TEXT & FILE ENCRYPTION & DECRYPTION

PROJECT SYNOPSYS

of

Bachelor of Technology
in
COMPUTER SCIENCE & ENGINEERING

BY

NAME:	ROLL NO:
Aditya Kundu	21CS011004
Anurag Karmakar	21CS011021
Arpan Halder	21CS011030
Biraj Naskar	21CS011043
Debalina Ghosh	21CS011045

4th YEAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JIS UNIVERSITY, KOLKATA

TEXT & FILE ENCRYPTION & DECRYPTION

BTech Final Year Project

Submitted in partial fulfillment of the
Requirements for the award of the degree
Of

Bachelor of Technology

in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BY

NAME:	ROLL NO:
Aditya Kundu	21CS011004
Anurag Karmakar	21CS011021
Arpan Halder	21CS011030
Biraj Naskar	21CS011043
Debalina Ghosh	21CS011045

4th YEAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JIS UNIVERSITY, KOLKATA

CERTIFICATE

We hereby certify that the work which is being presented in this Project Report entitled **"Text and File Encryption and Decryption"** in partial fulfillment of the requirements for the award of Bachelor **of Technology in Computer Science and Engineering** and submitted to the **Department of Computer Science and Engineering** of JIS University is an authentic record of our own work carried out during a period from July 2024 to Nov 2024 under the supervision of Mr. Suprativ Saha, Asst. Professor, CSE Department.

The matter presented in this thesis has not been submitted by us for the award of any other degree elsewhere.

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Suprativ Saha, Asst. Professor

Guide Name

Date:

HOD CSE Department

Signature of External

ACKNOWLEDGEMENT

We wish to take this opportunity to convey our sincere thanks to all who in various ways have contributed in the success of this project.

Prior to all I would like to thank Mr. Suprativ Saha for initiating us on to this project. His constant source of guidance, support & inspiration, perspective insight, friendly cooperation, valuable suggestion & advice, which not only helped to complete this project in such a nice way but will also act as stepping stone for our professional career. It has been a great pleasure learning and working under him.

We would like to thank all the staff member of computer science department (C.S.E) for their co-operation and constant motivation.

Thanks,

Aditya Kundu – 21CS011004
Anurag Karmakar – 21CS011021
Arpan Halder – 21CS011030
Biraj Naskar – 21CS011043
Debalina Ghosh – 21CS011045

CONTENTS

Title page	**1-2**
Certificate	**3**
Acknowledgement	**4**
Abstract	**6-7**
Project Overview	**7**
Functional Description	**8-11**
Code Analysis	**11-12**
Implementation Details	**12-13**
Results and Benefits	**13-14**
Future Developments	**14-15**
Project Diagrams	**15**
Conclusion	**16**

ABSTRACT

Cryptography and data security is a field focused on protecting information during its lifecycle—storage, transmission, and use. Its foundations lie in principles such as confidentiality (preventing unauthorized access), integrity (ensuring data remains unaltered), authentication (verifying user/system identities), and non-repudiation (preventing denial of actions).

The ultimate vision of this domain is to build a secure digital ecosystem where sensitive data is protected against evolving threats. Key goals include:

- **Enhancing Security**: Continually improving encryption techniques to combat emerging vulnerabilities.
- **Accessibility**: Bridging the gap between advanced cryptographic solutions and end-user usability.
- **Proactive Defense**: Anticipating and mitigating threats, such as those posed by quantum computing.
- **Awareness and Adoption**: Educating users and organizations on implementing robust security measures.

The Text & File Encryption and Decryption project directly aligns with the core principles of cryptography and data security, a critical domain dedicated to safeguarding sensitive information.

This project aligns with the domain in the following ways:

1. **Confidentiality and Integrity**:
   Through AES-256 encryption in CBC and GCM modes, the project ensures that sensitive text and files are securely encoded, protecting them from unauthorized viewing or tampering.
2. **Authentication and Access Control**:
   Secure password hashing (with salts) and a retry/lockout mechanism prevent brute-force attacks, enforcing access control measures critical to data security.
3. **Usability and Practicality**:
   The integration of cryptographic processes into an intuitive GUI makes advanced security accessible to users of all expertise levels, encouraging adoption in real-world scenarios.

The system allows users to input plaintext or select files for encryption, transforming them into secure, encoded formats that can only be decrypted with the correct password. To enhance security, the application employs strong password management, including validation rules and password hashing with salts using the **SHA-256** algorithm. This ensures that user passwords are stored securely and protected from unauthorized access. Additionally, the software implements a retry and lockout mechanism to prevent brute-force attacks.

