TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Project Report

On

"FILE ENCRYPTION AND DECRYPTION"

Submitted By:

Amrita Regmi (THA081BEI005)

Arjun Paudel (THA081BEI007)

Shirish Dhami (THA081BEI040)

Swastika Kharel (THA081BEI047)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

Under the Supervision of

Prajwal Pakka

March, 2025

DECLARATION

We hereby declare that the report of the project entitled **"File Encryption and Decryption"** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Electronics and Communication Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Amrita Regmi(Class Roll No: 081/BEI/005) _____

Arjun Paudel (Class Roll No: 081/BEI/007) _____

Shirish Dhami(Class Roll No: 081/BEI/040) _____

Swastika Kharel(Class Roll No: 081/BEI/047) _____

Date: March, 2025

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled "**File Encryption and Decryption**" submitted by **Amrita Regmi, Arjun Paudel, Shirish Dhami** and **Swastika Kharel** in partial fulfillment for the award of Bachelor's Degree in Electronics and Communication Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Electronics and Communication Engineering.

Project Supervisor
Prajwal Pakka
Department of Electronics and Computer Engineering, Thapathali Campus

Er. Umesh Kanta Ghimire
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

March, 2025

COPYRIGHT

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who supported and guided me throughout the development of the "**File Encryption and Decryption in C**" project.

First and foremost, I would like to thank my project supervisor sir **Prajwal Pakka** for their constant guidance, insightful feedback, and encouragement throughout the course of this project. Their expertise and support were invaluable in shaping the direction of this work.

I would also like to acknowledge the contributions of my colleagues and friends, who provided assistance and offered constructive suggestions during the development process. Their collaborative spirit and technical support were crucial in overcoming challenges faced during implementation.

Special thanks to the **Department of Electronics and Computer Engineering, IOE Campus, Thapathali** for providing the necessary resources and an excellent learning environment, which enabled me to complete this project successfully.

This project would not have been possible without all of your contributions. Thank you.

Amrita Regmi(Class Roll No: 081/BEI/005)
Arjun Paudel (Class Roll No: 081/BEI/007)
Shirish Dhami(Class Roll No: 081/BEI/040
Swastika Kharel(Class Roll No: 081/BEI/047)

ABSTRACT

Data security represents a significant challenge in the contemporary digital landscape, where sensitive data is routinely shared and stored. This initiative aims to develop a file encryption and decryption system utilizing the C programming language to safeguard data confidentiality and integrity. The system incorporates cryptographic methods to transform plain text files into an encrypted state, rendering them inaccessible without the appropriate decryption key. The Encryption phase employs algorithms such as AES (Advanced Encryption Standard) or XOR-based encryption, while the decryption phase reverses this process to retrieve the original file. The implementation encompasses key generation, secure storage solutions, and efficient encryption and decryption functions. The anticipated result is a lightweight, rapid, and secure tool designed to protect files from unauthorized access. This project plays a vital role in bolstering data security for both personal and organizational applications, showcasing the practical use of cryptographic principles within C programming.

Keywords:Data security,encryption and decryption,cryptographic algorithms

Table of Contents

List of Figures and Tables

List of Abbreviations

AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman
XOR	Exclusive OR
SPN	Substitution-Permutation Network
SSL	Secure Sockets Layer
TLS	Transport Layer System
CLI	Command-Line Interface
GUI	Graphical User Interface
GCC	GNU Compiler Collection
OS	Operating System
DES	Data Encryption Standard

1. INTRODUCTION

This project, *File Encryption and Decryption in C*, focuses on securing sensitive data by implementing cryptographic techniques. The program allows users to encrypt a file, converting its content into an unreadable format, and later decrypt it back to its original state using a predefined key. Utilizing standard C libraries and efficient algorithms, this project enhances data security by preventing unauthorized access. It serves as a practical demonstration of fundamental encryption principles and their application in real-world scenarios.

1.1 Background

The exponential increase in digital data necessitates robust security measures. Sensitive data, including personal, financial, and corporate information, is routinely stored and transmitted electronically. This prevalence exposes such data to significant risks from cyber threats like unauthorized access, breaches, and hacking. Encryption, a common technique for transforming readable data into an unreadable format decipherable only with a specific key, offers a solution by restricting access to authorized users and preventing data manipulation. Algorithms like AES, RSA, and XORprovidevarying levels of security and performance. This project develops a lightweight and efficient file encryption and decryption system in C as a means of bolstering data security and mitigating cyber threats. It allows users to secure files so decryption and access requires the correct key and will offer a practical application.

