**A PROJECT REPORT**

**on**

**“CIPHERLOCK - YOUR ROBUST ENCRYPTION AND DECRYPTION TOOL”**

**Submitted to**

**KIIT Deemed to be University**

**In Partial Fulfillment of the Requirement for the Award of**

**BACHELOR’S DEGREE IN**

**COMPUTER SCIENCE AND SYSTEMS ENGINEERING**

**BY**

**CHETAN HARSHIT SINGH**

**SUDIPA MANDAL**

**AYUSHI SUBHADARSHINI**

**RITWIKA DE**

2028017

2028036

2028053

2028174

**UNDER THE GUIDANCE OF**

**DR. JAYANTA MONDAL**

****

**SCHOOL OF COMPUTER ENGINEERING**

**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

**BHUBANESWAR, ODISHA - 751024**

**May 2023**

KIIT Deemed to be University

School of Computer Engineering

Bhubaneswar, ODISHA 751024



CERTIFICATE

This is certify that the project entitled

“CIPHER LOCK”

submitted by

CHETAN HARSHIT SINGH

SUDIPA MANDAL

AYUSHI SUBHADARSHINI

RITWIKA DE

2028017

2028036

2028053

2028174

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Sci-ence & Systems Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2022-2023, under our guidance.

Date: / /

Dr. Jayanta Mondal

Project Guide

**Acknowledgements**

We are profoundly grateful to **Dr.Jayanta Mondal** of **School of Computer Engineering** for her expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. .....................

CHETAN HARSHIT SINGH

SUDIPA MANDAL

AYUSHI SUBHADARSHINI

RITWIKA DE

**ABSTRACT**

**Keywords:**

**Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Introduction | | | 1 |
| 2 | Basic Concepts/ Literature Review | | | 2 |
|  | 2.1 | Cloud Computing | | 2 |
|  | 2.2 | Blockchain | | 3 |
|  | 2.3 | Distributed File System | | 4 |
|  | 2.4 | Interplanetary File System | | 5-7 |
| 3 | Problem Statement / Requirement Specifications | | | 8 |
|  | 3.1 | Project Planning | | 8 |
|  | 3.2 | Project Analysis (SRS)................. | | 9 |
|  | 3.3 | System Design ………………….. | | 10 |
|  |  | 3.3.1 | Design Constraints …… | 10 |
|  |  | 3.3.2 | System Architecture (UML) / Block Diagram … | 11-12 |
| 4 | Implementation | | | 13 |
|  | 4.1 | Methodology / Proposal ........................... | | 13 |
|  | 4.2 | Testing / Verification Plan ……………. | | 13-15 |
|  | 4.3 | Result Analysis / Screenshots …………. | | 15-17 |
|  | 4.4 | Quality Assurance …………………….. | | 15-16 |
| 5 | Standard Adopted | | | 18 |
|  | 5.1 | Design Standards . . . . . . . . . . . . . . . | | 18 |
|  | 5.2 | Coding Standards . . . . . . . . . . . . . . | | 19 |
|  | 5.3 | Testing Standards . . . . . . . . . . . . . . . | | 20 |
| 6 | Conclusion and Future Scope | | | 21 |
|  | 6.1 | Conclusion ……………………….. | | 21 |
|  | 6.2 | Future Scope ………………………. | | 21-22 |
| References | | | | 7 |
| Individual Contribution | | | | 24-27 |
| Plagiarism Report | | | | 28 |

List of Figures

|  |  |  |
| --- | --- | --- |
| 1. | A blockchain structure | 3 |
| 2.  3.  4.  5.  6.  7.  8. | Unique Node ID in the form of a hash  Blockchain-based data storage  Working of the distributed hash table  General workflow of the program  Class Diagram for Decentralized Cloud Storage Using Blockchain  Working of Vault 3.0  Snapshot of the Solidity Program | 4  7  11  12  12  17  20 |

**Chapter 1**

**Introduction**

**Chapter 2**

**Basic Concepts**

**Chapter 3**

**Problem Statement**

Cipherlock Tool aims to address the growing need for a secure and user-friendly encryption and decryption application. In an era of increasing digital threats and data privacy concerns, individuals and businesses require a reliable solution to safeguard sensitive information. Existing tools often lack versatility or user-friendliness. Cipherlock Tool seeks to bridge this gap by offering a Tkinter-based interface with five powerful algorithms, including Base64, Caesar Cipher, XOR Cipher, AES, and DES. This project responds to the demand for an intuitive, multi-algorithm encryption tool, empowering users to protect their data effortlessly across various security levels.

