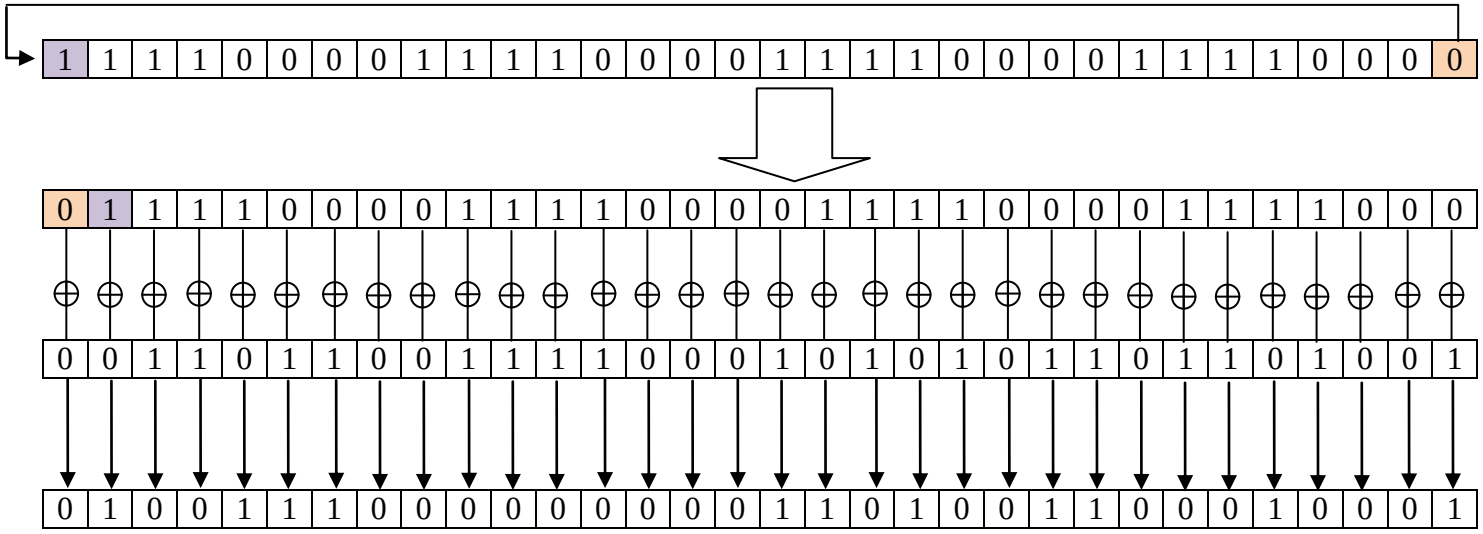
It is trivial that when two or more people communicate through a channel, the underlying framework must protect the data that is being sent among communicating people. This process can be done by the usage of cryptographic algorithms. Cryptographic algorithms consist of two parts: *encryption* and *decryption*. 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. This algorithm will be used for *decryption* as well.

To encrypt a message, your program should processes the input *plaintext* with the input *key*. Your program, first, will optionally apply bitwise rotation (circular rotation) to the *key* based on an option selected from the GUI. The options for key rotation are either (i) one-bit toward right, or (ii) one-bit toward left, or (iii) no rotation. Then, your program will apply bitwise XOR operation between the plaintext and the (rotated) key in order to generate the *ciphertext*.

For decryption, same operation as the encryption should be followed; but this time, your program should XOR input *ciphertext* and the optionally rotated key. Since XOR is a reversible operation (i.e. XOR of XOR yields the same data), this operation would yield original plaintext. A schematic example of encryption is given in the next page.

Your program will work with unsigned integer (`unsigned int`) values. Remember that unsigned integers are 32-bit values.

The following figure gives an example for encryption operation by rotating the key one bit toward right. In this example, the key entered is **4042322160** and the plaintext is **913877865**.



KEY → 4042322160 in decimal

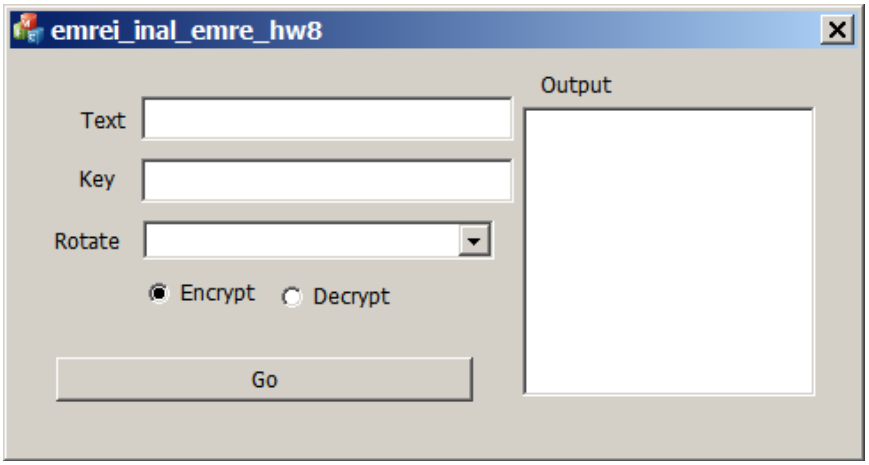
KEY becomes 2021161080 after Rotating one bit right

