**CIPHERVAULT (Using Fernet Symmetric Encryption – AES Algorithm)**

**Project report in partial fulfillment of the requirement for the award of the degree of**

**Bachelor of Technology**

**In**

**CST & CSIT**

**Submitted By**

Shivsundar Bera Enrollment No. 12021002023003

Soumyajit Patra Enrollment No. 12021002022004

Sayandeep Mondal Enrollment No. 12021002023071

Hriteesha Pramanik Enrollment No. 12021002022005

Arindam Roy Enrollment No. 12021002022116

Swarnadeep Roy Enrollment No. 120210020222061

**Under the guidance of**

Prof. Dr. Subhalaxmi Chakraborty

Department of Computer Science and Technology & Computer Science and Information Technology (CST & CSIT)



UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

University Area, Plot No. III – B/5, New Town, Action Area – III, Kolkata – 700160.

**CERTIFICATE**

This is to certify that the project titled **CipherVault** **(using Fernet Symmetric Encryption – AES Algorithm)** submitted by **Shivsundar Bera (University Roll No. 12021002023003), Soumyajit Patra (University Roll No. 12021002022004 )**, **Sayandeep Mondal (University Roll No. 12021002023071), Hriteesha Pramanik (University Roll No. 12021002022005)**, **Arindam Roy (University Roll No. 12021002022116) and Swarnadeep Roy (University Roll No. 120210020222061)** students of UNIVERSITY OF ENGINEERING and MANAGEMENT, KOLKATA, in partial fulfillment of requirement for the degree of Bachelor of Computer Science and Technology, is a bonafide work carried out by them under the supervision and guidance of Prof. Dr. Subhalaxmi Chakraborty during 6th Semester of academic session of 2021 – 2025. The content of this report has not been submitted to any other university or institute. I am glad to inform that the work is entirely original and its performance is found to be quite satisfactory.

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

Signature of Guide

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

Signature of Head of the Department

**ACKNOWLEDGEMENT**

We would like to take this opportunity to thank everyone whose cooperation and encouragement throughout the ongoing course of this project remains invaluable to us.

We are sincerely grateful to our guide Prof. Dr. Subhalaxmi Chakraborty of the Department of CST & CSIT, UEM, Kolkata, for his/her wisdom, guidance and inspiration that helped us to go through with this project and take it to where it stands now.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Shivsundar Bera

Soumyajit Patra

Sayandeep Mondal

Hriteesha Pramanik

Arindam Roy

Swarnadeep Roy

**TABLE OF CONTENTS**

**ABSTRACT ........................................................................................ 1**

**CHAPTER – 1: INTRODUCTION .................................................................. 2**

**CHAPTER – 2: LITERATURE SURVEY**

**2.1 Project Structure ................................................................................ 3**

**2.2 File Description ................................................................................... 4**

**2.3 Secret Key ............................................................................................ 4**

**CHAPTER – 3: PROBLEM STATEMENT ..................................................... 5**

**CHAPTER – 4: PROPOSED SOLUTION ....................................................... 6**

**CHAPTER – 5: EXPERIMENTAL SETUP AND RESULT ANALYSIS**

**5.1 Experimental Setup ............................................................................. 7**

**5.2 Result Analysis ..................................................................................... 7**

**CHAPTER – 6: CONCLUSION & FUTURE SCOPE**

**6.1 Conclusion ........................................................................................... 10**

**6.2 Future Scope ....................................................................................... 10**

**BIBLIOGRAPHY ............................................................................................... 11**

**Abstract:**

This project aims to develop a web application that allows users to securely encrypt and decrypt files using **symmetric encryption algorithms**. The application provides a user-friendly interface for uploading files, performing encryption or decryption operations, and ensuring the security of sensitive data. It leverages **Streamlit** for building the web interface and the cryptography library for implementing encryption and decryption functionalities. This project provides a more robust and customizable solution for encryption and decryption needs, especially in scenarios where data security and compliance are critical.

