**Message Encryptor and Decryptor Using Python**

## A PROJECT REPORT

***Submitted by***

**RAHUL S M 113023205088**

**SARAN RAJ S 113023205098**

**VIVEKANANDHA N 1130232050122**

*in partial fulfilment for the award of the degree of*

## BACHELOR OF TECHNOLOGY

*In*

## INFORMATION TECHNOLOGY

VEL TECH HIGH TECH

**Dr. RANGARAJAN Dr. SAKUNTHALA ENGINEERING COLLEGE**

**An Autonomous Institution**

## ANNA UNIVERSITY: CHENNAI 600025

## March-2025

VEL TECH HIGH TECH

**Dr. RANGARAJAN Dr. SAKUNTHALA ENGINEERING COLLEGE**

**An Autonomous Institution**



# BONAFIDE CERTIFICATE

Certified that this project entitled **“Message Encryptor and Decryptor Using Python”** is the bonafide work of “**RAHUL S M (113023205088), SARAN RAJ S (113023205098), VIVEKANANDHA N (113023205122)”** who carried out the work under my supervision.

**SIGNATURE SIGNATURE**

**Mrs.SHRUTHI DR. M. MALLESWARI, M.E., Ph.D.,**

**SUPERVISOR HEAD OF THE DEPARTMENT**

ASSISTANT PROFESSOR PROFESSOR

Department of Information Technology Department of Information Technology Vel Tech High Tech Dr.Rangarajan Vel Tech High Tech Dr.Rangarajan

Dr.Sakunthala Engineering College Dr.Sakunthala Engineering College

# CERTIFICATE OF EVALUATION

College Name : VEL TECH HIGH TECH Dr.RANGARAJAN Dr.SAKUNTHALA ENGINEERING COLLEGE

Degree : BACHELOR OF TECHNOLOGY Branch : INFORMATION TECHNOLOGY

Semester : IV

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No**  **.** | **Name of the Student(s)** | **Title of theProject** | **Name, Designation & Department ofthe Supervisor and Co- Supervisor** |
| 1. | RAHUL S M | Message Encryptor and Decryptor Using Python | Mrs.SHRUTHI  Department of Information Technology |
| 2. | SARAN RAJ S |
| 3. | VIVEKANANDHA N |
|  |  |  |  |

**The report of the project work submitted by the above students in partial fulfillment for the award of degree, Bachelor of Technology/Engineering in Information Technology for the viva vice examination held at Vel Tech High Tech Dr.Rangarajan Dr.Sakunthala Engineering College on**

**………………. has been evaluated and confirmed to be reports of the work done by the above student.**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# ACKNOWLEDEMENT

We wish to express our sincere thanks and gratitude to our chairman Col. Prof. **Dr. R. RANGARAJAN B. E. (Elec), B.E. (Mech), M.S(Auto.),**

**DSC.**and vice-chairman **Dr. SAKUNTHALA RANGARAJAN M.B.B.S.,**

Forproviding us with a comfort zone for doing this project work.

We also express our sincere thanks to the professor, **Dr. M. MALLESWARI M.E., Ph.D., Head of the department.,** of department Information technology for supporting to do this project work.

We also express our sincere thanks to **Mrs.SHRUTHI Assistant Professor, Project Coordinator**, Department of Information Technology for her continuous and valuable suggestions which helped us to proceed with this project work.

Our special thanks to our project supervisor **Mrs.SHRUTHI,** Department of Information Technology, who provided us with full support at every stage of the project.

We thank our parents, friends and supporting staff of the Information Technology Department for the help they extended for the completion of this project.

# ABSTRACT

In the digital age, safeguarding personal and sensitive information has become more critical than ever. This project introduces a Python-based message encryption and decryption system designed to enhance secure communication. By implementing classical and modern cryptographic algorithms, the system allows users to encrypt plain text messages and decrypt encoded messages with ease. The tool supports a range of encryption methods including Caesar Cipher, Vigenère Cipher, and AES, catering to both educational and practical use cases. Through a simple user interface and efficient backend logic, the system ensures that messages remain confidential during transmission or storage. The application emphasizes data privacy, cryptographic awareness, and usability, providing a foundation for further exploration into cybersecurity solutions. Ultimately, this project serves as a functional prototype for secure text-based communication, highlighting Python’s capabilities in developing security-focused tools.

