PROJECT REPORT ON

"TEXT FILE ENCRYPTION AND DECRYPTION"

SUBMITTED TO:

SHREE. KAKSAHEB HEERALAL MAGANLAL CHAUDHARI ARTS, COMMERCE AND SCIENCE COLLEGE, NANDURBAR - 425412
(ACCREDITED WITH GRADE-B BY NAAC)

SUBMITTED BY:

MISS. CHAUDHARI JANHAVI NANDLAL
PRN: 2022015400038272

GUIDED BY:

PROF. PRIYANKA AGALE

IN PARTIAL FULFILLMENT OF DEGREE OF

BACHELOR OF COMPUTER APPLICATIONS (BCA)

KAVAYATRI BAHINABAI CHAUDHARI NORTH MAHARASHTRA UNIVERSITY, JALGAON.

For the Academic Year 2024-2025

SHREE. KAKSAHEB HEERALAL MAGANLAL CHAUDHARI ARTS, COMMERCE AND SCIENCE COLLEGE, NANDURBAR

Title & Organization: - "Text File Encryption and Decryption"

Course name : - TYBCA

Student Name : - Chaudhari Janhavi Nandlal

Student PRN : - 2022015400038272

Email Id : - janhavichaudhari118@gmail.com

Mobile No : - 8010184837

Academic Year : - 2024 - 2025

Guide Name : -Prof. Priyanka Agale

Email of Guide :-agalepriyanka24@gmail.com

Mobile No of Guide: - 9860972509

CERTIFICATE

This is to certify that **Miss. Chaudhari Janhavi Nandlal** has completed his project entitled "**Text File Encryption and Decryption**" for practical fulfillment of **Bachelor of Computer Applications** for K.B.C. North Maharashtra University, Jalgaon under our guidance during academic year 2024-2025.

This performs and the system is up to the mark and we are satisfied with the same.

Dr. Chandrakant Wani	**Prof. Sunil More**	**Prof. Priyanka Agale**
(Principal)	**(H.O.D)**	**(Project Guide)**

External Examiner-1 External Examiner-2

ACKNOWLEDGEMENT

I have great pleasure in submitting this project report on "**Text File Encryption and Decryption"** for Bachelor of Computer Applications.

In completion of this project work , Firstly I will honest and sincere to our Principal **Dr. Ravindra Chaudhari** ,Vice Principal **Dr. Chandrakant Wani** and head of department represented by **Prof. Sunil More** for her time & kind cooperation and providing require facilities .

Our special acknowledgment to guide **Prof. Priyanka Agale** for their valuable guidance kind suggestion, constant encouragement and excellent co-operation and valuable helps.

We also like to pay humble gratitude to our parents for providing and moral support for completion of this project work.

I wish to express our heartiest feeling of love and affection of our parents molts our life, encourages moral support and provide me valuable opportunities for completion of this project. Also thanking to those who support me directly and indirectly to develop such project.

Miss. Chaudhari Janhavi Nandlal
(TY-BCA SEM VI)

2024-25

INDEX

CHAPTER-I

INTRODUCTION TO PROJECT

Introduction to Project :-

1.Overview of the Project

With the growing reliance on digital data, securing sensitive information has become a critical concern. Cyber threats such as hacking, data breaches, and unauthorized access have highlighted the need for effective security measures.

This project focuses on **text file encryption and decryption**, a fundamental method to protect data from unauthorized access. **Encryption** converts readable text (plaintext) into an unreadable format (ciphertext) using a cryptographic algorithm. **Decryption** reverses the process, converting ciphertext back into plaintext when access is granted.

The project will implement various encryption techniques such as:

- **Symmetric Encryption (AES, DES, Blowfish):** Uses a single key for both encryption and decryption.
- **Asymmetric Encryption (RSA, ECC):** Uses a public key for encryption and a private key for decryption.

This system will ensure that only authorized users with the correct decryption key can access encrypted files. It will be designed for **secure storage and transmission of text files** across different platforms.

Key Features of the Project:

✓ Secure **encryption and decryption** of text files.
✓ **User authentication** to prevent unauthorized access.
✓ Implementation of **strong cryptographic algorithms**.
✓ **Cross-platform compatibility** (Windows, Linux, MacOS).
✓ Secure **key management system** for encryption keys.
✓ **User-friendly interface** (command-line tool or GUI).

The project is relevant for **individual users, businesses, and organizations** looking to protect sensitive information from cyber threats.

2. Importance of Encryption and Decryption

Why is Encryption Important?

Encryption is crucial in today's digital world for **data protection and cybersecurity**. Without encryption, sensitive information can be easily accessed by attackers, leading to **identity theft, financial fraud, or data manipulation**.

Key Benefits of Encryption:

🔒 **Data Confidentiality:** Ensures only authorized users can read the file.
🛡 **Data Integrity:** Prevents unauthorized modifications or tampering.
✅ **User Authentication:** Allows only verified users to access encrypted content.
🌍 **Secure Communication:** Enables safe file sharing over the internet.
📂 **Compliance with Security Standards:** Helps organizations meet legal and regulatory requirements such as **GDPR, HIPAA, and ISO 27001**.

Why is Decryption Important?

Decryption allows authorized users to **retrieve original data** when needed. Without decryption, the encrypted data remains unreadable.

Key Benefits of Decryption:

✓ **Access Control:** Ensures that only those with valid keys can read the file.
✓ **Data Recovery:** Allows retrieval of important data securely.
✓ **Secure Information Exchange:** Helps in **secure file transfers, banking transactions, and password management**.

Encryption and decryption are widely used in:

- **Online banking and financial transactions**
- **Secure email communication**
- **Cloud storage and data backup**
- **Military and government data protection**

Without encryption, confidential data is at **high risk of cyber threats**, making encryption an essential tool for modern security systems.

3. Objectives and Expected Outcomes

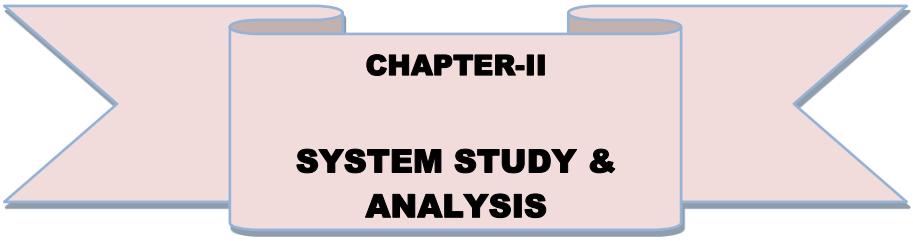
Objectives of the Project:

1. **Develop a secure encryption and decryption system for text files** to protect sensitive information.
2. **Implement strong cryptographic algorithms** (AES, RSA, Blowfish) for data security.
3. **Ensure user authentication** to restrict access to encrypted files.
4. **Provide a simple and efficient user interface** for file encryption and decryption.
5. **Enable secure file transmission** over networks without risk of unauthorized access.
6. **Manage encryption keys securely**, ensuring they are not exposed or lost.
7. **Analyze the performance of different encryption techniques** based on speed, security, and efficiency.

Expected Outcomes of the Project:

✅ **Successful encryption and decryption of text files** using selected cryptographic algorithms.
✅ **Enhanced data security and privacy**, preventing unauthorized access.
✅ **Secure key management** to avoid key loss or misuse.
✅ **User-friendly tool** for encrypting and decrypting files easily.
✅ **Performance evaluation of encryption methods**, helping users choose the most effective algorithm.
✅ **Implementation of best security practices**, ensuring compliance with data protection standards.

This project will be valuable for anyone dealing with **sensitive data**, such as **businesses, students, government agencies, and IT professionals**. By implementing encryption, we can significantly reduce the risks associated with **data breaches, cyber-attacks, and unauthorized file access**.



