Manzoor Ahmed

Gaurav Gupta

CS 265 Project 1

10/10/2016

LETTER TO THE TEMPLARS – PART 3

Letter to the Templars uses block ciphers to encrypt and decrypt messages with a key of 7 digits. Encryption method with Letter To the Templars works the following: the plain text is split into blocks of the same length as the key, and each block is encrypted and decrypted separately.

Consider the plaintext “Letter to the Templars --” and the key 2431576, as illustrated in Figure 1. To encrypt the plain text with the key, first the plaintext is removed of spaces, and split into blocks. Each block consists of the same length of characters as the key length; in this case one block is 7 characters, since the key has only 7 digits. Thus, splitting the plaint text into blocks becomes “Lettert otheTem plars--". After characters are swapped with the digits of the key, for example L is moved to position 2, e to position 4 and so on. Figure 2.0 illustrates the swapping process.

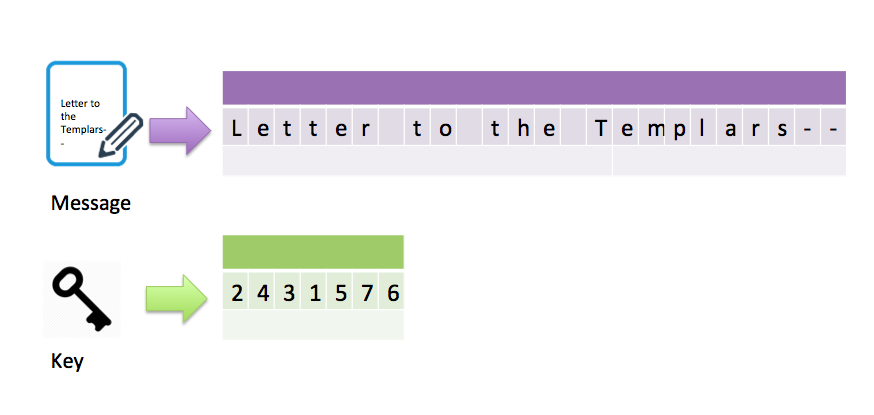


Figure 1 a simple message and the key

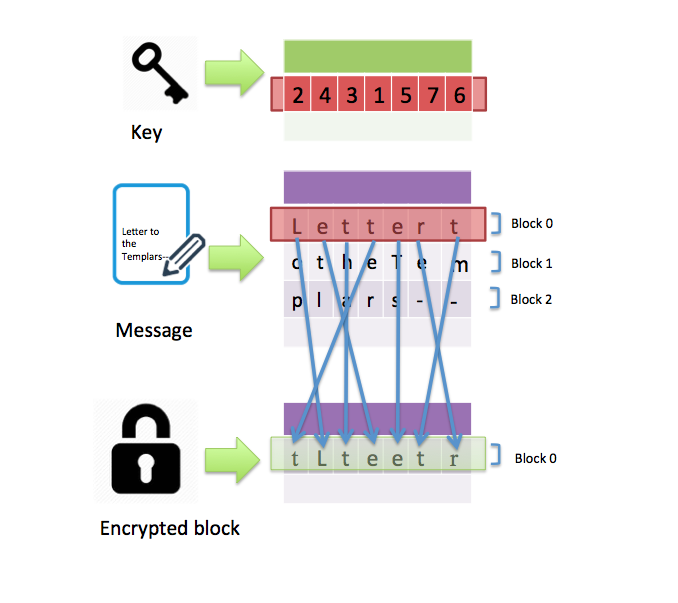


Figure 2 swapping process

**Program Run**

Encryption and decryption algorithm of Letter To the Templars is merely swapping of characters, and each digit of the key dictates the swapping position. In like manner, we will demonstrate that our program main.c uses the above encryption and decryption algorithm, and it is important to realize that it is written to perform operations on any .txt file, not just to decrypt and encrypt *only* Letter To the Templars.

Since encrypting and decrypting also involves Trudy, we have also lessened her work by providing her with our nextGen crackerCipher.c, which she can run to perform a brute force attack against the algorithm.

With this in mind, first we will run the program to decrypt Letter To the Templars, and then encrypt. At the same time, we will show the implementation of the algorithm, and finally we will perform an attack against an encrypted file to obtain a key.