**3.1 Project Planning**

To execute the development of the CipherLock tool project, the following steps should be followed:

1. Define Project Scope:

Clearly outline the objectives and deliverables of the CipherLock tool, identifying the necessary resources for development, and ensuring a well-defined project scope aligned with its goals.

2. Gather Requirements:

Document user and stakeholder requirements in a specification document, prioritize and analyze them to ensure alignment with project goals, providing a foundation for the development process.

3. Develop Project Plan:

Create a detailed project plan outlining tasks, timelines, and resource requirements, incorporating milestones, dependencies, and a risk management plan to guide the CipherLock tool's development.

4. Design System Architecture:

Architect the technical details of the CipherLock tool, defining the software, hardware, and network infrastructure to establish a solid foundation for development.

5. Develop System Components:

Implement the various components of the CipherLock tool, including encryption algorithms and user interface, ensuring they align with the specified requirements.

6. Test the System:

   Conduct rigorous testing, including unit, integration, and system testing, to verify that the CipherLock tool meets the specified requirements and functions seamlessly.

7. Deploy the System:

Deploy the CipherLock app on a suitable platform, such as mobile app stores or desktop environments, ensuring proper configuration for optimal performance.

8. Provide User Training and Support:

Offer comprehensive user training and support to ensure users can effectively navigate and utilize the CipherLock tool, enhancing user experience.

9. Maintain and Update the System:

Implement regular maintenance and updates to keep the CipherLock tool secure, efficient, and aligned with the latest technological advancements, ensuring ongoing reliability and relevance.

**3.2 Project Analysis**

**3.2.1 Project Perspective:**

* The purpose of this project is to provide a robust solution to customers to use strong encryption and decryption methods to protect their data.
* The CipherLock tool is designed to provide a user-friendly interface for customers to easily encrypt/decrypt their data using their preferred algorithm.

**3.2.2 Project Functions:**

* Encryption Functions:
* base64\_encrypt(data: str) -> str: Base64 encryption for the input string.
* caesar\_encrypt(data: str, shift: int) -> str: Caesar Cipher encryption with a specified shift value.
* xor\_encrypt(data: str, key: str) -> str: XOR Cipher encryption using a provided key.
* aes\_encrypt(data: str, key: bytes) -> str: AES encryption with a specified key.
* des\_encrypt(data: str, key: bytes) -> str: DES encryption with a specified key.
* Decryption Functions:

base64\_decrypt(data: str) -> str : Decodes a Base64-encoded string.

* caesar\_decrypt(data: str, shift: int) -> str : Decrypts a Caesar Cipher encrypted string with a specified shift value.
* xor\_decrypt(data: str, key: str) -> str : Decrypts an XOR Cipher encrypted string using a provided key.
* aes\_decrypt(data: str, key: bytes) -> str : Decrypts an AES-encrypted string using a specified key.
* des\_decrypt(data: str, key: bytes) -> str : Decrypts a DES-encrypted string using a specified key.
* User Interface Functions:
* choose\_algorithm(algorithm: str): Updates the selected algorithm based on user choice.
* encrypt\_button\_action() : Initiates the encryption process based on the selected algorithm and user input.
* decrypt\_button\_action() : Initiates the decryption process based on the selected algorithm and user input.
* reset\_button\_action() : Clears all input in the text box.
* Integration Functions:
* Functions that integrate selected algorithms with user input for encryption and decryption processes.
* Orchestrates the flow between the GUI and encryption/decryption functions.

**3.2.3 Functional Requirements:**

* User Interface

R1: Initialize Tkinter GUI

Input:Application startup

Output: Tkinter GUI with input text boxes, algorithm selection dropdown, and buttons for choosing algorithm, encryption, decryption, and reset.

R2: Algorithm Selection

Input:User interacts with the application

Output: Dropdown menu displaying encryption algorithms; application updates the algorithm variable upon selection.

R3: Input and Output

Input: User inputs text or loads text from a file

Output: Result of encryption or decryption displayed in the output text box.

R4: Buttons

Input: User clicks buttons

Output:

- "Choose Algorithm" button triggers the algorithm selection dropdown.

- "Encrypt" button initiates encryption based on the selected algorithm and user input.

- "Decrypt" button initiates decryption based on the selected algorithm and user input.

- "Reset" button clears all input in the text box.

