File Encryption using Fernet Cipher

By Aastha Kumar (21BCE5067)

***Abstract*—Data security is a paramount concern in the digital age, especially with the proliferation of multimedia files such as images, videos, and audio recordings. In this project, we explore the implementation of the Fernet Cipher for encrypting various types of files to ensure their confidentiality and integrity. The Fernet Cipher, known for its simplicity and effectiveness, offers robust encryption capabilities, making it an ideal choice for securing sensitive data. This report presents the methodology, findings, and implications of our project, underscoring the importance of file encryption in safeguarding digital assets.**

Keywords — File Encryption , File Decryption, Fernet Cipher , AES-CBC, Secret Key

I. INTRODUCTION

With the exponential growth of digital data, the need for robust encryption techniques has become increasingly critical. File encryption plays a pivotal role in protecting sensitive information from unauthorized access and tampering. The Fernet Cipher, a symmetric encryption algorithm based on symmetric key cryptography, offers a straightforward yet powerful solution for encrypting files of various formats. Unlike asymmetric encryption methods, the Fernet Cipher employs a single secret key for both encryption and decryption, simplifying the encryption process while maintaining strong security guarantees.

In this project, we aim to explore the efficacy of the Fernet Cipher in encrypting multimedia files, including images, videos, and audio recordings. By leveraging the simplicity and efficiency of the Fernet Cipher, we seek to develop a practical and reliable solution for securing digital assets against unauthorized access and data breaches. Through a systematic approach, we evaluate the performance and security of our encryption scheme, highlighting its strengths and limitations in real-world scenarios.

II. PROBLEM STATEMENT

In today's digital landscape, the security of multimedia files, including images, videos, and audio recordings, is of utmost importance due to the proliferation of digital data and the ever-present threat of unauthorized access and data breaches. Traditional encryption methods may not be well-suited for securing multimedia files efficiently, leading to challenges in maintaining confidentiality and integrity while ensuring compatibility across different file formats and platforms.

The problem at hand is to develop a robust and practical solution for encrypting multimedia files effectively, addressing the specific requirements and challenges associated with their encryption and decryption. This solution should offer strong security guarantees while minimizing computational overhead and ensuring seamless integration with existing file management systems and workflows.

Furthermore, the solution should be accessible to users with varying levels of technical expertise, allowing them to encrypt and decrypt multimedia files with ease, without compromising security. By addressing these challenges, the project aims to contribute to the advancement of data security practices and enhance the protection of multimedia assets in digital environments.

III. OBJECTIVE

1. User-Friendly Encryption/Decryption:
- Develop a user-friendly interface for encrypting and decrypting messages.
- Allow users to easily input messages and obtain the encrypted or decrypted output.
2. Cross-Platform Compatibility:
- Create a Python-based tool that is compatible with various operating systems, ensuring accessibility for a wide range of users.
3. Secure Message Transmission:
- Facilitate secure transmission of encoded messages, reducing the risk of unauthorized access during communication.
4. Documentation and User Guide:
- Provide comprehensive documentation and a user guide to assist users in understanding the tool's functionalities and usage.

IV. LITERATURE SURVEY

[1] Performance of Fernet and Aes Algorithms with Existing Encryption Techniques
Authors: Pronika, Supriya P.Panda, S.S. Tyagi
Year: 2022
Abstract:
This paper compares the performance of the AES and Fernet algorithms with different sizes of data files (1KB, 10KB, 100KB, and 1MB). The time taken by the AES algorithm is almost half as compared to the Fernet algorithm when the file size is 1KB.

[2] Comparative Analysis of Cryptographic Algorithms in Securing Data
Authors: Taylor, Onate E., Emmah, Victor T.
Year: 2018
Abstract:
This paper presents a comparative analysis of modern cryptographic algorithms, including the Fernet cipher. The algorithms are compared based on computation time, memory utilization, and security level.

[3] Securing the Data in Cloud Using Fernet Technique
Authors: Chinni Prashanth, D Bala Sai Teja, V Lavanya
Year: 2022
Abstract:
This work discusses securing data in the cloud using the Fernet method. The file is uploaded and downloaded using a cryptographic process, which involves encryption and decryption. Data access is made available to privileged users.

[4] Cloud Storage Security using Firebase and Fernet Encryption
Authors: Dhruv Sharma, C. Fancy

Year: 2022

Abstract:

This paper discusses encryption using the AES algorithm and file encryption. Adding an extra layer of encryption, data can be secured. If, in some way or the other, the data fall into the wrong hands, even in that condition, the attacker will not be able to open the data as it has double protection and a doubly encrypted layer.