**1**

**Chapter 1: Introduction**

In an era where data security is paramount, the need for robust encryption solutions has never been more pronounced. The Secure File Encryption and Decryption Web Application represents a beacon of trust and resilience in the realm of digital security. At its core, this project endeavors to empower users with the ability to safeguard their sensitive data through the seamless application of symmetric encryption algorithms. The Secure File Encryption and Decryption Web Application is a testament to the fusion of cutting-edge technology and user-centric design. Its primary objective is to furnish users with a platform where they can encrypt and decrypt files with unparalleled ease and security. Leveraging the prowess of Streamlit for crafting intuitive web interfaces and the cryptography library for implementing battle-tested encryption methodologies, this application epitomizes the convergence of simplicity and sophistication.

**2**

**Chapter 2: Literature Survey**

**2.1 Project Structure**

Delving into the project's anatomy unveils a meticulously crafted framework designed to encapsulate functionality, elegance, and extensibility. From the foundational main.py script orchestrating the application's logic to the nuanced style/request.css file, every component serves a distinct purpose in harmonizing form and function. The secret.key file emerges as a sentinel of security, safeguarding the encryption key with unwavering vigilance.

**Files:**

1. **main.py**: The main Python script containing the code for the Streamlit web application.
2. **style/request.css:** A local CSS file for custom styling of the web interface.
3. **secret.key:** A file to store the symmetric encryption key generated by the application.
4. **Dependencies:** Python 3.x, Streamlit, Pillow (PIL), requests, cryptography.

**Dependencies:**

Drawing upon a constellation of dependencies, the project stands as a testament to the collaborative spirit of open-source software. Python 3.x lays the groundwork for computational prowess, while Streamlit provides the canvas upon which the application's user interface flourishes. Pillow (PIL), requests, and cryptography converge to bestow the project with image processing capabilities, HTTP functionality, and cryptographic prowess, respectively.

1. **Python 3.x**: Python is a high-level programming language known for its simplicity and readability. Version 3.x is the latest major release series of Python, offering various improvements and new features over previous versions.
2. **Streamlit**: Streamlit is an open-source Python library used for creating web applications with minimal effort. It allows developers to build interactive and data-driven applications using simple Python scripts.
3. **Pillow (PIL)**: Pillow, also known as the Python Imaging Library (PIL), is a library for opening, manipulating, and saving many different image file formats. It provides powerful image processing capabilities and is widely used in Python applications for tasks such as resizing, cropping, and enhancing images.
4. **Requests**: Requests is a Python library used for making HTTP requests. It simplifies the process of sending HTTP requests and handling responses, making it easier to interact with web services and APIs.
5. **Cryptography**: Cryptography is a Python library that provides cryptographic recipes and primitives. It offers various cryptographic algorithms and protocols for securing data, including symmetric encryption, asymmetric encryption, hashing, and digital signatures. It's widely used for implementing secure communication, data protection, and authentication mechanisms in Python applications.

**3**

**2.2 File Description**

1. **main.py:** The main Python script containing the code for the Streamlit web application.
2. **Imports:** Import necessary libraries including PIL, requests, Streamlit, and cryptography.
3. **Functions:**
4. **load\_lottieurl(url):** Function to load Lottie animation from a URL.
5. **local\_css(file\_name):** Function to apply custom CSS styling from a local file.
6. **generate\_key():** Function to generate a symmetric encryption key and save it to Secret Key.
7. **load\_key():** Function to load the encryption key from Secret.key file or generate a new one if not found.
8. **encrypt(filename, key):** Function to encrypt a file using the provided encryption key.
9. **decrypt(filename, key):** Function to decrypt a file using the provided encryption key.
10. **main():** Main function containing the logic for the Streamlit web application.
11. **Web Application:** Defines the layout and functionality of the web application using Streamlit. Includes sections for header, purpose, encryption/decryption options, projects, and contact information. Allows users to upload files for encryption or decryption, perform the respective operations, and display success/error messages.
12. **style/request.css:** Contains custom CSS styles for the Streamlit web application, enhancing the visual appearance and user experience.
13. **secret.key:** Stores the symmetric encryption key generated by the application. It is used for both encryption and decryption processes, ensuring data security.