* Encryption and Decryption Functions

R5: Base64 implementation

Input: String data for encryption or Base64-encoded data for decryption

Output:

- For encryption: Base64-encoded string

- For decryption: Decoded string

R6: Caesar Cipher implementation

Input: String data and shift value for encryption or Caesar Cipher encrypted data and shift value for decryption

Output:

- For encryption: Caesar Cipher encrypted string

- For decryption: Decrypted string

R7: XOR Cipher implementation

Input:String data and key for encryption or XOR Cipher encrypted data and key for decryption

Output:

- For encryption: XOR Cipher encrypted string

- For decryption: Decrypted string

R8: AES implementation

Input:String data and key for encryption or AES encrypted data and key for decryption

Output:

- For encryption: AES encrypted string

- For decryption: Decrypted string

R9: DES implementation

Input:String data and key for encryption or DES encrypted data and key for decryption

Output:

- For encryption: DES encrypted string

- For decryption: Decrypted string

**3.2.4 Non-functional Requirements:**

N1: Performance Requirements

* The encryption and decryption processes shall be executed efficiently, ensuring timely results for typical user inputs.

N2: Externel Interface Requirements

* The user interface of the bot should be eye-catching.
* The user interface should be user friendly

N3: Error Handling

* The application shall employ robust error-handling mechanisms, providing informative messages to users in case of unexpected issues.

**3.3 System Design**

**3.3.1 Design Constraints**

**3.3.1.1 Software Constraints**

* The project relies on external libraries for the implementation of encryption/decryption algorithms (e.g., cryptography library for AES and DES).
* The availability and compatibility of external libraries may pose constraints. Compatibility issues with specific versions of these libraries could affect the application's functionality.

**3.3.1.2 Hardware Constraints**

* The performance of encryption algorithms, especially complex ones like AES and DES, may be constrained by the processing power of the hardware. Low-powered devices may experience slower encryption and decryption speeds.

**3.3.1.3 Environmental Constraints**

* Operating System Compatibility: The application's performance may vary across different operating systems due to dependencies on Tkinter. OS-specific behaviors can impact the user experience consistency.
* Dependency Management: Compatibility issues with external libraries, crucial for encryption algorithms, may affect the application. Ensuring consistent library versions and compatibility is essential for proper functionality.

