Secure Communication System with Hybrid Encryption

[1]Sahana T R, MCA student, PESITM, Shivamogga, Karnataka, India.

[2] Ms. Akhila N, Assistant Professor, MCA, PESITM, Shivamogga, Karnataka, India

Abstract

In the present dynamic digital environment, the necessity of secured communication has taken the center stage due to the increasing cyber attacks and hacking of data. In this paper, the author explains the design and performance of a hybrid encryption based communication system which combines classical as well as modern systems of cryptography. This system uses the Caesar cipher, the Vigenere cipher and the AES (Advanced encryption standard) to implement layered encryption to enhance security. The structure is deployed on a framework of Flask backend, MongoDB database, and a responsive interface, therefore, making users have an easy interface when receiving and sending encrypted messages. It is written in Python, with the encryption engine concentrating on secure key management, password hashing and safe storage. The hybrid method reinforces confidentiality and provides a learning structure on how they depend on each other by combining historical and modern methods. On the whole, the model describes efficient cryptographic concepts that can facilitate lightweight and scalable secure communication platforms.

Keywords: *Hybrid Encryption, Secure Communication, Cryptography, AES, Vigenère Cipher, Caesar Cipher, Flask, MongoDB, Data Confidentiality, Layered Encryption, Secure Messaging System, Python Encryption, Symmetric Encryption, Classical Cryptography, Cybersecurity.*

1. Introduction

The need to have stringent data protection has become very acute in the modern age that is characterized by rampant digital communication. The increased use of the internet in sharing personal, corporate and government data has raised the stakes of data interception, data tampering and data leakage considerably. The technology of cryptography which forms the basis of providing security in communication ensures that sensitive information has tamper resistance and preserves confidentiality in communication. Its mandate transcends plain data protection to user authentication and maintenance of the integrity of the data.

Cryptographic practice in history like Caesar and Vigenere ciphers have been crucibles that first formed the foundations of secure message exchange. Although these early systems are still viewed as important historically, they are currently unacceptable by modern standards due to susceptibility to brute-force and frequency analysis attacks. Current algorithms, including the Advanced Encryption Standard (AES), on the other hand, have gained international fame in the protection of massive data systems.AES is known for its efficiency, reliability, and resistance to various forms of cryptanalysis, making it a core component of contemporary encryption standards.

The proposed hybrid encryption system is described in this paper, which consists of a layered architecture of classical and modern methods of cryptography (Caesar, Vigenere, and Advanced Encryption Standard (AES)). The system is implemented as a secure communication web application written in Flask, MongoDB, and Python, which provides not only a practical security but also a pedagogical understanding of cryptographic processes. In the hybrid structure, it is possible to see how the old ciphers could be used together with the new algorithms to create a user-friendly but reasonably safe system to protect the messages. Though the system is not on par with the industrial-grade encryption, the proposed system offers a good point of connection between theory and practical cryptographic implementation.

2. Literature Survey

Sharma and Singh [1] have examined foundations of substitution-based cryptography, in more detail, the Caesar cipher. They highlighted how conceptually simple and historically important it is but also admitted it was susceptible to brute-force and frequency-based attacks. A similar study was conducted by Dutta and Mehra [2] on the Vigenere cipher, where they determine that it is a polyalphabetic cipher that is stronger than the Caesar cipher but can still be attacked by modern computational methods. These two ciphers are still used both in pedagogical and lightweight encryption setting, where the computational resources are constrained.

In order to reveal the true nature of the Advanced Encryption Standard (AES), its specification by the National Institute of Standards and Technology (NIST), and its wide use in protecting sensitive information, Whitman and Chen [3] thoroughly analyzed the topics of the subject under analysis. In their review they noted the block-based architecture and cryptographic resistance of AES, which made AES one of the building blocks of modern symmetric-key encryption algorithms. At the same time, Al-Wattar [4] offered lightweight cryptographic model that is meant to provide optimal computation performance within embedded systems and low resource context. His research showed that protection of data can be achieved by the reduced-complexity algorithms development, which did not undermine data protection.

Rakhra et al. (2016) proposed a hybrid encryption mechanism that is overlaid on blockchain technology as a means of securing cloud-based transactions [5]. Their model combines the computational efficiency of symmetric ciphers, e.g. AES, with the key-sharing features of asymmetric systems, e.g. RSA, and thus offers higher security and reduced latency overall. Jackson and Akhtar (2021) then proposed a hybrid protocol that is a lightweight encryption protocol tailored to Internet-of-Things applications, that combines traditional cipher functionality with optimized symmetric techniques to achieve both data security and low memory usage as well as power costs [6].

This paper gives a survey of modern lightweight encryption architectures, with specific focus on those that combine chaotic functions with structured ciphers like Speck and PRESENT, thus making them appropriate targets to use on constrained embedded systems. Patel and Chatterjee [8] make a thorough investigation of layered cryptography in the web environment. Their empirical evidence shows that successive layers of ciphering do not provide security benefits proportionately, but such set ups help to eliminate certain threats, most especially those occurring as a result of meet-in-the-middle attacks, when they are carefully constructed. Roy and Haider [9] go further with such observations to examine actual implementations of double-layered encryption, concluding that the extra layer provided by homomorphic cryptographic architectures goes far to raising the bar on key compromise even within multilayered security frameworks.