Two encryption modes are supported—**AES-CBC (Cipher Block Chaining)** and **AES-GCM (Galois/Counter Mode)**—allowing users to choose between traditional encryption with padding and a modern approach with built-in data authentication. The encrypted output is encoded in Base64 for easier transmission and storage, while decrypted files and text are restored to their original form with minimal effort.

The project emphasizes usability, with features such as a progress bar for long-running tasks, file dialogs for easy file selection, and clear instructions for password setup and validation. By combining advanced cryptographic practices with an intuitive interface, this application bridges the gap between technical complexity and user accessibility.

In essence, the **Text & File Encryption and Decryption Tool** provides a reliable, secure, and user-friendly platform for protecting sensitive information. It demonstrates the practical implementation of modern cryptography while addressing real-world challenges such as usability, password security, and resilience against common attack vectors. This project serves as a vital tool for individuals and organizations aiming to safeguard their data in an increasingly vulnerable digital world.

PROJECT OVERVIEW

This project implements a **text and file encryption/decryption utility**. The utility emphasizes both security and usability, featuring password validation, lockout mechanisms, and progress indicators.

The **Text & File Encryption and Decryption Tool** is a robust and user-friendly application designed for securely encrypting and decrypting sensitive text and files. It uses the **Advanced Encryption Standard (AES)** algorithm, a widely recognized and trusted encryption standard, ensuring that data remains confidential and tamper-proof. The application is built with **Python** and features a graphical user interface (GUI) powered by **Tkinter**, making it accessible even for non-technical users.

The core idea of the project is to enable users to securely store, transmit, and retrieve confidential information using encryption. The project emphasizes strong password management, secure encryption modes, and an intuitive interface to simplify the overall experience while maintaining high-security standards.

FUNCTIONAL DESCRIPTION

The **Text & File Encryption and Decryption** project is a comprehensive application designed to perform secure encryption and decryption operations on text and files. Below is a detailed functional breakdown of the project, categorized into its key features and processes:

1. Password Management

1.1 Password Setup

- **Functionality**:
  - Allows users to set a master password for the application.
  - Enforces a robust password policy:
    - Minimum 8 characters.
    - At least one uppercase letter, one lowercase letter, one digit, and one special character.
  - Displays guidelines for password creation in the GUI.
- **Implementation**:
  - Validates the password based on the predefined rules.
  - Hashes the password using **SHA-256** with a random salt for secure storage.
  - Prevents storage of plaintext passwords.

1.2 Password Verification

- **Functionality**:
  - Ensures that only authorized users can perform encryption or decryption tasks.
  - Compares the provided password against the securely stored salted hash.
- **Implementation**:
  - Extracts the salt from the stored hash and verifies the entered password using the same salt.

1.3 Retry and Lockout Mechanism

- **Functionality**:
  - Limits the number of incorrect password attempts to prevent brute-force attacks.
  - Implements a lockout period after exceeding the allowed number of retries.
- **Implementation**:
  - Tracks failed attempts and initiates a lockout timer when retries are exhausted.
  - Notifies users of lockout status and remaining time.

2. Text Encryption and Decryption

2.1 Encryption

- **Functionality**:
  - Encrypts user-provided text using the **AES** algorithm.

    - Supports two encryption modes:
        - **AES-CBC (Cipher Block Chaining)**: Requires padding to align text with block size.
        - **AES-GCM (Galois/Counter Mode)**: Provides encryption and authentication with an integrity tag.
    - Outputs encrypted text in **Base64** format for readability and compatibility.
- **Implementation**:
    - Derives a 256-bit encryption key from the password using **SHA-256**.
    - Generates a random initialization vector (IV) for AES-CBC or nonce for AES-GCM.
    - Uses AES encryption with the selected mode to generate ciphertext.
    - Concatenates the IV/nonce with the ciphertext and encodes it in Base64.

2.2 Decryption

- **Functionality**:
    - Decrypts the encrypted text back into its original form using the correct password.
    - Verifies the integrity of the encrypted data (in AES-GCM mode).
- **Implementation**:
    - Decodes the Base64 input to extract the IV/nonce and ciphertext.
    - Uses the same password-derived key for decryption.
    - Removes padding (if AES-CBC) and returns plaintext.

3. File Encryption and Decryption

3.1 File Encryption

- **Functionality**:
    - Encrypts selected files to secure their contents.
    - Outputs the encrypted file in a binary format with a `.enc` extension.
- **Implementation**:
    - Reads the file data in binary mode.
    - Generates a random IV/nonce for encryption.
    - Encrypts the file data using the same process as text encryption.
    - Saves the encrypted file with the IV/nonce prepended.