**3.3.2 System Architecture**

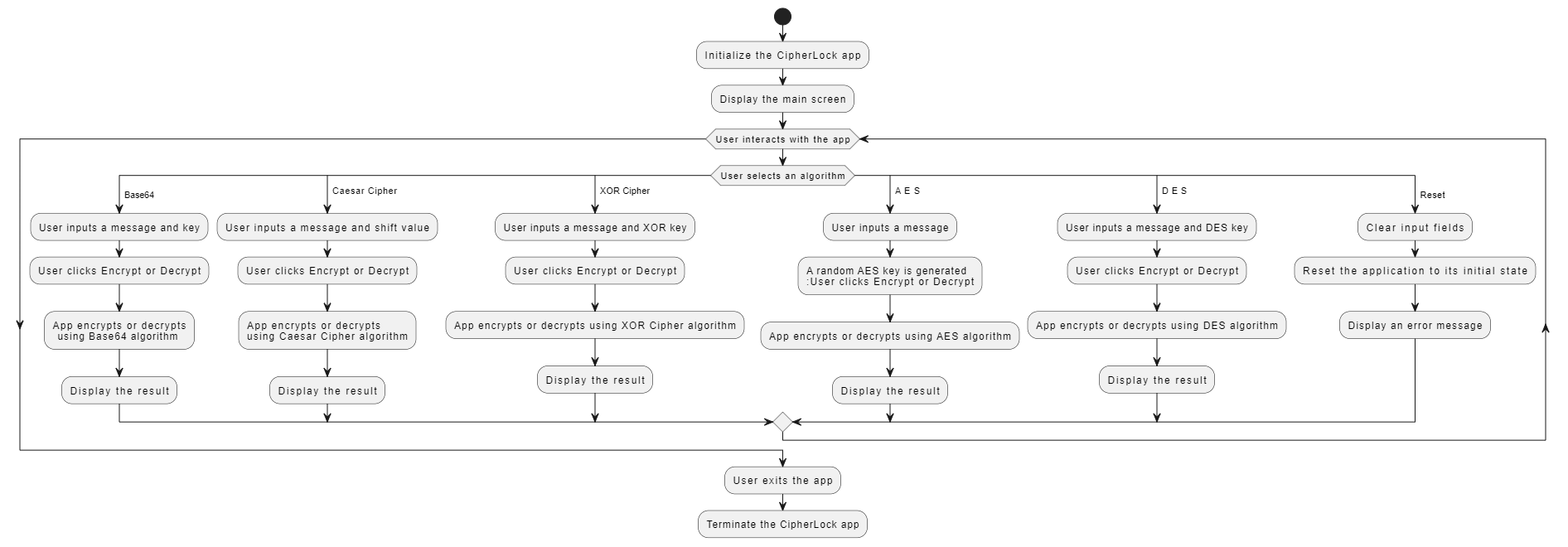


Fig 3 : General work flow of the program

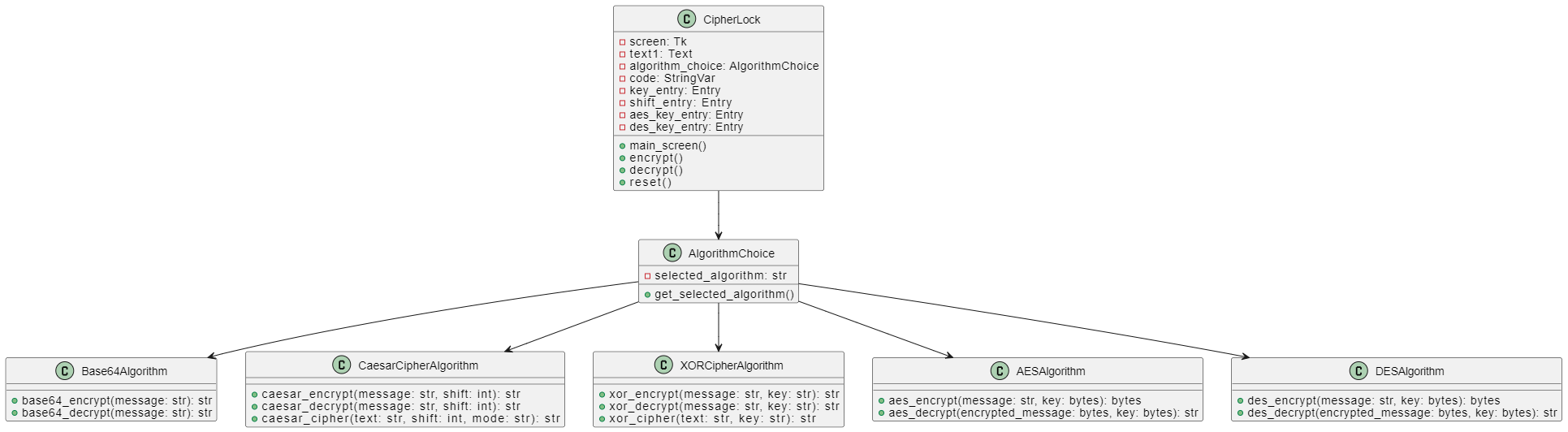


Fig 4 : Class diagram of the CipherLock tool

Flow diagrams of each algorithm:

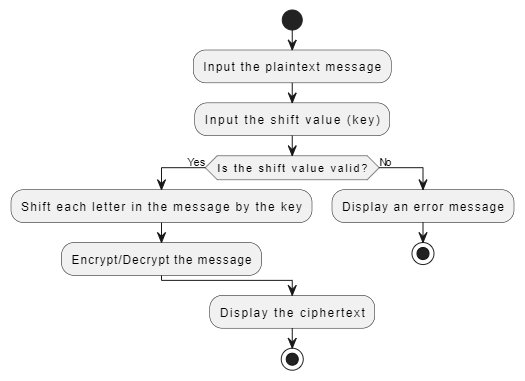


Fig 5 : Flow diagram of Caesar cipher algorithm

**Chapter 4**

**Implementation**

**4.1 Methodology**

**4.2 Testing**

**4.3 Result Analysis**

**4.4 Quality Assurance**

**Chapter 5**

**Standards Adopted**

**5.1 Design Standards**

**5.2 Coding Standards**

While creating the project, we adhered to the following coding standards to guarantee consistency and readability in the code:

* Naming Conventions:

Use descriptive and meaningful names for variables, functions, and classes.

Use camelCase for variables and functions, and PascalCase for class names.

Avoid using single-letter variable names except for simple counters or iterators.

* Comments and Documentation:

Provide clear and concise comments to explain complex logic, algorithms, and non-trivial solutions.

Use inline comments sparingly, focusing on explaining complex or non-obvious code segments.

Document the purpose, input/output, and usage of important functions and classes.

* Formatting and Indentation:

Maintain consistent and clear code formatting throughout the project.

Use consistent indentation (e.g., tabs or spaces) for improved readability.

Limit line lengths to improve code clarity, typically 80-120 characters per line.