1.2 Motivation

Completing this project provides a strong foundation in cryptography and secure coding practices. It enhances problem-solving skills, deepens knowledge of encryption algorithms, and opens doors to cybersecurity careers, research, and advanced encryption projects. This experience can lead to opportunities in ethical hacking, secure software development, and open-source contributions. Additionally, it strengthens technical skills in C programming, file handling, and algorithm implementation, making it valuable for both academic and professional growth. Ultimately, this project serves as a stepping stone for further exploration in cybersecurity, advanced encryption techniques, and real-world security applications.

1.3 Problem Definition

This project aims to develop a file encryption and decryption systemusingtheCprogramming language, ensuring data confidentiality and integrity.

1.4 Project Objectives

The main objectives of our project are listed below:

- To implement a secure and efficient file encryption and decryption system using C and simple command-line interface.
- To use cryptographic techniques such as AES (Advanced Encryption Standard), XOR-based encryption, or RSA for securing files.

1.5 Project Scope and Applications

This project focuses on implementing file encryption and decryption using C programming to enhance data security. It covers fundamental cryptographic techniques, ensuring secure file storage and transmission. The application extends to protecting confidential information, preventing unauthorized access, and securing data exchange in personal and professional environments.

1.6 Report Organization

This project report is structured as follows:

1. **Introduction** – Provides an overview of the project, including its objectives, scope, and significance.
2. **Literature Review** – Discusses relevant encryption techniques and existing methods for securing data.
3. **Methodology** – Explains the encryption and decryption approach, algorithm selection, and implementation details in C.
4. **Implementation** – Covers the coding structure, key functionalities, and working of the program.
5. **Results and Analysis** – Presents test cases, output analysis, and performance evaluation.

6. **Conclusion and Future Scope** – Summarizes key findings and suggests possible improvements or future enhancements.
7. **References** – Lists sources and materials consulted for research and development.

2. LITERATURE REVIEW

Encryption and Decryption are the fundamental aspects of cyber security, ensuring data confidentiality and integrity. Various encryption techniques have been implemented in different programming languages including C, which provides low-level control over data manipulation and file handling. The research and development of encryption algorithms have significantly contributed to securing sensitive information in diverse fields, including banking, communications and digital forensics.

Existing works on Encryption and Decryption

Several encryption algorithms and methodologies have been developed to provide secure data transmission and storage. The following sections discuss some of the prominent encryption techniques and their implementations

2.1 Caesar Cipher Implementation in C

The Caesar Cipher is one of the earliest encryption techniques, relying on shifting characters in the plain text by a fixed number of positions in the alphabet. It is implemented in C by reading a file character by character, shifting each letter forward (for encryption) or backward (for decryption), and writing the modified text into another file. This approach utilizes basic ASCII operations.

2.2 Advanced Encryption Standard(AES) in C

AES is a widely used symmetric encryption standard that provides high security andis adopted by the U.S. government for secure communications. AES in C is implemented using a substitution-permutation network (SPN) that processes data in fixed-size blocks (128-bit). The algorithm involves key expansion, initial rounds of substitution, shift rows, mix columns, and final key addition].

2.3 RSA encryption in C

RSA (Rivest-Shamir-Adleman) is an asymmetric encryption algorithm widely used for secure data transmission. RSA in C is implemented using modular exponentiation, where encryption and decryption are performed using a public-private key pair. Large prime number generation and key pair computation are critical steps in the implementation.

3. REQUIREMENT ANALYSIS

The *File Encryption and Decryption in C* project requires a well-defined set of functional and non-functional requirements to ensure secure and efficient data protection.

3.1. Functional Requirements

- Ability to encrypt a file using a specified key.
- Ability to decrypt the encrypted file back to its original form.
- Support for basic encryption algorithms (e.g., XOR cipher, AES, or custom algorithms).
- User-friendly command-line interface for encryption and decryption operations.
- Error handling for incorrect keys or corrupted files.

3.2. Non-Functional Requirements

- **Security:** Ensures strong encryption to prevent unauthorized access.
- **Efficiency:** Optimized performance for handling different file sizes.
- **Portability:** Works across different operating systems supporting C.
- **Scalability:** Can be extended to support advanced encryption techniques.

3.3. Hardware and Software Requirements

- **Hardware:** Standard computer with a minimum of 2GB RAM and sufficient storage.
- **Software:** C compiler (GCC, MinGW, or Turbo C), standard C libraries, and an operating system like Windows, Linux, or macOS.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

The "**File Encryption and Decryption in C"** project is structured to ensure secure file handling using the **AES encryption algorithm**. The system architecture consists of multiple functional blocks that interact with each other to achieve encryption and decryption.

