Computer Architecture and Assembly Programming

CSE 3120 2013FL   Final Project   November 28, 2013

```
   __________  ______  ____________  ______    _  __
  / ____/ __ \ \/ / __ \/_  __/ __ \/  |/  /   | |/ /
 / /   / /_/ /\  / /_/ / / / / / / / /|_/ / /| | |   /
/ /___/ _, _/ / / ____/ / / / /_/ / /  / / ___ |/   |
\____/_/ |_| /_/_/     /_/  \____/_/  /_/_/  |_/_/|_|
```

Encryption made in ASM

By

Iordanis Fostiropoulos

Instructor: Dr. Liam Mayron

Introduction

CryptoMax, is an attempt to create maximum security for file that contain sensitive information. The name CryptoMax is the concatenation of the word **Crypto**graphy and **Max**imum, hence, CryptoMax. This encryption/decryption program is meant to be extremely safe and capable of working in real life situations. However the program at this phase has many flaws, which, within the three weeks period of the project, were only possible to be identified but not fixed completely. The flaws do not relate with the functionally in any way but with the limits of the program and the security it can provide.

The motivation for such a program came as an interest towards cryptographic algorithms and how they work on the low level end. It was very insightful to work and have the ability to realize that everything is eventually consisted of bytes. Through this project it was possible to gain better understanding of what a file really means and how it can be manipulated. In addition to that, the project's complexity required extreme debugging which helped into a better understanding of the memory management and provoked discussion with other ASM programmers.

Implementation

The program consists of 2 main files and one resource file. The two main files are Lib.asm and main.cpp. Lib.asm contains assembly procedures that help with the implementation of the task, which is to encrypt a file. The main.cpp contains the user interface and some higher level interaction with the user such as windows dialogues. The resource file, HeaderArt.art, contains ASCII art that is meant to enhance the aesthetics of the program.

In order to assemble and compile the program, a Visual Studio project should be created with the settings provided on the course website. The settings should allow the compilation through Irvine library to allow C++/ASM interaction as demonstrated in class. The program should be loaded in visual studio and executed through the debug function. Otherwise it can be build using visual studio. It should be noted that if the program is build the HeaderArt.art file should be within the same folder with the executable. The program best runs through command line since it allows the user to receive better feedback.

The program lacks a lot of error checking due to the timely manner in which it had to be delivered. Many of the file streams are not tested as it would be expected and many insecurities exist which the user should be aware of. The program should best be run on the examples provided with a limited size of file as input. If the file is open at the same time by another program CryptoMax will most likely crash. The error checking was omitted mainly because in order to recover from such an error a better program design was required.

The first thing the user sees when he/she runs the program is a multiple menu. The user is first asked if they want to encrypt or decrypt a file. An open file dialog is instantly prompted to the user to select the file which they want to encrypt or decrypt. The options provided to the user by which they can choose the way they want to encrypt or decrypt their file are demonstrated in a table.

Encrypter				
Input	1. Console	2. Key File	3. Random Key	4. Mouse generated
Output	1. Console		2. Key File	
Decrypter				
Input	1. Console		2. Key File	

Functionality Table 1.0

The encrypter takes as input, key from the console, a file or generates a random key or mouse generated key. The keys generated are 255 bytes in size. The console input is undetermined and a passphrase can be used. The main feature implemented was a mouse generated key. It is basically more random than a random number generator because it is user driven. The user uses their mouse pointer to allow the program to collect multiple data points of the user's coordinates to use to create a key. This can happen over a small period of time like 20 seconds or almost instantly. To prevent keys that are easily reversed, the mouse must move at least 1 pixel for both x and y components to make another data collection. If the user keeps their mouse still the process might seem to take a while, but in fact it is waiting for the user input through the mouse. This method produces best results if the mouse is rapidly moved.

The program was designed so that the encrypter output can later on be used as the input for the decrypter. One exception should be made, that the console output created by the encrypter is in hexadecimal form and the user must convert it to ASCII if they wish to input it as a key-phrase. That decision was made due to the fact that many of the ASCII characters are not readable. However the console input on the encrypter is on ASCII form to make it easier to use a pass-phrase to encrypt a file. Regardless of that console output of the key is not suggested because it is easy to be lost or misinterpreted.

For each module of encrypter and decrypter the user can make multiple combinations of decisions allowing diversity in the user preferences. The decisions through the menu options are done by inputting number indicating each option as requested by the program. For ease of use, if an answer is not readable the last possible option is chosen.