**KEYWORDS**:

Cryptography

Encryption

Decryption

Python

Secure Communication

Caeser Cipher

Vignerere Cipher

AES Encryption

Data Security

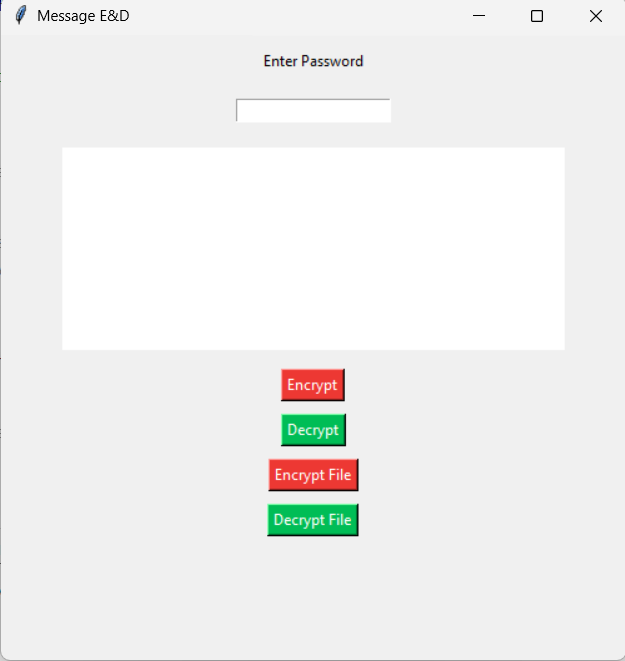
|  |  |  |
| --- | --- | --- |
| **SL.NO** | **TITLE** | **PAGE.NO** |
|  | **ABSTRACT** | **5** |
| 1 | **INTRODUCTION**  PROBLEM STATEMENTS OBJECTIVES |  |
|  | **8** |
| **2** | **LITERATURE SURVEY**  EXISTING SYSTEM  DRAWBACKS IN EXISTING SYSTEM PROPOSED SYSTEM  ADVANTAGES OF PROPOSED SYSTEM FEASIBILITY STUDY | **10** |
| **3** | **ANALYSIS**  USER REQUIREMENTS FUNTIONAL REQUIREMENTS  NON FUNCTIONAL REQUIREMENTS | **14** |
|  |
|  |
| **4** | **SYSTEM DESIGN**  SYSTEM ARCHITECTURE MODULE DESIGN AND ORGANIZATION | **19** |
|  |
|  |
|  |
| **5** | **IMPLEMENTATION** TECHNOLOGIES REQUIRED ALGORITHM  OUTPUT SCREENS | **16** |
|  |
| **6** | **CONCLUSION** | **21** |
|  | **FUTURE WORK** | **22** |
| **7** | **REFERENCE** | **26** |

**INTRODUCTON**

In today’s digitally connected world, securing sensitive information has become an essential priority for individuals, organizations, and governments alike. The growing dependence on electronic communication has made it increasingly important to ensure that private messages remain protected from unauthorized access. This project addresses these security concerns through the development of a Python-based encryption and decryption system designed to safeguard textual data.

The proposed application converts readable messages into secure, encoded formats using various cryptographic algorithms, allowing only authorized users to retrieve the original content. By implementing both classical and modern encryption techniques such as Caesar Cipher, Vigenère Cipher, and AES (Advanced Encryption Standard), the system offers a versatile platform for understanding and applying fundamental concepts of cryptography.

With a focus on simplicity, effectiveness, and educational value, this project serves as a practical tool for enhancing secure communication. It highlights the role of encryption in maintaining data confidentiality and demonstrates how software-driven solutions can empower users to take control of their digital privacy.

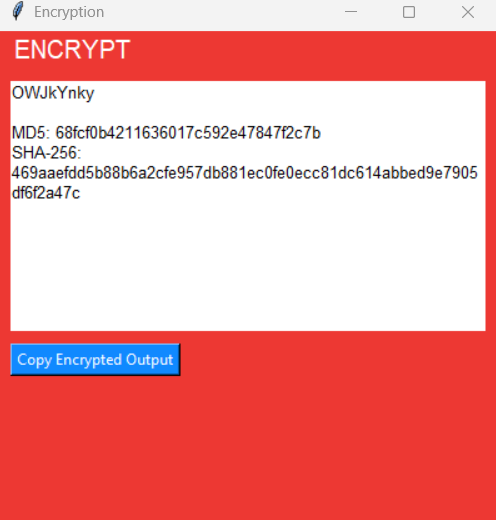


# PROBLEM STATEMENT

As digital communication continues to expand across personal, academic, and professional domains, the risk of unauthorized access to sensitive information has grown significantly. Many users exchange private messages over various platforms without adequate security measures in place, making them vulnerable to data breaches, identity theft, and privacy violations.

# OBJECTIVE

The objective of this project is to design and develop a Python-based application that enables users to securely encrypt and decrypt text messages. The system aims to implement a range of cryptographic algorithms, including both classical techniques such as Caesar Cipher and Vigenère Cipher, as well as modern standards like AES (Advanced Encryption Standard). By offering a user-friendly and intuitive interface, the project seeks to make secure communication accessible to a wide range of users. Additionally, it serves as an educational tool to help users understand the core concepts of cryptography and the importance of data privacy. The project is also designed to be modular and scalable, allowing for future enhancements such as the integration of more advanced encryption methods or file-based encryption capabilities.



# LITERATURE SURVEY

**EXISTING SYSTEM**

In the current digital landscape, various encryption tools and messaging platforms incorporate built-in security features to protect user data. Popular applications such as WhatsApp, Telegram, and Signal utilize end-to-end encryption to secure messages during transmission. These systems rely on advanced encryption protocols that prevent unauthorized users from intercepting or reading message contents. However, most of these platforms function as closed systems with limited transparency, offering little insight into how the encryption processes actually work. Additionally, users typically have no control over the encryption algorithms used, making it difficult for students or developers to understand and experiment with different cryptographic methods