**Decryption**

We will perform decryption on an existing file “cipher.txt” which contains the encrypted text for Letter To the Templars, as shown below.

ydonoaT4ethh1tpStefeo1e30brmKuinor7saighsgndareanwsaretr-n-aatrsityangdaalunopelaTmltwhnsirexptecoonuaoYibteceorrtedpuaet13nhoOocthft1e30broaBck(l7aiy)rdFodnvncadtasceitriceehrobeyusinngogllniqadssdseiauncfisoteadTctsyraleohbaerhmcesntasheadlelsrtstteealdmalotiransiovffietweitcsichhatehoeopgtretmonhennotadnrfeteobrmnieohtohegfnOtct3h11e30brotnosad7tilyrctolwtolfsitrenhotnsciuaninotctthedoeIrfyodwifltuaoyrouamoveesslreewihrtabnulelcredpendtwaneeaorrefv-t-assektecdanfeyoroalsvereufnaiadstoountleestdrneaoumtileorwlrznedaigclefoipTchereoaplsoePlinowlokayobctlhipPiuaVsdIhpoepevleoiwnhsdtregtasiellhwyhtePutpnedepuosrsuperwHileerehattrlstplnoeaFnctriucrcsheinnsadhtgeaiateossrcpetgrnhosnofudotriceehhneeitstnhaetvPhoptetlastfieciondsoenhiiutpuorspsyooufrtnloteyRethenhooothpfl-spur-erofayutnhthtiiawleonesnsaterymoudandobeLrrytouihw

According to the documentation, the above cipher was encrypted with 7-digit key, thus by using only the first 7 characters ydonoaT, we can generate the message Todayon , with the key 7526134. Subsequently, applying the key to each 7 characters, we can decrypt the entire file as demonstrated below, in Figure 3 and 4.

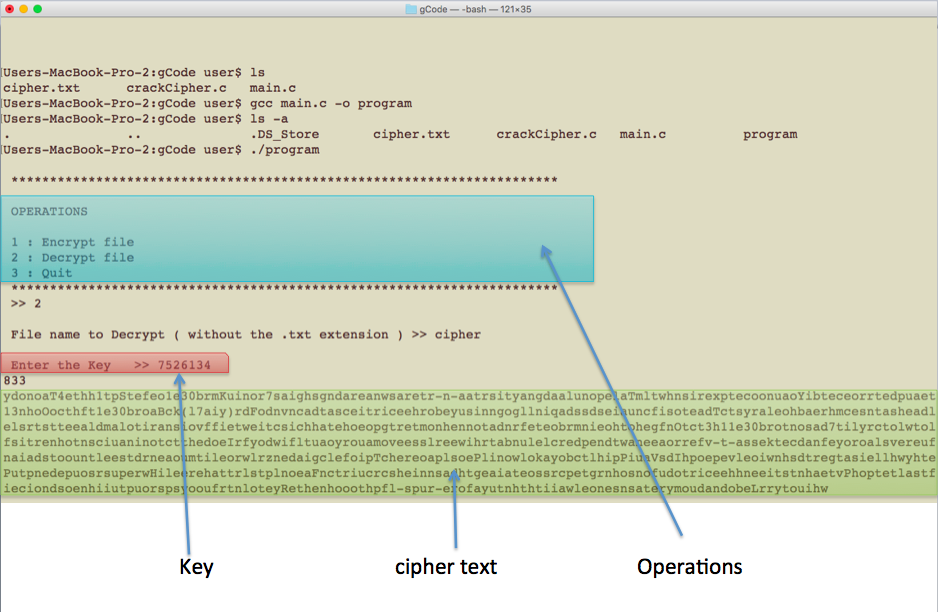


Figure 3.0 Decrypting cipher.txt

Below screen shot show that the user gives a file name to store decrypted message. After decryption, the user can see the content of decrypt.txt.

