**Project Report**

Encryption Algorithm with Modified Techniques

# Code:

INCLUDE irvine32.inc

.data

xamount dword ?

keycorrect byte "Key Correct, data decrypted", 0

keywrong byte "Wrong Key Entered, Redirecting to Main Menu...", 0

keydisplay byte "This is your private key: ",0

keyinput byte "Please enter your private key: ", 0

outputfile byte "decryptedtext.txt", 0

decryptedtext byte 500 dup (?)

output byte"Enter your data for Encryption: ",0

initial byte " \* WELCOME \*",0

op byte " OPTIONS ",0

choice byte "Enter your choice: ",0

choice1 byte "1) Encrypt the message ",0

choice2 byte "2) Decrypt the message ",0

choice3 byte "3) Exit ",0

Encrpyt byte "Encryption function called",0

correct byte" Please enter correctly. ",0

x dword ?

input dword 500 DUP(?)

ciphertext dword 500 DUP(?)

l dword ? ;length

key sdword ?

ans dword ?

str1 byte "Your Decrypted Data: ",0

thanks byte "Thank You ",0

filehandle\_writekey DWORD ?

filenamewritekey BYTE "key.txt", 0

filehandle\_encrypted DWORD ?

filenameencrpt BYTE "encryptedtext.txt",0

buffSize = 500

keyarray dword buffSize DUP(?)

filehandle\_readkey DWORD ?

s2 dword 500 DUP(?)

keyread BYTE buffSize DUP(?)

.code

main proc

start:

mov edx,offset initial

call writestring

mov ecx,2

LL:

call crlf

LOOP LL

mov edx,offset op

call writestring

call crlf

call crlf

mov edx,offset choice1

call writestring

call crlf

mov edx,offset choice2

call writestring

call crlf

mov edx,offset choice3

call writestring

call crlf

mov edx,offset choice

call writestring

mov eax,0

call readint

mov x,eax

cmp x,1

je E

cmp x,2

je D

cmp x,3

je Ex

mov edx,offset correct

call writestring

call crlf

jmp start

E:

call encryption

call readchar

call clrscr

jmp start

D:

call decryption

call crlf

jmp start

Ex:

exit

call dumpregs

main ENDP

encryption proc

mov l,0

call Randomize

mov eax,124

call RandomRange

mov dword ptr key,eax

add eax, 3

mov edx,offset keydisplay

call writestring

call writedec

call crlf

mov edx,offset output

call writestring

mov edx,offset input

mov ecx,sizeof input

call readstring

call lengthfun ;encryption

mov ecx, eax

mov esi,0

E1:

mov eax,0

mov edx,0

add key,3

mov eax,key

mov keyarray[esi\*type keyarray],eax

mov al, byte ptr input[esi]

add eax,key

mov ebx, 128

div ebx

mov eax,edx

add eax, 5

rol eax,9

TEST eax, 10000000000000000000000000000000b

jnz L6

mov ebx, 23567h

SHLD eax, ebx,1

L6:

TEST eax, 01000000000000000000000000000000b

jnz L5

mov ebx, 23567h

SHLD eax, ebx,1

L5:

TEST eax, 00100000000000000000000000000000b

jnz L4

mov ebx, 23567h

SHLD eax, ebx,1

L4:

rol eax, 2

TEST eax, 1h

jnz L3

add eax, 0

RCR eax, 1

L3:

sub eax, 2

ror eax, 1

TEST eax,10000000000000000000000000000000b

jnz L2

add eax, 0

RCL eax, 1

L2:

TEST eax,10000000000000000000000000000000b

jnz L1

SHL eax, 1

L1:

TEST eax,10000000000000000000000000000000b

jnz L7

SAL eax, 1

L7:

ror eax,2

add eax,89

mov ciphertext[esi\*type ciphertext], eax

inc esi

dec ecx

jnz E1

mov edi, offset ciphertext

mov esi, offset ciphertext

mov ebx, 2

invoke str\_length, addr ciphertext

mov ecx, eax

LL:

cld

lodsd

mul ebx

cld

stosd

Loop LL

mov edi, offset ciphertext

mov esi, offset ciphertext

call lengthfun

mov ecx, l

mov eax, 12

call randomrange

mov xamount, eax

f2:

cld

lodsd

xor eax, xamount

stosd

Loop f2

mov ecx,eax

mov esi,0

call crlf

call lengthfun

mov ebx,eax

mov edx, offset filenamewritekey

call CreateOutputFile

mov filehandle\_writekey, eax

mov eax, filehandle\_writekey

mov edx, OFFSET keyarray

;lengthof array in ebx

mov ecx,ebx

call WriteToFile

mov eax, filehandle\_writekey

call CloseFile

;this is for filing of encryted data

call lengthfun

mov ebx,eax

mov edx, offset filenameencrpt

call CreateOutputFile

mov filehandle\_encrypted, eax

mov eax, filehandle\_encrypted

mov edx, OFFSET ciphertext

;lengthof array in ebx

mov ecx,ebx

call WriteToFile

mov eax, filehandle\_writekey

call CloseFile

ret

encryption endp

decryption proc

mov eax, 0

mov edx,offset keyinput

call writestring

mov eax, keyarray[0]