An example of usage would be to run the program and select 1, for encryption and selection 3 for a randomized key and 2 for key output. A text file can be inputted with random characters written in it, no bigger than 1MB. The same file will be overwritten and the bytes will be displaced by a random offset stored in the key. The same file can be decrypted by running the program again and selecting 2 to enter the decrypter module, and 2 to input the key through a file. After selecting the same text file and the key file the original message will be retrieved.

Discussion

Many errors are related to the method used for the encryption and the program in general. First of all, the encryption used can be VERY unsafe. Usually program files have huge segments that contain the same byte value such as 00, over huge segments. Because the key is rotating and repeating over the file and displacing all the bytes in a repetitive way, in the empty segment of the file, it will be visible a repetitive pattern. That repetitive pattern can be reversed and the key can easily be found. One way to overcome this is by having a larger key than the empty data segment which can be hard to predict. There are many flaws of the program related to the user experience, such as the user can't change their options. The attempt was to make something very simple demonstrating the functionality of ASM and C++. Test cases that do not comply with the expected usage of the program were not taken into consideration (such as having a file locked for IO by another program). The program was only tested into a Windows 7 environment.

The underlying principle for the encryption is the Cipher method [1]. A file can be thought of, as a long string of bytes. Each byte represents a character. A key can be thought of as a long string of bytes as well. The algorithm works by receptively adding those two strings to encrypt and subtracting them to decrypt as shown in the figures 1 and 2. The mod of that number is taken to normalize the value to be able to be stored within 1 byte.