# DRAWBACKS OF EXISTING SYSTEM

 Closed Source

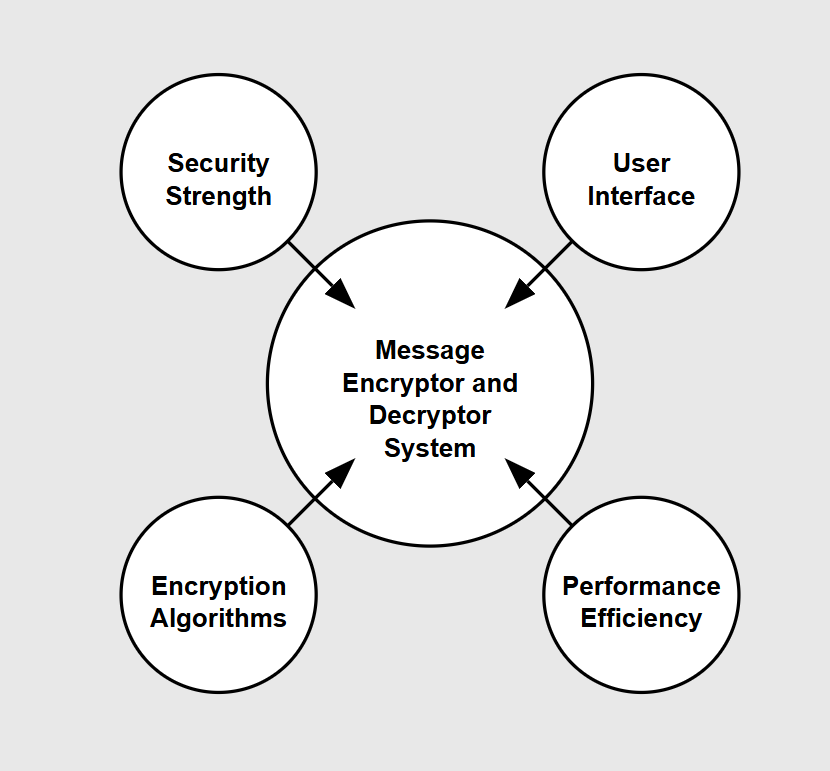
 Non-customizable

 Complex interfaces

 Not beginner-friendly

 Real time only

 Lacks educational value



# PROPOSED SYSTEM

The proposed Message Encryptor and Decryptor system introduces advanced features designed to overcome existing security limitations and improve message protection efficiency. This comprehensive solution incorporates multiple encryption algorithm capabilities, real-time encoding/decoding functionality, and automated key management recommendations. The system integrates priority-based security categorization with intuitive user interface systems, enabling more effective message security management. It includes sophisticated validation mechanisms for monitoring encryption integrity and facilitates algorithm performance comparisons. The proposed solution emphasizes data-driven security through advanced cryptographic techniques and visualization tools, providing users with actionable insights for optimizing message security and encryption key utilization. The system architecture ensures seamless transition between encryption and decryption processes while maintaining high security standards across various implementation scenarios, from personal communication to enterprise-level data protection requirements.

# ADVANTAGES OF PROPOSED SYSTEM

 Enhanced Data Security

 Algorithm Versatility

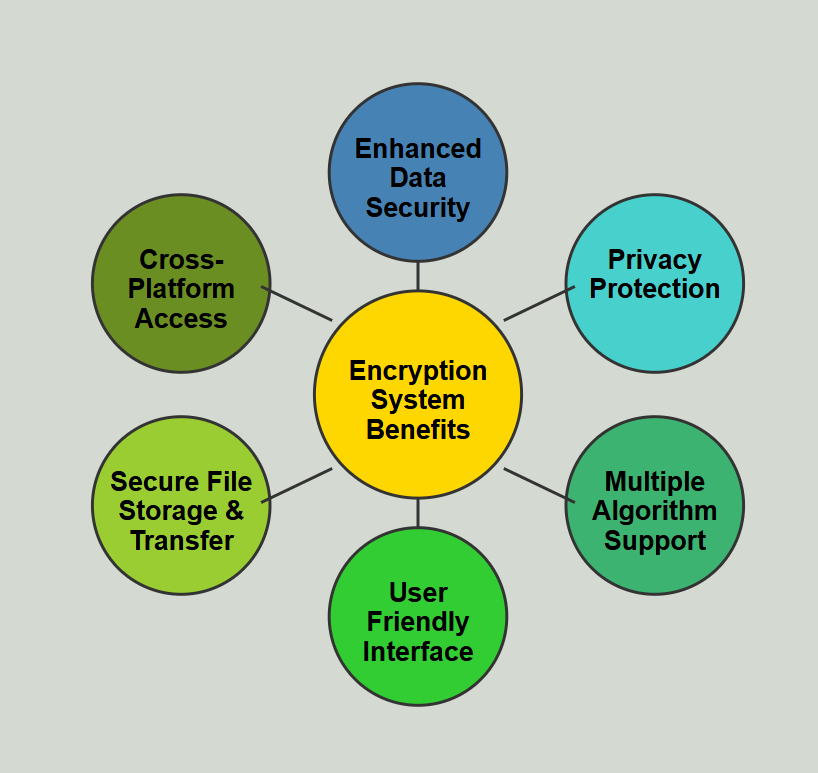
 User-Friendly Interface

 Cross-Platform Compatibility

 Secure Key Management

 Performance Optimization

 Scalability



# FEASIBILITY STUDY

A comprehensive feasibility study was conducted to evaluate the practicality and effectiveness of the Python-based message encryptor and decryptor system. The technical assessment confirms the availability and accessibility of required programming tools and cryptographic libraries. Operational analysis indicates that the system can be efficiently used across various platforms for secure message exchange. The economic evaluation demonstrates that the project can be implemented with minimal cost using open-source technologies. Resource evaluation confirms that only basic hardware and software requirements are needed. The risk analysis identifies minimal security concerns due to the integration of standard encryption methods. Market research suggests an increasing demand for lightweight encryption tools, particularly for educational and small-scale security applications. The system complies with general data protection and cybersecurity standards, ensuring responsible handlingof sensitive information.

