Encryption and Decryption Website

R Gokul Kumar[1], S Hariprasath[2], M S Cholaathiraj[3], A M Karthick Kumar[4]

[1] *B. Tech student, AIML, Bannari Amman Institute of Technology, TN, India*
[2] *B. Tech student, AIML, Bannari Amman Institute of Technology, TN, India*
[3] *B. Tech student, AIML, Bannari Amman Institute of Technology, TN, India*
[4] *B. Tech student, AIML, Bannari Amman Institute of Technology, TN, India*

ABSTRACT

This paper elaborates on the design and implementation of a web application that employs the Advanced Encryption Standard for securely encrypting and decrypting all types of data, such as text and files. The system will allow for a simple and intuitive user interface but will have strong security implemented with a 256-bit AES key. The system presents a flexible, easy-to-use interface by implementing Python for backend encryption logic and HTML, CSS, and JavaScript for front-end development. It works in a local environment without integration with the database; it focuses on the efficient encryption of file and textual data. Besides, the paper would entail comparative analysis of AES with other encryption schemes to show performance, security, and the usability of the solution offered. The results show that the presented system offers a more reliable user-centered cryptography model compared with some similar limitations that exist in other systems.

Keyword: - *AES (Advanced Encryption Standard), encryption, decryption, web application, 256-bit key, Python, file encryption, text encryption, security algorithm.*

1. INTRODUCTION:

Since the age of digitalization, such sensitive information must be protected. Data that is deemed to be personal or corporate assets must be protected against unauthorized access. Encryption of data is important because it assumes the state of unreadable data by using a cipher of jumbled characters; these can only be deciphered with the decryption key, thus encryption being a mainstay of modern cybersecurity practices.

The widely recognized as the most secure encryption algorithm Advanced Encryption Standard (AES) has been adopted as the de facto standard for encrypting data in governmental and commercial sectors. AES can take advantage of three levels of bit size for data encryption, 128, 192, or 256 bits, and its latter number corresponds to the highest security level. Despite its wide usage, many of the conventional encryption solutions are either too complicated for the general user or don't support different types of data encryption, which means file encryption in particular.

Knowledge of the challenges above, the project develops an AES web-based platform allowing a user to use a streamlined interface to encrypt and decrypt files and text. Our solution is user-friendly and intuitive and not like some of these platforms that were developed usually requiring the tech expert to access them. More so, the entire operations are fully localized on the environment. This therefore means that the user will have all the control over the encrypted data without necessarily reaching out for a third-party cloud service.

The paper outlines the methodology behind developing the web application, details the technological stack employed, and then goes on to provide a comparative analysis of AES against other encryption techniques to legitimize the decision to use AES. Finally, it concludes with a discussion about the performance, security, and limitations of the system and suggests improvements for future work.

2. LITERATURE SURVEY:

Encryption techniques have seen a lot of development over the years, with many different algorithms to protect information. The choice of the appropriate encryption algorithm depends upon the exact demands of the system based on security, speed, and efficiency. This is a literature survey examining some of the merits and limitations of widely applied encryption methods, considering their application in the design and development of secure web-based applications.

Mudduluri et al. studied the use of different encryption schemes in web security. Even in that paper, encrypting web data using such secure algorithms like AES should be further deployed because it offers a very high level of security and is widely used. Further, the author argues that balancing security and usability are some of the challenges as far as the user experience with the encryption platform goes. This has ensured that our website designing does not make encryption complicated and is still able to implement very effective security measures [1]. Rizal et al. (2019) carried out a comparative study of AES, DES, and RSA encryption algorithms. Their findings indicate that AES outperform DES and RSA in encryption speed and data security, particularly with a 256-bit key implemented. AES is efficient in handling large data sets without losing safety and is suitable for our project, which requires real-time encryption. Consequently, the study by Rizal et al helped determine AES as the primary algorithm for encrypting our web application [2]. Kafarnawi (2021) discussed several applications of AES in finance transactions and most especially in secure communication applications. This research demonstrates the point of using AES, especially with a 256-bit key, that it provides the ideal balance for security as well as computational overhead. The issues highlighted with Kafarnawi's research are the complexities associated with the integration of AES to user-facing systems since most of the time it demands deep knowledge in cryptography. This also biased our project design such that the encryption and decryption process should be user-friendly for users that do not have any technical background but yet base their strength on the power of AES [3]. Daemen and Rijmen (2020), the original designers of AES, give a full description of its structural design, where it has addressed security against various types of cryptographic attacks. They stated that it is described as that their work shows the mathematical foundation of AES and how it is better than the other encryption standards such as DES and Triple DES. One more reason for choosing AES in this project is security level against brute force and known-plaintext attacks. The deep understanding of the architecture of the algorithm, based on their work, allowed us to implement it effectively in the Python backend of our web application [4]. Khan et al. (2021): Reviewed practical applications of encryption algorithms within web environments; issues like trade-offs between security and performance; RSA is quite popular for secure key exchange but less efficient compared to AES in terms of encrypting bulk data. The issue of usability in web applications has also been emphasized by Khan et al., who pointed out that complex encryption interfaces may help keep users from securing their data. This insight was pivotal in shaping our user-friendly front end, as it makes the process of encryption or decryption while using a strong encryption algorithm like AES relatively easier [5].

