A Project Report

on

File Encryptor in C++

Submitted in partial fulfillment of the requirement of PROJECT

PROJECT-I (BCE3009)

of

Bachelor in Computer Engineering

**Submitted to**

****

Purbanchal University

Biratnagar, Nepal

**Submitted By**

<Saraswoti Rokaya> <University Symbol #>

<Salim Shrestha> <University Symbol #>

<Aayush Kumar Mallik> <University Symbol #>

**KANTIPUR CITY COLLEGE**

Putalisadak, Kathmandu

2025

A Project Report

on

File Encryptor in C++

Submitted in partial fulfillment of the requirement of PROJECT

PROJECT- I (BCE3009)

of

Bachelor in Computer Engineering

**Submitted to**

Purbanchal University

Biratnagar, Nepal

**Submitted By**

< Saraswoti Rokaya > <University Symbol #>

< Salim Shrestha > <University Symbol #>

< Aayush Kumar Mallik > <University Symbol #>

**Project Supervisor**

**Mr. Kiran Khanal**

**Senior Lecturer**

**KANTIPUR CITY COLLEGE**

Putalisadak, Kathmandu

2025

Abstract  
  
This project presents a lightweight file encryption tool developed in C++, aimed at enhancing data confidentiality. The application utilizes symmetric encryption techniques like Caesar Cipher and XOR for securing both text and binary files. Designed for speed and simplicity, the tool features a user-friendly interface and supports file integrity checks and error handling. The project serves as a practical introduction to cryptography while promoting awareness of secure data handling practices.

The growing need for secure data storage and transmission has made file encryption an essential aspect of digital security. This project, “**File Encryptor in C++”**, focuses on the development of a lightweight yet effective tool to encrypt and decrypt files using standard cryptographic algorithms. The application is built using C++ due to its performance efficiency and control over system-level operations. It supports both symmetric encryption and basic XOR-based techniques for demonstration purposes. Users can securely encrypt text or binary files, store them safely, and decrypt them when required using a private key or password. The program also includes error handling, file integrity checks, and a simple command-line interface for ease of use. This project aims to provide a practical understanding of cryptography principles while ensuring data confidentiality and integrity.

# ACKNOWLEDGEMENT

We would like to express heartfelt gratitude to all those who supported us throughout the development of this project titled **“File Encryptor in C++”**. First and foremost, We would like to sincerely thank our project supervisor, **Mr. Kiran Khanal**, for his invaluable guidance, encouragement, and constructive feedback during every stage of this project. His deep knowledge and constant motivation were instrumental in shaping this work. We also thankful to the faculty members and staff of the **Department of Computer Engineering** for providing a supportive academic environment and the necessary infrastructure to complete this project. Our sincere appreciation also goes to my friends and classmates who provided insightful suggestions and encouragement throughout the development process. Last but not least, we would like to thank my family for their unwavering support, understanding, and patience during the course of this project.

# DECLARATION

We hereby declare that the project report entitled **“File Encryptor in C++”**, submitted in partial fulfillment of the requirements for the degree of **Bachelor of Engineering in Computer Engineering**, is the result of our original work carried out under the supervision of **Mr. Kiran Khanal**. This work has not been submitted previously for the award of any degree, diploma, or similar title at any other institution or university. In accordance with academic and ethical standards, proper acknowledgements have been given wherever the work of others has been referenced.

Salim Shrestha

Saraswoti Rokaya

Aayush Kumar Mallik

Date: - 2025

# SUPERVISOR’S APPROVAL

This is to certify that the major project entitled “**File Encryptor in C++”** undertaken and successfully demonstrated by **Salim Shrestha , Saraswoti Rokaya and Aayush Kumar Mallik**, has been completed under my guidance. This project is submitted as partial fulfillment of the requirements for the degree of **Bachelor of Engineering in Computer Engineering** under **Purbanchal University**. Throughout the duration of the project, the students have shown dedication, strong technical skills, and a clear understanding of the subject matter. Their performance during the development and presentation of the project reflects their readiness to take on professional responsibilities in the field. I hereby approve this project for certification by the concerned authority.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr. Kiran Khanal

Senior lecturer

Date :-

# CERTIFICATE FROM DEPARTMENT

This is to certify that, following the Supervisor’s Approval and Examiners’ Acceptance, the project entitled “**File Encryptor in C++”**, submitted by **Salim Shrestha , Saraswoti Rokaya and Aayush Kumar Mallik**, has been officially approved as a partial fulfillment of the requirements for the degree of **Bachelor of Engineering in Computer Engineering** under **Purbanchal University**. The department acknowledges the students’ efforts and successful completion of the project.