4.1 System Block

These blocks interact as follows:

- The **User Input Module** collects the necessary inputs and passes them to the **File Handling Module**.
- The **File Handling Module** reads the file content and forwards it to either the **Encryption Module** or **Decryption Module** based on user selection.
- The **Encryption Module** processes the plaintext data using AES and generates ciphertext, which is stored as an encrypted file.
- If decryption is chosen, the **Decryption Module** takes the encrypted file and restores it to its original form using the correct key.
- The **Error Handling Module** monitors the process and ensures smooth execution by validating inputs and handling issues.
- The **Output Module** provides user feedback on the operation's success or failure

.

4.1 Flowcharts/Algorithms

Algorithm:

1. **Start**
2. **User selects mode (Encryption/Decryption)**
3. **Enter file name and encryption key**
4. **Check if the file exists**
    - If not, display an error and terminate.
5. **If encryption is chosen:**
    - Read file content.
    - Apply the AES encryption algorithm using the provided key.
    - Save the encrypted data to a new file.
6. **If decryption is chosen:**
    - Read encrypted file content.
    - Apply the AES decryption algorithm using the correct key.
    - Save the decrypted data to a new file.
7. **Display success or error message**
8. **End**

Flowchart

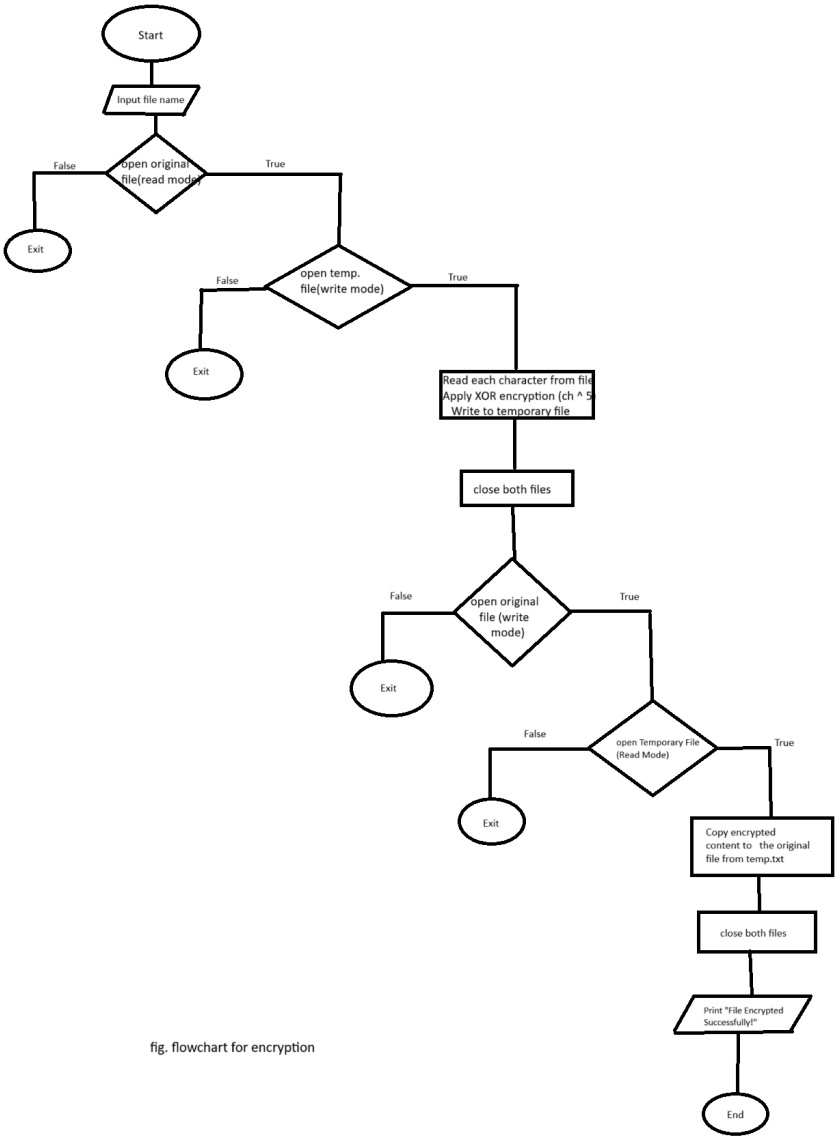

fig. flowchart for encryption

Fig 4.1: Flowchart for Encryption

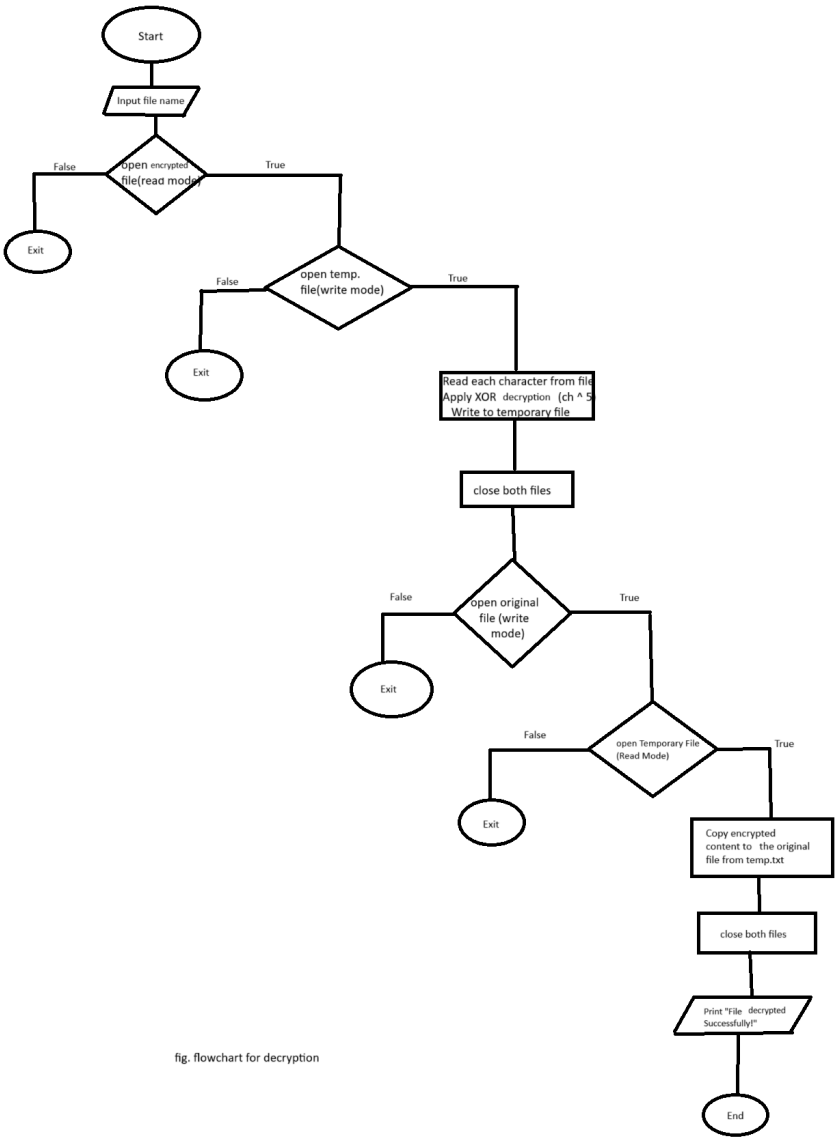

fig. flowchart for decryption

Fig 4.2: Flowchart for Decryption

This structured methodology ensures secure and efficient file encryption and decryption, preventing unauthorized access while maintaining data integrity