1. Problems in Existing Data Security Methods

With the rapid advancement of technology, data security has become a significant concern. Many traditional data protection methods are **ineffective against modern cyber threats**. Below are some of the critical issues in existing data security techniques:

1.1 Weak Password-Based Protection

- Many users rely on simple **password-protected files** (e.g., ZIP files, Microsoft Word documents).
- These passwords can be **easily cracked using brute force attacks** or **dictionary attacks**.
- Passwords are often **stored insecurely** or shared improperly, leading to unauthorized access.

1.2 Lack of Strong Encryption

- Some systems use **outdated or weak encryption algorithms**, such as DES (Data Encryption Standard), which are **vulnerable to modern computing power**.
- Some applications **store data in plaintext**, making it easy for attackers to access and modify it.

1.3 Data Breaches and Cyber Threats

- Cybercriminals exploit **unprotected files** and networks to steal sensitive data.
- **Man-in-the-Middle (MITM) attacks** occur when unencrypted data is transmitted over a network, allowing attackers to intercept it.
- **Ransomware attacks** encrypt files and demand a ransom, making data inaccessible to users.

1.4 Unauthorized Access and Insider Threats

- Employees or unauthorized users can **access confidential files** due to **poor access control mechanisms**.
- Lack of **audit trails and user activity logs** makes it difficult to track security breaches.

1.5 Data Integrity Issues

- Without encryption, attackers can **modify files** without detection, leading to **data corruption or manipulation**.
- **Malware and viruses** can infect files, leading to irreversible damage.

1.6 Issues in Data Storage and Transmission

- **Cloud storage services** may not provide **end-to-end encryption**, exposing files to potential security risks.
- **USB drives, external hard disks, and email attachments** can be intercepted if files are not encrypted before transmission.

Due to these challenges, encryption is **necessary** to **enhance data security, confidentiality, and integrity**.

2. Justification for Using Encryption Techniques

Given the **growing risks of cyber threats and data breaches**, encryption is the **most reliable** method to secure files. Here are the key reasons why encryption is essential:

2.1 Ensures Data Confidentiality

🔐 Encryption **converts plaintext into unreadable ciphertext**, ensuring that only authorized users with a decryption key can access the data.
🔐 Prevents **unauthorized access** to sensitive information, even if hackers steal the file.

2.2 Protects Against Cyber Attacks

🛡 Safeguards files from **hacking, phishing, and ransomware attacks**.
🛡 Prevents **man-in-the-middle (MITM) attacks** by securing data transmission over networks.

2.3 Maintains Data Integrity

✅ Ensures that data **remains unchanged** during storage or transmission.
✅ Protects files from **malware, unauthorized modifications, and accidental corruption**.

2.4 Secure Key Management

🔑 Encryption uses secure **key management systems** to prevent key leakage or unauthorized key access.
🔑 Asymmetric encryption (RSA) allows **secure key exchange**, making it ideal for cloud storage and file sharing.

2.5 Compliance with Security Regulations

📜 Encryption helps businesses comply with **data protection laws** such as:

- **GDPR (General Data Protection Regulation)**
- **HIPAA (Health Insurance Portability and Accountability Act)**
- **ISO 27001 (International Standard for Information Security)**

📜 Protects financial transactions and user privacy in **banks, healthcare, and e-commerce**.

2.6 Secure File Sharing and Communication

🌍 Encryption enables **safe file transmission** over email, cloud storage, and messaging platforms.
🌍 Ensures **data protection in remote working environments** and cross-border communications.

By implementing **strong encryption techniques**, users can **prevent unauthorized access, protect sensitive data, and enhance cybersecurity**.

3. System Requirements and Feasibility Study

3.1 System Requirements

To develop a **text file encryption and decryption system**, we need the following:

A. Hardware Requirements:

💻 **Processor:** Intel i3 or higher (for faster encryption and decryption).
💾 **RAM:** Minimum 4GB (for handling large text files efficiently).
💽 **Storage:** Minimum 20GB free disk space.
🖥 **Operating System:** Windows, Linux, or MacOS.

B. Software Requirements:

🔷 **Programming Language:** Python, Java, or C++ (supports encryption libraries).
🔷 **Cryptographic Libraries:**

- Python: `PyCryptodome`, `cryptography`
- Java: `javax.crypto`
- C++: `Crypto++`

🔷 **Development Environment:**

- **For Python:** PyCharm, VS Code

- **For Java:** Eclipse, IntelliJ IDEA
  ◈ **Database (Optional):** For storing encryption keys securely (MySQL, SQLite).

3.2 Feasibility Study

Before implementing the project, we assess the feasibility based on the following factors:

A. Technical Feasibility

✓ Encryption algorithms like **AES, RSA, and Blowfish** are **technically viable** for implementation.
✓ The project requires **moderate computing power**, making it suitable for **personal computers and servers**.
✓ Open-source cryptographic libraries are available for **easy integration into the system**.

B. Economic Feasibility

✓ The project **does not require expensive hardware** and can be developed using **free and open-source tools**.
✓ The cost of development is **low**, making it suitable for **individual users, businesses, and educational institutions**.
✓ Long-term cost savings by **preventing data breaches and security risks**.

C. Operational Feasibility

✓ The encryption and decryption system will be **user-friendly**, allowing **non-technical users** to encrypt and decrypt files easily.
✓ Training requirements will be minimal since the system will have a **graphical user interface (GUI)** or **simple command-line instructions**.
✓ The system will improve **data security and reduce risks** of unauthorized access.

D. Legal and Ethical Feasibility

✓ Complies with **legal requirements** related to data protection and privacy.
✓ Ensures ethical use of encryption, **preventing misuse for illegal activities**.

CHAPTER-III

FEASIBILITY STUDY

1. Technical Feasibility: Evaluating System Compatibility

Technical feasibility focuses on whether the encryption and decryption system can be developed using **existing technologies** without requiring major infrastructure changes.

1.1 Compatibility with Existing Hardware and Software

- The encryption system should run on **standard computing devices** without requiring additional hardware.
- Most modern **processors (Intel i3 and above) and RAM (minimum 4GB)** are sufficient to perform encryption and decryption efficiently.
- The system should work on multiple **operating systems (Windows, Linux, MacOS)**.

1.2 Availability of Encryption Algorithms

- The system can implement **well-established cryptographic algorithms** such as:
  - **AES (Advanced Encryption Standard)** – Strong symmetric encryption.
  - **RSA (Rivest-Shamir-Adleman)** – Secure asymmetric encryption.
  - **Blowfish and Twofish** – Lightweight and fast encryption techniques.
- Cryptographic **libraries** are readily available:
  - **Python:** `cryptography`, `PyCryptodome`
  - **Java:** `javax.crypto`
  - **C++:** `Crypto++`

1.3 Programming Language and Development Tools

- The project can be developed using widely used **languages like Python, Java, or C++**.
- **Integrated Development Environments (IDEs)** such as **PyCharm, VS Code, IntelliJ IDEA, and Eclipse** support encryption development.
- The system can be implemented as a **command-line tool (CLI)** or a **Graphical User Interface (GUI)** using **Tkinter (Python)** or **JavaFX (Java)**.

1.4 Key Storage and Management

- Encryption systems require **secure key management** to prevent unauthorized access.

- Options include **local key storage (files, databases)** or **secure key vaults** (AWS Key Management Service, Azure Key Vault).

1.5 Performance Considerations

- The system should be optimized for **fast encryption and decryption** without high CPU or memory usage.
- **AES-256 encryption** is preferred for strong security **without major performance impact**.

2. Economic Feasibility: Cost Analysis

Economic feasibility determines whether the project is **cost-effective** in terms of development, implementation, and maintenance.