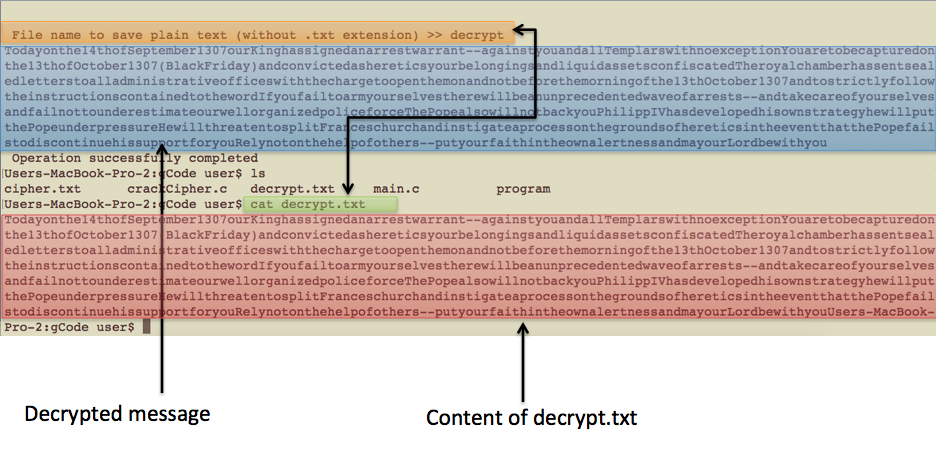


Figure 4.0 Decrypted message saved in decrypt.txt

After adding spaces manually to the content of decrypt.txt file, we obtain the following message:

Today on the 14th of September 1307 our King has signed an arrest warrant--against you and all Templars with no exception

You are to be captured on the 13th of October 1307(Black Friday) and convicted as heretics your belongings and liquid assets confiscated

The royal chamber has sent sealed letters to all administrative offices with the charge to open them on and not before the morning of the

13th October 1307 and to strictly follow the instructions contained to the word If you fail to arm yourselves

there will be an unprecedented wave of arrests--and take care of yourselves and fail not to underestimate our well organized police force

The Pope also will not back you Philipp IV has developed his own strategy he will put the Pope under pressure

He will threaten to split Frances church and instigate a process on the grounds of heretics in the event that the Pope fails to discontinue his support for you

Rely not on the help of others--put your faith in the own alertness and may our Lord be with you

**Decryption Source Code**

In the following screen shots, vital details are highlighted for decryption algorithm.