[5] Proceedings of the National Conference on Emerging Computer"

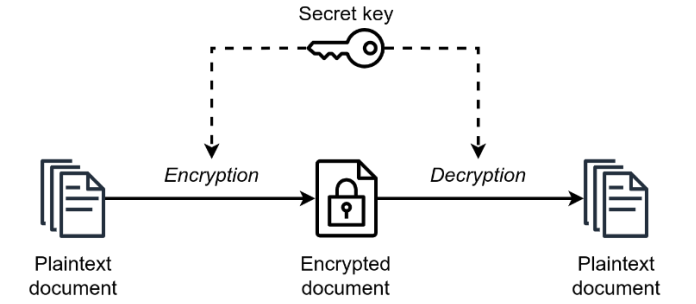
Authors: Ajin S, Dr. Bijimol T K

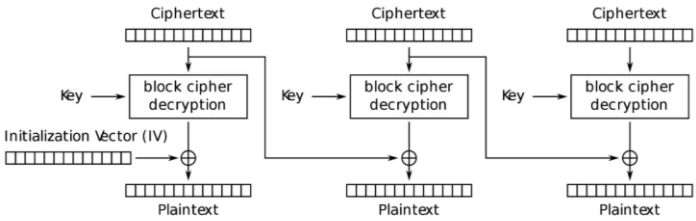
Year: 2022

Abstract:

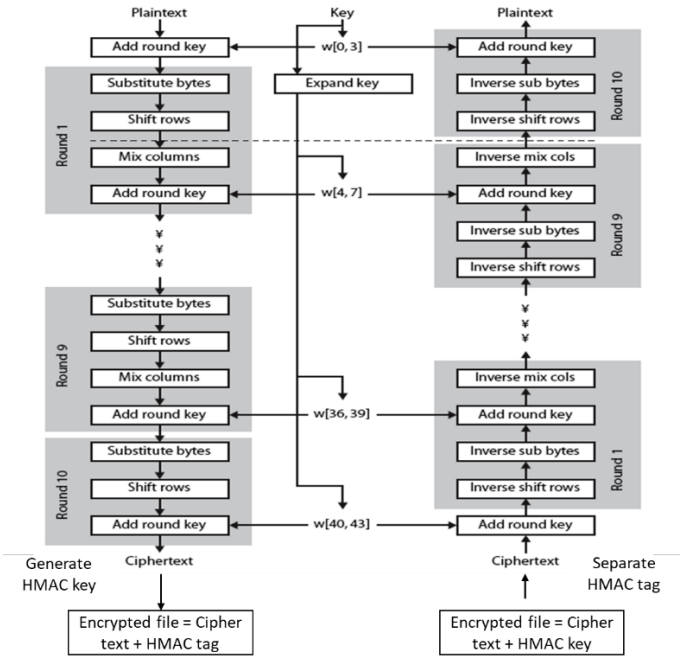
This paper provides a brief description of the Fernet key encryption.