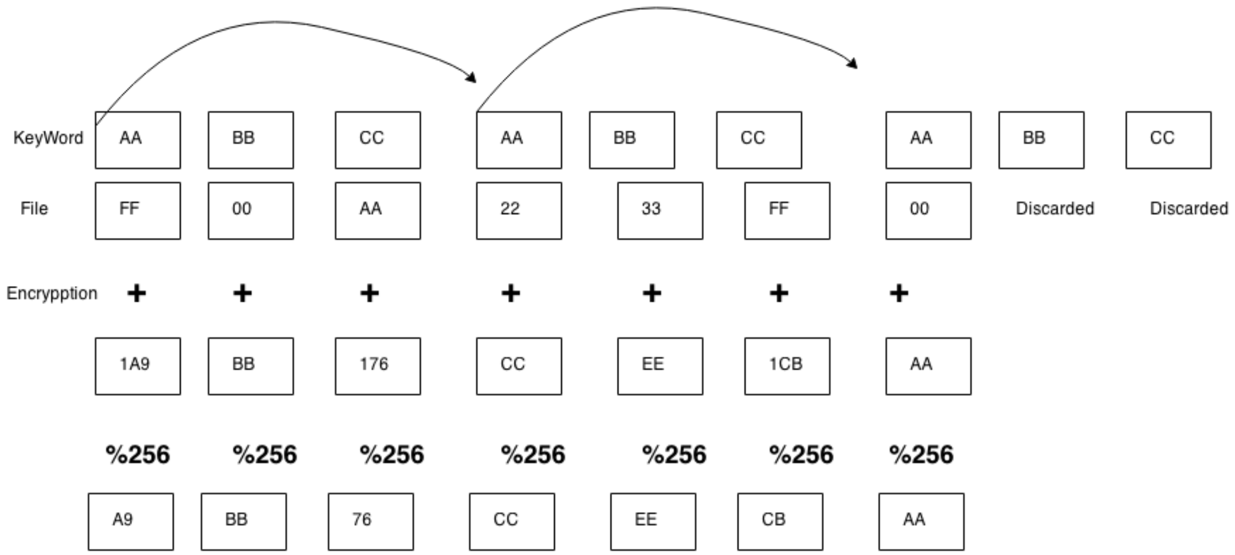


Figure 1: Encryption algorithm

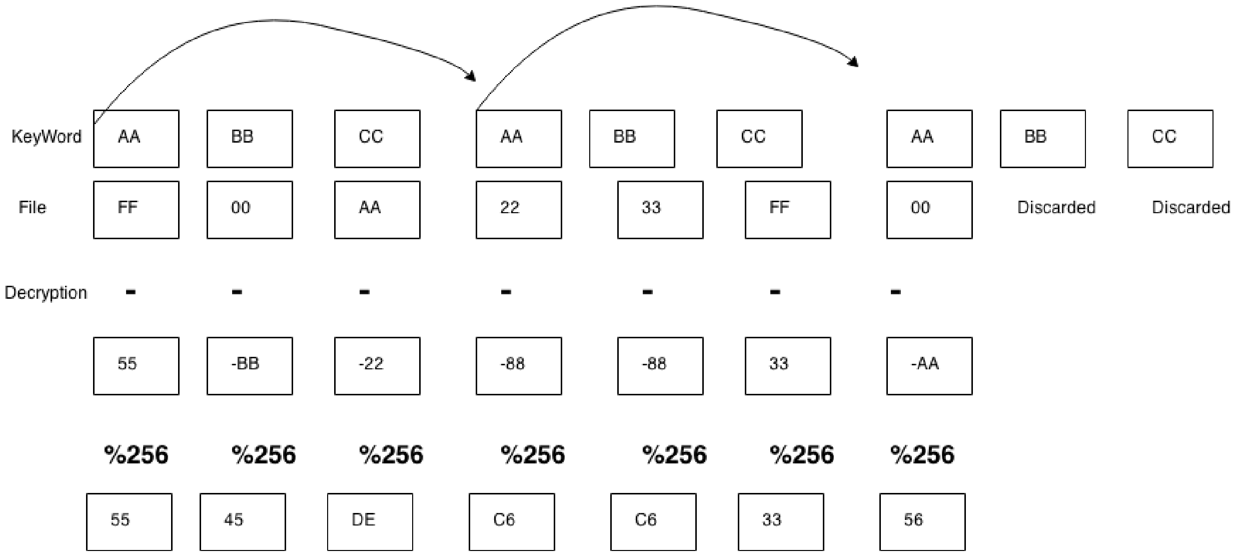
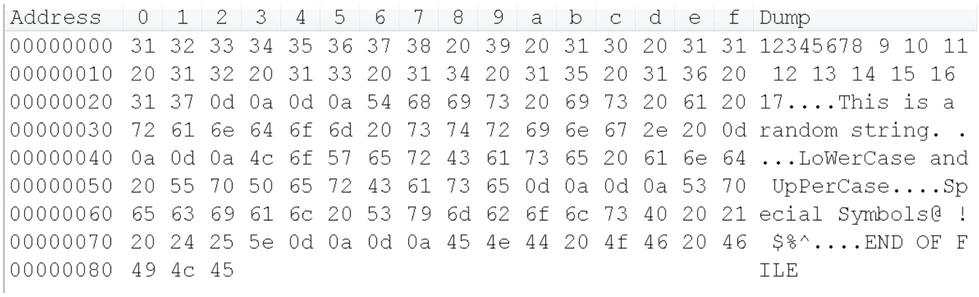


Figure 2: Decryption algorithm

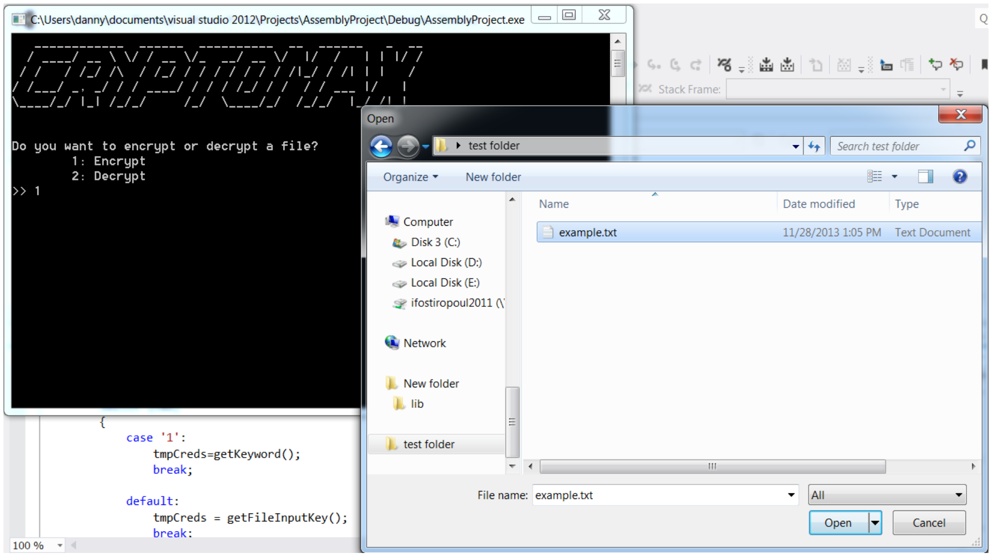
A short demonstration of the algorithm above was made into a screenshot sequence. An original file is created with random characters but the file can be of any form such as executable. The file was encrypted with a key and decrypted back with the same key. The same output was received.