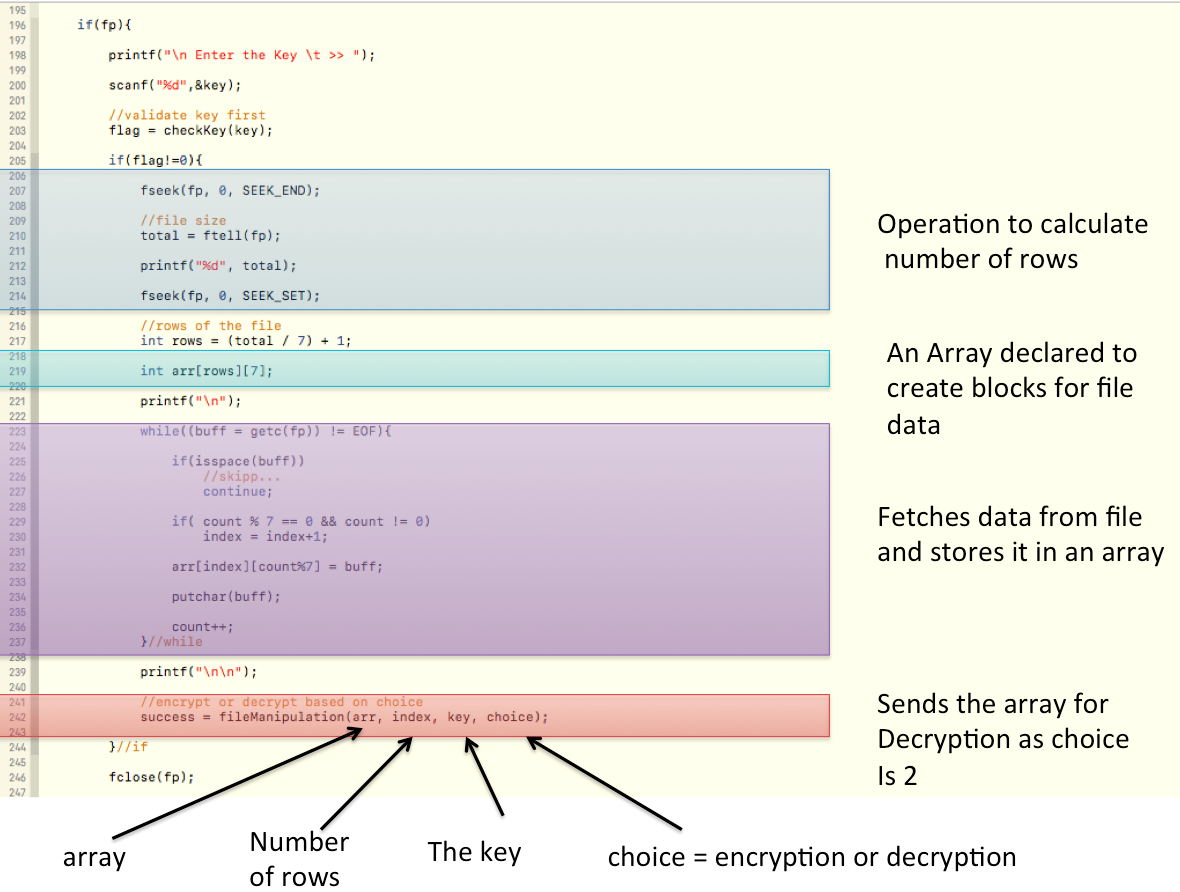


Figure 5 File data is split into row and columns

Next, user is prompted to enter a file name to save plain text, and if the file exists data is stored, if not the program generates error message. In the next step, a block of 7 characters are swapped and saved to the file, and this process continues until all the blocks are swapped and saved into the file. Figure 6.