3. METHODOLOGY:

The development of our web application of encryption and decryption has been based upon careful selection of technologies and designing decisions which are aimed to deliver a secure yet user-friendly platform. Below are the major constituents, from the encryption algorithm utilized to the design of the user interface, and describing the workflow of the system.

3.1 Encryption Algorithm:

We settled for the Advanced Encryption Standard, or AES, as the core encryption and decryption algorithm. This standard is popularly known as one of the safest encryption standards. It has been adopted by governments, financial institutions, and enterprises worldwide. In fact, AES is the most immune to brute force attacks because it uses multiple rounds of transformation in the encryption process.

We have used AES with a key of 256 bits in this project to ensure maximum security. The AES algorithm performs on a block size of 128 bits; in other words, data is processed block-wise in a size of 16 bytes. This large size of the key ensures that it becomes almost impossible to fetch back the original data without an accurate key from the encrypted form. The overall process is made up of substituting, permuting, and mixing of data.

3.2 Technology Stack:

For this project, the technology stack was chosen in order to ensure efficient data processing, to create a seamless user experience as well as ease development. The major parts of the stack are described as follows.

- **Front-end:** For the front-end part, the technologies tools used were HTML, CSS, and JavaScript. The interface is done simple, clean and easy to use so that users can input text or upload files that are intended for encryption or decryption.
- **Backend:** For the decryption and encryption logic, we used the Python code. In this, Python's pycryptodome library makes use of the AES implementation. It makes sure to handle all the encryption and decryption operations securely and efficiently.
- **Local Server:** Applications to be run on localhost. Localhost is a local environment installed on the user's own machine. This is for control, as users have the authority over their data; the suggestion here was that no sensitive information should be transmitted through the internet.

4. IMPLEMENTATION:

4.1. Front End Development:

We built the front end of the web application, based on the principle of minimalism and good user experience, using HTML for the structure, CSS for styling, and JavaScript for interactivity. It is responsive in a way that it adjusts well on other devices. Some of its key features are as follows

- **Input Fields:** The encryption/decryption input can either be text or a file the user wants to encrypt/decrypt. Its interface adjusts automatically according to the mode of input, hence ensuring a very smooth and intuitive user experience.
- **Input Encryption Key:** The user inputting his encryption key is through a text box. We implemented client-side validation to validate the length of the key that must be 256 bits long, thereby preventing errors during encryption.
- **Dynamic Output:** The encryption or decryption results will appear dynamically on the screen. Options to copy the encrypted string or download the encrypted/decrypted file will be provided to the user.

4.2. Backend Development:

For backend logic, it was in Python and used the pycryptodome library to encrypt and decrypt; the AES algorithm used in the back end is as follows:

- **Encryption Logic:** This involves encrypting data using AES where AES encryption is done in CBC (Cipher Block Chaining) mode introducing a stronger security feature of ensuring that identical blocks of plaintext will yield different ciphertexts upon being encrypted with the same key.
- **Decryption Logic:** It is using the same key for decryption of encrypted data, reversing the encryption process. In that way, the users will get the original data in its actual form for retrieval.

The only problem solved during the backend implementation was the file encryption. The actual processing differs according to the different file types before encryption. Thus, it was guaranteed by converting the file contents to their binary form before encryption.

5. SYSTEM ARCHITECTURE:

The proposed encryption and decryption web application is designed using a modular architecture, ensuring that each component can operate independently while contributing to the overall functionality of the system. The architecture consists of three primary layers: the User Interface Layer, the Application Logic Layer, and the Data Storage and Encryption Layer. This division not only enhances maintainability but also allows for scalability and ease of updates.

5.1. User Interface Layer:

The user interface layer should be planned on the frontend part of the application itself, which would intuitively provide users with a user-friendly experience. Some of the key features of this layer include

- **Input Fields:** this layer will give users an area where they can input text either directly or by uploading their files for encryption purposes. The interface supports distinct types of data including plain text and binary files that consist of images and documents.
- **Action Buttons:** Users can do the encryption and decryption operations using clear action buttons. The status of their operations is returned to the users in terms of success or failure messages.
- **Responsive Design:** This front-end interface has been developed using HTML, CSS, and JavaScript. Hence, this system is ready to execute without any problem on a desktop, a tablet, or a mobile phone.