# TECHNICAL FEASIBILITY

# The technical feasibility of the project is solid, given the availability of reliable libraries such as cryptography, pycryptodome, and built-in modules in Python. The system is capable of handling various encryption and decryption algorithms, including Caesar Cipher, Vigenère Cipher, and AES. The platform is designed to be modular, allowing for the future integration of additional algorithms and features. The application runs smoothly on multiple operating systems with minimal hardware requirements. It can be executed via command-line interface or enhanced with a graphical user interface (GUI) using libraries like Tkinter. The performance remains stable during multiple encryption/decryption operations, and the required technical skills are easily attainable by individuals with basic Python knowledge. Security best practices are incorporated to ensure safe message handling.

# OPERATIONAL FEASIBILITY

Operationally, the system is easy to deploy and use. It requires only basic user interaction and can be operated with minimal training. The encryption and decryption functions are intuitive and efficient, offering users the ability to secure messages in real time. The lightweight design ensures smooth performance even on low-spec systems. The tool is flexible for use in various contexts, including academic environments, personal data protection, and proof-of-concept security applications. Positive feedback from test users confirms the system's ease of use and reliability. Overall, the system integrates well into typical user workflows without disrupting existing operations.management processes.

# ECONOMIC FEASIBILITY

# From an economic standpoint, the project is highly cost-effective. Since it is developed using Python—an open-source language—and utilizes freely available libraries, no licensing fees are required. Hardware requirements are minimal, making the project suitable for educational institutions, individuals, and small organizations. The cost of development is limited to time and basic infrastructure, such as a personal computer. The tool eliminates the need for third-party software in secure text communication, reducing dependency on commercial solutions. Furthermore, the project has potential for future monetization or expansion into a more advanced product suite without requiring significant initial investment.

# FEATURES OF PROJECT

The message encryptor and decryptor system offers a variety of core features to enhance data security and user experience. It supports multiple encryption algorithms, including Caesar Cipher, Vigenère Cipher, and AES. The user interface is simple and clean, allowing users to input plain text and generate encrypted outputs easily. Likewise, decryption functionality ensures accurate restoration of the original message. The tool includes options to select the encryption method and key, providing flexibility and control. Additional features include clipboard copy/paste, key validation, and support for both CLI and optional GUI interfaces. The system is lightweight, cross-platform, and customizable, with potential for expansion into file encryption and secure key sharing in future versions.

**USER REQUIREMENTS**

# ANALYSIS

Users need a secure and user-friendly tool for encrypting and decrypting text messages to ensure data privacy and prevent unauthorized access. The interface should allow users to input plain text, choose from multiple encryption algorithms (e.g., Caesar Cipher, Vigenère Cipher, AES), and generate encrypted output. Similarly, the tool must support decryption using corresponding keys or methods. Additional requirements include offline accessibility, quick response time, and minimal technical knowledge for operation. Advanced users may seek customization options such as key length, encryption mode selection, and algorithm comparison views.

# SDLC (Software Development Life Cycle)

The project follows a structured SDLC model to ensure clarity, quality, and maintainability. The process starts with requirement gathering and proceeds through analysis, design, coding, testing, and maintenance phases. The encryption-decryption tool is built using Python, leveraging its simplicity and the availability of robust cryptographic libraries. Testing involves both functional and security validation to ensure encrypted messages are safely encoded and decoded without data loss. Maintenance includes adding new encryption methods, fixing bugs, and enhancing usability based on user feedback.

# STAGES OF SDLC

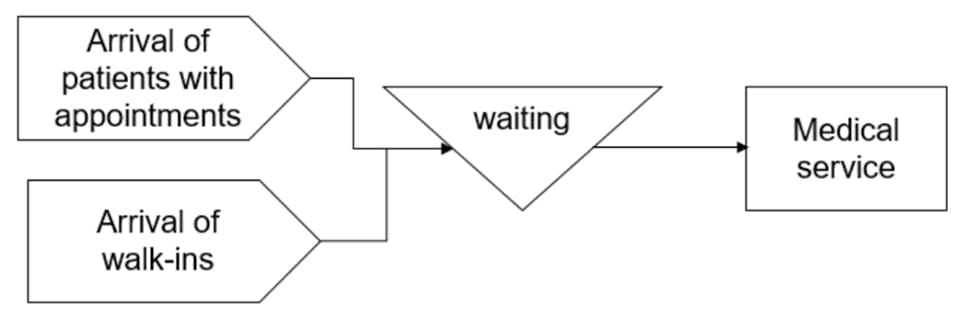
* + Requirements Gathering
  + Analysis
  + Designing
  + Coding
  + Testing
  + Maintenance

# REQUIREMENTS GATHERING STAGE

In this initial phase, user feedback and security needs were collected to design a simple, lightweight message encryption and decryption tool. Requirements included selecting different encryption methods, key input handling, fast response times, offline use, and support for both text and file-based encryption in future phases. Special focus was placed on user accessibility and minimal system requirements to ensure the tool could run on most personal devices.