We commend their work and wish them continued success in all their future endeavors.

Official Stamp:

Name:

Post:

Date:-

## Introduction

In the digital era, securing sensitive information has become more important than ever. The File Encryptor in C++ project is designed to provide a reliable way to protect confidential data by converting readable files into an unreadable format using encryption algorithms. This ensures that even if unauthorized users gain access to the files, they cannot understand or misuse the information without the correct decryption key or password.

Developed using C++, this project focuses on simplicity, speed, and effectiveness. It allows users to encrypt and decrypt various types of files through a user-friendly command-line interface. The project implements both basic XOR encryption for learning purposes and more secure methods such as AES (Advanced Encryption Standard) for practical use. The File Encryptor not only enhances understanding of file handling and cryptography but also promotes awareness of digital security in real-world applications.

**Overview**

The File Encryptor project is a desktop-based application developed in C++ that enables users to securely encrypt and decrypt files to protect sensitive data from unauthorized access. With growing concerns over data privacy and cyber security, this tool offers a practical solution for safeguarding personal or confidential information by applying encryption algorithms to file content. Users can select a file, choose an encryption method, and provide a key or password to convert readable data into an unreadable format (cipher text), which can only be restored to its original form using the correct key.

The application primarily supports symmetric encryption techniques, such as a basic XOR cipher and may be extended to implement more advanced methods like AES. It includes features like file validation, key verification, and error handling to ensure a smooth and secure user experience. Designed with simplicity and efficiency in mind, this project demonstrates how encryption works at a fundamental level and provides a solid foundation for students to explore real-world applications of cryptography and data protection.

Problem Statement

In the current digital landscape, data is frequently shared and stored across various platforms, often without adequate protection. Many users, especially individuals and small organizations, rely on basic file storage methods that do not include any form of encryption. As a result, sensitive files are vulnerable to unauthorized access, data breaches, and cyber-attacks. Existing commercial encryption tools may be costly, complex, or require internet connectivity and advanced technical knowledge, making them less accessible to general users or students.

Moreover, some open-source or built-in file encryption solutions offer limited customization, lack transparency in their encryption processes, or provide minimal control over key management. This creates a significant gap for users who need a lightweight, efficient, and easy-to-use encryption tool for protecting personal or confidential data. There is a clear need for a standalone, platform-independent file encryptor that prioritizes security, simplicity, and control addressing the shortcomings of current systems while being accessible for educational and practical use.

Objectives  
  
- \*\*Confidentiality\*\*: Prevent unauthorized access to file contents.  
- \*\*Data Integrity\*\*: Ensure files are not altered during storage or transfer.  
- \*\*Access Control\*\*: Restrict file access to authorized users only.  
- \*\*Secure Storage\*\*: Safeguard sensitive data stored on disk.  
- \*\*Secure Transmission\*\*: Ensure protection of files during network transfer.

1. Confidentiality:

* Ensure that the contents of a file cannot be understood by unauthorized users.
* Encryption transforms readable data (plaintext) into unreadable data (ciphertext).

1. Data Integrity:

* Prevent undetected modification of the file during storage or transmission.
* While encryption alone doesn’t guarantee integrity, it's often paired with checksums or hashes for this purpose.

1. **Access Control:**

* Only authorized users with the correct decryption key or password can access the file’s original content.

1. **Secure Storage:**

* Protect sensitive data stored on disk, such as user credentials, personal information, or business documents.

1. **Secure Transmission:**

* Ensure the file remains protected during network transmission, preventing eavesdropping.

**Features**

1. Authentication

* Ensures integrity and authenticity using GCM or HMAC.

1. Initialization Vector

* Random IV for each encryption session to ensure randomness (especially for AES in CBC/CTR mode).

1. Stream Encryption

* Encrypt/decrypt file in chunks to avoid loading large files into memory.