3.2 File Decryption

- **Functionality**:
    - Decrypts files encrypted by the application, restoring them to their original state.
    - Supports integrity checks for AES-GCM encrypted files.
- **Implementation**:
    - Reads the encrypted file and extracts the IV/nonce and ciphertext.
    - Decrypts the file data using the selected AES mode and password-derived key.
    - Saves the decrypted content to a new file.

4. Graphical User Interface (GUI)

4.1 Text Area and Input Fields

- **Functionality**:
  - Provides a text box for users to input text for encryption or decryption.
  - Includes an input field for entering the password.
- **Implementation**:
  - Uses Tkinter widgets such as `Text` and `Entry` for input fields.
  - Hides the password input for security using the `show="*"` attribute.

4.2 File Dialogs

- **Functionality**:
  - Allows users to select files for encryption or decryption.
  - Enables users to specify the save location for processed files.
- **Implementation**:
  - Uses Tkinter's `filedialog` module for file selection and save operations.

4.3 Progress Bar

- **Functionality**:
  - Displays progress during encryption and decryption tasks to indicate ongoing operations.
- **Implementation**:
  - Creates a `Progressbar` widget that runs asynchronously while tasks are performed in a separate thread.

4.4 Dropdown for Encryption Mode Selection

- **Functionality**:
  - Allows users to choose between **AES-CBC** and **AES-GCM** modes.
- **Implementation**:
  - Implements a Tkinter `Combobox` to display selectable encryption modes.
  - Updates the application's encryption mode based on the user's selection.

4.5 Buttons for Operations

- **Functionality**:
  - Provides buttons to trigger specific actions:
    - **Encrypt Text**: Encrypts the text in the input box.
    - **Decrypt Text**: Decrypts the encrypted text in the input box.
    - **Encrypt File**: Encrypts the selected file.
    - **Decrypt File**: Decrypts the selected encrypted file.
    - **Reset**: Clears all input fields and text areas.
- **Implementation**:
  - Maps each button to its respective functionality using event handlers.

5. Security Features

5.1 Strong Encryption

- Uses AES with 256-bit keys, a high-security standard suitable for modern encryption needs.

5.2 Password Hashing with Salt

- Ensures that passwords are never stored in plaintext and are resistant to dictionary attacks.

5.3 Lockout Mechanism

- Protects against brute-force attacks by temporarily disabling access after multiple failed attempts.

5.4 Integrity Checks

- Verifies the authenticity of encrypted data using tags in AES-GCM mode.

6. Error Handling and User Notifications

- Provides user-friendly error messages for issues such as:
  - Invalid password attempts.
  - Incorrect or corrupted encrypted data.
  - Missing input text or file selection.
- Uses `messagebox` to display notifications and error alerts to users.

This functional description demonstrates the project's ability to provide a secure, efficient, and user-centric platform for data encryption and decryption while adhering to modern cryptographic standards and best practices.

CODE ANALYSIS

3.1 Modules Used

Module	Purpose
`os`	Generates secure random salts for password hashing.
`hashlib`	Computes SHA-256 hashes for password security and AES key derivation.
`base64`	Encodes and decodes binary data for secure storage.
`binascii`	Error handling during binary encoding/decoding.
`time`	Implements retry lockout mechanisms with timestamp comparisons.
`threading`	Executes background tasks without blocking the GUI thread.
`re`	Validates passwords against regex-based security rules.
`tkinter`	Builds the GUI for user interaction.
`Crypto.Cipher`	Provides AES encryption and decryption operations.

Module	Purpose
`Crypto.Util`	Manages padding/unpadding for AES in CBC mode.

3.2 Key Components

Password Manager

- **Password Hashing and Validation**:
  - Uses a random 16-byte salt concatenated with SHA-256 hash of the salt and password.
  - Stored format: `salt + hash`.
  - Prevents dictionary attacks by ensuring the same password produces unique hashes.
- **Password Validation**:
  - Enforces rules for length, uppercase, lowercase, digit, and special character inclusion.
  - Provides user-friendly error messages for violations.

Encryption and Decryption

- **AES Modes**:
  - **CBC**: Requires Initialization Vector (IV). Data is padded to match block size (16 bytes).
  - **GCM**: Provides authenticated encryption by generating a unique tag.
- **Key Derivation**:
  - Derives a 32-byte AES key using SHA-256 of the user's password.
- **Progress Bar**:
  - Indicates encryption or decryption tasks in a non-blocking manner.