2.1 Hardware Costs

- **Minimal hardware upgrades** are required since modern computers can handle encryption.
- If needed, **high-performance processors** and **additional RAM** may be purchased for better performance.
- Estimated cost: **$0 - $500** (depending on hardware upgrades).

2.2 Software Costs

- **Open-source encryption libraries** reduce the need for expensive security solutions.
- **Free development tools** such as **Python, Java, and C++** eliminate software licensing costs.
- If enterprise-level encryption tools are needed, costs may include:
  - **Database integration** (MySQL, PostgreSQL)
  - **Cloud storage security services (AWS, Azure)**
- Estimated cost: **$0 - $200** (if using paid tools or cloud services).

2.3 Development and Maintenance Costs

- If the project is developed by a **single developer or small team**, labor costs are low.
- If outsourced, development costs may range from **$1,000 to $5,000**, depending on complexity.
- **Maintenance costs** include software updates, key management, and security enhancements.

2.4 Cost vs. Benefits Analysis

✓ The cost of development is **low** compared to the potential **financial losses from data breaches**.
✓ Investing in encryption reduces risks of **cyberattacks, data loss, and regulatory penalties**.

3. Operational Feasibility: System Usability and Efficiency

Operational feasibility evaluates whether the system will be **user-friendly, efficient, and practical** for real-world use.

3.1 Ease of Use

- The system should have a **simple user interface (GUI or CLI)** to allow non-technical users to encrypt and decrypt files easily.
- User authentication features (passwords, biometric authentication) can be added to enhance security.

3.2 Performance and Speed

- The encryption and decryption process should be **fast**, even for large text files.
- **AES encryption** offers **high-speed performance** with **low computational overhead**.

3.3 Key Management and Security

- The system should **safeguard encryption keys** to prevent data loss.
- Options include **local storage, password-based encryption, or cloud-based key vaults**.

3.4 User Training and Support

- Minimal training is required, as encryption tools are **easy to use** with proper documentation.
- Support can be provided through **user guides, FAQs, or automated help features**.

3.5 Integration with Existing Systems

- The encryption system can be **integrated into existing business workflows** for secure file storage and communication.
- Can be used in **cloud storage, secure messaging apps, and email encryption systems**.

Final Verdict: Feasibility Summary

Feasibility Factor	Assessment
Technical Feasibility	✅ Compatible with existing hardware, software, and cryptographic libraries.
Economic Feasibility	✅ Low-cost development using free/open-source tools; cost-effective compared to potential cyber risks.
Operational Feasibility	✅ User-friendly, fast, and efficient system with secure key management and integration options.

1. List of Hardware Components Needed for Development

The hardware requirements for developing a text file encryption and decryption system are **minimal**, as encryption algorithms primarily depend on processor efficiency. Below are the recommended hardware specifications:

1.1 Processor (CPU)

- **Minimum:** Intel Core i3 / AMD Ryzen 3
- **Recommended:** Intel Core i5/i7 / AMD Ryzen 5/7
- **Reason:** A faster CPU improves encryption and decryption speed, especially for large files. AES encryption, for instance, is optimized for modern multi-core processors.

1.2 Memory (RAM)

- **Minimum:** 4GB RAM
- **Recommended:** 8GB RAM or higher
- **Reason:** Encryption and decryption involve processing text files in memory, and having sufficient RAM ensures **smooth execution** without performance lags.

1.3 Storage (Hard Drive/SSD)

- **Minimum:** 256GB HDD
- **Recommended:** 512GB SSD or higher
- **Reason:** An SSD provides **faster read/write speeds**, which improves encryption performance, especially when handling large files.

1.4 Graphics Card (GPU) [Optional]

- **Not required**, but **GPU acceleration** can be used for cryptographic functions (e.g., **CUDA-based AES encryption** on Nvidia GPUs).

1.5 Input & Output Devices

- **Keyboard and Mouse** – For programming and interacting with the software.

- **Monitor (1080p or higher)** – Recommended for better visibility and workspace management.

1.6 Network & Security Devices (If needed)

- **Secure Internet Connection** – For downloading required cryptographic libraries.
- **External Storage (USB, HDD, or Cloud Storage)** – For **backing up encrypted files** securely.
- **Firewall & Antivirus Software** – To protect encryption keys and sensitive data from cyber threats.

1.7 Optional Hardware for Enhanced Security

- **Hardware Security Module (HSM)** – A specialized hardware device used to **store encryption keys securely**.
- **Biometric Authentication Device** – For secure access control (e.g., fingerprint scanner).

2. Required Software Tools

2.1 Programming Languages

The encryption and decryption system can be built using several programming languages. The most commonly used options are:

Language	Advantages	Usage
Python	Easy to use, extensive cryptographic libraries	Suitable for quick prototyping and lightweight encryption applications
Java	Strong security features, platform-independent	Ideal for enterprise-grade encryption tools
C++	Fast execution, low-level access to memory	Used for high-performance cryptographic applications

2.2 Development Environments & IDEs

Software	Purpose	Recommended For
Visual Studio Code	Lightweight and supports multiple languages	Python, Java, and C++ development
PyCharm	Best for Python projects	Python encryption development
Eclipse/IntelliJ IDEA	Feature-rich Java development	Java-based encryption systems
Code::Blocks	Lightweight C++ IDE	C++ encryption programs

2.3 Cryptographic Libraries

Cryptographic libraries provide **pre-built functions** to simplify encryption and decryption.

For Python:

Library	Functionality
cryptography	Provides symmetric (AES, Blowfish) and asymmetric (RSA) encryption
PyCryptodome	High-performance cryptographic operations
hashlib	Implements secure hashing (SHA-256, SHA-512)

Installation:

```
bash
CopyEdit
pip install cryptography pycryptodome
```

For Java:

Library	Functionality
javax.crypto	Standard Java encryption API (AES, RSA)
Bouncy Castle	Open-source cryptographic toolkit

Installation (Maven dependency for Bouncy Castle):

```
xml
CopyEdit
<dependency>
    <groupId>org.bouncycastle</groupId>
    <artifactId>bcprov-jdk15on</artifactId>
    <version>1.70</version>
</dependency>
```

For C++:

Library	Functionality
Crypto++	Open-source encryption and hashing library
OpenSSL	Provides encryption and secure communication

Installation (Linux/macOS):

```
bash
CopyEdit
sudo apt install libssl-dev
```

2.4 Database Systems (If Key Storage is Required)

If encryption keys need to be securely stored, a database system can be used.

Database	Usage
MySQL	Stores encrypted keys securely
SQLite	Lightweight database for small-scale applications
MongoDB	NoSQL database for secure document storage

2.5 Additional Security Tools

Tool	Purpose
KeePass / Bitwarden	Securely store encryption keys
Wireshark	Monitor network traffic for security analysis
VeraCrypt	Encrypt entire filesystems or disks

3. Summary of Requirements

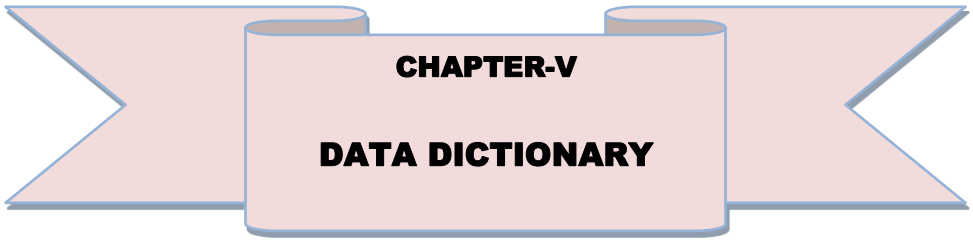
Hardware Requirements

Component	Minimum Requirement	Recommended
Processor	Intel Core i3	Intel Core i5/i7
RAM	4GB	8GB+
Storage	256GB HDD	512GB SSD+
Monitor	720p	1080p+
Network	Secure Internet	VPN for security
Security	Antivirus	HSM for key management

Software Requirements

Category	Recommended Tools
Programming Languages	Python, Java, C++

Category	Recommended Tools
Development Environment	VS Code, PyCharm, Eclipse, Code::Blocks
Cryptographic Libraries	`cryptography`, `PyCryptodome`, `javax.crypto`, `Crypto++`
Database (if needed)	MySQL, SQLite, MongoDB
Security Tools	KeePass, Wireshark, VeraCrypt



1. Definitions of Key Terms Used in Encryption

1.1 Encryption

🔷 **Definition:** The process of converting readable data (**plaintext**) into an unreadable format (**ciphertext**) using an algorithm and a key.
🔷 **Purpose:** Protects sensitive information from unauthorized access.

