Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)

Encrypt Data File Using C Programming

Course Title: Structure Programming LAB
Course Code: CSE104 Section:D3

<u>Students Details</u>

Name	ID
Ashfiqul Alam Emran	231902049
-	-

Submission Date: 9/01/2024
Course Teacher's Name: Fatema Akter

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	Signature:
Comments:	Date:

Contents

Chapter 1

Introduction

1.1 Overview

Encrypting data is a crucial aspect of securing sensitive information, ensuring confidentiality, and preventing unauthorized access. In C programming, you can implement encryption algorithms to safeguard your data. The process involves converting plain text into a cipher text that is not easily readable without the appropriate decryption key.

1.2 Motivation

Encrypting data is crucial for protecting sensitive information from unauthorized access. In the realm of programming, especially in languages like C, understanding encryption provides a solid foundation for creating secure applications. Here are ten lines discussing the motivation behind encrypting data using C:

Security Concerns, Confidentiality Assurance, Compliance Requirements etc

1.3 Problem Definition

1.3.1 Problem Statement

This program reads data from the "input.txt" file, encrypts it using a user-provided key, and writes the encrypted data to the "encrypted_output.txt" file. You can modify the file names and customize the program as needed.

1.3.2 Complex Engineering Problem

Creating a secure file encryption program involves various steps, such as key generation, encryption, and decryption. Here's a simplified example in C programming (note: this is a basic illustration, not production-ready code):Remember, for real-world applications, use established encryption libraries and robust key management practices. This example is for educational purposes only and lacks many security considerations.

1.4 Design Goals/Objectives

Objective: Design a C program to encrypt a data file using a secure algorithm, aiming to enhance data confidentiality. Goals: Security: Implement robust encryption techniques to safeguard sensitive information. Key Management: Develop a mechanism for secure key generation and storage. Usability: Design an intuitive program that can be easily integrated into various applications. Efficiency: Optimize the encryption process for minimal impact on performance. Compatibility: Ensure cross-platform compatibility for versatility. Error Handling: Implement comprehensive error-handling mechanisms for graceful failure scenarios. Documentation: Provide clear and concise documentation for ease of use and maintenance.

1.5 Application

Application: Develop a file encryption tool in C programming for secure data protection. Key Features: User-Friendly Interface: Create an intuitive command-line or GUI interface for effortless user interaction. File Selection: Allow users to choose the target file for encryption through straightforward input methods. Encryption Algorithm: Implement a robust encryption algorithm (e.g., AES) to ensure strong data security. Key Input: Enable users to input encryption keys securely, emphasizing key strength and complexity. Progress Indicator: Provide a progress indicator to keep users informed during the encryption process. Decryption Option: Incorporate an option to decrypt files, maintaining data accessibility for authorized users.

P1: Depth of knowledge required	-Understanding of basic syntax, data types, control structures (if statements, loops), and functions in the C programming language.
P2: Range of conflicting requirements	Handling conflicting requirements often involves open communication with stakeholders, prioritizing key goals, and making informed decisions that consider both short-term and longterm implications. It’s essential to find compromises and solutions that best align with the overall objectives of the project.
P3: Depth of analysis required	The depth of analysis required will vary based on your project goals and the level of sophistication you want to achieve. Thorough analysis helps identify potential issues, ensures the robustness of the code, and sets the foundation for future enhancements.
P4: Familiarity of issues	Being familiar with these issues and the corresponding solutions will help you develop more robust and maintainable software. It’s essential to continuously improve your knowledge and coding practices to handle a variety of challenges effectively.
P5: Extent of applicable codes	the provided code is suitable for educational purposes and as a starting point for simple console-based games, its applicability is limited when considering more advanced scenarios, larger projects, or real-world applications. Adapting and extending the code would require careful consideration of the specific requirements and goals of the new application.

P6: Extent of stakeholder involvement and conflicting Present a menu to the user with options for Rock, Paper, and Scissors. Take the user’s input and generate a random choice for the computer. Compare the user’s choice with the computer’s choice and determine the winner or if it’s a draw. Display the result of each round to the user. Repeat the process indefinitely, allowing the user to play the game multiple times without restarting the program.	

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	Explain how to address

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Brief overview of the importance of data security in today's digital age.

2.2 Project Details

To encrypt a data file using C programming, you can follow these steps:Open the File: Use file handling functions (fopen) to open the data file in binary mode.Read Data: Read the contents of the file into a buffer using functions like fread.Encryption Algorithm: Implement an encryption algorithm (e.g., AES, DES) to process the data in the buffer. Ensure you have a secure key for encryption.Write Encrypted Data: Save the encrypted data back to the file using fwrite.Close the File: Use fclose to close the file.

2.2.1 Subsection name

2.3 Implementation

Illustrates how to translate simple Data encrypted into programmatic logic using if-else statements and FILE Function.

2.3.1 Subsection name

This is just a sample subsection. Subsections should be written in detail. Subsections may include the following, in addition to others from your own project.