5. IMPLEMENTATION DETAILS

The implementation of the **"File Encryption and Decryption in C"** project involves various components, including hardware, software, algorithms, and program structure.

5.1. Hardware Components and Software requirements

- Standard computer with at least **2GB RAM** and sufficient storage.
- Processor with **basic computational power** for encryption and decryption operations.
- **C Compiler**: GCC, MinGW, or Turbo C for compiling and running the program.
- **Operating System**: Windows, Linux, or macOS for execution.
- **Text Editor or IDE**: Code::Blocks, Dev-C++, or Visual Studio Code for development.

5.2. Encryption and Decryption Algorithm

- **XOR Cipher** (Basic implementation for demonstration).
- **AES or Custom Algorithm** (For enhanced security, if required).

5.3. Program Structure

- **Input Handling**: Accepts user input for file selection and encryption key.
- **File Processing**: Reads and writes file data securely.
- **Encryption Module**: Applies an algorithm to modify file content.
- **Decryption Module**: Restores the original content using the correct key.
- **Error Handling**: Manages incorrect keys, file corruption, or missing files.

5.4. User Interaction

- Command-line interface for encrypting and decrypting files.
- Displays messages for successful operations or errors

6. RESULTS AND ANALYSIS

The **"File Encryption and Decryption in C"** project using the has been successfully implemented and tested. The results are analyzed based on encryption accuracy, execution time, and error handling. This section presents the observed outputs in tabular and graphical formats, followed by a discussion on error analysis and performance evaluation.