mov key, eax

call readint

cmp eax, key

jz testkey

mov edx, offset keywrong

call writestring

call crlf

jnz notapprove

mov edx,offset correct

call writestring

call crlf

testkey:

mov edx, offset keycorrect

call writestring

call crlf

mov edx,offset str1

call writestring

approve:

mov edi, offset ciphertext

mov esi, offset ciphertext

call lengthfun

mov ecx, l

Lx:

cld

lodsd

xor eax, xamount

stosd

Loop Lx

mov edi, offset ciphertext

mov esi, offset ciphertext

mov ebx, 2

invoke str\_length, addr ciphertext

mov ecx, eax

f:

cld

lodsd

mov edx, 0

div ebx

cld

stosd

Loop f

mov eax,0

mov ebx,0

mov ecx,0

mov edx,0

mov edx,OFFSET filenamewritekey

call OpenInputFile

mov filehandle\_readkey, EAX

mov edx, OFFSET keyread

mov ecx, buffSize

call ReadFromFile

call lengthfun

mov ecx,eax

mov esi,0

stc

sub eax, 0

D:

mov eax, ciphertext[esi\*type ciphertext]

sub eax,89

rol eax,2

jz L11

SAR eax, 1

;call dumpregs

L11:

jz L66

SHR eax, 1

;call dumpregs

L66:

jc L22

RCR eax, 1

;call dumpregs

L22:

rol eax, 1

add eax, 2

;call dumpregs

jc L33

RCL eax, 1

;call dumpregs

L33:

ror eax, 2

;call dumpregs

jc L44

mov ebx, 0

SHRD eax, ebx, 3

L44:

ror eax,9

sub eax, 5

mov ebx,dword ptr keyarray[esi\*type keyarray]

mov dword ptr key, ebx

sub eax,key

mov edx, 0

mov ebx, 128

div ebx

mov eax,edx

mov decryptedtext[esi\*type decryptedtext], al

call writechar

inc esi

LOOP D

call crlf

call lengthfun

mov ebx, eax

mov edx,offset outputfile

call createoutputfile

push eax

mov ecx, ebx

mov edx,offset decryptedtext

call writetofile

pop eax

call closefile

notapprove:

call readchar

call clrscr

ret

decryption endp

lengthfun PROC

INVOKE Str\_length,ADDR input

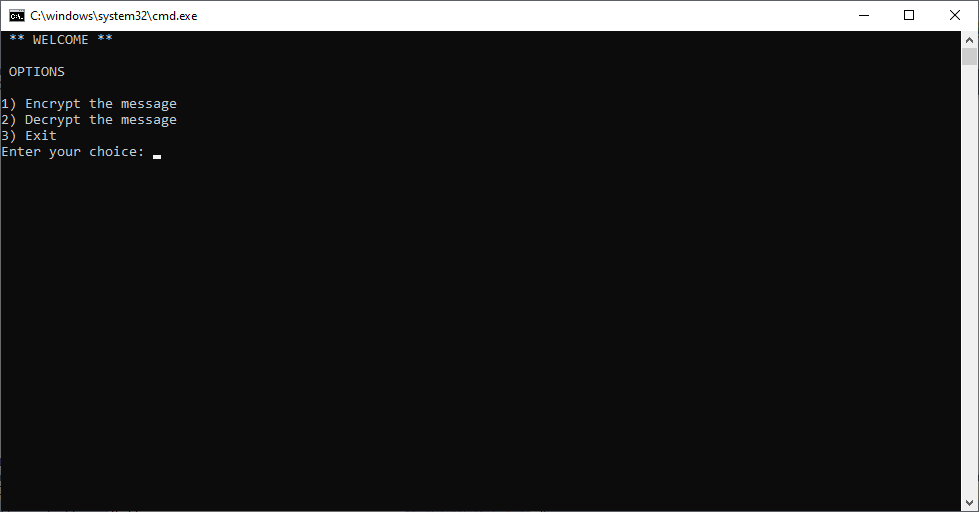
mov l,eax

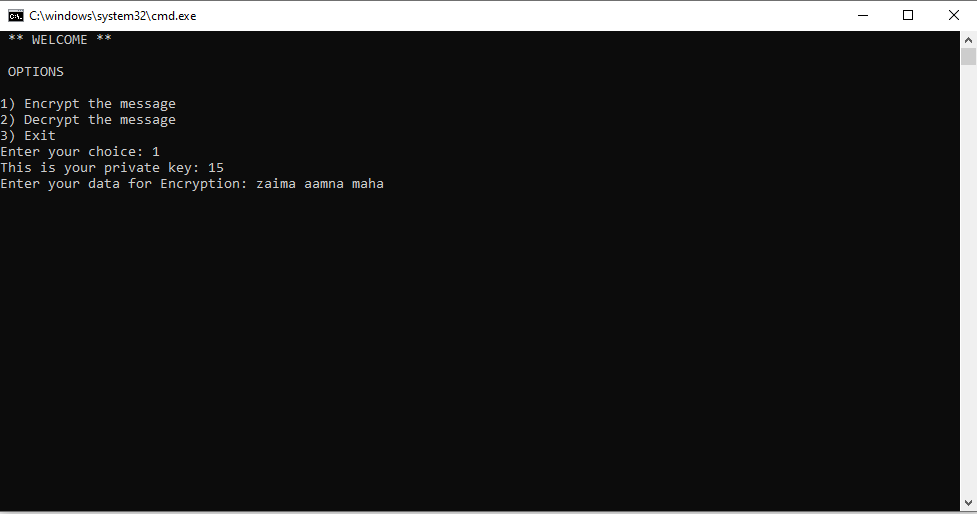
ret

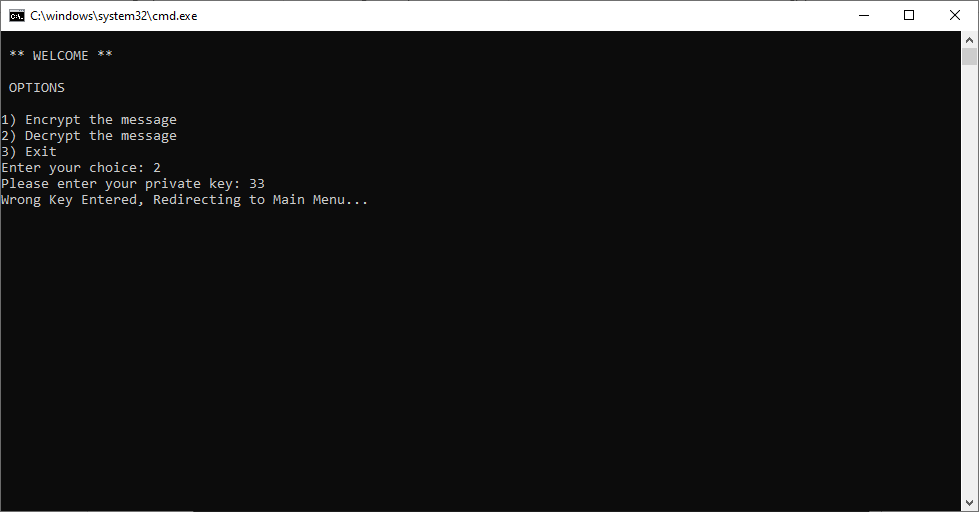
lengthfun ENDP

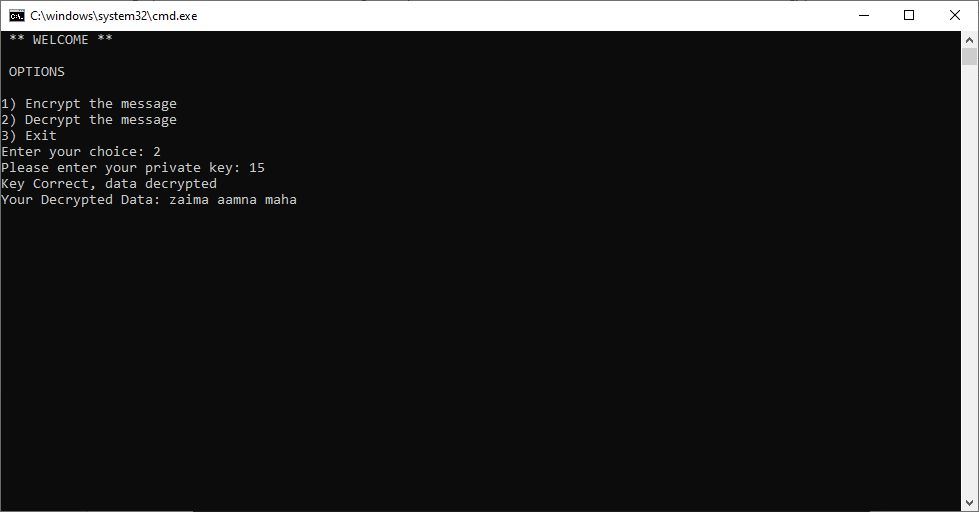
end main

# Output:









**Discussion:**

1. **Objectives:**

Create an efficient encryption algorithm based on a modified version of the Caesar cipher and logical operations.

1. **Implementation:**

**Encrytpion/Decryption:**

We used a randomly generated key for the Caesar cipher, we incremented (changed) the value of the key for each unit of the input being encrypted. We used rotates and shifts with bit-testing, as well as xor’d the data. We also used string primitives for convenience (lodsd and stosd) along with str\_length.

**Filing:**

We stored the encrypted data into a file. After being generated, the key is stored in a file and later read from it to decrypt data.

1. **Improvements:**

We discussed during the demo that extending the key would make it harder to break if a third person tried to decipher the text.