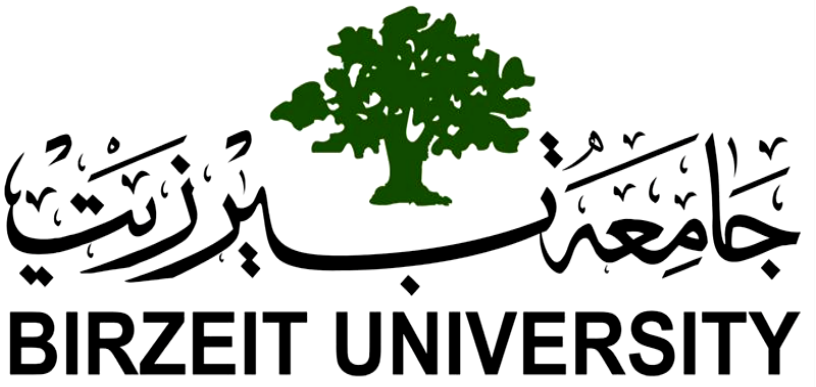


**Faculty of Engineering & Technology**

**Electrical & Computer Engineering Department**

**LINUX LABORATORY ENCS3130**

**Project #1**

# Shell Scripting Project – Text Message Encryption and Decryption

**Prepared by**:

Hasan Hamed 1190496

Shorooq Ngjar 1192415

**Instructor**: Dr. Mohammad Jubran

**Assistant**: Eng. Ibrahim Angass

**Section**: 1

**Date:** 14/8/2022

# Table of Contents

# Table of Figures

# Encryption

## 1. Step 1: Generate key

### 1.1. Code

```
sum=0                       #intilaizing counter with 0

while read -n1 c; do                    # reading the plain text file character by character

        if [[ "$c" == "a" || "$c" == "A" ]]; then

                sum=$((sum+1))

        elif [[ "$c" == "b" || "$c" == "B" ]]; then             #adding the index of each character to the sum

                sum=$((sum+2))

        elif [[ "$c" == "c" || "$c" == "C" ]]; then

                sum=$((sum+3))

        elif [[ "$c" == "d" || "$c" == "D" ]]; then

                sum=$((sum+4))

        elif [[ "$c" == "e" || "$c" == "E" ]]; then

                sum=$((sum+5))
```

```
elif [[ "$c" == "f" || "$c" == "F" ]]; then
        sum=$((sum+6))
elif [[ "$c" == "g" || "$c" == "G" ]]; then
        sum=$((sum+7))
elif [[ "$c" == "h" || "$c" == "H" ]]; then
        sum=$((sum+8))
elif [[ "$c" == "i" || "$c" == "I" ]]; then
        sum=$((sum+9))
elif [[ "$c" == "j" || "$c" == "J" ]]; then
        sum=$((sum+10))
elif [[ "$c" == "k" || "$c" == "K" ]]; then
        sum=$((sum+11))
elif [[ "$c" == "l" || "$c" == "L" ]]; then
        sum=$((sum+12))
elif [[ "$c" == "m" || "$c" == "M" ]]; then
        sum=$((sum+13))
elif [[ "$c" == "n" || "$c" == "N" ]]; then
        sum=$((sum+14))
elif [[ "$c" == "o" || "$c" == "O" ]]; then
        sum=$((sum+15))
elif [[ "$c" == "p" || "$c" == "P" ]]; then
        sum=$((sum+16))
elif [[ "$c" == "q" || "$c" == "Q" ]]; then
        sum=$((sum+17))
elif [[ "$c" == "r" || "$c" == "R" ]]; then
        sum=$((sum+18))
elif [[ "$c" == "s" || "$c" == "S" ]]; then
        sum=$((sum+19))
```

```
        elif [[ "$c" == "t" || "$c" == "T" ]]; then
                sum=$((sum+20))
        elif [[ "$c" == "u" || "$c" == "U" ]]; then
                sum=$((sum+21))
        elif [[ "$c" == "v" || "$c" == "V" ]]; then
                sum=$((sum+22))
        elif [[ "$c" == "w" || "$c" == "W" ]]; then
                sum=$((sum+23))
        elif [[ "$c" == "x" || "$c" == "X" ]]; then
                sum=$((sum+24))
        elif [[ "$c" == "y" || "$c" == "Y" ]]; then
                sum=$((sum+25))
        elif [[ "$c" == "z" || "$c" == "Z" ]]; then
                sum=$((sum+26))
        else                                    #since there is no non-alpha character, whenever the
loop reads a space or newline it returns the sum of each word
        key=$((sum%256))                        # finding the key of each sum
        echo $key >> keys.txt                   # redirecting the key of each word to keys file
        sum=0                                   #reintilaizing the sum to 0 to use for next words
        fi
done < "$inputfile"
Decimalkey=`sort -n keys.txt | tail -1`         # soring keys file and returning the tail of it which is the
max of sums mod 256 which is our key in decimal
```