V. SYSTEM DESIGN

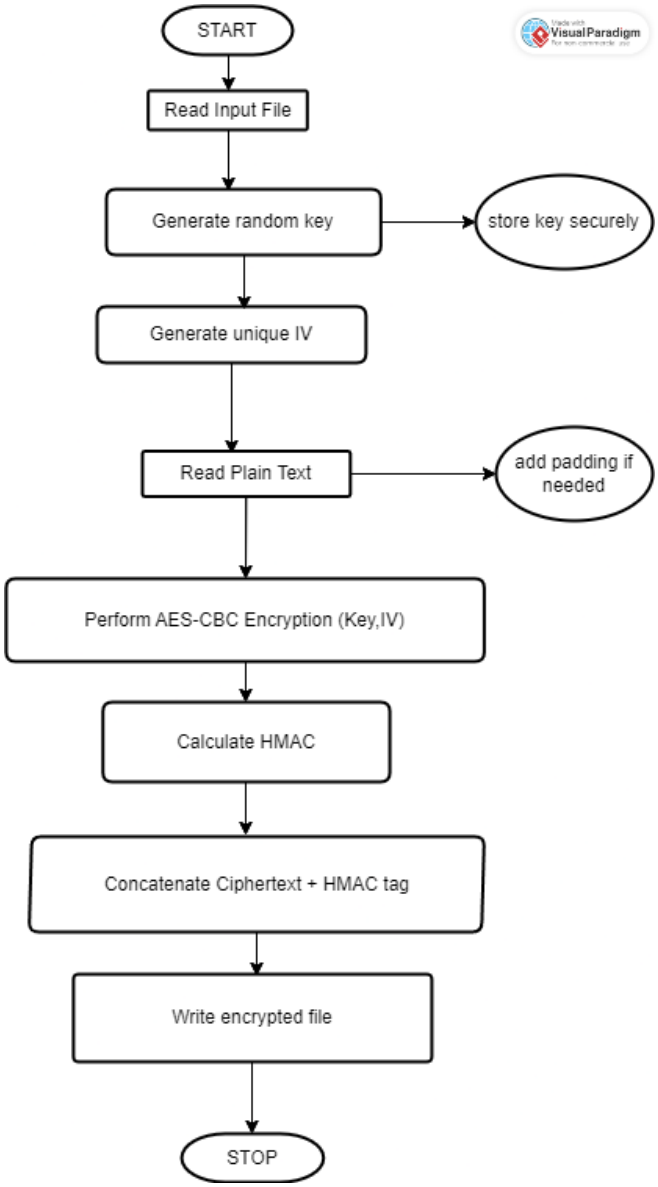




Cipher Block Chaining (CBC) mode decryption



VI. FLOWCHART



1. Input: Audio/Video/Image data to be encrypted.
2. Key Management: Fernet randomly generates a 32-byte key.
3. Initialization Vector (IV) Generation: Fernet internally creates a unique, random 16-byte IV for each encryption operation.
4. File Preparation:

- Read File: Read the Audio/Video/Image data into memory.
- Padding (Optional): If the file size is not a multiple of the AES block size (16 bytes), add padding bytes.

5. Encryption: Encrypt the file data using AES-CBC with the key and IV.

6. HMAC Calculation: Calculate the HMAC tag using the cipher text and a separate HMAC key (can be the same as the encryption key).

7. Combine Results: Append the HMAC tag to the end of the combined data (IV + encrypted data).

8. Write Encrypted File: Write the combined output (encrypted data + HMAC tag) to a new file.

9. Outputs: Encrypted version of the input file

VII. TERMINOLOGY

Fernet Cipher = symmetric encryption + authentication of data. It makes use of 256-bit AES in CBC mode and an HMAC with SHA-256.

AES-CBC:

It is a fast, symmetric-key encryption algorithm. Fernet uses 256-bit AES keys because it is considered the most secure.

AES is a block cipher. If you have a large amount of data, it cannot be encrypted all at once. First, it needs to be divided into blocks that have a fixed length, and each block is processed separately.

The result of the XOR operation is then run through the many steps of the AES block cipher, alongside the key. This produces a block of ciphertext, which is then XORed with the next block of the plaintext that needs to be encrypted. This means that in the second block, as well as all subsequent blocks, the output from the previous block acts in much the same way as the initialization vector did for the first block.

HMAC:

AES-CBC doesn't have in-built authentication. HMAC stands for hash-based message authentication code. HMACs use secret keys and hash functions in a special way so that a recipient can verify the authenticity and integrity of data that they have received.

VIII. SOFTWARE SPECIFICATION

1. Front-End Requirements:

HTML:

- Ability to create a user interface for file selection (audio, video, image).
- Ability to display options for encryption and decryption.
- Ability to handle user input for the encryption key.
- Ability to display progress indicators or success/error messages during encryption/decryption.

CSS:

- Ability to style the user interface for a visually appealing and user-friendly experience.
- Ability to ensure responsiveness across different screen sizes.

JavaScript:

- Ability to handle user interactions with the interface elements (file selection, button clicks, etc.).
- Ability to communicate with the back-end server using techniques like AJAX or WebSockets for file upload/download and encryption/decryption requests.
- Libraries: Consider using JavaScript libraries for file handling (like FileReader API) and user interface enhancements (like progress bars or modal windows).

2. Back-End Requirements:

Server-Side Scripting Language:

You'll need a server-side scripting language like Python or Node.js to handle the actual encryption/decryption logic.

Cryptography Library:

A library like cryptography (Python) or crypto-js (JavaScript) is required to implement the Fernet cipher for encryption and decryption operations.

File I/O Libraries:

Libraries for handling file uploads, downloads, and manipulation on the server-side are necessary.