**2.3 secret.key**

The Secret.key file serves as the cornerstone of the Secure File Encryption and Decryption Web Application, representing a pivotal element in the realm of data security. Its primary function is to securely house the symmetric encryption key generated by the application, which is indispensable for safeguarding the confidentiality and integrity of user files. In the landscape of encryption, the concept of symmetric cryptography revolves around the utilization of a single, shared key for both encrypting and decrypting data. This symmetric encryption key, akin to the heart of a digital fortress, is the linchpin of security within the application's cryptographic framework. The Secret.key file embodies the essence of data protection, acting as a fortified repository where the encryption key is diligently stored. This key is a testament to trust, as it forms the bedrock upon which the application's encryption and decryption processes rest. Its sanctity must be upheld at all costs, as any breach or compromise could potentially lead to the exposure of sensitive information. Security considerations surrounding the Secret.key file extend beyond mere storage; they encompass a multifaceted approach to key management. Robust access controls, stringent permissions, and encryption-at-rest protocols are but a few measures employed to fortify its defenses against malevolent adversaries. Moreover, the Secret.key file epitomizes resilience through redundancy. Regular backups ensure that the encryption key remains impervious to the vagaries of data loss, guaranteeing the continuity of encryption and decryption operations. The Secret.key file is not merely a static artifact; it is a testament to the application's commitment to data security and privacy. Its custodianship underscores the application's dedication to preserving the sanctity of user data, ensuring that sensitive information remains shielded from prying eyes. In essence, the Secret.key file embodies the ethos of trust, resilience, and security within the Secure File Encryption and Decryption Web Application. As the custodian of the encryption key, it stands as a sentinel against the forces of digital malfeasance, safeguarding the sanctity of user data with unwavering resolve.

**4**

**Chapter 3: Problem Statement**

This way offers a lot of protection to the files.

1. **Security Level:** This program uses symmetric encryption algorithms like Fernet, providing strong encryption to safeguard your files. .rar, on the other hand, primarily offers file compression with optional password protection. While password protection in .rar can offer some level of security, it might not be as robust as encryption algorithms used in dedicated encryption tools.
2. **Flexibility:** This program allows you to choose the encryption algorithm and customize encryption settings according to your security requirements. Additionally, it offers functionalities like encryption key management and secure file transmission, which might not be available or as customizable in .rar.
3. **Use Cases:** This program is specifically designed for encryption and decryption purposes, making it suitable for scenarios where data security is paramount, such as sensitive document storage, secure file sharing, or compliance with data protection regulations. .rar, on the other hand, is primarily used for file compression and archiving, with password protection as an additional feature.
4. **Openness and Transparency:** This program is built on open-source libraries and provides transparency regarding the encryption algorithms and methods used, allowing users to verify the security of their files. While .rar is a widely used file format, the specific encryption methods and their security might not be as transparent or customizable.
5. **Secure communication:** Enable secure transmission of sensitive information over untrusted networks.
6. **Data protection:** Safeguard sensitive data stored on devices or transmitted between systems.
7. **Compliance requirements:** Help organizations comply with industry regulations for data protection and encryption.
8. **Educational tool:** Teach users about encryption algorithms and best practices.
9. **Demonstration and Testing:** Test and demonstrate encryption algorithms or implementations for research or training purposes.
10. **Personal privacy:** Protects personal information and maintain privacy online.
11. **Collaborative work:** Facilitates secure collaboration and file sharing among individuals or teams.

**5**

**Chapter 4: Proposed Solution**

The Secure File Encryption and Decryption Web Application provides a robust solution for securely encrypting and decrypting files, catering to various use cases such as data protection, secure communication, and compliance requirements. By leveraging Streamlit and cryptography libraries, the application offers a simple yet powerful interface for users to safeguard their sensitive data effectively. This comprehensive project file format provides detailed insights into the project structure, functionalities, execution steps, additional notes, future enhancements, and conclusion, offering a holistic view of the Secure File Encryption and Decryption Web Application project.