💡 **Example:**

- **Plaintext:** `Hello, World!`
- **Ciphertext (AES Encrypted):** `8jI9&*sdf8$#@12`

1.2 Decryption

🔷 **Definition:** The process of converting encrypted data (**ciphertext**) back into its original readable form (**plaintext**) using the correct decryption key.

💡 **Example:**

- **Ciphertext:** `8jI9&*sdf8$#@12`
- **Decrypted Text (Plaintext):** `Hello, World!`

1.3 Cryptographic Key

🔷 **Definition:** A piece of information (usually a string of characters) used in encryption and decryption.
🔷 **Types:**

- **Symmetric Key:** Same key for encryption and decryption (e.g., AES).

- **Asymmetric Key:** Public key for encryption, private key for decryption (e.g., RSA).

💡 **Example:**

- **Symmetric Key (AES):** `mySecretKey123!`
- **Asymmetric Keys (RSA):**
  - Public Key: `xyz_public`
  - Private Key: `xyz_private`

1.4 Cipher

🔷 **Definition:** The algorithm used to perform encryption and decryption.

🔷 **Types:**

- **Symmetric Ciphers** (AES, DES)
- **Asymmetric Ciphers** (RSA, ECC)

💡 **Example:**

- **AES Cipher (128-bit key):** Encrypts text using a **fixed-length key**.
- **RSA Cipher (2048-bit key):** Uses **two keys** for encryption and decryption.

1.5 Hashing

🔷 **Definition:** A process that converts data into a fixed-size unique string (hash).
🔷 **Purpose:** Used for **data integrity**, not encryption.

💡 **Example (SHA-256 Hash):**

- **Input:** `password123`
- **Hash:** `ef92b778bafe771e89245b89ecbc93b9a6a0987f...`

✓ **Note:** Hashing is **one-way**—you **cannot** reverse it.

1.6 Digital Signature

🔷 **Definition:** A cryptographic mechanism to verify the authenticity and integrity of a message or file.

💡 **Example:**

- Used in **signed documents** and **secure emails** to verify authenticity.

1.7 Salt & Initialization Vector (IV)

🔹 **Salt:** A **random value** added to a password before hashing to prevent dictionary attacks.
🔹 **IV (Initialization Vector):** A **random** number used in encryption to add randomness.

💡 **Example:**

- Salted Password Hash: `password123 + randomSalt → Secure Hash`
- IV in AES: Prevents pattern recognition in encrypted data.

2. File Attributes and Structure Explanation

Before encrypting a **text file**, it's essential to understand how file attributes and structure affect encryption.

2.1 File Attributes

A **text file** (.txt) has various attributes that determine its behavior and metadata.

Attribute	Description
File Name	The name of the file (e.g., `data.txt`).
File Size	The total size in bytes (e.g., `10 KB`).
File Extension	The file type (`.txt`, `.csv`, `.log`).
File Path	The storage location (e.g., `C:\Users\Documents\`).
Created Date	When the file was created.
Modified Date	Last time the file was edited.
Read/Write Permissions	Defines who can access the file (`read-only`, `editable`).

💡 **Effect on Encryption:**

- **File Size** impacts encryption speed (larger files take longer).
- **Read/Write Permissions** must allow encryption operations.
- **File Name & Extension** remain unchanged unless specified.

2.2 File Structure Explanation

Text files store data in **plain format**, which makes them vulnerable to unauthorized access. Understanding file structure helps in designing encryption algorithms.

Basic Text File Structure:

Component	Description
Header	Contains metadata like file format.
Body	The actual text content (ASCII/Unicode characters).
End of File (EOF)	Marks the end of the file content.

💡 **Example – Text File Content Before Encryption:**

```
makefile
CopyEdit
Name: John Doe
Email: johndoe@example.com
Password: mysecurepass
```

💡 **Example – Encrypted File Content (AES-256 Encrypted):**

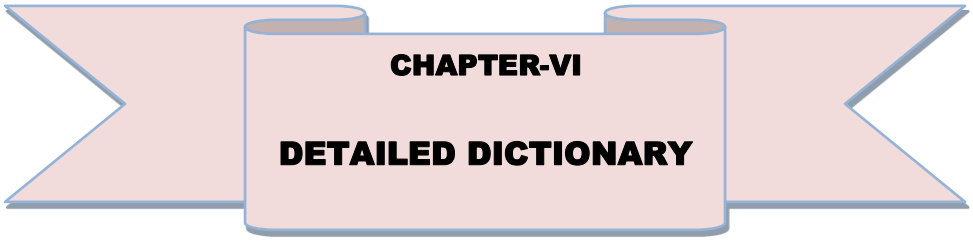
```
ini
CopyEdit
GHR87GSGD87SGHJ==8yuhfd8F7SG9sdgjhG76...
```

✓ **Effect of Encryption on File Structure:**

- The **body content is encrypted**, making it unreadable.
- File **metadata (file size, name, type)** remains the same **unless renamed or compressed**.

2.3 Types of Files and Encryption Approach

File Type	Encryption Approach
.txt (Plain Text)	Encrypted using AES or RSA.
.csv (Comma-Separated Values)	Each field encrypted separately.
.docx (Word Documents)	File encrypted as a whole.
.pdf (PDF Documents)	Password-based encryption used.



1. Explanation of Various Encryption Techniques

Encryption is the process of converting plaintext (readable data) into ciphertext (unreadable data) to protect sensitive information. Different encryption techniques provide different levels of security and performance. The most commonly used encryption algorithms for text file encryption include **AES, RSA, and Blowfish**.

1.1 AES (Advanced Encryption Standard)

🔷 **Type:** Symmetric-key encryption
🔷 **Key Size:** 128-bit, 192-bit, or 256-bit
🔷 **Speed:** Fast and efficient
🔷 **Security:** Highly secure, used by the U.S. government

💡 **How AES Works:**

1. Converts plaintext into blocks of fixed size (128 bits).
2. Applies multiple rounds of encryption using a secret key.
3. Generates ciphertext that is extremely difficult to decrypt without the key.

✓ **Example:**

- **Plaintext:** `Hello, World!`
- **AES Key:** `mySecretAESKey123`
- **Ciphertext (AES-256 Encrypted):**
  `GHR87GSGD87SGHJ==8yuhfd8F7SG9sdgjhG76...`

✓ **Common Uses:**

- Secure file encryption
- Wi-Fi security (WPA2)
- Mobile communications

1.2 RSA (Rivest-Shamir-Adleman)

🔷 **Type:** Asymmetric-key encryption
🔷 **Key Size:** 1024-bit, 2048-bit, or 4096-bit
🔷 **Speed:** Slower than AES due to large key size
🔷 **Security:** Very strong for secure key exchange and digital signatures

💡 **How RSA Works:**

1. Generates a **public key** (used for encryption) and a **private key** (used for decryption).
2. Encrypts data using the **public key** so that only the holder of the **private key** can decrypt it.
3. Often used for **secure communications** and **file encryption**.