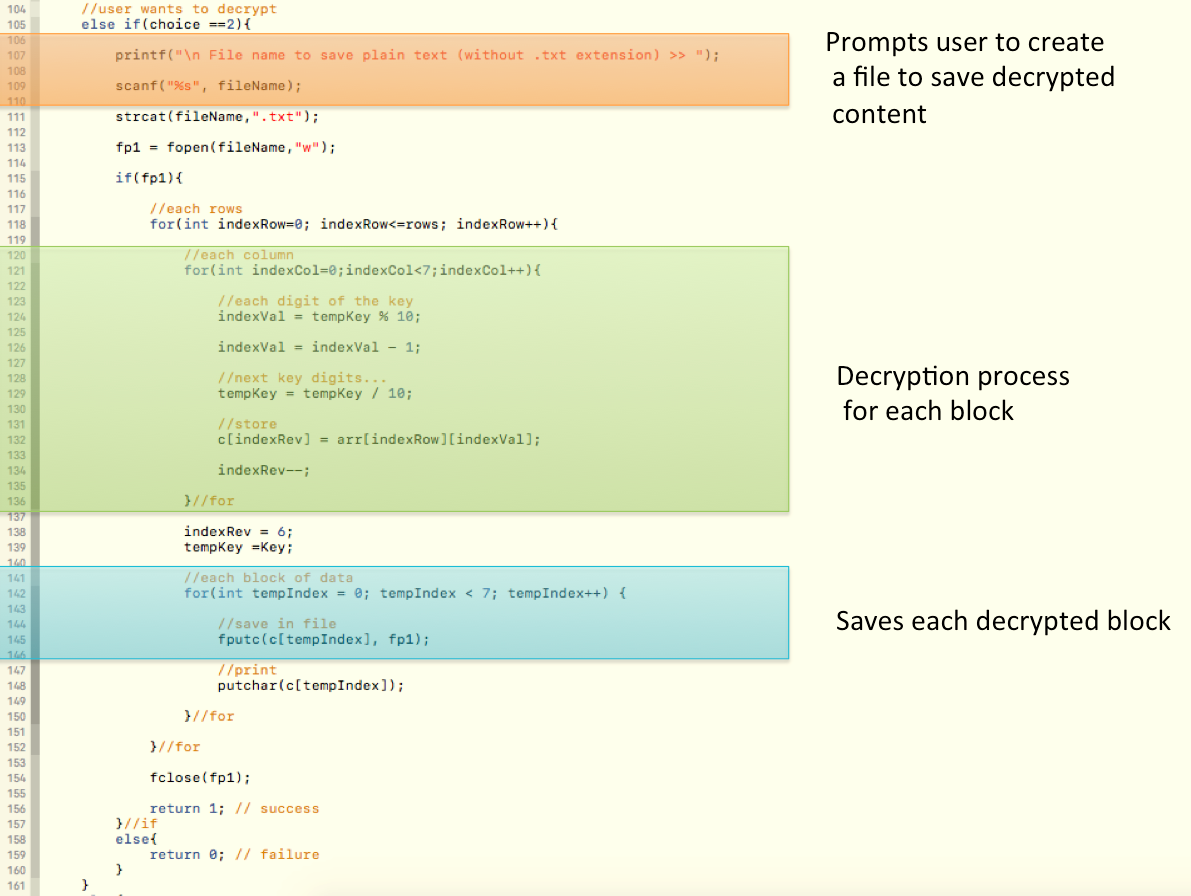


Figure 6 Decryption source code

**Encryption**

As mentioned before, for encryption the swapping of the characters are reversed based on the digits of the key. Since decrypt.txt has the plain text already, the same key is used to encrypt the data. In the following program run will demonstrate how the plain text gets encrypted back.

In the screen shot below Figure 5, user provides the file name to encrypt, and if that file exists then the data is read for encryption, if the file does not exists then an error message is generated. Next, the user is asked to provide a key to encrypt with.

The program checks the key for validation, and if the key is valid it proceeds with encryption, if the key is invalid, it just prints an error message without giving the user another change to entry the key again. This also increases program security, since the user has to run the program again, instead of trying different keys.

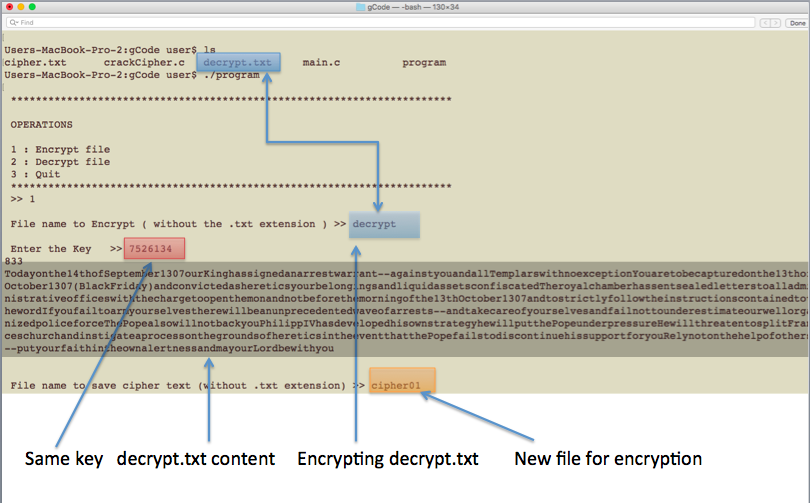


Figure 5 Encrypting decrypt.txt

The next screen shot shows that a new file **cipher01.txt** file is generated based on the user’s input, because that is where the user saves the encrypted data, so cat cipher01.txt shows it’s content.



Figure 6 Encrypting decrypt.txt

**Encryption Source Code**

In the following screen shot, vital details are highlighted for encryption algorithm.