Solution:

1. Develop a web application with an intuitive and user-friendly interface for seamless file encryption and decryption operations.
2. Implement symmetric encryption algorithms, such as Fernet encryption, to provide strong data security while maintaining efficiency and ease of use.
3. Securely manage encryption keys using the Secret.key file to ensure the confidentiality and integrity of encrypted data.
4. Utilize the Streamlit framework to build the web application, enabling rapid development and deployment of interactive data-driven applications.
5. Leverage the cryptography library in Python to implement encryption and decryption functionalities, ensuring compliance with industry-standard cryptographic protocols.
6. Provide visual feedback to users through animations, progress indicators, and status messages to enhance the user experience and guide users through the encryption/decryption process.
7. Offer customization options for users to tailor the encryption/decryption process to their specific requirements, including choosing encryption algorithms, key sizes, and encryption modes.
8. Design the application architecture for scalability and performance to accommodate varying workloads and user demands, ensuring responsive and reliable operation under different scenarios.
9. Provide comprehensive documentation and user support resources, including tutorials, FAQs, and community forums, to assist users in understanding and effectively using the application.
10. Commit to ongoing maintenance, updates, and enhancements to address emerging security threats, incorporate user feedback, and improve the overall functionality and usability of the application.
11. Empower users to protect their sensitive data from unauthorized access and data breaches through robust encryption mechanisms.
12. Streamline the process of managing encrypted files by providing an easy-to-use interface for encryption, decryption, and key management.
13. Enable users to take control of their data security and privacy by offering accessible tools for secure file encryption and decryption.
14. Assist organizations and individuals in meeting compliance requirements for data protection and encryption, ensuring adherence to industry regulations and standards.
15. Serve as an educational resource for users to learn about encryption algorithms, best practices, and security principles, fostering a culture of digital literacy and cybersecurity awareness.
16. Foster a community of users, developers, and security enthusiasts to share knowledge, collaborate on improvements, and contribute to the advancement of data security practices.

**6**

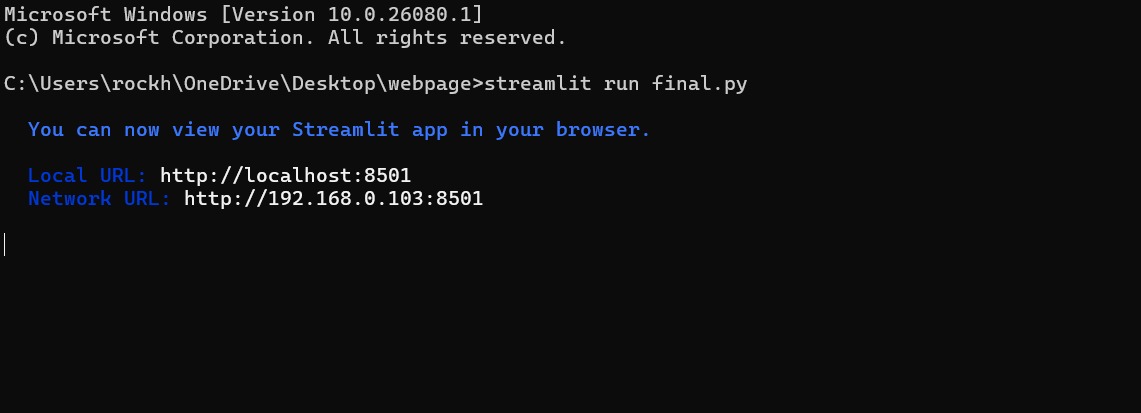
**Chapter 5: Experimental Setup & Result Analysis**

* 1. **Experimental Setup:**

1. **Setup:** Clone the repository or download the project files to your local machine.
2. **Installation:** Install the required dependencies (Python 3.x, Streamlit, Pillow, requests, cryptography) using pip or conda.
3. **Execution:** Run the main.py script using Python (python main.py).
4. **Access:** Access the web application through the provided local host address (usually http://localhost:8501).
5. **Usage:** Use the web interface to upload files, perform encryption or decryption operations, and view the results.
6. **Exploration:** Explore other sections of the application such as purpose, projects, and contact for additional information and functionalities.
7. **Additional Notes:** Ensure that the secret.key file is present in the same directory as the main.py script for proper functioning of encryption and decryption. Customize the application further by modifying the CSS styles, adding additional features, or integrating with external services as required. Consider deploying the application to a web hosting service for public access and usability.
   1. **Result Analysis:**