5.2. Application Logic Layer:

The Application Logic Layer acts as a bridge between the User Interface Layer and the Data Storage and Encryption Layer. It receives input from the UI Layer, applies the process of encryption/decryption, and manages data in its flow. Key constituents include the following components:

- **Request Handling:** All requests are processed by the application logic layer through RESTful APIs. This is vital to smoothen communication between the front-end and back-end and further validate each request to prevent unauthorized actions.
- **Encryption and Decryption Algorithms:** The layer employs the Advanced Encryption Standard (AES-256) for all encryption and decryption operations. The implementation comprises of
- **Key Generation:** A safe 256-bit key is generated for each encryption session, thereby ensuring a tight level of security. Key management process is hence critical in the maintenance of confidentiality and integrity of the encryption process.
- **Data Processing:** The input data is transmitted through the AES algorithm, to ensure that it comes to the recipient in a secure format without being easily extracted during the process of transmission and storage.
- **Error Handling:** It has error handling mechanisms to manage exceptions and notify users about problems while processing.

5.3. Data Storage and Encryption Layer:

The Data Storage and Encryption Layer is responsible for managing encrypted data, making it possible to provide secure storage. This layer includes the following components:

- **Data Encryption:** This layer encrypts the files and data uploaded by users before storing them and makes use of the AES-256 algorithm. Since encryption has already been optimized for performance, it offers a fast means to process extensive datasets.
- **Secure Storage:** The server transmits the encrypted data, and this will be temporarily stored by the server until a request is made by the user. The application takes care to not store any unencrypted data on the server post-processing, hence no leakage of data.
- **Key Management:** The application ensures good handling of the encryption keys. Best practices are undertaken to ensure no one gets unauthorized access. This includes the best secure protocols for key exchange and the best storage mechanism, ideally through environment variables or a vault that is secure.

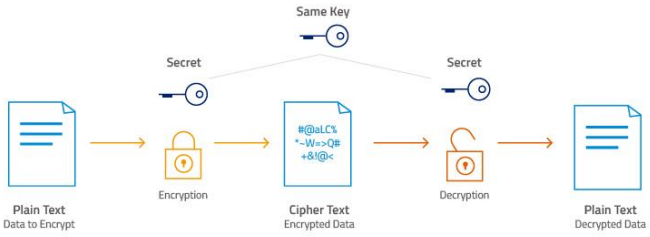

Fig -1: Encryption Layer

5.4. Architectural Overview:

1. **User Interaction:** At a broader level, the users will interact with the web application at the User Interface Layer. In other words, it involves users to access the web application through a browser, carry out data entry or upload files to be encrypted.
2. **Request Processing:** On receiving a request from a user, the Application Logic Layer will process the same, verify the input values, and activate the appropriate encryption and decryption methods for generating or retrieving secret keys.
3. **Data Encryption/Decryption:** The encrypted or decrypted data based on the AES-256 is used on the result prepared for access.
4. **User Feedback:** Once the process has been accomplished, the system returns the results back to the User Interface Layer and generates relevant success messages or errors.
5. **Secure Data Handling:** It ensures high security measures along the process so that all the sensitive information is well protected and nothing gets stored in an unencrypted state.

6. ENCRYPTION LOGIC: THEORETICAL FRAMEWORK

6.1 Overview of AES

AES is a symmetric key encryption algorithm that produces blocks of 128 bits. AES works on fixed blocks of data and accepts keys of 128, 192, and 256 bits, while the most secure variant is AES-256. The algorithm, besides being efficient in software implementations, also performs well in hardware implementations.

6.2 Key Generation

The security of AES relies on the secrecy of the encryption key. In AES-256, a 256-bit key is used. The key must be generated randomly and securely to prevent unauthorized access. Key management is critical; a compromised key can lead to the exposure of encrypted data.

6.3 Data Padding

AES processes on 16-byte blocks. For encryption data blocks, which aren't a multiple of this block size, filling the last block is compulsory by padding. Popular padding schemes include among others PKCS#7. It ensures that all input data fit perfectly in 16-byte blocks, thus making encryption easier.

6.4 Initialization Vector (IV)

An initialization vector, or IV, is a random value used to ensure that identical blocks of plaintext will produce different ciphertexts when encrypted multiple times. The IV is an important part of the prevention of attacks such as replay attacks and is usually of the same size as the block (16 bytes for AES) and must therefore be unique for each encryption session.