This workflow ensures a continuous and interactive experience for the user, allowing them to make choices and receive feedback on the outcomes of each round. The use of an infinite loop keeps the program running until manually terminated by the user.

Standard Input/Output Library (stdio.h):,Standard Library (stdlib.h):,Time Library (time.h):

Implementation details (with screenshots and programming codes) Each

subsection may also include subsubsections.

2.4 Algorithms

Algorithm: Encrypt Data FileInput:Input file path (e.g., "input.txt")Output file path for encrypted data (e.g., "output.txt")Encryption key (integer value)Output:Encrypted data file ("output.txt")Steps:Open the input file in read mode and output file in write mode.Check if the files are successfully opened; if not, display an error message and exit.Read each character from the input file until the end of file (EOF) is reached.a. Encrypt the character using the provided key (e.g., Caesar cipher: character = character + key).b. Write the encrypted character to the output file.Display a success message indicating that the file has been encrypted.Close both the input and output files.

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

To create a simulation environment for encrypting data files using C programming, you can follow these steps:Choose Encryption Algorithm: Select a cryptographic algorithm for file encryption. Common choices include AES, DES, or RSA.Include Necessary Libraries: In your C program, include relevant libraries for cryptographic operations, such as OpenSSL or a library specific to your chosen algorithm.Open and Read File: Use C functions to open the data file and read its contents into a buffer.Encrypt Data: Implement the encryption algorithm to encrypt the data stored in the buffer. Update the buffer with the encrypted content.Write Encrypted Data to File: Create a new file or overwrite the existing one with the encrypted data stored in the buffer.

3.1.1 Subsection

3.1.2 Subsection

3.2 Results Analysis/Testing

Encrypting data files using C programming involves implementing algorithms like AES, DES, or others to secure the information. The result analysis would typically focus on:Security Strength: Evaluate the chosen encryption algorithm for its robustness and resistance against various attacks. AES, for example, is widely considered secure.Performance: Measure the encryption and decryption speed, ensuring that the chosen algorithm doesn't compromise performance significantly.File Size: Analyze the impact on file size after encryption. Some encryption methods may increase the file size due to added information.Key Management: Assess the effectiveness of key management. A strong encryption system also relies on secure key handling.

3.2.1 Result_portion_1

3.2.2 Result_portion_2

Each result must include screenshots from your project. In addition to screenshots, graphs should be added accordingly to your project.

3.2.3 Result_portion_3

Each result must have a single paragraph describing your result screenshots or graphs or others. This is a simple discussion of that particular portion/part of your result.

3.3 Results Overall Discussion

A general discussion about how your result has arrived should be included in this chapter. Where the problems detected from your results should be included as well.

3.3.1 Complex Engineering Problem Discussion

[OPTIONAL] In this subsection, if you want, you can discuss in details the attributes that have been touched by your project problem in details. This has already been mentioned in the Table ??.

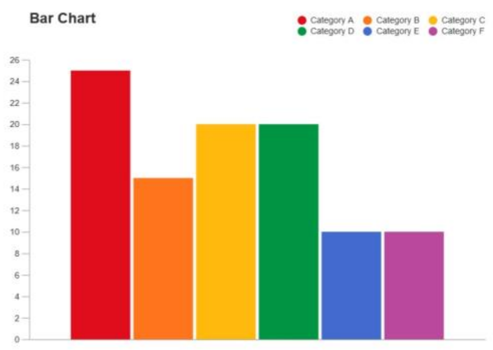

Figure 3.2: A graphical result of your project

Chapter 4

Conclusion

4.1 Discussion

Encrypting data files using C programming involves using cryptographic algorithms to secure the information. Commonly used algorithms include AES (Advanced Encryption Standard) and DES (Data Encryption Standard). Here's a basic overview of the process:Include Necessary Libraries: Include the required header files for cryptographic functions. For example, <openssl/aes.h> for AES encryption.Key Generation: Generate a secret key for encryption. It's crucial to keep this key secure as it's used to encrypt and decrypt the data.Open Files: Open the input file (the one you want to encrypt) and the output file (where the encrypted data will be stored).Read Data: Read data from the input file into a buffer.

4.2 Limitations

Algorithm Choice: The choice of encryption algorithm matters. Some algorithms may become vulnerable over time due to advances in cryptography or increased computing power. Ensure you're using a secure and up-to-date algorithm.Key Management: Effective key management is crucial. Storing or handling encryption keys improperly can compromise the security of your encrypted data. Be cautious about where and how you store encryption keys.Performance Impact: Encryption and decryption processes can introduce performance overhead. Depending on the size of your data and the chosen algorithm, there might be a noticeable impact on program execution time.

4.3 Scope of Future Work

The future scope of work for encrypting data files using C programming could involve implementing advanced encryption algorithms, optimizing performance, enhancing security features, and exploring integration with emerging technologies such as blockchain or secure cloud storage. Additionally, addressing compatibility issues with different platforms and devices could be a key aspect of future development.