**Significance**

A file encryptor in C++ is a program that protects the contents of a file by turning it into unreadable code using encryption techniques. This is important because it keeps sensitive information like passwords, personal data, or private documents safe from hackers or unauthorized users. Only someone with the correct key or password can unlock (decrypt) the file and read its original content.

Using C++ for file encryption is powerful because it gives full control over how files are handled and how secure the encryption process is. C++ also runs very fast, making it a good choice for encrypting large files or building secure systems like password managers, secure backups, or confidential file transfer tools.

Scope  
  
This project aims to develop a simple file encryptor using C++ that allows users to securely encrypt and decrypt files to protect their data from unauthorized access. It uses Caesar Cipher and XOR encryption to convert contents of a file into unreadable text and then restore it using the correct password or key. The program supports text or binary files and provides an interactive interface, focusing on simplicity, portability, and practical use.

This project aims to develop a simple file encryptor using C++ that allows users to securely encrypt and decrypt files to protect their data from unauthorized access. It will use a basic encryption algorithm (like AES) to convert the contents of a file into unreadable text and then restore it using a correct password or key. The program will support text or binary files and allow users to choose files from their system through a command-line interface. It will be designed to work on most operating systems and will focus on protecting personal or sensitive data in a lightweight and user-friendly way.

Limitations  
  
- No support for network/cloud encryption.  
- Uses fixed encryption algorithm.  
- Password must be remembered (no secure storage).  
- May fail on large files due to memory limits.  
- Command-line only (no GUI).  
- No protection against advanced attacks (e.g., brute-force).

 The program does not support network or cloud-based file encryption.

 It uses a fixed encryption algorithm and does not allow algorithm switching.

 Does not store or manage keys securely key/password must be remembered by the user.

 It may not work properly with very large files due to memory usage.

 No user interface only works through command-line input.

 ️ Does not provide protection against advanced attacks like side-channel or brute-force unless combined with strong passwords.