File Operations

- **Encryption**:
  - Reads file data, encrypts using AES, and writes encrypted content with IV or nonce prepended.
- **Decryption**:
  - Reads encrypted file, extracts IV/nonce, and decrypts using the same AES key.

IMPLEMENTATION DETAILS

4.1 Security Features

1. **Salted Password Hashing**:
   - Prevents precomputed hash attacks (e.g., rainbow tables).
2. **AES Encryption**:
   - AES is a symmetric encryption standard widely regarded as secure.

3. **Password Policies**:
    - Enforces strong password rules to enhance resistance against guessing attacks.
4. **Retry Lockout**:
    - Implements a 30-second lockout after 3 consecutive failed password attempts.

4.2 User Interface Design

1. **Password Setup**:
    - A dedicated window allows users to set and validate their encryption password.
    - Includes detailed instructions for password requirements.
2. **Main Screen**:
    - Text area for input and a combobox for encryption mode selection.
    - Buttons for text/file encryption and decryption with reset functionality.
3. **Progress Indicators**:
    - Enhances user experience by showing task progress.

4.3 Challenges Addressed

1. **Padding for CBC Mode**:
    - Managed via `Crypto.Util.Padding` to ensure plaintext aligns with AES block size.
2. **File Handling**:
    - Supported various file formats with user-specified save paths.
3. **Error Handling**:
    - Provided detailed error messages for encryption/decryption failures.

RESULTS & BENEFITS

- **Results**:
    - Successfully encrypts and decrypts text and files using AES-256 in both CBC and GCM modes.
    - Implements robust password protection with hashing and salting, and enforces strong password policies.
    - Provides an intuitive GUI with progress indicators and clear notifications for enhanced usability.
    - Handles various file types and ensures secure data sharing through Base64 encoding.
    - Offers error handling, lockout mechanisms, and customizable encryption modes for versatility.

- **Benefits**:
    - **Enhanced Security**: Ensures confidentiality and integrity of sensitive data.
    - **Ease of Use**: Simplifies encryption processes with a user-friendly interface.
    - **Flexibility**: Supports text and file encryption for personal and professional use.
    - **Unauthorized Access Prevention**: Employs password protection and lockout features.
    - **Future-Ready**: Modular design allows for further enhancements like additional encryption modes or features.

This project provides a secure and accessible tool for protecting data, ideal for individual and small-business use cases.