The original file was created with a text editor as shown below.



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000  31 32 33 34 35 36 37 38 20 39 20 31 30 20 31 31 12345678 9 10 11
00000010  20 31 32 20 31 33 20 31 34 20 31 35 20 31 36 20  12 13 14 15 16
00000020  31 37 0d 0a 0d 0a 54 68 69 73 20 69 73 20 61 20 17....This is a
00000030  72 61 6e 64 6f 6d 20 73 74 72 69 6e 67 2e 20 0d random string. .
00000040  0a 0d 0a 4c 6f 57 65 72 43 61 73 65 20 61 6e 64 ...LoWerCase and
00000050  20 55 70 50 65 72 43 61 73 65 0d 0a 0d 0a 53 70  UpPerCase....Sp
00000060  65 63 69 61 6c 20 53 79 6d 62 6f 6c 73 40 20 21 ecial Symbols@ !
00000070  20 24 25 5e 0d 0a 0d 0a 45 4e 44 20 4f 46 20 46  $%^....END OF F
00000080  49 4c 45                                        ILE
```

The encrypter was executed and the file was selected



C:\Users\danny\documents\visual studio 2012\Projects\AssemblyProject\Debug\AssemblyProject.exe

```
Do you want to encrypt or decrypt a file?
        1: Encrypt
        2: Decrypt
>> 1
```

```
            {
                case '1':
                    tmpCreds=getKeyword();
                    break;

                default:
                    tmpCreds = getFileInputKey();
                    break;
```

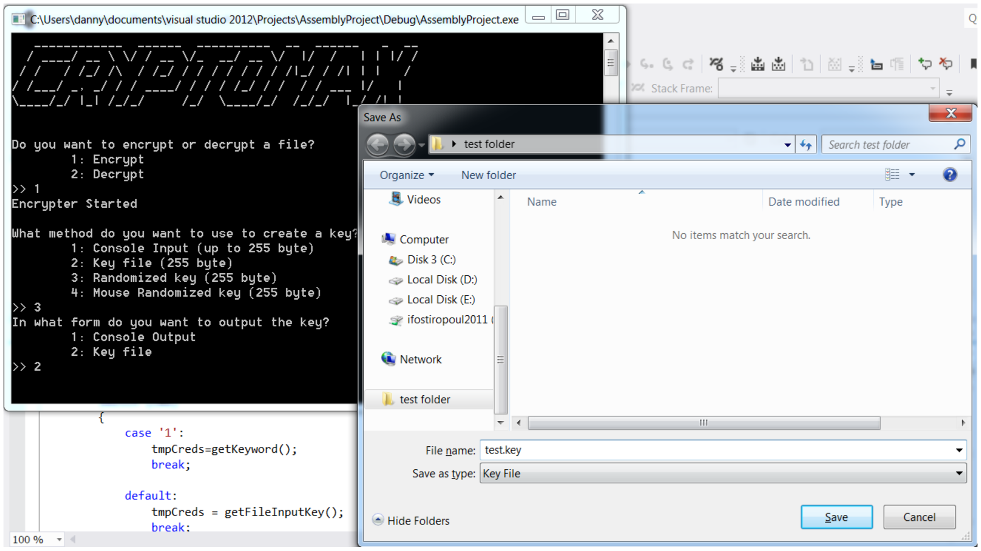
Open

Name	Date modified	Type
example.txt	11/28/2013 1:05 PM	Text Document

File name: example.txt

All

A random key was generated and it was saved on the same folder



C:\Users\danny\documents\visual studio 2012\Projects\AssemblyProject\Debug\AssemblyProject.exe

```
Do you want to encrypt or decrypt a file?
        1: Encrypt
        2: Decrypt
>> 1
Encrypter Started

What method do you want to use to create a key?
        1: Console Input (up to 255 byte)
        2: Key file (255 byte)
        3: Randomized key (255 byte)
        4: Mouse Randomized key (255 byte)
>> 3
In what form do you want to output the key?
        1: Console Output
        2: Key file
>> 2
```