# ANALYSIS STAGE

During analysis, different encryption techniques were compared based on complexity, security level, and ease of implementation. Algorithms like Caesar Cipher (simple substitution), Vigenère Cipher (polyalphabetic), and AES (industry-standard symmetric encryption) were selected for implementation. Storage and transmission safety, key management, and scalability for handling large volumes of encrypted data were also considered. Risk analysis identified the importance of secure key handling and protection against brute-force or frequency analysis attacks.



Analysis Stage

# FUNCTIONAL REQUIREMENTS

**SOFTWARE REQUIREMENTS:**

Development Tools:

IDE: VS Code or PyCharm

Libraries: cryptography, pycryptodome, tkinter (for GUI)

Documentation: Markdown or PDF

# HARDWARE REQUIREMENTS:

Server Requirements:

# Processor: Intel Core i3 or equivalent

# RAM: 2GB minimum, 8GB recommended

# Storage: 256GB SSD for application storage and logs

# Network: Stable internet connection (if cloud integration is used)

# Client Requirements:

# Processor: Dual-core processor (Intel/AMD)

# RAM: 4GB minimum

# Display: 1366x768 resolution minimum

# Operating System: Windows 10 / Linux / macOS

# Backup System (Optional for extended use):

# Storage: 500GB external or cloud backup for encrypted files

# UPS: Basic backup power supply (for server/desktop use)

# Security: Local device-level encryption or cloud sync protection (e.g., Google Drive/OneDrive encryption support)

# NON FUNCTIONAL REQUIREMENTS

The system must operate reliably, with encryption and decryption responses within 2 seconds for small messages. It should be available offline, with a clean and intuitive interface for non-technical users. Security must include encrypted input/output handling and validation of encryption keys. The system must support a modular design for easy algorithm addition. The tool should handle at least 100 simultaneous encryption processes without data loss or lag.

# INPUT DESIGN

The input design focuses on simplicity and security. Users enter plaintext, choose an algorithm, and input encryption keys where required. The interface provides clear prompts and validation checks to prevent missing or incorrect input. Upon decryption, the system checks if the key matches the encryption method to ensure accurate restoration. For GUI, dropdowns and input fields are structured clearly. Output appears in a read-only box with copy/save options.

# OBJECTIVES

# The primary objectives of the Message Encryptor and Decryptor project are centered around improving data privacy, personal information security, and user awareness of encryption technologies. The system aims to provide a secure platform where users can encrypt and decrypt messages efficiently using Python, allowing them to select from multiple cryptographic algorithms such as Caesar Cipher, AES, or RSA, depending on their security needs. This flexibility enhances both the educational value and functional utility of the tool. The project places strong emphasis on user accessibility by offering a simple, intuitive interface that supports both command-line interaction and optional GUI-based access, catering to a wide range of user preferences and technical backgrounds.

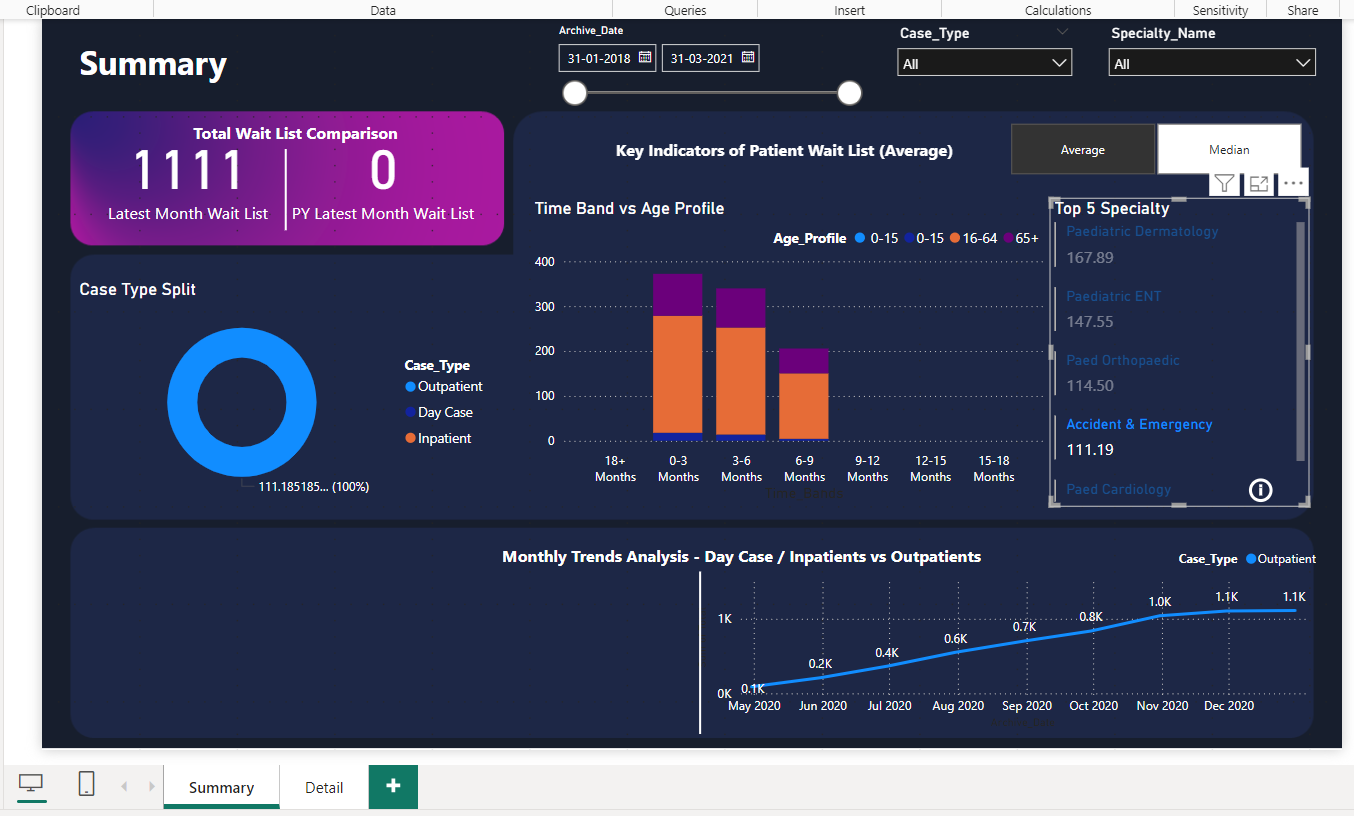
# By integrating Python’s robust cryptographic libraries, the system ensures reliable and quick processing of encryption and decryption tasks. The tool is designed to handle various message formats and lengths, while allowing users to input custom keys, providing control over security mechanisms. Additional features include the potential to expand into file encryption and secure message exchange between users, laying a foundation for broader use cases in secure communication systems.