Security Considerations:

Secure key storage: The Fernet key must be securely stored on the server-side. Consider techniques like hashing and salting the key before storage.

Secure communication: Ensure secure communication between the front-end and back-end using HTTPS to protect data transmission.

IX. IMPLEMENTATION

1. We will first select an audio file by clicking on the "choose file' button

2. A pop up menu will open where we can browse and select the file we wish to encrypt.

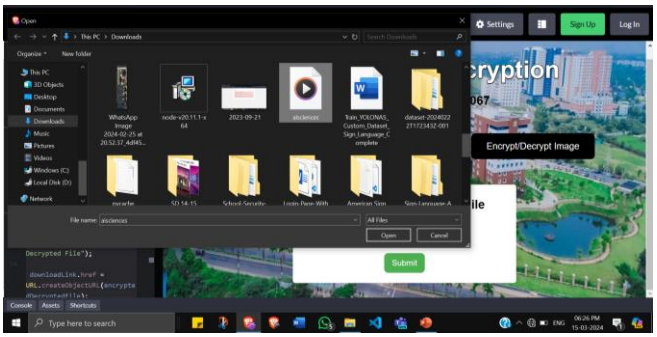
3. Then click on the "Submit" button.

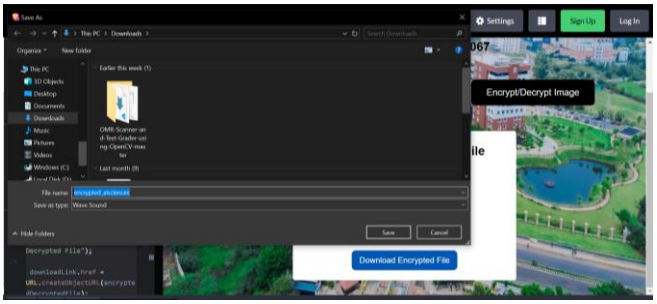
4. The code will take some time to encrypt the file depending on its size and will then show you an option to download the encrypted file.

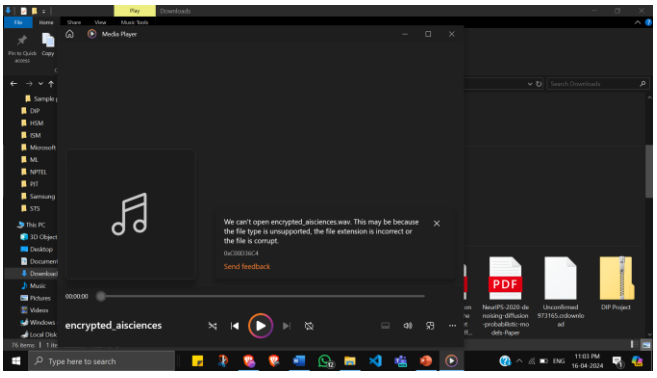
5. A pop up menu will open where we can browse and select the folder/destination in which we want to save the encrypted file.

6. The same process is to be followed while decryption. Just make sure the decryption button is enabled.

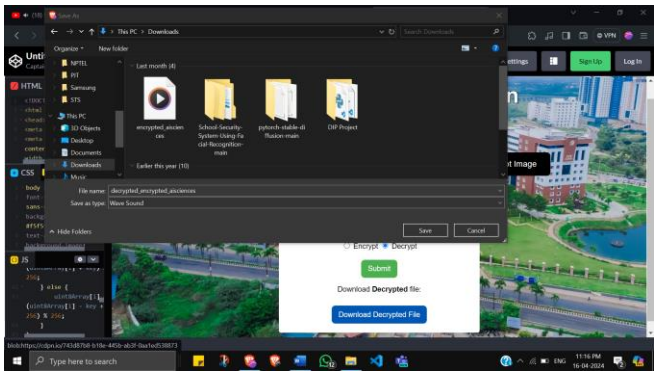
Encryption Process:

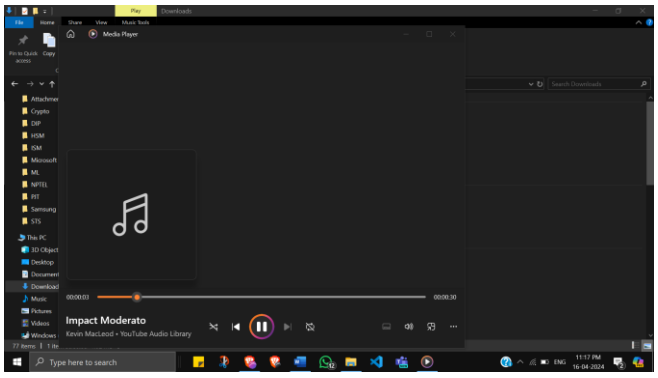






Decryption Process:





As you can see, the encrypted audio file has been restored to its original format. You can perform the same encryption/decryption processes for other file formats like video and image.