```
            {
                case '1':
                    tmpCreds=getKeyword();
                    break;

                default:
                    tmpCreds = getFileInputKey();
                    break;
```

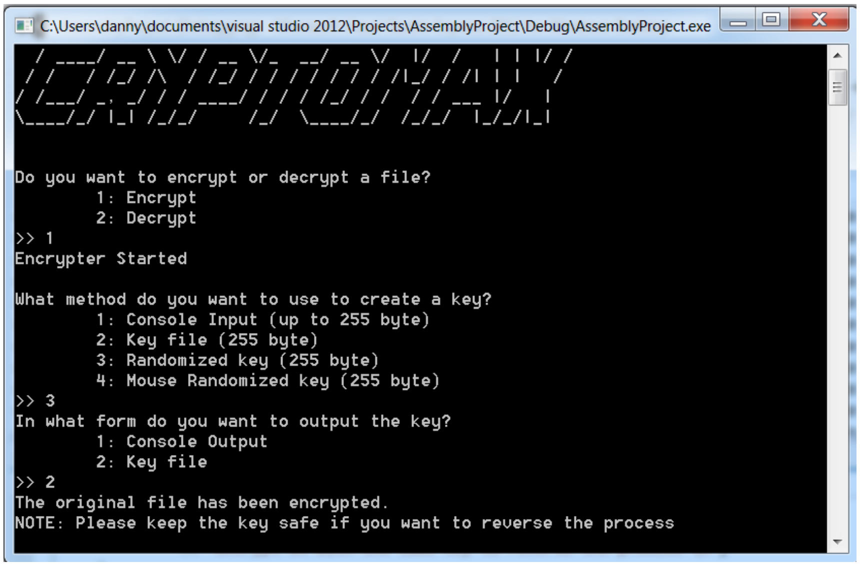
Save As

No items match your search.

File name: test.key

Save as type: Key File

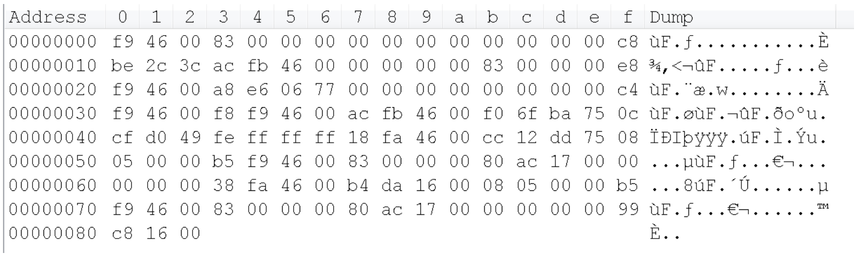
The final message of the program is displayed.



C:\Users\danny\documents\visual studio 2012\Projects\AssemblyProject\Debug\AssemblyProject.exe

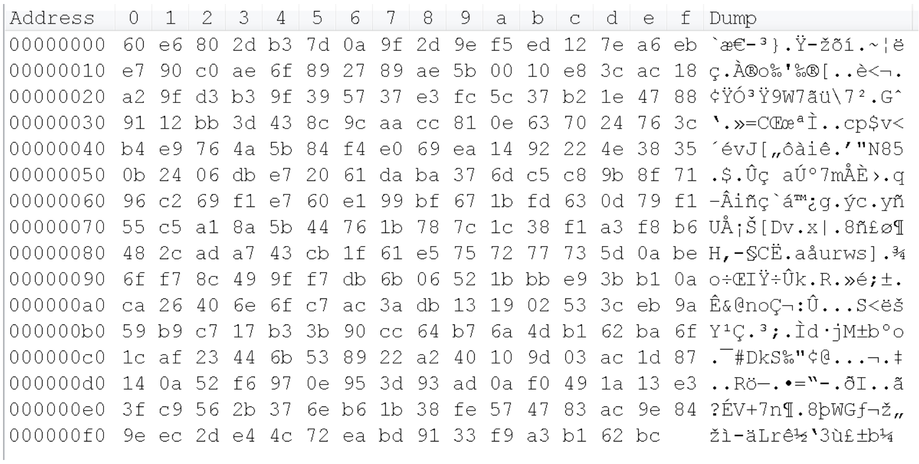
```
  / ____/ __ \ \/ / __ \/_  __/ __ \/   |/  /   | | |/ /
 / /   / /_/ /\  / /_/ / / / / / / / /|_/ / /| | |   /
/ /___/ _, _/ / / ____/ / / / /_/ / /  / / ___ |/   |
\____/_/ |_| /_/_/     /_/  \____/_/  /_/_/  |_/_/|_|


Do you want to encrypt or decrypt a file?
        1: Encrypt
        2: Decrypt
>> 1
Encrypter Started