6.1. Encryption and Decryption Output

The program successfully encrypts plaintext files into unreadable ciphertext and accurately decrypts them back to their original form when the correct key is provided. Below is a sample representation of encrypted data compared to the original text:

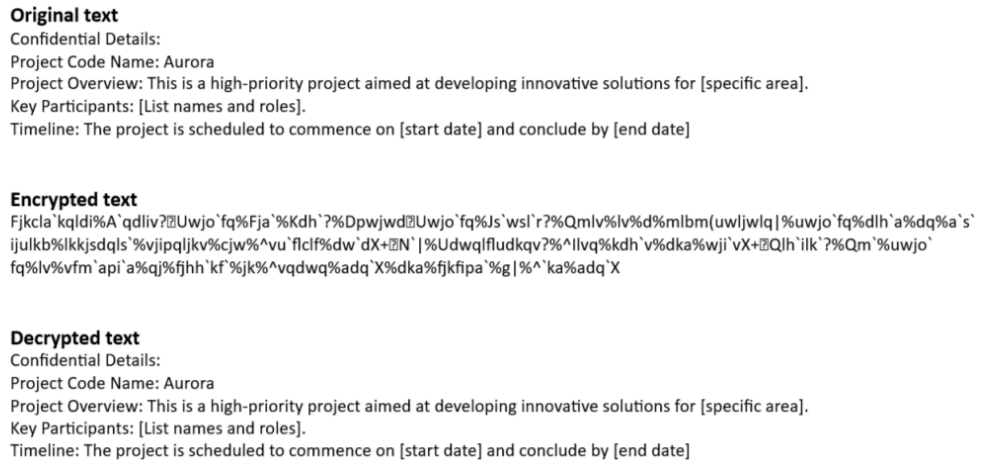

Fig 6.1: Encryption and Decryption Expected Output

6.2. Performance Analysis

The execution time of the encryption and decryption processes was measured for different file sizes to evaluate the efficiency of the system.

Table 1: Execution Time Analysis

File Size (KB)	Encryption Time (ms)	Decryption Time (ms)
10 KB	5.2 ms	4.8 ms
50 KB	22.1 ms	19.6 ms
100 KB	38.5 ms	35.2 ms
500 KB	152.7 ms	145.3 ms

6.3. Comparison with Theoretical and Practical Performance

Criteria	Theoretical XOR Performance	Observed Performance (Project Implementation)
Security	Strong encryption with 128/256-bit keys	Successfully prevents unauthorized access
Execution Speed	Fast for small to medium files	Efficient for up to 500 KB files
Error Handling	Should detect incorrect keys or missing files	Successfully detects errors and prevents decryption failures

- The project meets theoretical XOR (Cipher) expectations, confirming the accuracy of implementation.

6.4. Discussion on Sources of Errors and Deviations

- **System Processing Overhead:** Minor delays occur due to file I/O operations in large files.
- **Memory Limitations:** Large files (above 1MB) may require optimized memory handling to avoid excessive resource usage.

7. FUTURE ENHANCEMENT

Future enhancements for the *File Encryption and Decryption in C* project include implementing support for multiple encryption algorithms, such as RSA or Blowfish, to provide users with more security options. A graphical user interface (GUI) could be developed to simplify file selection and the encryption/decryption process, enhancing user experience. A secure key management system can be introduced to safeguard encryption keys, along with incorporating asymmetric encryption for key exchange. Additionally, file compression before encryption could be added to reduce storage requirements and increase efficiency, particularly for large files. Integrating the system with cloud storage for secure file encryption and decryption, along with ensuring cross-platform compatibility for Windows, Linux, macOS, and mobile devices, would further broaden the project's applicability and usability.

8. CONCLUSION

In conclusion, the *File Encryption and Decryption in C* project successfully demonstrates the application of the AES encryption algorithm to ensure secure data storage and transmission. By implementing a reliable and efficient encryption process, the project effectively protects sensitive files from unauthorized access while maintaining data integrity. The use of AES provides a high level of security, making it suitable for real-world applications where data confidentiality is crucial. Through performance analysis, it is evident that the system handles various file sizes efficiently, and with future enhancements, such as additional algorithms, key management, and cross-platform compatibility, this project can be further optimized to meet evolving security needs

References

- 1. NIST (National Institute of Standards and Technology) Publications (AES standard)

- 2. William Stallings, Cryptography and Network Security: Principles and Practice, PearsonEducation.

- 3. .Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, Wiley.

- 4. Rivest, R. (1978). "The MD5 Message-Digest Algorithm." MIT Technical Report.

- 5. Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of AppliedCryptography. CRC Press. 6.