



Encryption & Decryption



Report on File Encryption and Decryption

Individual project

Module Name: ST4061CEM Programming and Algorithms 1

BSc. (Hons) Ethical Hacking and Cyber Security,

Softwarica college of IT and E-commerce Coventry University

Intake:

September 2023

Submitted by:

Name: Shrijesh Pokharel

CUID: 14806685

SUID: 230329

Submitted to:

Name: Suman Shrestha

Acknowledgement

I am grateful to Softwarica College for giving me the amazing opportunity to work on my project report, which focuses on developing a C programming tool for file cryptography. Express my gratitude and appreciation to my module teacher for kindly lending their time, offering invaluable support, and offering insightful suggestions to boost the caliber of my course. In closing, let me express my gratitude to my friends for their program recommendations and assistance in improving my report.

Abstract

This C program shows the Caesar Cipher method that works for both encryption and decryption. It is a kind of file processing utility. After the user enters the names of the input and output files, the software either encrypts or decrypts the input file's contents, depending on what they choose. The “processFile” function, which handles opening the files, applying the given offset to each character, and writing the processed information to the output file, encapsulates the essential functionality.

The user-friendly interface allows users to choose between encryption and decryption, as well as produce a new output file for the processed material. This illustrates the fundamental implementation of the Caesar Cipher. The offset utilized in the encryption and decryption operations is a fixed value (100 for encryption and -100 for decryption). Error handling is included into the application to alert users to any problems with file operations.

For instructional reasons, this code acts as a fundamental example, showing how to handle files, input from the user, and apply a basic encryption/decryption technique. Because of its modular design, which facilitates comprehension, it is appropriate for beginning programming classes.

Contents

Acknowledgement	2
Abstract.....	3
I. Introduction	5
1) Background	5
II. Strengths and Weaknesses.....	5
III. Objectives	6
IV. Literature review:	6
V. Methodology	7
1) Algorithm	7
2) Procedure.....	8
a) Libraries	8
b) File handling	8
c) Statement.....	8
d) Interface	9
e) Case	9
f) Main function	10
g) processFile function	11
3) Output.....	12
VI. Future improvements:.....	14
VII. Conclusion	14
VIII. Reference.....	14

I. Introduction

1) Background

This report provides a detailed overview of the processes involved in encrypting and decrypting using the Caesar Cipher. It explores the methodology, implementation, and applications of this classic cryptographic algorithm, while discussing its strengths and weaknesses. During wars Caesar cipher was used by Julius Caesar of Rome to keep confidentiality. This encryption technique involves shifting each letter in the plaintext by a fixed number of positions up or down the alphabet. The recipient can decrypt the message by reversing the process and shifting the letters back to their original positions.

II. Strengths and Weaknesses

On the plus side, one noticeable benefit is its simplicity. Furthermore, the Caesar Cipher is renowned for its quickness. Both of these are computationally fast, allowing for faster coding and decoding of messages. Furthermore, the Caesar Cipher is easily implementable in a variety of computer languages, making it versatile and adaptable to diverse software systems.

However, the Caesar Cipher does have several serious flaws that should be considered. One of its main weakness is its vulnerable to brute-force assaults. Because the key space is very tiny, with just 26 potential keys when using the English alphabet, an attacker can try all of them until the proper one is discovered. This renders the Caesar Cipher vulnerable to determined opponents with significant processing power. Another problem is the Caesar Cipher's lack of security in comparison to contemporary standards. The original message can be revealed by using cryptanalysis techniques like frequency analysis to crack the encryption. Because of this, the Caesar Cipher is no longer suitable for encrypting sensitive data in the modern world, since more robust techniques are easily accessible. Finally, only a certain number of key values can be supported by the Caesar Cipher due to its constrained key space. Due to this restriction, there are fewer encryption versions accessible, which lowers the security and protection level that the encryption can provide overall.

III. Objectives

The main goal of cryptography is to keep security, integrity, authenticity of data while managing the keys. By accomplishing these goals, cryptography contributes to the establishment of secure communication and transactions in a number of areas, such as secure electronic transactions, secure data storage, and data transfer via networks.