What method do you want to use to create a key?
        1: Console Input (up to 255 byte)
        2: Key file (255 byte)
        3: Randomized key (255 byte)
        4: Mouse Randomized key (255 byte)
>> 3
In what form do you want to output the key?
        1: Console Output
        2: Key file
>> 2
The original file has been encrypted.
NOTE: Please keep the key safe if you want to reverse the process
```

The file after it was encrypted



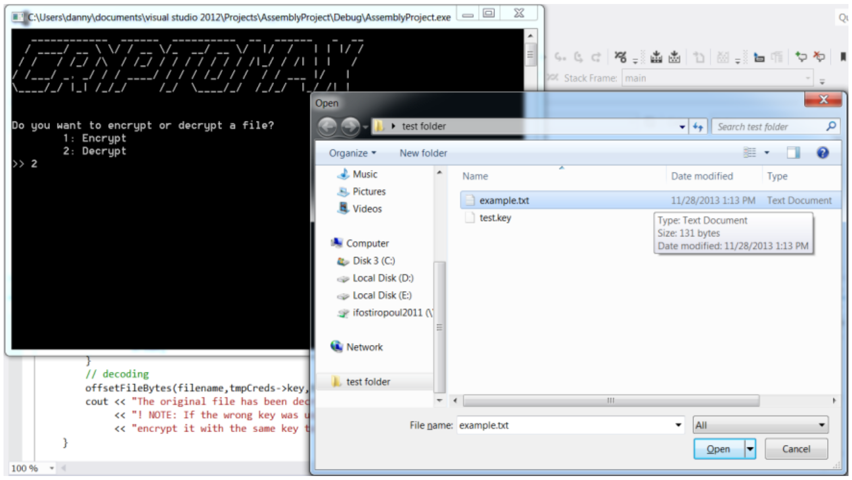
```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 f9 46 00 83 00 00 00 00 00 00 00 00 00 00 00 c8 ùF.ƒ...........È
00000010 be 2c 3c ac fb 46 00 00 00 00 00 83 00 00 00 e8 ¾,<¬ûF.....ƒ...è
00000020 f9 46 00 a8 e6 06 77 00 00 00 00 00 00 00 00 c4 ùF.¨æ.w........Ä
00000030 f9 46 00 f8 f9 46 00 ac fb 46 00 f0 6f ba 75 0c ùF.øùF.¬ûF.ðoºu.
00000040 cf d0 49 fe ff ff ff 18 fa 46 00 cc 12 dd 75 08 ÏÐIþÿÿÿ.úF.Ì.Ýu.
00000050 05 00 00 b5 f9 46 00 83 00 00 00 80 ac 17 00 00 ...µùF.ƒ...€¬...
00000060 00 00 00 38 fa 46 00 b4 da 16 00 08 05 00 00 b5 ...8úF.´Ú......µ
00000070 f9 46 00 83 00 00 00 80 ac 17 00 00 00 00 00 99 ùF.ƒ...€¬......™
00000080 c8 16 00                                        È..
```

The key that was used, 255 bytes long



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 60 e6 80 2d b3 7d 0a 9f 2d 9e f5 ed 12 7e a6 eb `æ€-³}.Ÿ-žõí.~¦ë
00000010 e7 90 c0 ae 6f 89 27 89 ae 5b 00 10 e8 3c ac 18 ç.À®o‰'‰®[..è<¬.
00000020 a2 9f d3 b3 9f 39 57 37 e3 fc 5c 37 b2 1e 47 88 ¢ŸÓ³Ÿ9W7ãü\7².Gˆ
00000030 91 12 bb 3d 43 8c 9c aa cc 81 0e 63 70 24 76 3c ‘.»=CŒœªÌ..cp$v<
00000040 b4 e9 76 4a 5b 84 f4 e0 69 ea 14 92 22 4e 38 35 ´évJ[„ôàiê.’"N85
00000050 0b 24 06 db e7 20 61 da ba 37 6d c5 c8 9b 8f 71 .$.Ûç aÚº7mÅÈ›.q
00000060 96 c2 69 f1 e7 60 e1 99 bf 67 1b fd 63 0d 79 f1 –Âiñç`á™¿g.ýc.yñ
00000070 55 c5 a1 8a 5b 44 76 1b 78 7c 1c 38 f1 a3 f8 b6 UÅ¡Š[Dv.x|.8ñ£ø¶
00000080 48 2c ad a7 43 cb 1f 61 e5 75 72 77 73 5d 0a be H,-§CË.aåurws].¾
00000090 6f f7 8c 49 9f f7 db 6b 06 52 1b bb e9 3b b1 0a o÷ŒIŸ÷Ûk.R.»é;±.
000000a0 ca 26 40 6e 6f c7 ac 3a db 13 19 02 53 3c eb 9a Ê&@noÇ¬:Û...S<ëš
000000b0 59 b9 c7 17 b3 3b 90 cc 64 b7 6a 4d b1 62 ba 6f Y¹Ç.³;.Ìd·jM±bºo
000000c0 1c af 23 44 6b 53 89 22 a2 40 10 9d 03 ac 1d 87 .¯#DkS‰"¢@...¬.‡
000000d0 14 0a 52 f6 97 0e 95 3d 93 ad 0a f0 49 1a 13 e3 ..Rö—.•=“-.ðI..ã
000000e0 3f c9 56 2b 37 6e b6 1b 38 fe 57 47 83 ac 9e 84 ?ÉV+7n¶.8þWGƒ¬ž„
000000f0 9e ec 2d e4 4c 72 ea bd 91 33 f9 a3 b1 62 bc    žì-äLrê½‘3ù£±b¼
```