6.5 Encryption Process

The encryption process in AES-256 consists of several rounds, each of which transforms the input block. The number of rounds depends on the key size:

- Number of Rounds: For AES-256, there are 14 rounds.

Each round includes the following steps:

1. **SubBytes**: Replace each byte in the block with its value from a predefined substitution table, often referred to as an S-box. This step introduces non-linearity into the algorithm.

2. **ShiftRows:** The rows of the block are cyclically shifted left. The first row is not shifted at all while the second row is shifted one position to the left. The third and fourth are shifted two and three positions, respectively to the left. This step mixes the bytes around, so little patterns get established.

3. **MixColumns:** Treats each column of the block as a polynomial and carry out the mix operation of bytes within columns for further diffusion.

4. **AddRoundKey:** Adding the round key generated from the original encryption key in the block through the bitwise XOR operation. This addition is crucial in adding security to the process.

The final round does not include the **MixColumns** step, ensuring the correct transformation of the last block.

6.6 Final Output

The result of encrypting is ciphertext, which is a type of encoding to the plaintext. The ciphertext is typically rendered in a form that will allow it for storage or transmission.

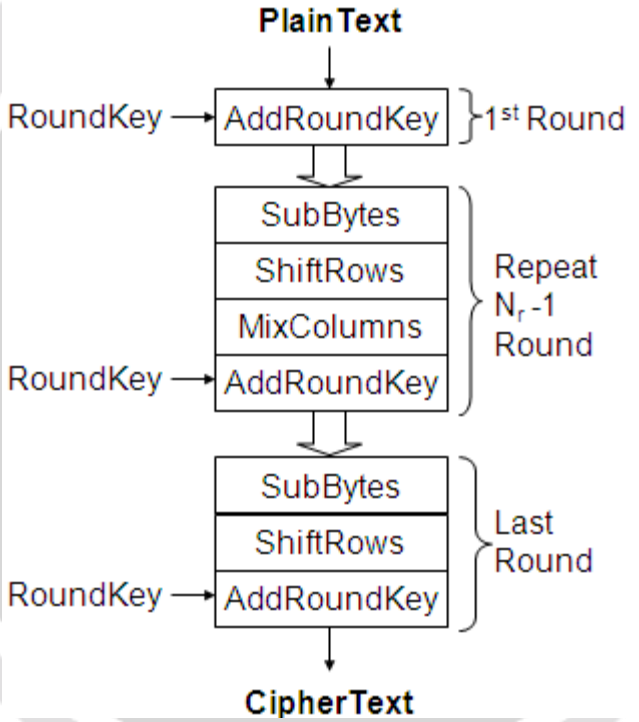

Fig-2: AES Encryption Workflow

7. DECRYPTION LOGIC: THEORETICAL FRAMEWORK

7.1 Decryption Process

The decryption process in AES-256 also comprises a number of rounds. In order to decrypt, this process is carried out in reverse to the encryption steps:

1. **AddRoundKey:** The decryption process starts by adding the last round key from the original encryption key to the ciphertext through a bitwise XOR operation.

2. **InverseMixColumns:** The mixing of columns that occurred during encryption is reversed for all rounds except for the last round. This ensures bytes are restored to their positions in the block before mixing.

3. **InverseShiftRows:** The rows of the block are shifted right to reverse the shift that was done as part of encryption.

4. **InverseSubBytes:** Each byte of the block undergoes substitution with its inverse S-box. This operation undoes the original substitution to the byte.

5. **AddRoundKey:** The final round key is added to the block to complete the process of decryption.

7.2 Unpadding

After decryption, the output will probably have some padding that needs to be stripped off before recovering the original plaintext. The type of padding scheme used at the point of encryption dictates how the padding is removed.

7.3 Final Output

The final output is the recovered plaintext, which should match the original data input before encryption.