Using cryptography one can be safe from unwanted access and interpretation. In order to prevent unwanted changes in data cryptography uses method like digital signature and message authentication code. Digital signature, crypto key, certificate to validate the identity of person or entitled involve in transaction or communication, by doing this trust with user is created.

IV. Literature review:

The topic of file encryption and decryption has made great progress, with several studies looking into various elements of cryptography approaches and implementation methodologies. Several research have focused on establishing strong algorithms and frameworks for protecting digital data. Notable works in the field of file encryption and decryption use a wide range of programming languages, including Java and Python.

1. Bouncy Castle Library: A well-known Java cryptographic library that offers thorough support for a wide range of encryption techniques. It provides a number of cryptographic functions, such as file encryption and decryption, to protect sensitive data's integrity and secrecy.
2. Java Cryptography Extension(JCE) a vital part of java platform makes it easier to apply encryption and decryption methods allowing programmers to perform safe solution for files encryption and decryption. It add reliability of cryptography by supporting wide range of algorithm.
3. Cryptography Module This Python library gives programmers access to high-position cryptographic savages so they may include encryption and decryption features into their operations. Its straightforward and secure armature enhances the responsibility of train protection.
4. cryptography Module This Python library gives programmers access to high- position cryptographic savages so they may include encryption and decryption features into their operations. Its straightforward and secure armature enhances the responsibility of train protection.

V. Methodology

1) Algorithm

1. Start
2. Create an function to read the file
3. Open file input in read
4. Open file output in write mode
5. Read and write character
6. Close file
7. Main function
8. Take input from above function
9. Use switch to choose between 1 or 2
10. Use if statement to encrypt and decrypt
11. end

2) Procedure

a) Libraries

```
#include <stdio.h>
```

The code begins by including the necessary header file `stdio.h`, which provides input-output functions.

b) File handling

```
FILE* inputFile = fopen(inputFileName, "r");  
FILE* outputFile = fopen(outputFileName, "w");
```

This lines of code helps to run the file located in device and let c language modify and manipulate them.

c) Statement

```
if (inputFile == NULL || outputFile == NULL)  
{  
    printf("Error opening files!\n");  
    return;  
}  
int ch;  
while ((ch = fgetc(inputFile)) != EOF)  
{  
    ch += offset;  
    fputc(ch, outputFile);  
}  
fclose(inputFile);  
fclose(outputFile);  
printf("%s' success\n", inputFileName);
```

These line of code give an instruction to show error if the file name or file is empty and manipulate the file if it is correct and display success.

d) Interface

```
char inputFileName[30];
char outputFileName[30];
int choose;
printf("Enter the file to open: ");
scanf("%s", inputFileName);
printf("Enter the file to create: ");
scanf("%s", outputFileName);
printf("\nChoose an option:\n");
printf("1. Encryption.\n");
printf("2. Decryption.\n");
printf(": ");
scanf("%d", &choose);
```

These line help in showing the user to insert the file name in order to run the above command.

e) Case

```
switch (choose)
{
    case 1:
        processFile(inputFileName, outputFileName, 100);
        break;
    case 2:
        processFile(inputFileName, outputFileName, -100);
        break;
    default:
        printf("Errors!!!\n");
        break;
}
```

The user chooses from this collection of code whether to encrypt or decrypt.

f) Main function

```
int main()
{
    char inputFileName[30];
    char outputFileName[30];
    int choose;
    printf("Enter the file to open: ");
    scanf("%s", inputFileName);
    printf("Enter the file to create: ");
    scanf("%s", outputFileName);
    printf("\nChoose an option:\n");
    printf("1. Encryption.\n");
    printf("2. Decryption.\n");
    printf(": ");
    scanf("%d", &choose);
    switch (choose)
    {
        case 1:
            processFile(inputFileName, outputFileName, 100);
            break;
        case 2:
            processFile(inputFileName, outputFileName, -100);
            break;
        default:
            printf("Errors!!!\n");
            break;
    }
    return 0;
}
```

The main() function is defined as the entry point of the program. It declares variables i and x of type int and an array str of type char to store the input string. After obtaining the string input, the program presents a menu to the user, displaying two options: encryption and decryption. The user is expected to enter the corresponding option by inputting either 1 or 2. Scanf() function is used to store the input given by the user. switch statement is used than to perform either encrypt or decrypt. If x is equal to 1 (encryption), the code enters the corresponding case 1: block. For each character, it adds 3 to its ASCII value, effectively shifting it three positions up the alphabet.