The decryption module was executed and the same file was selected



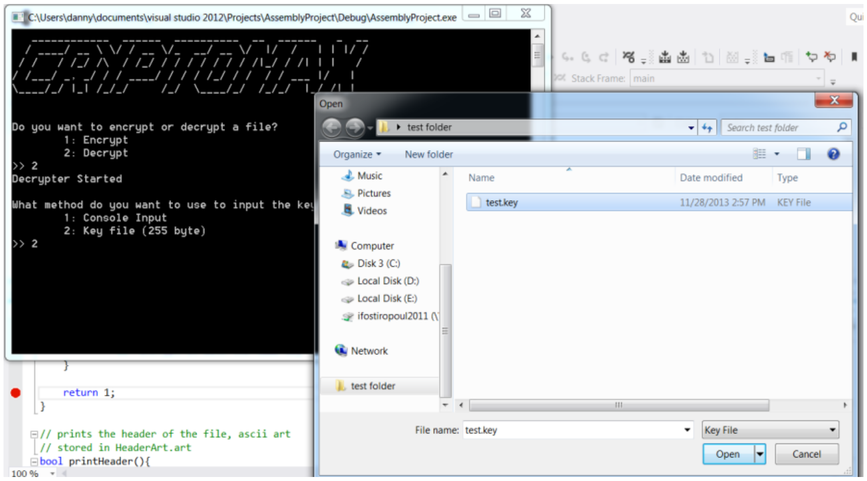
C:\Users\danny\documents\visual studio 2012\Projects\AssemblyProject\Debug\AssemblyProject.exe

```
   ____________  ______  __________  __  ______   _  __
  / ____/ __ \ \/ / __ \/_  __/ __ \/ / |/ /   | | |/ /
 / /   / /_/ /\  / /_/ / / / / / / / /|_/ / /| | |   /
/ /___/ _, _/ / / ____/ / / / /_/ / /  / / ___ |/   |
\____/_/ |_| /_/_/     /_/  \____/_/  /_/_/  |_/_/|_|


Do you want to encrypt or decrypt a file?
        1: Encrypt
        2: Decrypt
>> 2
```

```
        }
        // decoding
        offsetFileBytes(filename,tmpCreds->key,
        cout << "The original file has been dec
             << "! NOTE: If the wrong key was u
             << "encrypt it with the same key t
    }
```

File name: example.txt

The key was selected again.