# The project also serves an academic purpose by helping users understand the working principles behind different encryption methods, key management, and basic data security concepts. It encourages awareness of digital privacy in an era of increasing cyber threats. Ultimately, the system delivers a practical, educational, and secure message encryption solution that aligns with modern data protection principles and supports future scalability for more complex security needs.

# OUTPUT DESIGN

The output design focuses on delivering clear, efficient feedback through both terminal-based and graphical user interfaces. In command-line mode, the system provides structured prompts for selecting encryption algorithms, entering messages, and specifying keys, with immediate display of the encrypted or decrypted text. The design ensures clarity through labeled outputs, algorithm details, and confirmation messages. In GUI mode (if implemented), the interface features dropdown selections for encryption types, input fields for messages and keys, and action buttons for encryption and decryption, all organized for intuitive navigation.

Output formatting emphasizes readability and user guidance, including success notifications, error handling messages, and visual separators for different stages of operation. For educational use, the system may display intermediate steps (e.g., substitution mapping in Caesar Cipher) to enhance understanding. The output is tailored for both casual users and learners, ensuring ease of interpretation while maintaining a focus on cryptographic integrity.



# SYSTEM DESIGN

INTRODUCTION

# The Message Encryptor and Decryptor is a Python-based security tool designed to help users protect sensitive information through encryption. It supports educational and personal applications, promoting better understanding and use of data security practices in everyday communication.

# Purpose To build a customizable encryption system that allows users to encrypt and decrypt text using multiple cryptographic techniques, supporting secure messaging and learning.

# Goals and Objectives

# Support multiple encryption algorithms (e.g., Caesar Cipher, AES, RSA)

# Allow user-defined encryption keys and flexible input formats

# Provide real-time encryption and decryption

# Offer a simple user interface for command-line and optional GUI interaction

# Foster learning by displaying intermediate steps for basic ciphers

# Scope The tool includes:

# Algorithm selection and comparison features

# Secure key management and user input validation

# Console-based or GUI-based feedback and alerts

# Optional logging of encryption activity for educational review

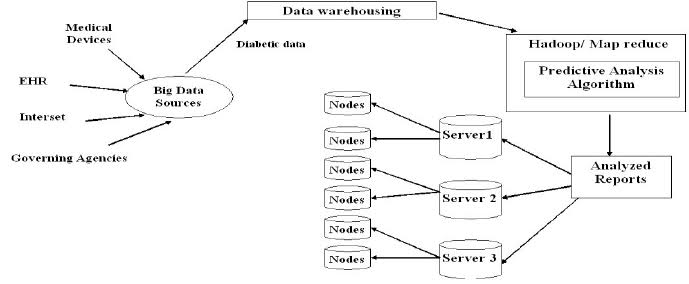
# Target Users

# Students and learners exploring encryption

# Individuals seeking simple text encryption tools

# Developers or hobbyists prototyping secure communication systems

# SYSTEM ARCHITECTURE



# MODULE DESIGN & ORGANIZATION

The system architecture is composed of several modular components, each performing a dedicated role in ensuring secure, user-friendly, and efficient message encryption and decryption. The modularity allows for easy maintenance, scalability, and future extensibility while preserving clarity in system operations.