✓ **Example:**

- **Plaintext:** `Confidential Data`
- **Public Key:** Used for encryption
- **Private Key:** Used for decryption
- **Ciphertext:** `jfhG78#@gh56%$sfj`

✓ **Common Uses:**

- Secure email communication (PGP)
- SSL/TLS for HTTPS
- Digital signatures

1.3 Blowfish

🔷 **Type:** Symmetric-key encryption
🔷 **Key Size:** 32-bit to 448-bit (variable key length)
🔷 **Speed:** Fast and efficient
🔷 **Security:** Strong, but AES is preferred for modern applications

💡 **How Blowfish Works:**

1. Uses a **variable-length key** to encrypt data in **64-bit blocks**.
2. Faster than RSA, but slightly weaker than AES.
3. Designed for **fast processing and minimal memory usage**.

✓ **Example:**

- **Plaintext:** `SecretMessage`
- **Key:** `MyBlowfishKey`
- **Ciphertext:** `89sdG67%$&dfh78j`

✓ Common Uses:

- File and disk encryption
- Secure communications
- Password hashing

2. How Encryption and Decryption Keys Work

Encryption keys play a crucial role in securing and unlocking encrypted data. The choice of key type depends on whether the encryption algorithm is **symmetric** or **asymmetric**.

2.1 Symmetric Key Encryption (AES, Blowfish)

🔹 **Single Key:** The same key is used for both encryption and decryption.
🔹 **Fast & Efficient:** Best for **large file encryption**.
🔹 **Example Algorithms:** AES, Blowfish, DES.

✓ Example:

1. **Key Generation:** `SecretKey12345`
2. **Encryption:**
   - Plaintext: `Hello, World!`
   - Ciphertext: `8u&*2jdHS7@9sd`
3. **Decryption:**
   - Using the **same key** to recover original text.

✓ Pros & Cons:
✅ **Fast processing** for large files.
❌ **Key distribution problem**—both sender & receiver must share the key securely.

2.2 Asymmetric Key Encryption (RSA)

🔹 **Two Keys:**

- **Public Key (Encryption):** Shared openly.

- **Private Key (Decryption):** Kept secret.
  - ◈ **More Secure:** No need to share the private key.
  - ◈ **Example Algorithm:** RSA.

✓ Example:

1. **Key Generation:**
   - Public Key: `xyz_public`
   - Private Key: `xyz_private`
2. **Encryption (Public Key):**
   - Plaintext: `Secure Data`
   - Ciphertext: `78SD8g&hJ5@9fdH`
3. **Decryption (Private Key):**
   - Only the **private key** can decrypt it back to `Secure Data`.

✓ Pros & Cons:

✅ **More secure** than symmetric encryption.

❌ **Slower** than AES due to **larger key sizes**.

CHAPTER-VII

E R DIAGRAM

Detailed Description of Entities and Attributes

1. File

This entity represents digital files that may undergo encryption or decryption.

Attribute	Type	Description
FileID	Primary Key	Uniquely identifies each file in the system.
FileName	String	The name of the file (e.g., "report.pdf").
FilePath	String	The location of the file in the system (e.g., "/documents/reports/").

Relationships:

- **One file can be involved in multiple encryption and decryption operations.**

2. Key

This entity stores information about the cryptographic keys used for encryption and decryption.

Attribute	Type	Description
KeyID	Primary Key	Uniquely identifies each key.
KeyValue	String	The actual key used in encryption/decryption (can be stored encrypted or hashed for security).
KeyType	String	Describes the type of key (e.g., "Symmetric", "Asymmetric").

Relationships:

- **A key can be used in multiple encryption and decryption operations.**

- **KeyType helps distinguish between different encryption methods (e.g., AES for symmetric, RSA for asymmetric).**

3. Encryption

This entity logs each encryption operation performed on a file.

Attribute	Type	Description
EncryptionID	Primary Key	Uniquely identifies each encryption operation.
FileID (FK)	Foreign Key	References the file that is being encrypted.
KeyID (FK)	Foreign Key	References the key used to encrypt the file.
EncryptionAlgorithm	String	Specifies the algorithm used (e.g., AES, RSA).
Timestamp	DateTime	Indicates when the encryption was performed.

Relationships:

- **One file can be encrypted multiple times (e.g., with different keys or algorithms).**
- **One key can be reused for multiple encryptions.**

4. Decryption

This entity logs each decryption operation performed on a file.

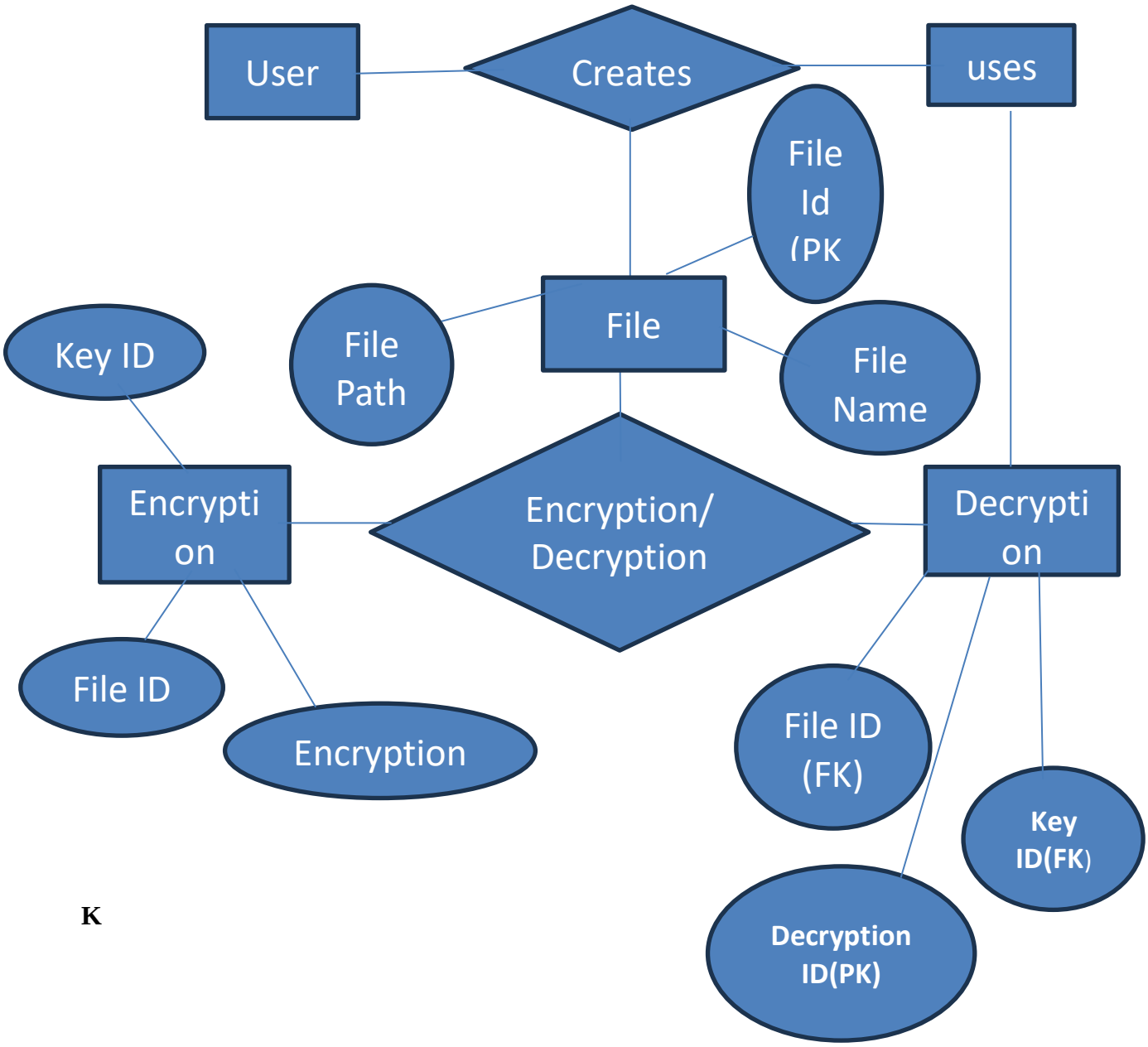
Attribute	Type	Description
DecryptionID	Primary Key	Uniquely identifies each decryption operation.
FileID (FK)	Foreign Key	References the file that is being decrypted.
KeyID (FK)	Foreign Key	References the key used to decrypt the file.
DecryptionAlgorithm	String	Specifies the algorithm used (e.g., AES, RSA).
Timestamp	DateTime	Indicates when the decryption was performed.