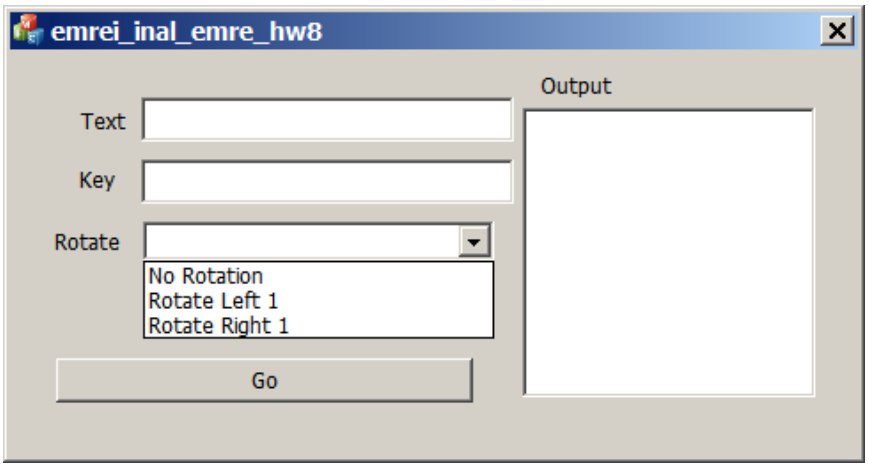
PLAINTEXT → 913877865 in decimal

CIPHERTEXT becomes 1308676881 after encryption

GUI Design

GUI of the Encryption/Decryption tool must be designed as shown in the following figures.





The caption of the GUI should be given according to naming convention of the projects "SUCourseUserName_YourLastname_YourName_HWnumber". In the GUI snapshot above, the caption of the GUI is given according to this convention: emrei_inal_emre_hw8. Of course, you will use your own information.

The encryption tool should contain 2 *EditControl* objects labeled **"Text"** and **"Key"**. These labels are *StaticText* objects as shown in the figure above. For encryption, **"Text"** will hold your plaintext value; and for decryption, **"Text"** will hold your ciphertext value as unsigned integers. In your program, you must implement the necessary input checks for both **"Text"** and **"Key"** such that: (i) they are not empty, and (ii) they do not contain any characters other than digits.

A *ComboBox* object whose caption is **"Rotate"** should be in your GUI, under the abovementioned *EditControl* objects. This *ComboBox* object should contain 3 different rotation types: **"Rotate Right 1", "Rotate Left 1"** and **"No Rotation".** The caption of this *ComboBox* object needs to be a *Static Text* object. In your program, you have to make sure that user has selected at least one of these types in order to continue with encryption or decryption.

2 *RadioButton* objects whose caption is **"Encrypt"** and **"Decrypt"** should be in your GUI, following *ComboBox* object. Only one of the *RadioButton* objects shall be available for choosing at a given time. You have to make sure that one of these *RadioButton* objects is selected. *(Hint: You can set an integer value for radio buttons in order to get which one of them is checked. Therefore you can get a "-1" if none of them is selected. Actually, setting an integer value instead of Boolean ensures that one of the RadioButtons grouped together is always selected. You can refer to GUI_Document.pdf given together with Lab13 material on the web site in order to see how you can set an integer value to RadioButtons).*

The GUI should contain a *ListBox* object labeled **"Output"**. Output for encryption and decryption will be added to this *ListBox* using the following formats:

Encryption: *PLAINTEXT* encrypted with *KEY* using *ROTATETYPE*: *CIPHERTEXT*
Decryption: *CIPHERTEXT* decrypted with *KEY* using *ROTATETYPE*: *PLAINTEXT*

You should replace *PLAINTEXT*, *CIPHERTEXT*, *KEY* and *ROTATETYPE* with the values that the user has selected for the particular operation. The message type to be displayed in the *ListBox* will be selected by the *RadioButton* objects mentioned above ("Encrypt" or "Decrypt"). Please note that this *ListBox* is created with a low width on purpose. Your program should handle both horizontal and vertical overflow in the *ListBox* object carefully; showing scrollbars when needed. *(Hint: see the example that we developed in the lab to enable horizontal and vertical scroll bars. Please recall that you should also write some code to enable horizontal scroll bar).*

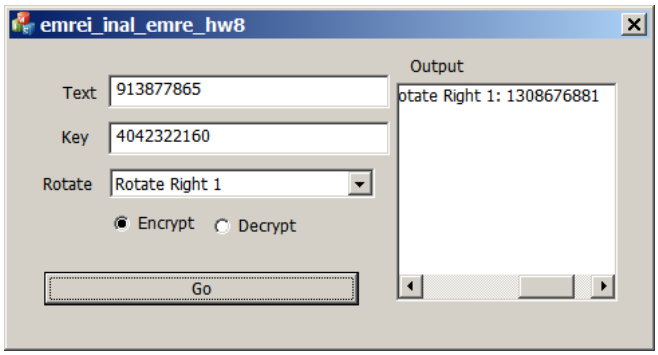
Important! If any of the abovementioned input checks for *EditControl*, *ComboBox* and *RadioButton* fail, you should add a message which has the text *"Wrong Input. Try Again."* into your Output *ListBox*.