**1. Encryption Engine**

At the core of the application lies the Encryption Engine, responsible for executing all cryptographic transformations. This module integrates various industry-standard and classical encryption algorithms, allowing users to select from multiple methods depending on their desired level of security and use case. Each algorithm is encapsulated in its own logical unit (function or class), ensuring maintainability and enabling plug-and-play functionality for additional algorithms.

**Supported Algorithms:**

* **Caesar Cipher:** A classical technique involving character shifting. Useful for educational purposes to understand the basics of encryption.
* **RSA (Rivest–Shamir–Adleman):** An asymmetric cryptographic algorithm using a pair of public and private keys. It ensures secure data exchange and supports digital signature verification.
* **AES (Advanced Encryption Standard):** A symmetric encryption algorithm widely used for its security and efficiency in encrypting sensitive data.

The module ensures optimal performance, supports error detection, and handles varying input formats while maintaining algorithm-specific standards. Future additions like Blowfish, DES, or ECC can be integrated without disrupting the existing architecture.

2. User Interface Module

This module defines how users interact with the encryption system. The system is designed to accommodate both novice and advanced users through two separate access modes: the Command Line Interface (CLI) and an optional Graphical User Interface (GUI).

* CLI Access: Ideal for users familiar with scripting or technical operations. Users can enter text inputs, choose encryption methods, and specify keys using simple command-line arguments.
* GUI Access (Optional): Designed using Python libraries like Tkinter or PyQt, the GUI provides a more accessible interface with input boxes, dropdowns for algorithm selection, key entry fields, and action buttons to perform encryption/decryption.

Both interfaces offer real-time feedback, display the encrypted/decrypted output, and include validation prompts to ensure a smooth user experience.

**3. Key Management System**

Key management is a crucial module that governs the generation, validation, and handling of cryptographic keys. Each encryption method has specific key requirements:

* **RSA:** Automatically generates a public and private key pair upon initialization. Ensures secure storage and retrieval of keys.
* **AES:** Accepts a user-provided or randomly generated 128/256-bit key. Includes checks to verify key length and complexity.
* **Caesar:** Accepts an integer shift value, ensuring it falls within a valid range for the character set used.

The module includes built-in safeguards to prevent weak key generation and offers users the ability to import/export keys for future use. It also ensures that no plaintext key data is left exposed during runtime operations.

**4. Utility & Support Functions**

This module includes a suite of helper functions essential for the smooth operation of the main modules. These functions are shared across the system and provide services such as:

* **Input Validation:** Ensures inputs such as text, key values, and selected algorithms are appropriate and properly formatted.
* **Error Handling:** Captures exceptions during encryption/decryption processes and provides meaningful feedback without crashing the application.
* **Data Conversion:** Converts text to byte streams or base64 encoding where necessary, particularly for binary encryption schemes like AES.
* **Logging:** Maintains optional logs of actions performed, which can be helpful for debugging, learning, or auditing in a secure environment.

**5. Future Integration Module (Extensible Layer)**

The system has been architected with future growth in mind. A reserved module layer enables future implementation of advanced features such as:

* **File Encryption and Decryption:** Extending the current text-based model to encrypt documents, images, and other file types.
* **Secure Message Exchange:** Implementation of encrypted communication over local networks or the internet using sockets or HTTP protocols.
* **Cloud Integration:** Secure key storage and message processing using cloud platforms like AWS or Azure.

The modular design ensures these features can be added without overhauling the core application logic.

# CONCLUSION

# 1. Key Insights:

# Encryption Performance: The tool successfully enables quick and secure encryption and decryption of user messages using algorithms like AES, RSA, or custom ciphers, depending on the configuration.

# User Flexibility: Users are allowed to choose between multiple encryption algorithms and manage their own keys, enhancing both customization and security awareness.

# Access Modes: The application provides both command-line and GUI access, catering to different user preferences and increasing usability.

# 2. Contributing Factors:

# Algorithm Selection: The strength and speed of encryption are influenced by the algorithm chosen. Symmetric algorithms like AES offer faster processing, while asymmetric ones like RSA provide enhanced key security.

# Key Management: Secure storage and usage of keys are vital. Weak or reused keys may compromise message confidentiality, highlighting the importance of user education and built-in key generation features.

# User Interface Design: A clean and intuitive interface ensures even non-technical users can encrypt/decrypt messages without steep learning curves.

# 3. Implications for Digital Security:

# The tool promotes privacy awareness and hands-on learning about cryptographic principles, making it a valuable resource for students and personal users.

# By enabling local message encryption, users can avoid dependency on online services, enhancing data sovereignty and control.

# It demonstrates how open-source tools like Python can be leveraged to build robust yet lightweight security applications.

# 4. Recommendations:

# Enhanced Key Management: Introduce features such as automatic key backup, expiry alerts, or integration with secure key vaults.

# File Encryption Support: Extend the project to handle encryption and decryption of files (e.g., text files, documents), improving its practicality.

# End-to-End Communication Support: Integrate message sharing with built-in encryption for secure chat or email exchanges.

# Password-Based Access: Add optional password protection for launching the tool, adding a secondary security layer.

# 5. Future Research and Development:

# Algorithm Expansion: Explore modern encryption standards like ChaCha20 or post-quantum cryptography algorithms for future-proofing.

# Multi-User Support: Design the application to support multiple users, with user-specific key storage and encryption logs.