## Methodology

**Software Development Life cycle**

**Waterfall Model**

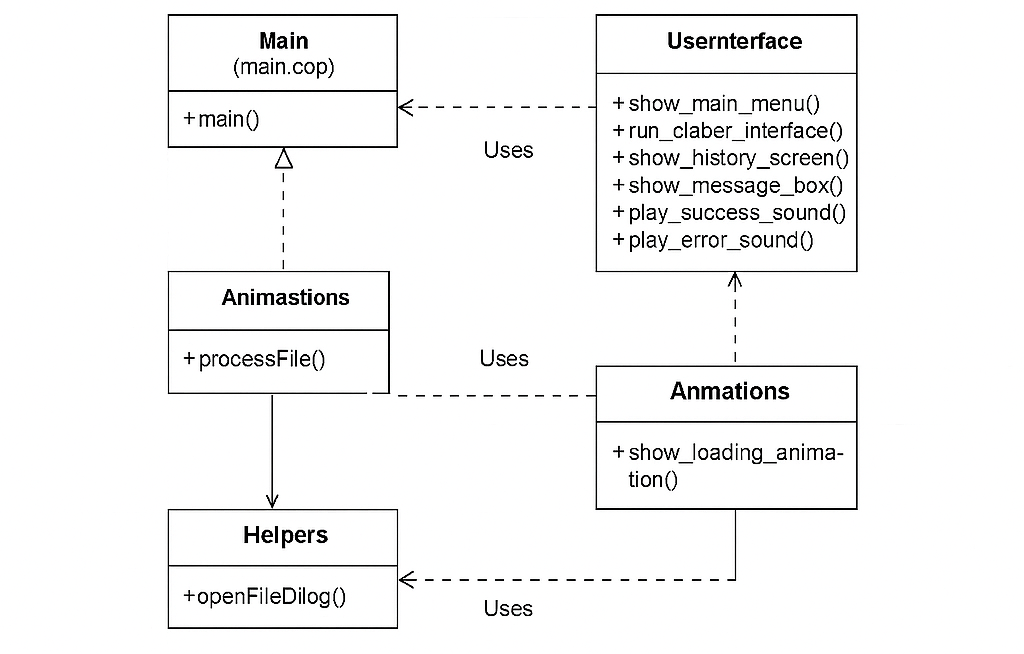
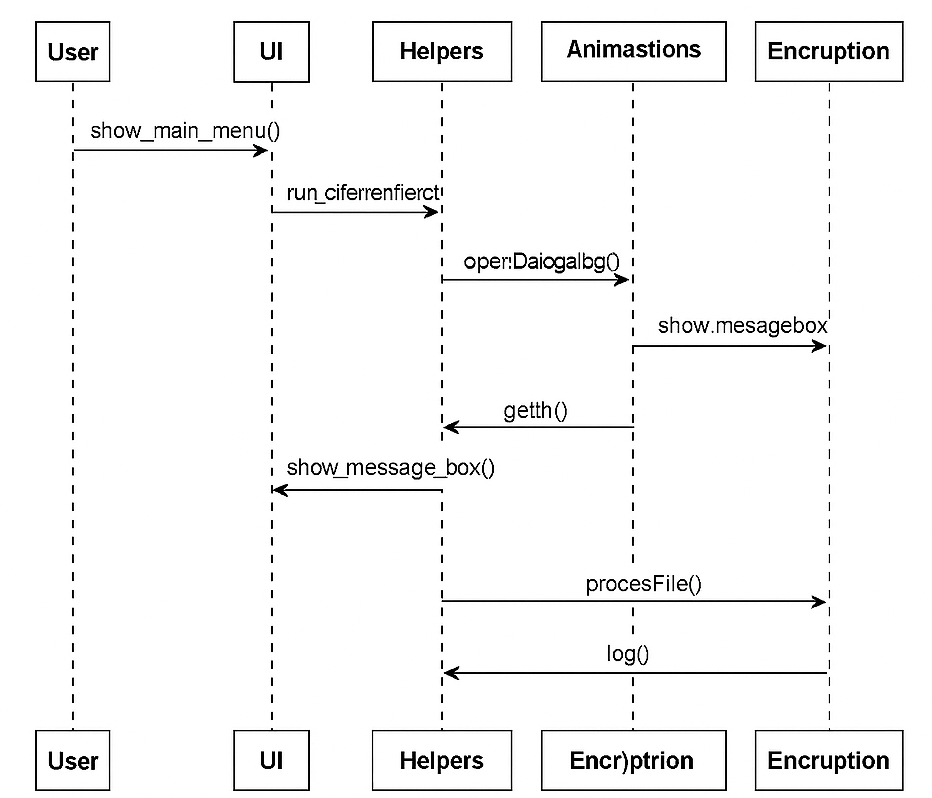
****

Fig 1: Waterfall Model

1. Requirement Analysis

* Define what the program should do: encrypt/decrypt a file using Caesar Cipher, take input/output file names, and shift key.
* Decide on supported file formats (e.g., .txt only).
* Define expected behavior for invalid inputs or characters.

1. System Design

* Design the structure of the program (functions for reading, encrypting, writing).
* Plan file handling, user input, and error checking.
* Decide how Caesar Cipher will be implemented (ASCII character shifting, handling upper/lowercase).

1. Implementation

* Write the program in C++.
* Create functions for:
* Reading input from a file
* Encrypting text using Caesar Cipher
* Writing to the output file

1. Testing

* Test with various shift keys and input files.
* Check encryption correctness and file handling robustness.
* Validate handling of special characters, empty files, etc.

1. Deployment

* Deliver the final executable or source code.
* Document usage instructions for the user.

1. Maintenance

* Fix any bugs found after deployment.
* Optional: Add new features like decryption, GUI, or support for other ciphers.

**Technologies and Tools used for the File Encryptor Project**

|  |  |  |
| --- | --- | --- |
| SN | TOOLS | PURPOSE |
| 1 | C++ | Core programming language |
| 2 | VS Code, DEV C | Write, debug, and run code |
| 3 | Compiler (GCC,MSVC) | Turn code into executable |
| 4 | Standard Libraries | File I/O, encryption logic |
| 5 | Operating system | Any major OS (Windows/Linux/macOS) |

Fig 2: Table

Figure: UML Class Diagram of the File Encryptor Project

Figure: Sequence Diagram of File Encrypt/Decrypt Workflow