### 1.2. Idea

The idea of generating the key depends on reading the plain text file character by character and adding the index of each character to a variable sum and then reinitializing the sum whenever there is a space or new line, and then finding the key of each word that is sum % 265 and then redirecting each key to a file and finding the max key using sort and tail.

### 1.3. Output

```
hasan@hasan-VirtualBox:~/Desktop$ ./SimpleEncryption
Text Message Encryption and Decryption Menu
1) Enter e For Encryption
2) Enter d For Decryption
3) Press q To Quit
e
Please input the name of the plain text file
myfile.txt
Plain Text Is:
Test test LOL HAHA
Hasan
Key in Decimal Is:
64
```

**Figure 1.1: Key Generation Output**

## 2. Step 2: Representing Key as an 8-bit binary number.

### 2.1. Code

Binarykey=$(echo 'obase=2;' ${Decimalkey}| bc ) #converting key from decimal to binary

Keyy=`printf "%0*d\n" 8 $Binarykey` #this command is important since the converted key is not 8 bit so it appends zeros to the left of the key untill its 8 bit

### 2.2. Idea

The decimal key was converted to binary using bc and was made 8 bit using printf by appending zeros to the left of the key until it's 8 bit.

### 2.3. Output

```
Key in Decimal Is:
64
Key In Binary Is :
01000000
```

**Figure 2.1: 8-bit Binary Key**

## 3. XOR between the key generated and each character and swaping the first 4-bit with the last four bit.

### 3.1. Code

while read -r line; do #since the message can be multi lines this loop goes for every line in the inputfile

```
echo -n $line | perl -lpe '$_=join " ", unpack"(B8)*"' | tr ' ' '\n' > PlainTextB.txt
```

#above command gets the line words and change the whole line to aasci with each characther is 8 bit binary according to aasci with each word aasci separeted by space and then it changes the space to newline and redirect it to PlainTextB.txt file to be used for the second loop

```
        while read -r Line; do                          #loop that goes for each character aasci in each
line to do the encryption

                XorResultD=`echo $(( 2#$Line ^ 2#$Keyy ))`      #xoring each character with the key

                XorResultB=$(echo 'obase=2;' ${XorResultD}| bc ) #converting xor result to binary since
the output of the xor is decimal

                LastFour=`printf "%0*d\n" 8 $XorResultB | cut -c5-8`    #getting the last four characters
of xor result

                FirstFour=`printf "%0*d\n" 8 $XorResultB | cut -c1-4`    #getting the first four characters
of xor result

                Encrypted=$LastFour$FirstFour                   #swapped xor result


                echo $Encrypted >> Encrypted.txt                #redirecting xor result of each
character to Encrypted.txt file

                done < PlainTextB.txt

        cat Encrypted.txt | tr '\n' ' ' | head -c-1 >> Cipher.txt  #this removes an extra space in the last of each
line that made me some troubles in decryption

        echo "" >> Cipher.txt                                   # newline for the new line encryption

        rm -r Encrypted.txt                                     #removing Encrypted.txt file to be used
for the later lines encryptions

done < $inputfile
```