If x is equal to 2 (decryption), the code enters the case 2: block. It performs a similar loop through each character of the input string. However, in this case, it subtracts 3 from the ASCII value of each character, shifting it three positions back down the alphabet. This process decrypts the string. The resulting decrypted string is then printed using printf().

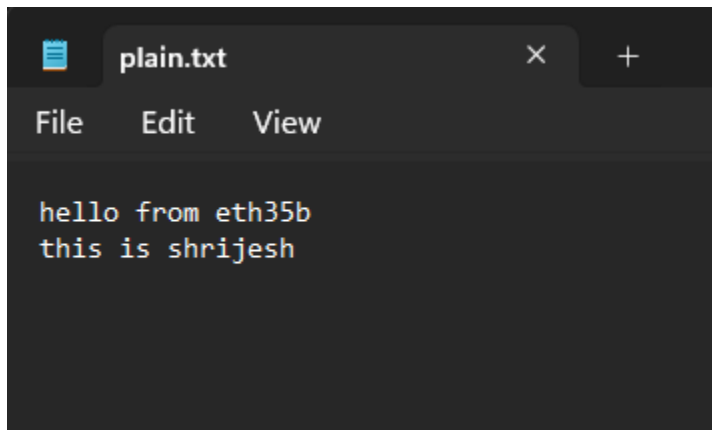
g) processFile function

```
void File(const char* inputFileName, const char* outputFileName, int offset)
{
    FILE* inputFile = fopen(inputFileName, "r");
    FILE* outputFile = fopen(outputFileName, "w");
    if (inputFile == NULL || outputFile == NULL)
    {
        printf("Error opening files!\n");
        return;
    }
    int ch;
    while ((ch = fgetc(inputFile)) != EOF)
    {
        ch += offset;
        fputc(ch, outputFile);
    }
    fclose(inputFile);
    fclose(outputFile);
    printf("'%'s' success\n", inputFileName);
}
```

If the user enters any value other than 1 or 2, the code enters the default: block and prints an error message using printf().

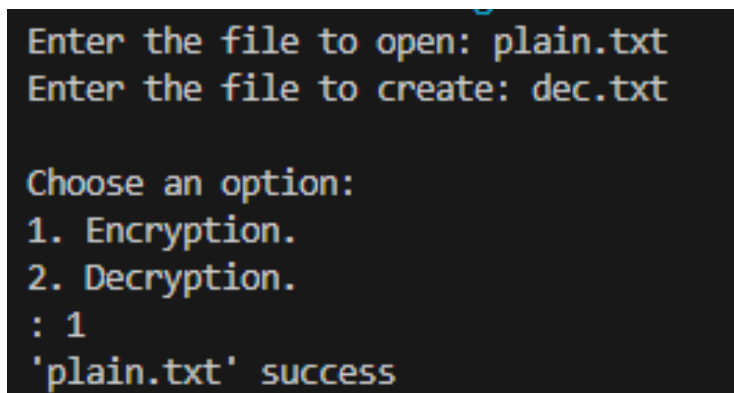
Finally, the return 0; statement ends the main() function and signifies successful program execution.

3) Output



```
plain.txt
File Edit View
hello from eth35b
this is shrijesh
```

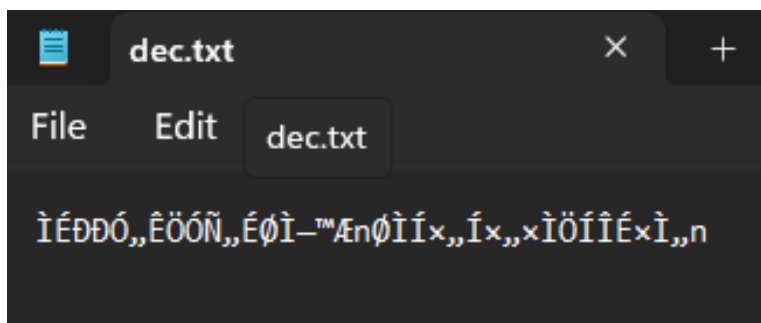
This is the plain text with text to be encrypted



```
Enter the file to open: plain.txt
Enter the file to create: dec.txt

Choose an option:
1. Encryption.
2. Decryption.
: 1
'plain.txt' success
```