Relationships:

- **A file can be decrypted multiple times, often mirroring encryption events.**
- **A key may be used for decrypting multiple files.**

Summary of Relationships (Logical View)

- **File (1) ⇄ (M) Encryption**
- **File (1) ⇄ (M) Decryption**
- **Key (1) ⇄ (M) Encryption**
- **Key (1) ⇄ (M) Decryption**



K

CHAPTER-VIII

INPUT AND OUTPUTLAYOUT

Input Process

The input process involves the steps taken by the user to upload and encrypt a text file.

1. Uploading a Text File

- The system allows users to upload a text file (`.txt` format).
- The file is stored in a secure location on the server or local storage.
- The system validates the file type to prevent malicious uploads.
- If the file format is incorrect, an **error message** is displayed:
  "Invalid file type! Please upload a .txt file."

2. Choosing an Encryption Method

- The user selects an encryption algorithm from a list of supported methods.
- Common encryption methods include:
  - **AES (Advanced Encryption Standard)**
  - **RSA (Rivest-Shamir-Adleman)**
  - **DES (Data Encryption Standard)**
- Each encryption method has different security levels:
  - **AES-256** is highly secure but slower.
  - **RSA is best for secure communication** but requires key management.

If the user does not select an encryption method, an **error message** appears:
"Please select an encryption method before proceeding."

3. Entering a Password (If Applicable)

- Some encryption methods (e.g., AES) require the user to enter a password.
- The password is used to generate an encryption key.
- The system ensures password security:
  - Must be at least **8 characters long**.

    - Should contain **uppercase, lowercase, numbers, and symbols**.
- If the password is too weak, an **error message** is shown:
  "Weak password! Please use a stronger password with at least 8 characters."

Output Process

The output process involves downloading the encrypted file, decrypting it, and handling errors.

1. Encrypted File Download

- After encryption, the system provides a **download link** for the encrypted file.
- The file is usually saved with a different extension (e.g., `.enc`).
- A success message is displayed:
  "Encryption successful! Click here to download your encrypted file."