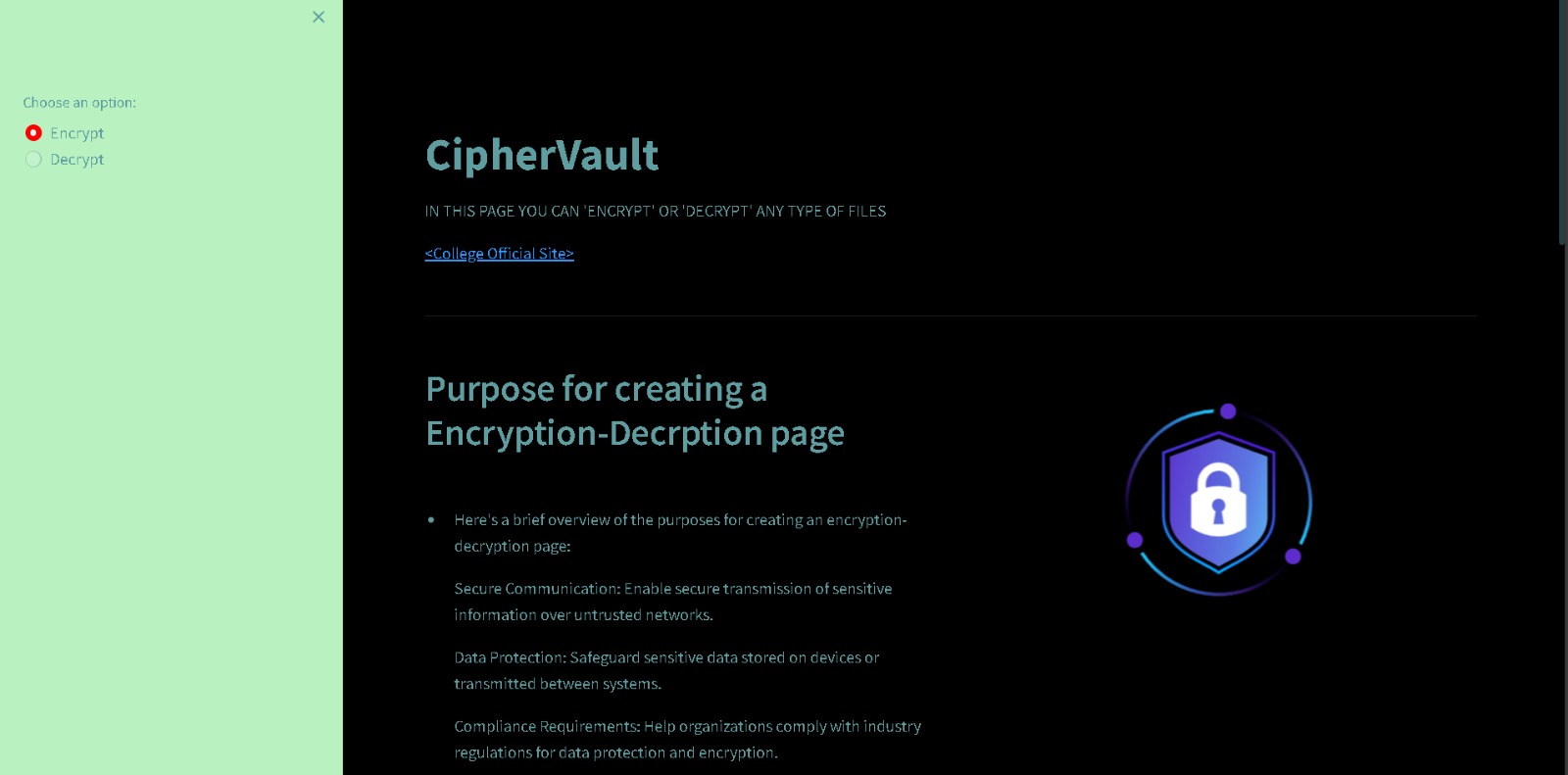
The program uses the Fernet symmetric encryption algorithm for encrypting and decrypting files. Fernet is a symmetric encryption algorithm based on the AES (Advanced Encryption Standard) algorithm in CBC (Cipher Block Chaining) mode with a SHA-256 HMAC (Hash-based Message Authentication Code) for authentication. It provides secure encryption and decryption of data using a shared secret key.