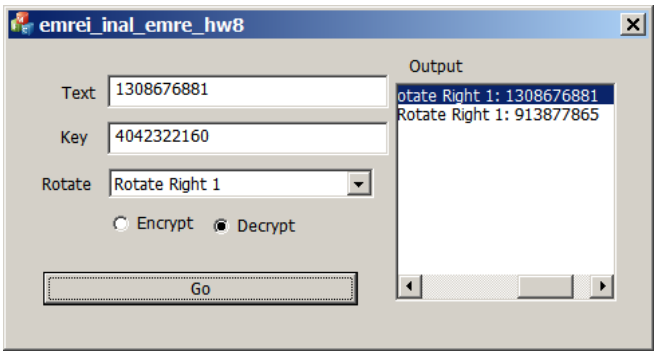
Finally, the GUI of your program should contain a *Button* named **"Go"**. When this *Button* is pressed, your program will take all the values the user has entered on the GUI, will compute the necessary encryption or decryption operation and generate a message for the *ListBox*. Of course, you will be able to press on **"Go"** button several times, creating different output lines based on the current values entered on the GUI.

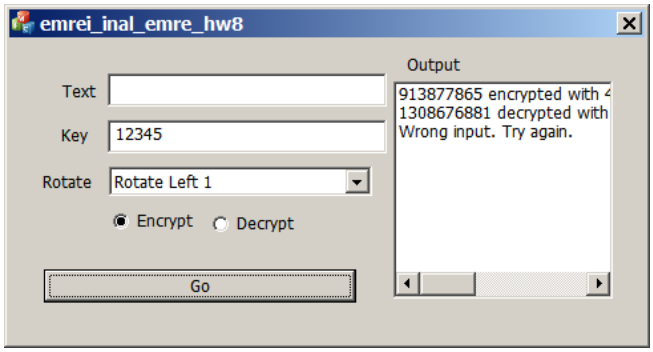
Remark: since you use bitwise operations on the data, you should use `unsigned int` for both key and plaintext/ciphertext. Since the data that you read via *EditControl* objects is of type `CString`, you must convert `CString` to `unsigned int` before encryption/decryption operations. Similarly, since you can only display `CString` in *ListBox* object, you must convert `unsigned int` values to `CString` as well. The ways of doing these conversions are explained in **GUI_document.pdf** file given together with Lab13 material on the web site.

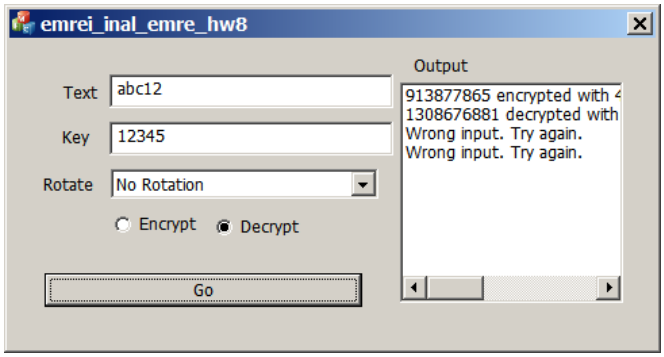
Sample Runs

Some sample snapshots are given below. However, the homework is better understood if you run the implemented version. In this homework zip package, we provide you an .exe file of the Encryption/Decryption Tool that we developed. You can play with this sample program and see how the program operates.









ATTENTION

Please see previous homeworks for the general rules that you have to follow.

Submission rules are similar to the previous homeworks, but there are some special considerations that you have to be very careful since this is a GUI (MFC) application.

- Use the same naming convention as in the previous homework assignments.
- Since this is not a console application, adding an existing cpp to a new project does not work. You have to develop your GUI application from scratch using the methods explained in the labs. Please check out the Lab13 link at the website for details.
- There are some other very important project files in addition to the cpp and header files. Thus please pay extreme attention that all project files exist in your zip file.
- Since the entire project would be very large (a couple of 100s of Mbytes), you will delete some files and folder while submitting. These are:
    - the large file with *.sdf* extension
    - the folder named *ipch*
    - all *debug* and *release* folders
- These files and folders, which are deleted while submitting, regenerate themselves when you open the solution again. Thus, there is no harm not to submit those. However, please make sure that without those files, your solution opens up and your program compiles and runs correctly.
- As in the other homework assignments, correct submission is part of your duty; if we cannot run your program, we cannot grade it.

Good Luck!
Emre İnal, Albert Levi