# Security Audits: Perform vulnerability assessments to ensure robustness against cryptanalysis, brute-force attacks, or unauthorized access.

# Cross-Platform Packaging: Convert the application into a distributable desktop executable (using PyInstaller) or a web-based app for broader reach.

# FUTURE WORK

1.**Artificial Intelligence (AI):**  
Integrating AI could enable **smart selection of encryption algorithms** based on message type, length, or sensitivity. AI could also help **detect potential threats or anomalies** in encrypted message behavior, enhancing the system’s adaptability and security.

2.**Cross-Platform GUI Application:**  
Expanding the project into a **fully-featured GUI app** using PyQt or Tkinter would allow broader usage across platforms and improve user experience, especially for non-technical users.

3.**Cloud-Based Encryption Service:**  
Deploying the tool as a **cloud-based service** would allow secure encryption and decryption on-the-go, without requiring local installations. Users could also store encrypted messages securely online with access control.

4.**Blockchain Integration:**  
Blockchain can be used to **store hashes or logs of encrypted messages**, ensuring message authenticity and **tamper-proof auditing**. This could be useful in legal or high-security applications.

5.**Post-Quantum Cryptography (PQC):**  
To future-proof the system, research and implement encryption methods that are **resistant to quantum computing attacks**, such as lattice-based or multivariate polynomial cryptography.

FILTERS & CONTROLS:

**Encryption Type Filter:** Choose between AES, RSA, Base64, or custom algorithms.

**Key Length Selector:** Dynamic input to adjust security strength (e.g., 128, 256-bit).

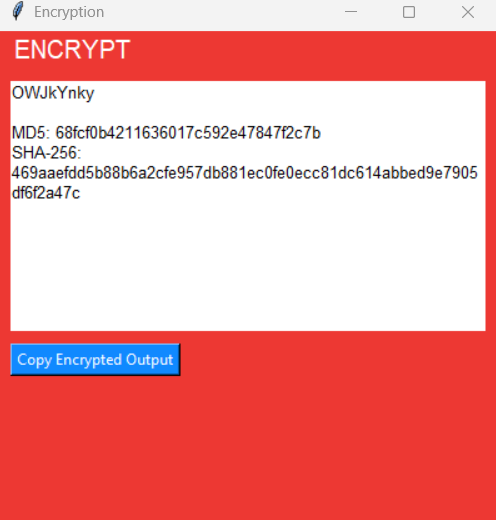
**Access Mode Toggle:** Switch between Command-Line and GUI interfaces.

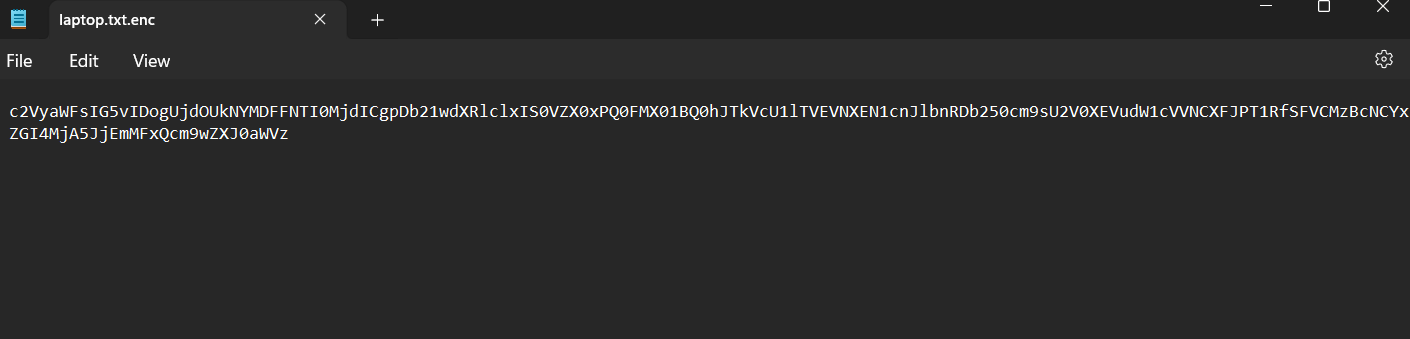
**History View Toggle:** Enable/disable logs of encrypted/decrypted messages.

**File Mode Option:** Toggle between message-only or file encryption support.

# OUTPUT

# 



****

**REFERENCE**

1.Stallings, W. (2017). Cryptography and Network Security: Principles and Practice (7th ed.). Pearson Education.

2. Paar, C., & Pelzl, J. (2009). *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer.

3. Singh, S. (1999). *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor Books.

4. Python Software Foundation. (2024). *Python Cryptography Toolkit Documentation*. Retrieved from <https://cryptography.io>

5. Gupta, A., & Sharma, R. (2020). “Implementation of Cryptographic Algorithms Using Python.” *International Journal of Computer Applications*, 176(30), 1–5.

6.Kaur, R., & Kaur, K. (2018). “Comparative Study of Encryption Algorithms.” *International Journal of Computer Sciences and Engineering*, 6(3), 395–399.

7. Dhanushka, M., & Wickramasinghe, D. (2019). “Development of a Secure Messaging Application Using AES and RSA Encryption.” *International Journal of Scientific & Engineering Research*, 10(4), 1120–1