So, we run the program:

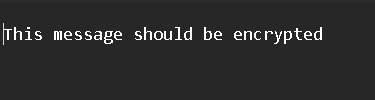


**7**

That opens up the browser and we get a page like this where we choose whether we want to encrypt a file or decrypt an encrypted file:



Here, we have to upload the file we want to encrypt. So, we can take maybe a text file whose contents are like this:



Now, we upload it to our website and click on “Encrypt”. Then, we can check the file in our local storage and it will look like this:



So, we can see that the file is encrypted.   
In the same way, we can again choose this encrypted file and upload it to our website and select the “Decrypt” option:

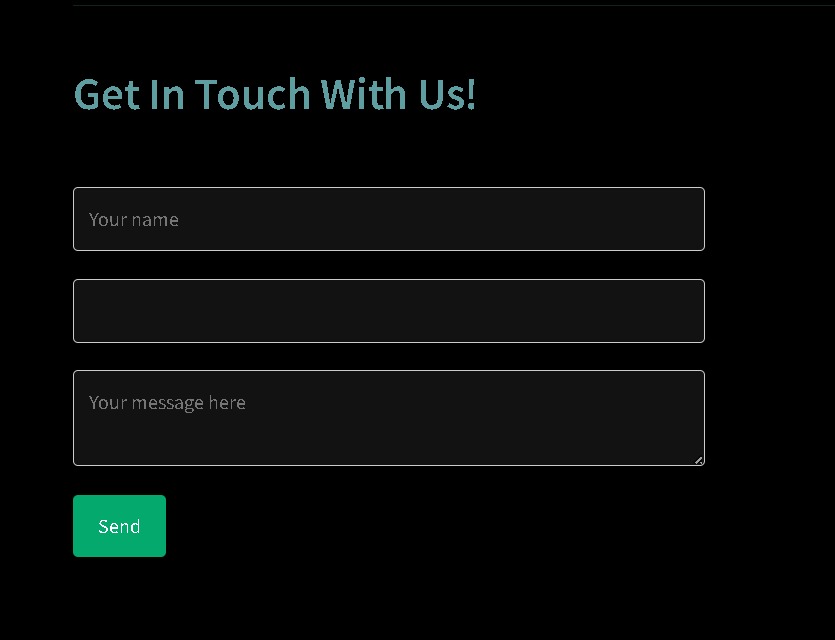


**8**

After decryption the file will again show its normal contents.

We can do this with any kind of file whether text, images or videos. For images and videos, the file format of the file would be such that it won’t be supported to show. But after decryption we can view their regular contents again.

Additionally, there are also means provided to contact the admin for any kind of queries regarding the service:



**9**

**Chapter 6: Conclusion & Future Scope**

* 1. **Conclusion:**

The Secure File Encryption and Decryption Web Application provides a robust solution for securely encrypting and decrypting files, catering to various use cases such as data protection, secure communication, and compliance requirements. By leveraging Streamlit and cryptography libraries, the application offers a simple yet powerful interface for users to safeguard their sensitive data effectively.

This comprehensive project file format provides detailed insights into the project structure, functionalities, execution steps, additional notes, future enhancements, and conclusion, offering a holistic view of the Secure File Encryption and Decryption Web Application project.

* 1. **Future Scope:**

Implement advanced encryption algorithms for stronger security. Add user authentication and authorization features to control access to encryption/decryption functionalities. Integrate with cloud storage services for seamless file management and encryption/decryption capabilities. Enhance the user interface with more interactive elements and visualizations. Provide options for batch processing and handling large files efficiently.

**10**

**Bibliography:**

Python cryptography library documentation: Cryptography - PyPI

Fernet encryption algorithm documentation: Fernet - cryptography documentation

Streamlit:

Streamlit official documentation: Streamlit Documentation

Streamlit GitHub repository for issues, discussions, and updates: Streamlit GitHub

Lottie Animations:

LottieFiles official website for exploring and downloading animations: LottieFiles

Streamlit Lottie extension documentation: Streamlit Lottie

**11**