* Error Handling and Logging:

Implement comprehensive error handling to manage exceptions and errors gracefully.

Use meaningful error messages and logs for effective troubleshooting and issue resolution.

Log important events and errors to facilitate debugging and maintenance.

* Modularity and Reusability:

Organize the code into modular components for improved maintainability and reusability.

Encapsulate reusable functionalities within separate modules or libraries to promote code reusability.

Minimize dependencies between modules to enhance code modularity.

* Testing and Quality Assurance:

Write unit tests for critical functions and modules to ensure their correctness and expected behavior.

Conduct regular code reviews to identify and address coding errors, inefficiencies, and inconsistencies.

By following these coding standards, we were able to ensure consistency in the code and make it more readable and maintainable.

**5.3 Testing Standards**

**Chapter 6**

**Conclusion and Future Scope**

**6.1 Conclusion**

In conclusion, CIPHERLOCK stands as a powerful and reliable encryption and decryption tool, fortified by the implementation of a comprehensive suite of algorithms. By understanding and incorporating Base-64, XOR Cipher, Caesar Cipher, AES, and DES, we have created a versatile solution that caters to diverse security needs.

CIPHERLOCK not only ensures the confidentiality of sensitive information but also adapts to the dynamic challenges of the digital landscape. With the seamless integration of both time-tested and state-of-the-art encryption methods, this tool is well-equipped to safeguard user data against a spectrum of potential threats.

In a world where data security is paramount, CIPHERLOCK emerges as a robust ally, offering users a straightforward yet potent means to protect their digital assets. Through the harmonious synergy of these algorithms, our project encapsulates the essence of a cutting-edge encryption tool—trustworthy, adaptive, and committed to ensuring the privacy of every user.

**6.2 Future Scope**

* Enhancement of the User Interface (UI):

Improve the interface's usability and intuitiveness.

During the encryption/decryption procedures, add progress indicators or status messages.

Provide users with tooltips or help sections to assist them in better comprehend each algorithm and its parameters.

* Additional Algorithms:

Consider including additional encryption techniques or modes of operation. For example, you might use RSA, Triple DES, Blowfish, and so forth.

Allow users to select alternative key sizes and modify algorithm settings.

* Key Personnel:

Install a secure key management system that allows users to produce, import, and export encryption keys.

Investigate key exchange techniques for securely sharing keys among users.

* Encryption and decryption of files:

Extend your project so that it can encrypt and decode full files rather than simply text.

Enable streaming encryption for big files.

* Password Security:

Allow users to establish passwords or passphrases for encryption, which adds a degree of protection.

To generate encryption keys from user passwords, employ key derivation functions.

* Digital Signatures and Hashing:

Integrate hashing methods to create text or file checksums or digital signatures.

Use digital signatures to validate the encrypted content's validity.

* Compression:

To conserve space, include the ability to compress text or files before encryption.

Choose an appropriate compression technique (for example, gzip) and include it in your project.

* Compatibility Across Platforms:

Create versions of your program for multiple operating systems (Windows, macOS, and Linux) to make it cross-platform.

Consider creating a web-based version that can be accessed from any device.

Consider developing a web-based version for accessibility from any device.

* Auditing and logging:

Enable logging to record encryption/decryption activities.

Install an auditing system to examine logs and track use.

* Handling Errors:

Improve error handling techniques such that useful messages are sent in the event of improper input or problems during the encryption/decryption process.

* Performance Enhancement:

Improve algorithm performance, especially for huge datasets.

Use parallel processing for encryption/decryption to take advantage of multi-core platforms.

* Localization:

Add multilingual support to your application to make it more accessible to a wider audience.

* Auditing for security:

Review and update your algorithms regularly to resolve any vulnerabilities or shortcomings.

To discover possible risks, conduct security audits or involve the community.

***References***

**PROJECT TITLE : CHATBOT USING PYTORCH**

Student Name - SUDIPA MANDAL

Student Roll number - 2028036

**Abstract:**

**Individual contribution and findings:**

Contributions

Findings

**Individual contribution to project report preparation:**

Full signature of the Supervisor: Full signature of student:

……………………………. …………………………….

**PROJECT TITLE : CHATBOT USING PYTORCH**

Student Name - RITWIKA DE

Student Roll number - 2028174

**Abstract:**

**Individual contribution and findings:**

Contributions

Findings

**Individual contribution to project report preparation:**

**Individual contribution for project presentation and demonstration:**

Full signature of the Supervisor: Full signature of student:

……………………………. …………………………….

TURNITIN PLAGIARISM REPORT