### 3.2. Idea

The idea is reading the file line by line and getting the asci code of each line using perl command which it's input a word and it output is the 8 bit asci code of each character separated by space. And for each character in each line it was Xored with the key and converted into binary since the output of the xor is decimal and the first four and last four bits were swapped using cut command and concatenation and for each line encrypted characters which were one in a line were separated by space and moved to the cipher file, and that will be done until all lines are read thus the cipher file will contain each line encryption from the input file.

### 3.3. Output

```
1 01000001 01010010 00110011 01000011 00000110 01000011 01010010 00110011 01000011 00000110 11000000 11110000 11000000 00000110 10000000 00010000 10000000 00010000
2 10000000 00010010 00110011 00010010 11100010 00000100
```

**Figure 3.1: Xoring and Swapping Output.**

## 4. Swapping Key and Adding it to Ciphertext

### 4.1. Code

```
LastFourKey=`echo $Keyy | cut -c5-8`                    #getting the last four characters of key

FirstFourKey=`echo $Keyy | cut -c1-4`                   #getting the first four characters of key

EncryptedKey=$LastFourKey$FirstFourKey                  #swapped key

LastLineNoKey=`tail -n -1 Cipher.txt`                   #returning the last line of the cipher

LastLineWithKey="$LastLineNoKey $EncryptedKey"          #appending the key with the
last line of the cipher which means the key will be the last character

cat Cipher.txt | head -n -1  > temp.txt ; mv temp.txt Cipher.txt          #since the cipher texts
contains the last line with no key this commands removes the last line

echo $LastLineWithKey >> Cipher.txt                                      # redirecting the
last line with key appended to the cipher text

echo "Plain Text Is: "

cat $inputfile

echo "Key in Decimal Is: "

echo "$Decimalkey"

echo "Key In Binary Is :"

echo "$Keyy"

echo "input the name of the cipher text file"           #the file that will contain the final encrypted
message

read cipherfile

cat Cipher.txt > "$cipherfile"                          #redirecting the encrypted message in
Cipher.txt to the user chosen cipher text

echo "Encrypted Message in Binary is: "

cat "$cipherfile" | tr -d " \t | tr '\n' ' '"           # printing the ciphertext of all lines with no spaces or
newlines

echo ""
```

**4.2. Idea**

The key was swapped using cut and concatenation and the last line was recovered using tail command and the encrypted key was concatenated to the last line and the last line with no key was replaced with the concatenated last line.

### 4.3. Output

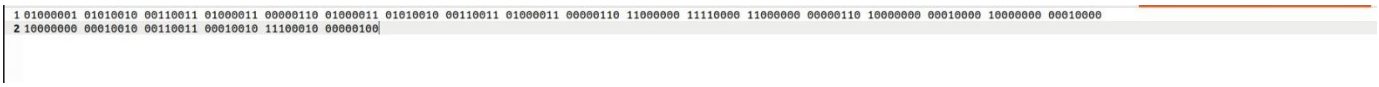


**Figure 4.1: Generated Cipher File with Swapped Key**

# Decryption

## 1. Getting Key and Swapping It

### 1.1. Code

```
Key=`cat $cipherfile | tr ' ' '\n' | tail -1`


#above command, since the key is the last character and the cipher file contains the encryption of each line and each character encrypted separeted by spaces the key will be the last line after changing spaces to new line and returning last line


LastFourKey=`echo $Key | cut -c5-8`                #getting the last four characters of key

FirstFourKey=`echo $Key | cut -c1-4`               #getting the first four characters of key

DecryptedKey=$LastFourKey$FirstFourKey                    #swapped key which is the original key

KeyD=$(echo 'ibase=2;obase=A;' ${DecryptedKey}| bc )              #converting key to decmimal
```

### 1.2. Idea

Since the cipher file contain every encrypted character for each line and the key is the last character and they are all separated by space the key was recovered using tr to newline and tail because the key is in the last line, and swapping was done using cut and concatenation and it was converted to decimal using bc.