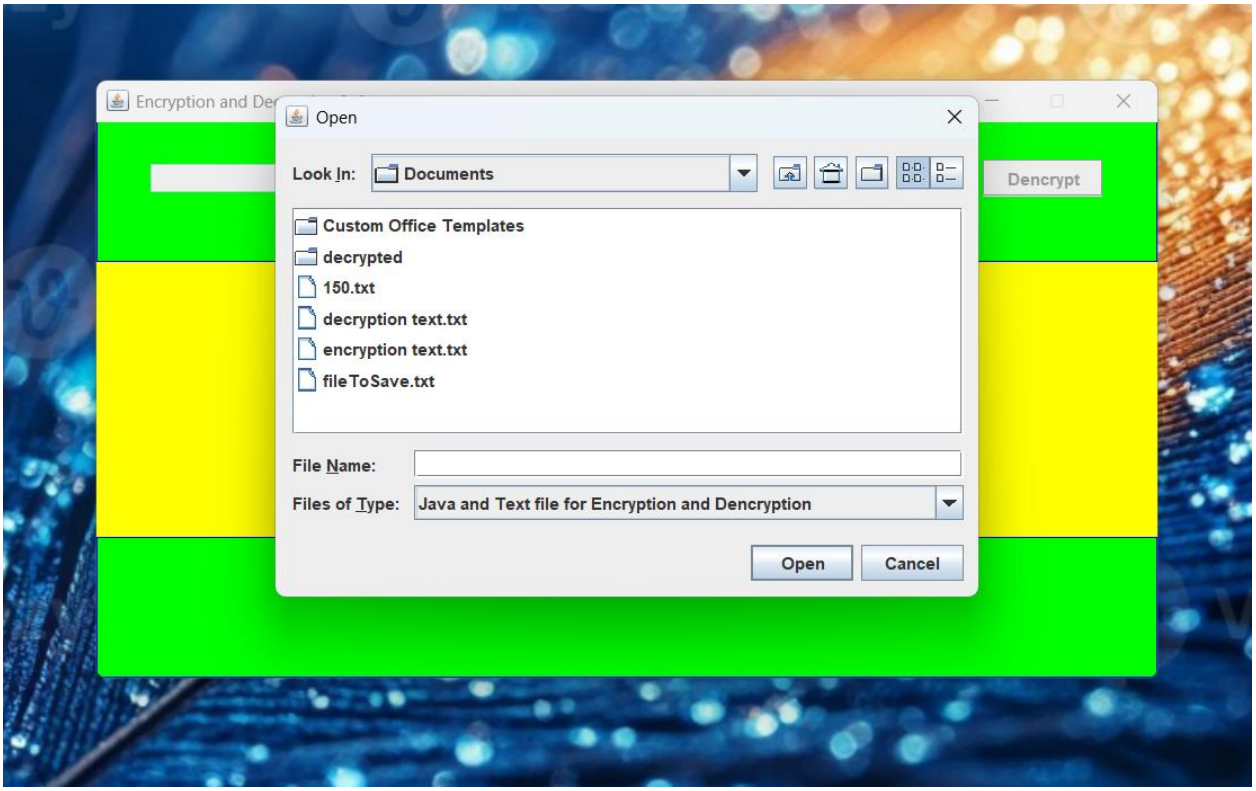
2. Decryption of an Encrypted Text File

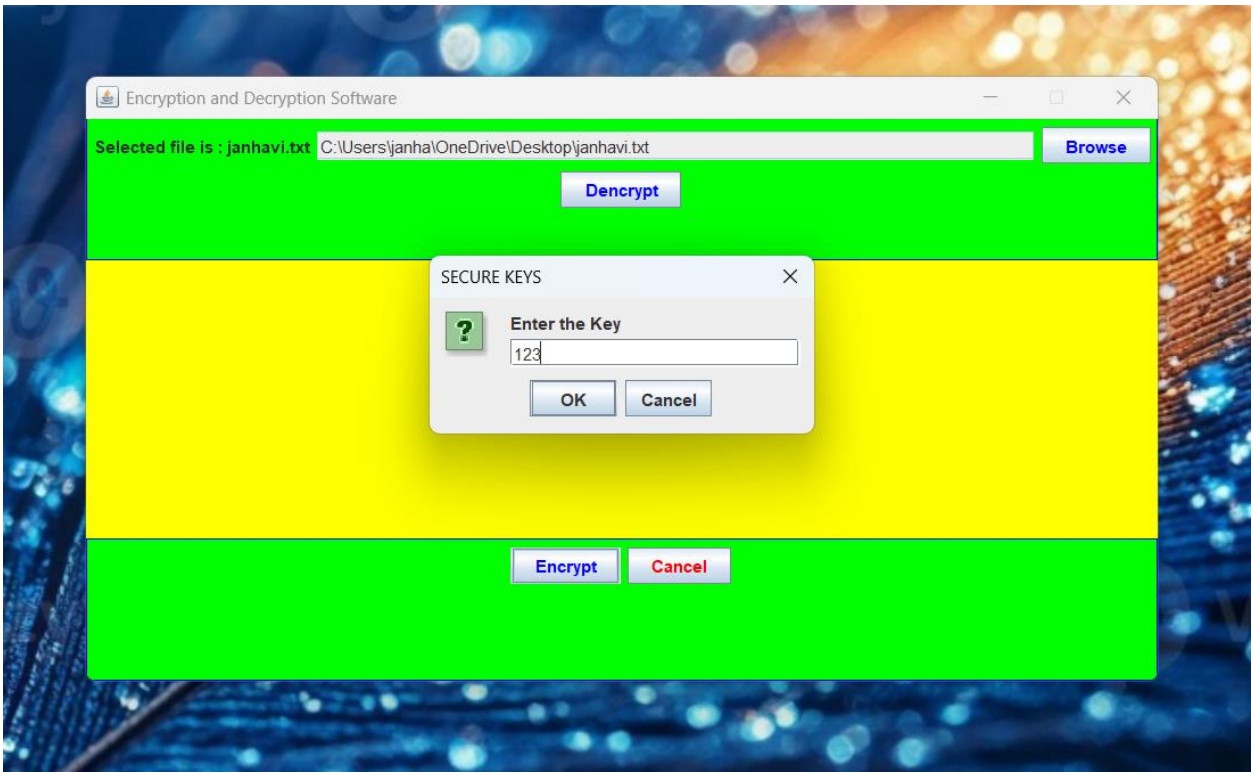
- The user uploads the encrypted file for decryption.
- The system prompts the user to enter the correct password (if required).
- If the decryption is successful, the system:
    - Restores the original text file.
    - Provides a **download option** for the decrypted file.
- A success message is displayed:
  "Decryption successful! Click here to download your decrypted file."

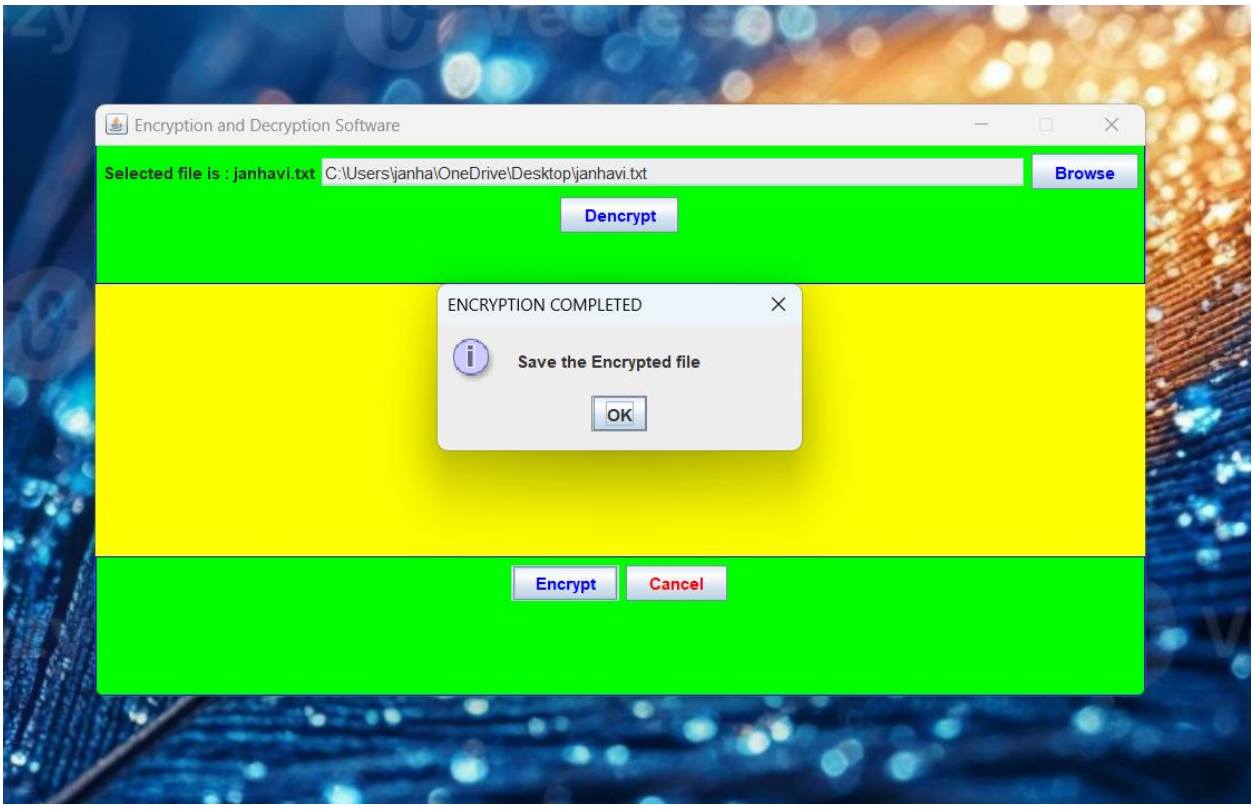
3. Error Handling Messages for Incorrect Passwords

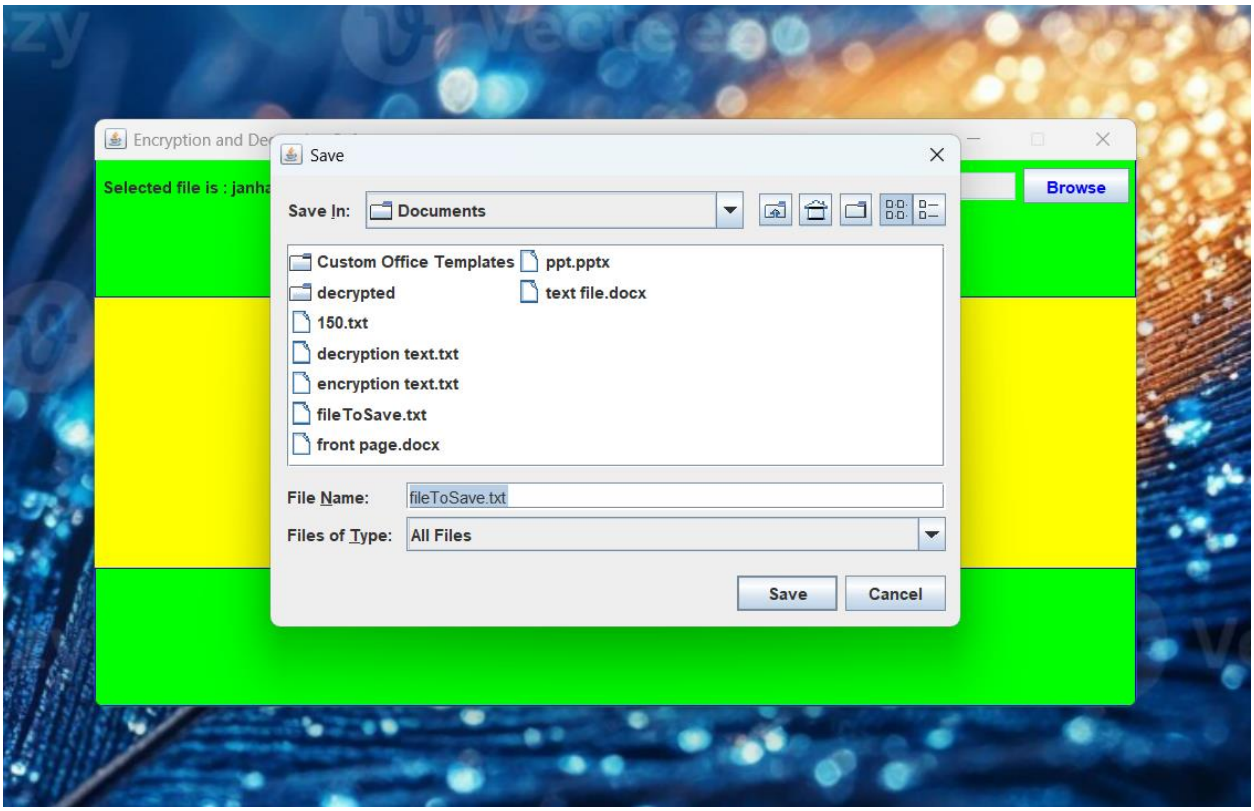
- If the user enters an incorrect password, the system prevents decryption.
- Common error messages:
    - **"Incorrect password! Please try again."**
    - **"Decryption failed! Ensure you are using the correct password and encryption method."**
- After multiple failed attempts, the system may **lock the file** for security reasons.