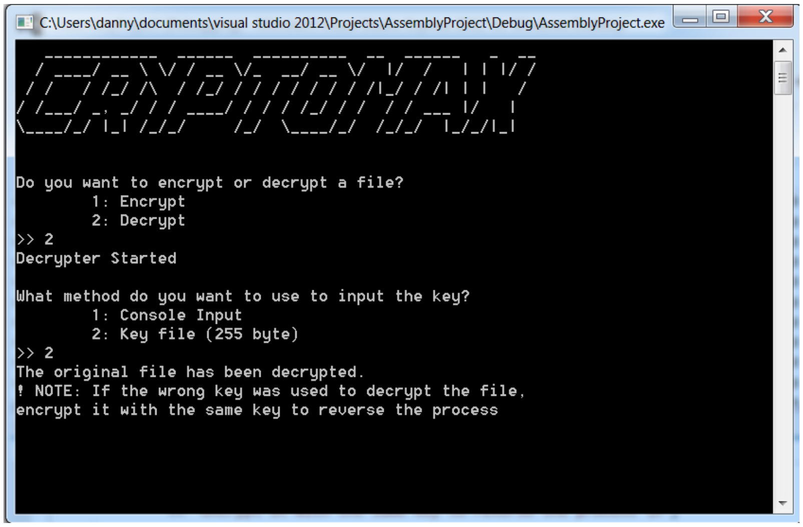


C:\Users\danny\documents\visual studio 2012\Projects\AssemblyProject\Debug\AssemblyProject.exe

```
   ____________  ______  __________  __  ______   _  __
  / ____/ __ \ \/ / __ \/_  __/ __ \/ / |/ /   | | |/ /
 / /   / /_/ /\  / /_/ / / / / / / / /|_/ / /| | |   /
/ /___/ _, _/ / / ____/ / / / /_/ / /  / / ___ |/   |
\____/_/ |_| /_/_/     /_/  \____/_/  /_/_/  |_/_/|_|


Do you want to encrypt or decrypt a file?
        1: Encrypt
        2: Decrypt
>> 2
Decrypter Started

What method do you want to use to input the key
        1: Console Input
        2: Key file (255 byte)
>> 2
```

```
    }

    return 1;
}

// prints the header of the file, ascii art
// stored in HeaderArt.art
bool printHeader(){
```

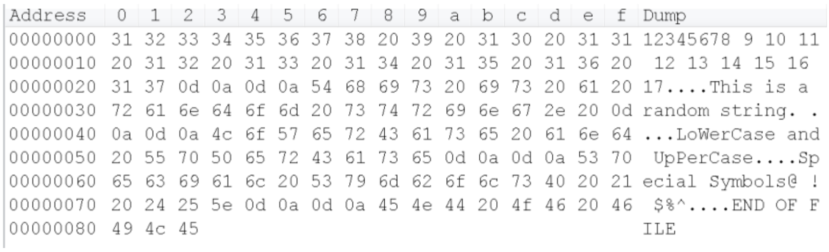
File name: test.key



C:\Users\danny\documents\visual studio 2012\Projects\AssemblyProject\Debug\AssemblyProject.exe

```
   ____________  ______  __________  __  ______   _  __
  / ____/ __ \ \/ / __ \/_  __/ __ \/ / |/ /   | | |/ /
 / /   / /_/ /\  / /_/ / / / / / / / /|_/ / /| | |   /
/ /___/ _, _/ / / ____/ / / / /_/ / /  / / ___ |/   |
\____/_/ |_| /_/_/     /_/  \____/_/  /_/_/  |_/_/|_|


Do you want to encrypt or decrypt a file?
        1: Encrypt
        2: Decrypt
>> 2
Decrypter Started

What method do you want to use to input the key?
        1: Console Input
        2: Key file (255 byte)
>> 2
The original file has been decrypted.
! NOTE: If the wrong key was used to decrypt the file,
encrypt it with the same key to reverse the process
```

The same file was outputted back.



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 31 32 33 34 35 36 37 38 20 39 20 31 30 20 31 31 12345678 9 10 11
00000010 20 31 32 20 31 33 20 31 34 20 31 35 20 31 36 20  12 13 14 15 16
00000020 31 37 0d 0a 0d 0a 54 68 69 73 20 69 73 20 61 20 17....This is a
00000030 72 61 6e 64 6f 6d 20 73 74 72 69 6e 67 2e 20 0d random string. .
00000040 0a 0d 0a 4c 6f 57 65 72 43 61 73 65 20 61 6e 64 ...LoWerCase and
00000050 20 55 70 50 65 72 43 61 73 65 0d 0a 0d 0a 53 70  UpPerCase....Sp
00000060 65 63 69 61 6c 20 53 79 6d 62 6f 6c 73 40 20 21 ecial Symbols@ !
00000070 20 24 25 5e 0d 0a 0d 0a 45 4e 44 20 4f 46 20 46  $%^....END OF F
00000080 49 4c 45                                        ILE
```

Conclusion

I was able to learn many things from this project. I was mainly able to improve my debugging skills and use the disassembler for the first time and have full knowledge of what the instructions were doing. Moreover I was able to understand better the model under which processes are executed and realize the memory limitations. I was able to understand assembly better and find real world applications for it. This project could have been done easier with C++, however the current approach allows for expansion into the current version. The expansion would make it easier for the program to manipulate bytes differently or implement different algorithms. Assembly is very powerful when it comes to handling single data and there is no need to worry about conversion between data types which would have been an issue with C++.

There are some flaws to the program that would not recommend it for as a real security tool. Future expansion could improve the current version and enhance it. Moreover the algorithm is light enough to be used on small microprocessors and devices that require a form of encryption. The applications and expansion are limitless.

Resources:

[1] http://en.wikipedia.org/wiki/Cipher