### 1.3. Output

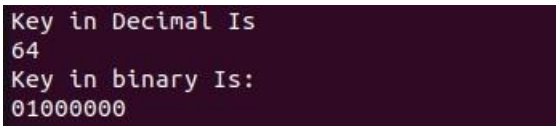


**Figure 1.1: Decrypted Key Output**

## 2. Swapping Characters from Encrypted File and Doing XOR

### 2.1. Code

```
while read -r line; do                  # a loop that goes for each line in the encrypted message since it might be multilined message

        echo $line | tr ' ' '\n' > EncryptedLines.txt


#above, taking each line and translating spaces to newlines and redirecting it to EncryptedLines.txt to be used in the next loop which means that each line contains encrypted char of each line


        while read -r Line; do                                  #a loop that goes on each encrypted character in each line

                LastFour=`echo -n $Line | cut -c5-8`            #getting the last four characters of each encrypted character

                FirstFour=`echo -n $Line | cut -c1-4`           #getting the first four characters of each encrypted character
```

```
        Decrypted=$LastFour$FirstFour                    #swapped character

        XorResultDecD=`echo $(( 2#$Decrypted ^ 2#$DecryptedKey ))`   #xoring the swapped character with the restored key

        XorResultDecB=$(echo 'obase=2;' ${XorResultDecD}| bc )       #converting xor result to binary since the output of the xor is decimal

        XorResultDecBF=`printf "%0*d\n" 8 $XorResultDecB`            #making the xor result 8 bits by adding extra zeros to the left of the result untill its 8 bit

        echo  $XorResultDecBF >> DecryptedText.txt                   # redirecting each decrypter character to DecryptedText.txt file

    done < EncryptedLines.txt

    cat DecryptedText.txt | tr '\n' ' ' | head -c-1 >> DecryptedLines.txt


#above, since each decrypted character for each line is in a line in DecryptedText.txt, this command translate the newline into spaces and redirect it to DecryptedLines.txt and (-c-1) to remove an extra space at the last of each line


    echo "" >> DecryptedLines.txt          #the next loop didn't work without this newline

    rm -r DecryptedText.txt                #removing DecryptedText.txt to be used in later lines decryption

done < EncryptedNoKey.txt

cat DecryptedLines.txt > DecryptedLinesFinal.txt   #redirecting DecryptedLines.txt to the DecryptedLinesFinal.txt

rm -r DecryptedLines.txt                   #removing DecryptedLines.txt to be used for later times

while read -r line; do

    echo $line | perl -lape '$_=pack"(B8)*",@F' >> Decrypted.txt


# above, since each line in DecryptedLinesFinal.txt contains the decrypted message for each line of the
```

inputfile this loop convert each line aasci to characters and redirect it to Decrypted.txt which is the input message

done < DecryptedLinesFinal.txt

echo "Key in Decimal Is"

echo "$KeyD"

echo "Key in binary Is: "

echo "$DecryptedKey"

echo "input the name of the plain text file"

read plaintext

cat Decrypted.txt > $plaintext

echo "Decrypted Message in Binary IS: "

cat DecryptedLinesFinal.txt | tr '\n' ' ' | tr -d " \t"  # decrypted message in binary without newlines or spaces

echo ""

echo "Decrypted Message IS: "

cat $plaintext

### 2.2. Idea

The idea here is to first remove the key from the encrypted file using head command and then reading line line and from each line read character character using the nested loop and for each character first and last four bits were swapped using cut and concatenation and then doing the xor and convert it to binary, and that will continue until all lines are decrypted, the output will be a file containing the decryption of each line in a separate line and then using the last loop it will loop for every line and convert each line from binary to characters using perl the output will be the decryption of the encryption which is the original message.