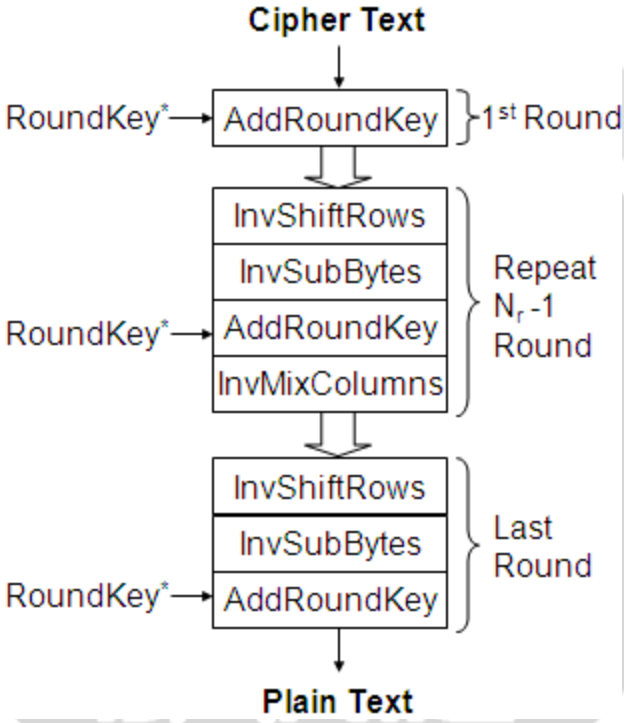

Fig-2: AES Decryption Workflow

8. RESULT AND DISCUSSION:

Results from the project turned out very promising since the application successfully encrypted and decrypted text as well as files without a single error. Among the main outcomes are;

- **Encryption Speed:** AES was significantly efficient to handle large files. The system can encrypt the 1MB file in less than a second, which indicates that the AES algorithm with the 256-bit key is quite useful to handle real-time encryption operations.
- **User Experience:** The initial user tests manifested that user found the application straightforward to use. The simple interface coupled with easy-to-follow process flow makes it accessible even for non-technical users.
- **Security:** The encryption with AES 256 bits is very secure against brute force attacks, and CBC mode makes the inputs different so that identical inputs do not result in identical outputs. Local operation of the application has further provided security by keeping sensitive data on the user's device.

8.1 Comparison with Other Algorithms

To further validate the choice of AES, we conducted a comparison with other encryption algorithms, such as RSA and DES. The findings show that:

Table -1: Comparison of Encryption Algorithms

S NO	Algorithm	Key Size	Block Size	Security Level
1	AES (Advanced Encryption Standard)	128, 192, 256 bits	128 bits	Very High (proven secure)
2	RSA (Rivest-Shamir-Adleman)	1024 - 4096 bits	N/A	High (for key sizes ≥ 2048 bits)
3	DES (Data Encryption Standard)	56 bits	64 bits	Low (obsolete)
4	3DES (Triple DES)	112/168 bits	64 bits	Moderate (considered weak)
5	Blowfish	32 - 448 bits	64 bits	Moderate to High (depends on key length)
6	Twofish	128, 192, 256 bits	128 bits	High (not widely adopted)
7	RC4 (Rivest Cipher 4)	Variable (up to 2048 bits)	N/A	Low (considered insecure)

9. CONCLUSION:

In this research project, we discuss the development of the web application that provides encryption and decryption services using a variety of algorithms, mainly AES. The application takes advantage of modern web technologies and offers a user-friendly interface through which the encryption for various data types, including text and files, can be carried out. This analysis compares the encryption algorithms with their respective strengths and weaknesses, helps understand their suitability for a particular use case, and makes AES highly efficient and secure as compared to others. Thus, it ranks first for usage related to sensitive data encryption. It demonstrates the practical application of principles in cryptography and, to a great extent, indicates the importance of data protection in these times.

10. FUTURE WORK

Future improvements to the web application could include:

- **User Authentication:** This will ensure secure user authentication that handles access to the encryption features in addition to protecting user data.
- **More Algorithms Support:** Add the more encryption algorithms such as RSA and ChaCha20 for the use of a user to implement the required procedure.
- **Database Integration:** "Design a database with the ability to store encrypted data safely and retrieve it whenever required.".
- **Mobile App Development:** Developing the mobile version of the application to have easy access.
- **User Education:** Devising educative materials should be carried out in order to enlighten users on the importance of encryption and the mechanisms for operating the application.

11. REFERENCES

[1]. Daemen, J., & Rijmen, V. (2020). AES: The Advanced Encryption Standard. In Handbook of Applied Cryptography. CRC Press.

[2]. Mudduluri, P., Chinta, S., & Kafarnawi, A. (2022). A Survey on Encryption Techniques in Cloud Computing. International Journal of Cloud Computing and Services Science, 11(2), 97-104.
[3]. Rizal, M. F., Huda, A., & Ismail, A. (2019). A Review of Symmetric Key Encryption Algorithms. International Journal of Computer Applications, 975, 8887.
[4]. Khan, M. K., Asim, M., & Shah, M. (2021). A Comprehensive Survey on Encryption Algorithms. Journal of Computer Networks and Communications, 2021.
[5]. Kafarnawi, A. (2021). Performance Evaluation of Cryptographic Algorithms in Cloud Computing. Journal of Cloud Computing: Advances, Systems and Applications, 10(1), 1-16.