To encrypt the text from plain.txt to dec.txt we selected 1st option (encryption)



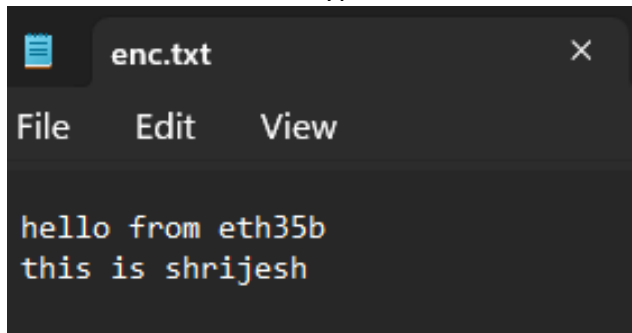
```
dec.txt
File Edit dec.txt
ÏÉÐÐÓ,,ÊÖÖÑ,,ÉØÏ-™ÆnØÏÍx,,Íx,,xÏÖÍÊxÏ,,n
```

Now again to decrypt this file we can run same command

```
Enter the file to open: dec.txt
Enter the file to create: enc.txt

Choose an option:
1. Encryption.
2. Decryption.
: 2
'dec.txt' success
```

This shows that the encrypted content in the file dec.txt has been decrypted into enc.txt.

A screenshot of a text editor window titled 'enc.txt'. The window has a dark theme and a menu bar with 'File', 'Edit', and 'View' options. The text content of the file is displayed in a monospaced font: 'hello from eth35b' on the first line and 'this is shrijesh' on the second line.

```
enc.txt
File Edit View
hello from eth35b
this is shrijesh
```

This is the decrypted file that we have created.

VI. Future improvements:

While this is an encryption decryption application, it is really easy. The program has several potential areas for growth. Future enhancements and features for this software include:

- a) Supports several encryption techniques.
- b) Graphical User Interface
- iii. Error handling
- c) Security upgrades (use of private and public keys)
- d) Performance optimization

VII. Conclusion

In conclusion, the Caesar Cipher is an encryption technique with historical significance that paved the way for modern cryptography. While it lacks the strength and security required for contemporary applications, it remains a valuable tool for educational purposes and basic data protection. The Caesar Cipher algorithm's perpetration in the C programming language exemplifies how it might be used in practice. As technology continues to evolve, it is crucial to understand the limitations of classical encryption methods like the Caesar Cipher and explore more robust alternatives for secure communication.

VIII. Reference

1. Admin. (n.d.). *Caesar cipher for binary data*. lora-grig.ru.
<https://lora-grig.ru/using-the-caesar-cipher-to-encrypt-and-decrypt-binary-data/>
2. *Basics of file handling in C*. (2023, October 19). GeeksforGeeks.
<https://www.geeksforgeeks.org/basics-file-handling-c/>
3. *C files - File handling and how to create files*. (n.d.). W3Schools Online Web Tutorials.
https://www.w3schools.com/c/c_files.php
4. *C program to encrypt and decrypt files*. (n.d.). CodesCracker: Learn Online Coding.
<https://codescracker.com/c/program/c-program-encrypt-file.htm>
5. *C program to encrypt and decrypt the string (Source code)*. (n.d.). Online tutorials for c programming, cplusplus, Java, Python.
<https://www.trytoprogram.com/c-examples/c-program-to-encrypt-and-decrypt-string/>
6. *Coding ninjas studio*. (n.d.). Coding Ninjas.
<https://www.codingninjas.com/studio/library/caesar-cipher>
7. *Encryption and decryption in C*. (n.d.). Code Review Stack Exchange.
<https://codereview.stackexchange.com/questions/275332/encryption-and-decryption-in-c>