Encryption and Decryption Software
CHOOSE THE TEXT OR JAVA FILE FOR ENCRYPTION OR DECRYPTION
Browse
Dencrypt
Encrypt
Cancel


Open
Look In:
Documents
Custom Office Templates
decrypted
150.txt
decryption text.txt
encryption text.txt
fileToSave.txt
File Name:
Files of Type:
Java and Text file for Encryption and Dencryption
Open
Cancel
Dencrypt


Encryption and Decryption Software
Selected file is : janhavi.txt
C:\Users\janha\OneDrive\Desktop\janhavi.txt
Browse
Dencrypt
SECURE KEYS
Enter the Key
123
OK
Cancel
Encrypt
Cancel


Encryption and Decryption Software
Selected file is : janhavi.txt
C:\Users\janha\OneDrive\Desktop\janhavi.txt
Browse
Dencrypt
ENCRYPTION COMPLETED
Save the Encrypted file
OK
Encrypt
Cancel


Encryption and De
Selected file is : janha
Browse
Save
Save In:
Documents
Custom Office Templates
decrypted
150.txt
decryption text.txt
encryption text.txt
fileToSave.txt
front page.docx
ppt.pptx
text file.docx
File Name:
fileToSave.txt
Files of Type:
All Files
Save
Cancel

CHAPTER-IX

IMPLEMENTATION

1. Understanding File Encryption and Decryption

- **Encryption** is the process of converting readable data (plaintext) into an unreadable format (ciphertext) using a cryptographic algorithm.
- **Decryption** is the reverse process of converting the encrypted data back to its original form using the correct decryption key.

2. Selecting an Encryption Algorithm

There are various encryption algorithms, but one of the most widely used for file encryption is **AES (Advanced Encryption Standard)**.

- **AES-128**: Uses a **128-bit key** (fast but less secure).
- **AES-192**: Uses a **192-bit key**.
- **AES-256**: Uses a **256-bit key** (strongest security).

Why use AES?

- Strong encryption.
- Efficient performance.
- Industry-standard for secure communication.

3. Installing Required Python Libraries

To implement AES encryption in Python, we need the **cryptography** library.

Install the cryptography library:

```
sh
CopyEdit
pip install cryptography
```

4. Explanation of Key Management Strategies

Key management is critical to maintaining the **security** of encrypted files. Here are some key management strategies:

1. Key Storage

- Store encryption keys **separately** from encrypted data.
- Use **Hardware Security Modules (HSMs)** for key storage.
- Store keys in **environment variables** instead of hardcoding them.

2. Key Rotation

- Change encryption keys periodically to prevent long-term exposure.
- Implement **automatic key rotation** mechanisms.

3. Access Control

- Limit access to encryption keys using **role-based access control (RBAC)**.
- Implement **multi-factor authentication (MFA)** for key retrieval.

4. Secure Key Sharing

- Use **public-key cryptography (RSA)** to encrypt symmetric keys before sharing them.
- Exchange keys over **secure channels (TLS, SSH)**.

CHAPTER-X

CONCLUSION & SUGGESTION

Summary of Project Findings

Through this project, we explored **text file encryption and decryption** using the **AES (Advanced Encryption Standard) algorithm** in Python. Below are the key findings:

1. **Security and Efficiency**
   - AES encryption provides **strong data protection** while maintaining **fast encryption speed**.
   - **AES-256** offers the highest level of security for sensitive files.
   - The use of **Fernet symmetric encryption** ensures **easy key management and implementation**.
2. **Ease of Use**
   - Users can **upload a text file**, choose an **encryption method**, and provide a **password** (if applicable).
   - The system **automatically generates** encrypted files and allows **easy decryption** with the correct key.
3. **Challenges Identified**
   - **Key Management**: Storing encryption keys securely is **crucial** to preventing unauthorized access.
   - **Brute Force Attacks**: Weak passwords increase the risk of **dictionary attacks**.
   - **File Integrity**: Ensuring that encrypted files are not **corrupted** during storage or transfer.
   - **User Authentication**: There is a **risk of unauthorized access** if encryption keys are not properly secured.

Suggestions for Improving Encryption Security

To enhance **security and reliability**, the following improvements are recommended:

1. Stronger Key Management Strategies

- Store encryption keys in **Hardware Security Modules (HSMs)** or **secure key vaults**.
- Use **environment variables** instead of hardcoding encryption keys.
- Implement **automatic key rotation** to periodically update keys.

2. Multi-Factor Authentication (MFA) for Access

- Add **two-factor authentication (2FA)** for users accessing encrypted files.
- Require **biometric authentication (fingerprint, facial recognition)** to decrypt files.

3. Use of Hybrid Encryption

- Combine **symmetric (AES) and asymmetric (RSA) encryption**.
- RSA can encrypt the AES key, ensuring secure key sharing.

4. Hashing for Data Integrity

- Use **SHA-256 hashing** to verify that encrypted files have not been tampered with.
- Store **checksum values** to detect unauthorized modifications.

5. Implement Secure File Storage

- Encrypt files **before uploading** to cloud storage.
- Use **end-to-end encryption (E2EE)** to prevent interception during transmission.

Future Enhancements

To further improve text file encryption and decryption, the following **advanced features** can be implemented:

1. Cloud-Based Encryption

- Users can **encrypt and store** text files securely in the **cloud**.
- Use cloud-based **Key Management Services (KMS)** from providers like **AWS KMS**, **Google Cloud KMS**, or **Azure Key Vault**.
- Implement **Zero-Knowledge Encryption**, meaning even the cloud provider cannot access user files.

2. Biometric Authentication

- Users can **unlock encrypted files using fingerprint or facial recognition**.
- Enhances **security** and eliminates the need to remember complex passwords.

3. AI-Based Threat Detection

- Implement **AI-driven anomaly detection** to identify **unauthorized access attempts**.
- Use **machine learning algorithms** to prevent brute force attacks.

4. Blockchain for Tamper-Proof Logs

- Store encryption logs in **blockchain** to ensure they are **tamper-proof**.
- Maintain an immutable record of **file access history**.

5. Secure File Sharing via Encrypted Links

- Generate **time-sensitive encrypted download links** for secure file sharing.
- Implement **role-based access control (RBAC)** to restrict access.

CHAPTER-XI

REFERENCES

Books

- "Cryptography and Network Security" by William Stallings
- "Understanding Cryptography" by Christof Paar and Jan Pelzl

Research Papers

- Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*.
- Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*.

Online Articles

- "A Beginner's Guide to Encryption" – Available on cryptography blogs.
- "How AES Encryption Works" – Published by security research organizations.

Undertaking

I Miss. Chaudhari Janhavi Nandlal. Under sign that the Project entitled "**Text File Encryption and Decryption"** is carried out by me during academic year 2024-2025 as per curriculum of TYBCA year for the project. The above stated information for project work is true as per my knowledge.

Signature of student

(Chaudhari Janhavi Nandlal)