IX. CRYPTANALYSIS

1. Weak Encryption Key:

Fernet relies on a secret key for encryption and decryption. If this key is weak (easily guessed or obtained through brute-force attacks), an attacker could potentially decrypt the files.

Mitigation: Enforce strong key generation practices. Use a cryptographically secure random number generator to create the key and ensure proper key length (typically 256 bits).

2. Key Storage and Management:

Improper key storage on the server can be a major security risk. If an attacker gains access to the server and the key is stored in plain text, they can decrypt all the files.

Mitigation: Implement secure key storage mechanisms. Hash and salt the key before storing it on the server. Consider using environment variables or a dedicated key management solution.

3. Brute-Force Attacks (Theoretical):

While computationally expensive, an attacker with significant resources could attempt a brute-force attack on shorter keys (less than 128 bits) by trying every possible combination until they find the correct one.

Mitigation: Enforce strong key length (ideally 256 bits) to make brute-force attacks impractical.

4. Side-Channel Attacks (Potential):

In rare cases, depending on the implementation, side-channel attacks might be possible. These exploit information leaks during encryption/decryption processes (timing variations, power consumption) to potentially guess the key.

Mitigation: This is a complex area, but using well-established cryptographic libraries like cryptography (Python) helps mitigate such risks as they are designed with security best practices in mind.

5. Social Engineering Attacks:

An attacker might try to trick users into revealing the encryption key through phishing emails or social manipulation. This isn't a weakness of Fernet itself but a human factor to consider.

Mitigation: Educate users about social engineering threats and emphasize the importance of keeping the encryption key confidential.

IX. CONCLUSION

This project demonstrates a file encryption/decryption web application using Fernet for audio, video, and image files. By leveraging HTML, CSS, and JavaScript for the front-end and a server-side scripting language with a cryptography library for the back-end, you can create a user-friendly tool for securing sensitive files.

X. FUTURE SCOPE

1. Integration with Cloud Storage: Instead of storing encrypted files on the server, consider integrating with cloud storage platforms like Amazon S3 or Google Cloud Storage for scalability and redundancy.
2. Advanced User Management: Implement user authentication and access control mechanisms to allow multiple users and restrict access to sensitive files.
3. Alternative Encryption Techniques: Explore integrating other encryption algorithms or key management solutions for more robust security needs.
4. Mobile Application Development: Develop a mobile application using frameworks like React Native or Flutter to extend functionality beyond web browsers.

XI. REFERENCES

[1] Performance of Fernet and Aes Algorithms with Existing Encryption Techniques by Pronika, Supriya P.Panda, S.S. Tyagi, 2022
[2] Comparative Analysis of Cryptographic Algorithms in Securing Data by Taylor, Onate E., Emmah, Victor T., 2018
[3] Securing the Data in Cloud Using Fernet Technique by Chinni Prashanth, D Bala Sai Teja, V Lavanya, 2022
[4] Cloud Storage Security using Firebase and Fernet Encryption by Dhruv Sharma, C. Fancy, 2022
[5] Proceedings of the National Conference on Emerging Computer by Ajin S, Dr. Bijimol T K, 2022
[6] Performance of Fernet and Aes Algorithms with Existing Encryption Techniques by Pronika, Supriya P.Panda, S.S. Tyagi, 2022
[7] How to decrypt a value from file, encrypted using Fernet by Stack Overflow community, 2020
[8] FERNET System by Zenodo community, 2021
[9] FERNET : Symmetric Key Encryption for Password Security by Zenodo community, 2021
[10] Performance of Fernet and Aes Algorithms with Existing Encryption Techniques by Pronika, Supriya P.Panda, S.S. Tyagi, 20221
[11] How to decrypt a value from file, encrypted using Fernet by Stack Overflow community, 20202
[12] Enhancing Security of Cloud Data through Encryption with AES and Fernet by Kaur, R., & Singh, B., 20213
[13] FERNET : Symmetric Key Encryption for Password Security by Zenodo community, 20214
[14] Cloud Storage Security using Firebase and Fernet Encryption by IJETT Journal, 20205