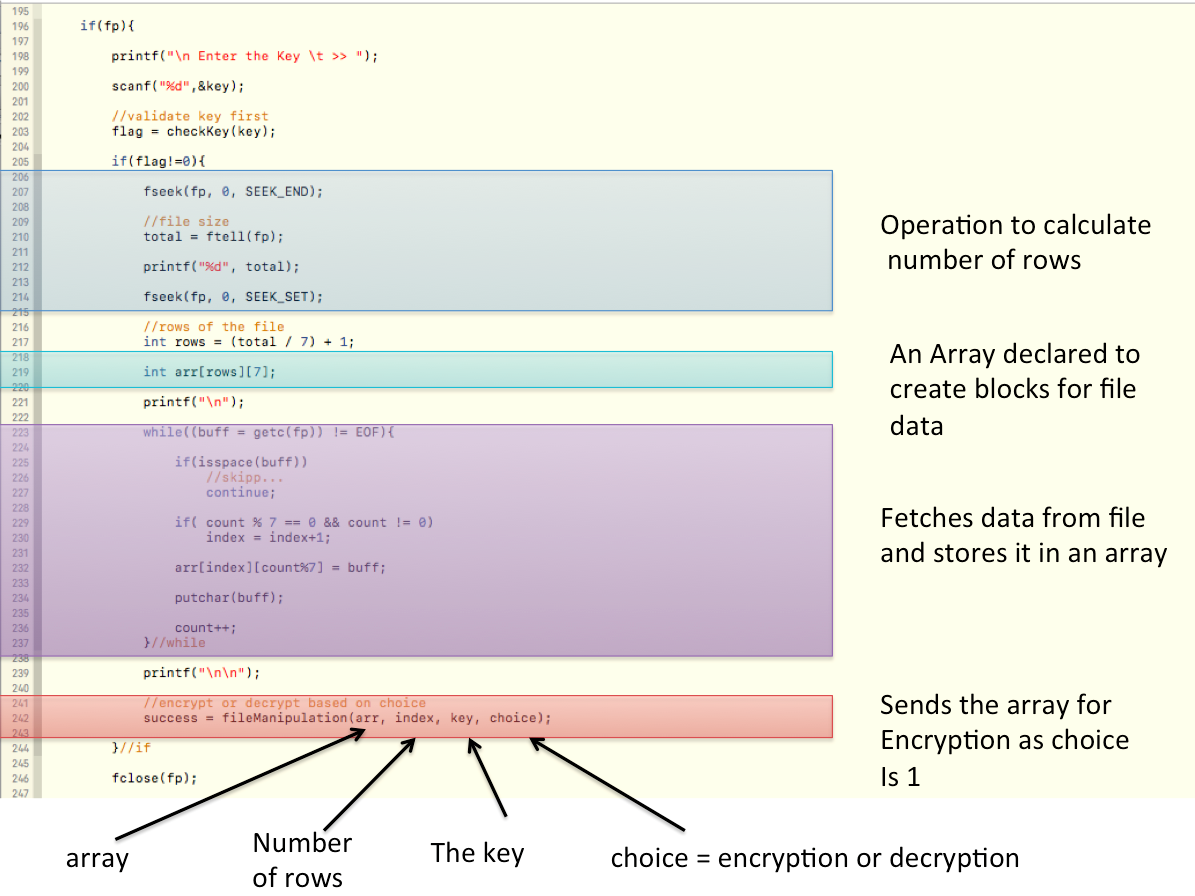


Figure 7 encryption source code for generating rows and columns

Next, the reverse steps of decryption algorithm are taken, user is prompted to enter a file name to save encrypted text, and if the file exists data is stored, if not the program generates error message. Each block of 7 characters are encrypted and saved to the file, and this process continues until all the blocks are swapped and saved into the file.

Figure 8 shows the encryption source code.