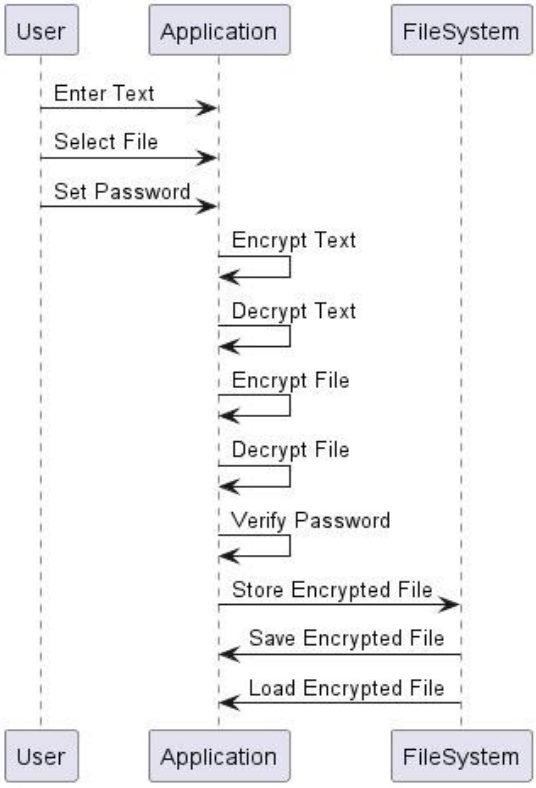
FUTURE IMPROVEMENTS

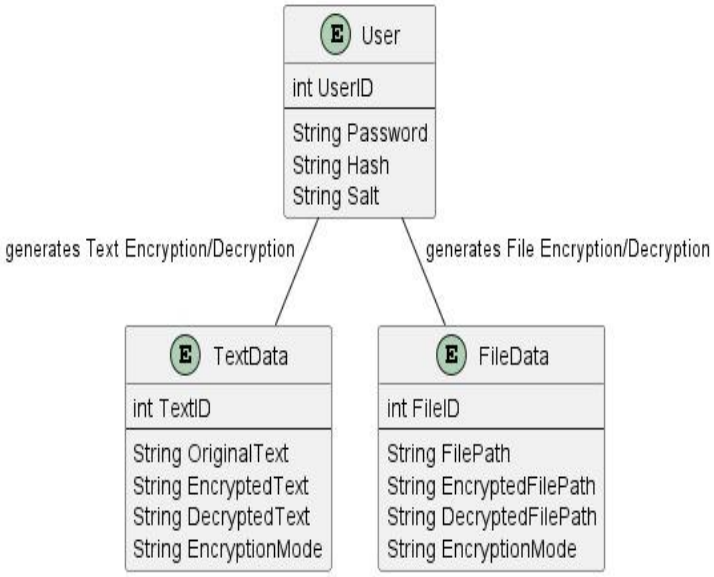
While the current implementation of the **Text & File Encryption and Decryption Tool** provides robust functionality and usability, there are several potential enhancements that could improve its effectiveness, user experience, and adaptability. Below are some suggestions for future improvements:

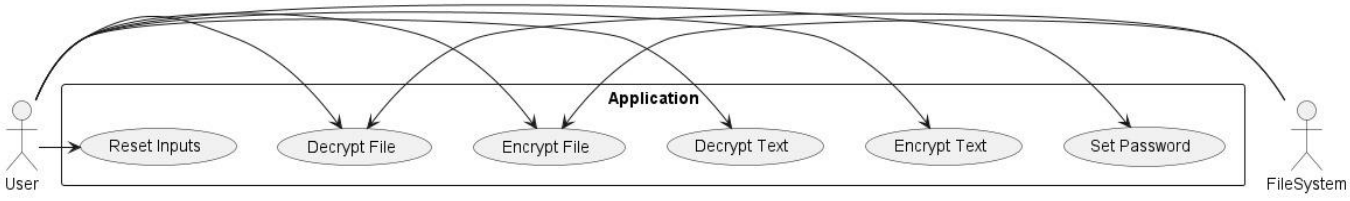
1. **Enhanced Authentication Mechanisms**:
    - **Biometric Authentication**: Integrating fingerprint or facial recognition for secure and convenient access.
    - **Multi-Factor Authentication (MFA)**: Adding a secondary layer of security, such as OTPs (One-Time Passwords) or email verification.
2. **Support for Additional Encryption Algorithms**:
    - Including other algorithms like RSA (asymmetric encryption).
    - Allowing users to choose encryption standards based on their specific needs.
3. **Batch File Processing**:
    - Introducing functionality for encrypting or decrypting multiple files simultaneously to save time.
    - Allowing directory-level encryption for larger-scale use cases.
4. **Improved Password Management**:
    - Incorporating password recovery options, such as security questions or encrypted backup files, to prevent permanent lockouts.
    - Adding a password strength meter to guide users during password creation.
5. **Enhanced User Interface**:
    - Designing a modern, responsive UI with improved aesthetics and accessibility for users with disabilities.
    - Introducing a dark mode option for better usability in low-light environments.

6. **Language Support**:
    - Providing multilingual support to cater to a global user base.
    - Allowing users to select their preferred language in the application settings.
7. **File Integrity Verification**:
    - Adding checksum verification to ensure the integrity of files before and after encryption or decryption.
    - Informing users if a file has been corrupted or tampered with.
8. **Advanced Error Handling**:
    - Implementing detailed error logs that guide users in troubleshooting issues.
    - Providing auto-recovery mechanisms for interrupted encryption or decryption processes.
9. **Education and Guidance**:
    - Adding a help section or tutorial to educate users on cryptographic practices and how to use the application securely.
    - Providing prompts or tooltips to explain complex features.

PROJECT DIAGRAMS







CONCLUSION

The **Text & File Encryption and Decryption** project provides a secure and user-friendly solution for protecting sensitive data. By leveraging AES-256 encryption in both CBC and GCM modes, it ensures robust confidentiality and data integrity for both text and files. The implementation of strong password policies, hashing with salting, and a lockout mechanism further enhances security, preventing unauthorized access effectively.

The graphical user interface simplifies encryption and decryption processes, making the tool accessible even to non-technical users. Features such as real-time progress bars, customizable encryption modes, and detailed error handling contribute to a seamless user experience. Additionally, support for file encryption extends its utility to diverse personal and professional use cases.

While the project achieves its core objectives, there remains significant potential for enhancements, including advanced authentication, cloud integration, and support for additional encryption standards. These improvements would enhance its adaptability, usability, and overall effectiveness.

In conclusion, this project serves as a reliable tool for safeguarding sensitive information, combining practical functionality with strong cryptographic principles. It provides a solid foundation for further development and adaptation to meet evolving security challenges.