### 2.3. Output

```
Decrypted Message in Binary IS:
01010100011001010111001101110100001000000111010001100101011100110111010000100000010011000100111101001100001000000100100001000001010010000100000101001000011000010111001101100001011011
Decrypted Message IS:
Test test LOL HAHA
Hasan
```

**Figure 2.1: Output of Decryption Process**

```
1 Test test LOL HAHA
2 Hasan
```

**Figure 2.2: Decryption File**

# Test Cases

## 1. Single Line

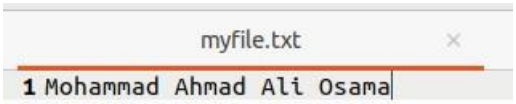


**Figure 1.1: Input File**

```
hasan@hasan-VirtualBox:~/Desktop$ ./SimpleEncryption
Text Message Encryption and Decryption Menu
1) Enter e For Encryption
2) Enter d For Decryption
3) Press q To Quit
e
Please input the name of the plain text file
myfile.txt
Plain Text Is:
Mohammad Ahmad Ali Osama
Key in Decimal Is:
68
Key In Binary Is :
01000100
input the name of the cipher text file
c.txt
Encrypted Message in Binary is:
1001000010110010110000100101001001010010100100101001010000000100100011001010000110000101001001001010010000000100100011001010000100000101101001001000110101100000111001101010010100100100101001001000100
Text Message Encryption and Decryption Menu
1) Enter e For Encryption
2) Enter d For Decryption
3) Press q To Quit
d
Please input the name of the cipher text file
c.txt
Key in Decimal Is
68
Key in binary Is:
01000100
input the name of the plain text file
d.txt
Decrypted Message in Binary IS:
0100110101011011110110100001100001011011010110110101100001011001000100000010000010110100001101101011000010110010000100000010000010110110001101001001000000100111011100110110000101101101011000001
Decrypted Message IS:
Mohammad Ahmad Ali Osama
Text Message Encryption and Decryption Menu
1) Enter e For Encryption
2) Enter d For Decryption
3) Press q To Quit
```

**Figure 1.2: Program Output**

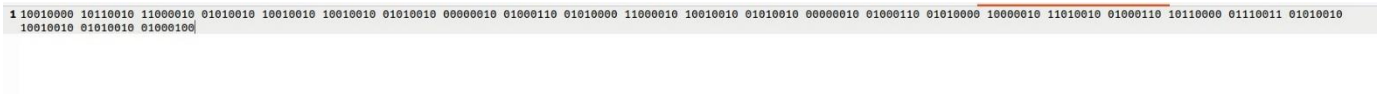


**Figure 1.3: Cipher File**


**Figure 1.4: Plain Text File**

## 2. Multiple Lines


**Figure 2.1: Input File**


**Figure 2.2: Program Output**

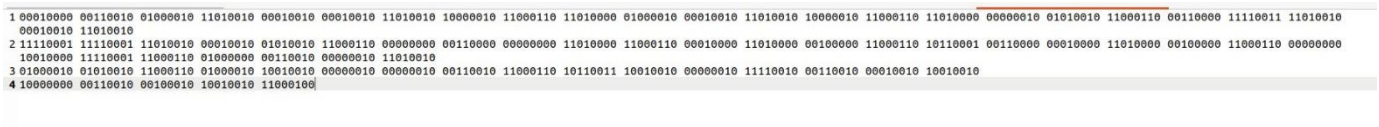


**Figure 2.3: Cipher File**

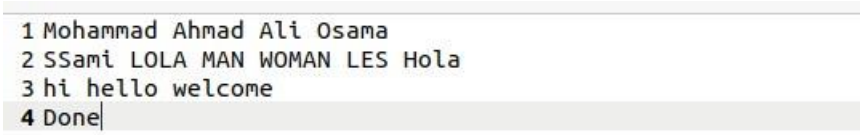


**Figure 2.4: Plaintext File**