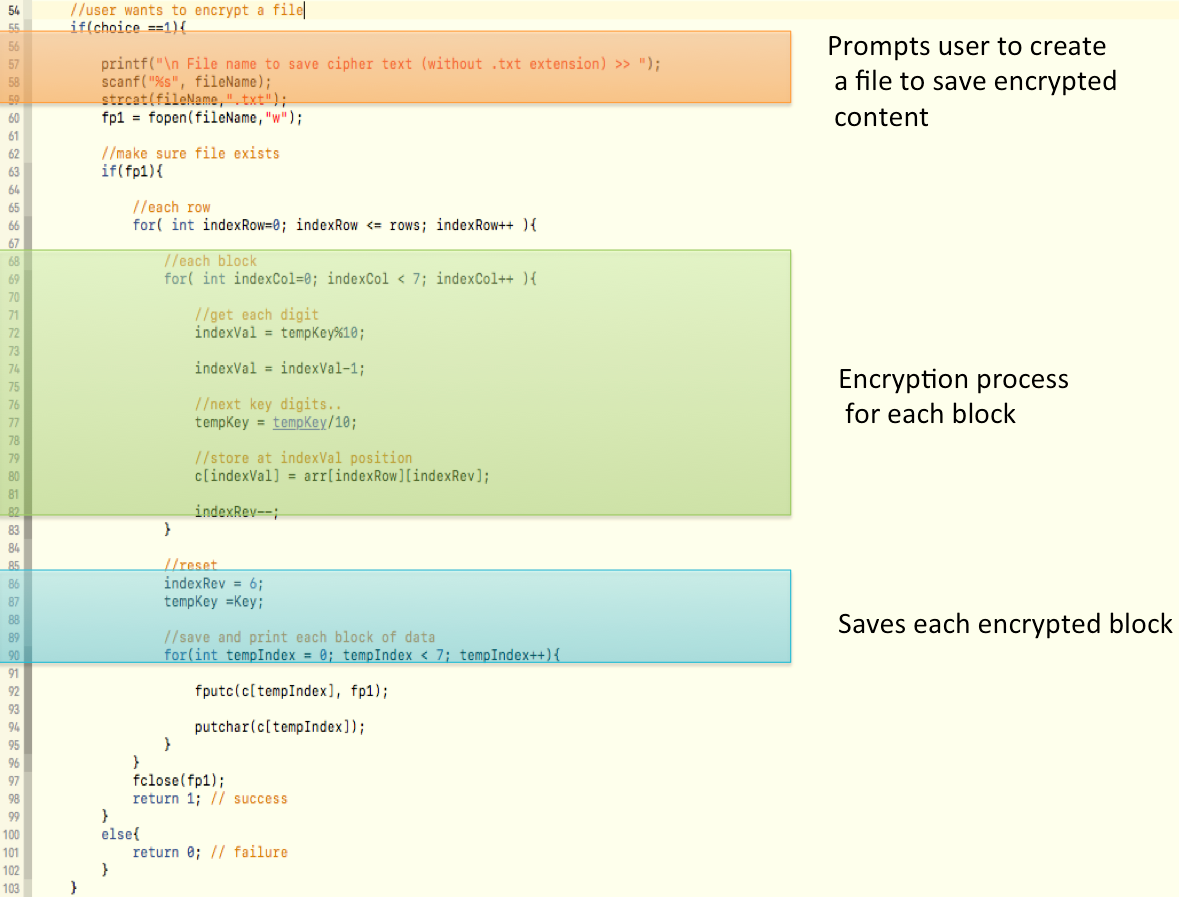


Figure 8 Encryption process source code

**Attack**

To perform an attack against the system, crackerCipher.c is implemented to generate 7 digit keys, because we know that the cipher was decrypted with 7-digit key; that is, the actual key is one of the keys within the range of 1234567 to 7654321. With each attempt, a different key is used to decrypt the cipher, and if a message is a legitimate message gives us the original key. Let us look at crackerCipher.c in action.

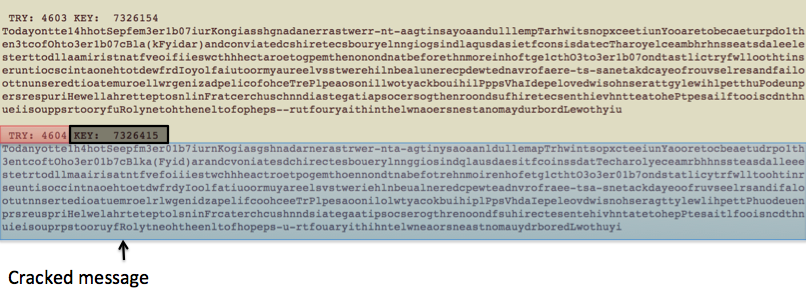
****

Figure 9 after try 4604 correct message and key is found

**Computational Experiment**

If we had a file containing 1000 characters, and splitting the file into 7 characters would give us142 blocks. Let us further assume, that it would talk 0.000001 second to encrypt each block. In addition, if we need to use every single key, then we would need to use 6419755 keys (the total number of possibilities for 7 digit key). Thus, using all keys would take 911.6 seconds to encrypt or decrypt 1000 characters.

Although, using every single key is wasting of time, and resources because an intruder like Trudy can eliminate most of the keys. Instead of using 6419755 keys she could just use 5040, enhancing optimization by tremendous factor of 99.92%.

The screen shot below show the difference between using all possible keys, and only using 5040 keys. (The screen shot shows 5039, since the try counter started at 0).