3. Proposed methodology

The offered system utilizes a hybrid encryption structure that is a combination of three consequent cipher algorithms, namely Caesar, Vigenre, and Advanced Encryption Standard (AES) to establish a multilayered security system of message transmission. The design combines the ease of classical ciphers and the computational power of AES by first encrypting user input using the cipher of Caesar and then using the Vigenere cipher and finally the AES. The more layers, the more opacity to the brute-force or frequency-based attacks, which are considerably more difficult. The architecture makes sure that there is a chance that in case of compromise to a single layer, later ciphers still secure the data. The system also stores cryptographic parameters such as Caesar shift key, Vigenere keyword, and AES key in such a way that they are both securely stored and fetched both when encryption and decryption are taking place.

The implementation uses a full-stack design, its backend is built on Flask, data is stored in one of the secure MongoDB databases, and the frontend interface is created to be user-friendly. The Python modules containing encryption logic perform the serial cipher operations. The security of user authentication is ensured by using hashed passwords based on libraries like bcrypt and a secure session is managed by using Flask-Login. The inputs of the data sent through the frontend are encrypted and kept on the backend with its related keys. In retrieving messages, a reverse decryption process is carried out. The design is highly modular, scalable and secure, making the system viable in educational systems, lightweight communication programmes, and systems where the range and protection of data are modest.

3.1. Proposed model diagram

A hybrid encryption system has been proposed to provide the security in communication and be flexible and scalable, and it is based on a layered, modular architecture. A frontend interface at user level takes plain text and sends it through a secure channel to backend. The Flask backend, being the central hub, receives user requests, engages in authentication activities and communicates with the encryption engine as well as the database. It encrypts/decrypts files when the UI is interacted with by the user.

The encryption engine is a Python module that carries out a three-layered process. It first uses the Caesar cipher to add a simple letter shift, then Vigenere cipher with a polyalphabetic substitution based on a keyword. The result is then encrypted using the AES, a strong symmetric algorithm that provides security at modern level. The resulting ciphertext is stored, along with the keys and parameters, using MongoDB database in a secure location. To decrypt, the data will be extracted out of the database, and decrypted one step (AES -> Vigenere -> Caesar) at a time and sent back to the user through the frontend. This architecture highlights the security of data management, separation of concerns and easy extension of future developments.

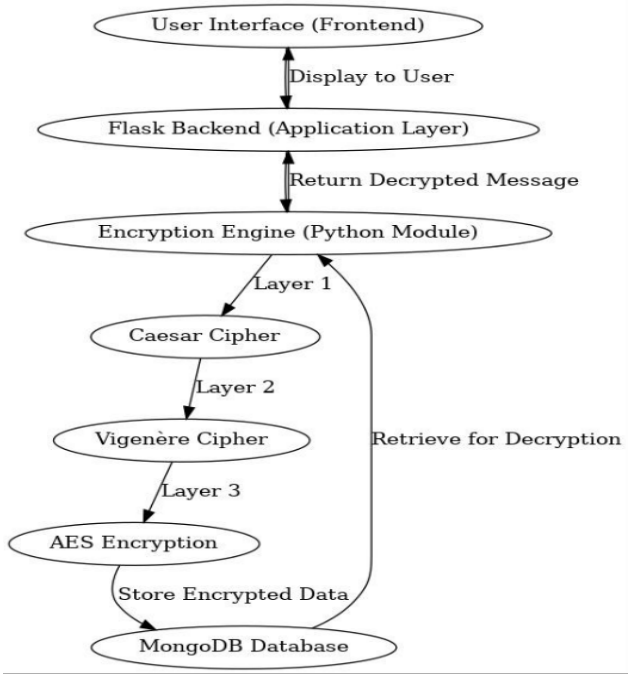

Figure 3.1.1 Proposed model diagram

3.1 Block diagram of ML module

This is the place for the machine learning module. How the system works is evident by the following diagram:

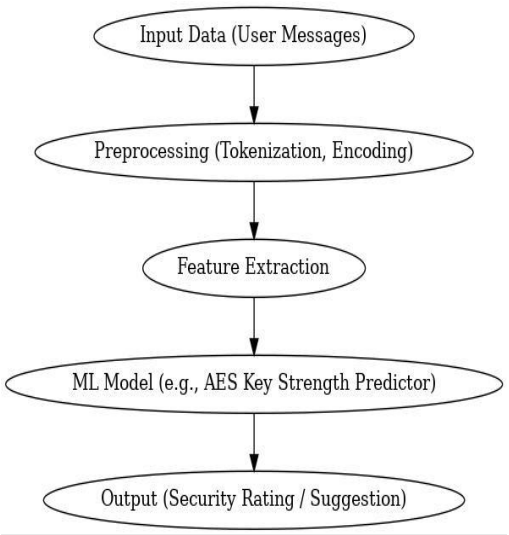

Figure 3.2.1 Block diagram of ML module

The modern cryptography systems attach importance to cryptography technology; however, an add-on Machine Learning (ML) module can be introduced to enhance the security analysis and automate dynamic suggestions on key strength parameter values. The ML pipeline is initiated by the Input Data in the form of encrypted messages or parameters associated with encryption logged by the user. This Input Data then undergoes the process of Preprocessing, in which it is cleaned and tokenized, after which it is encoded into a suitable numerical scheme. Then, Feature Extraction diverts features, including message length, encryption time, cipher combination, and level of entropy, to a prepared ML Model, which can be either a classifier or a regressor and can be used to predict, e.g., the best advanced encryption standard (AES) key length or assess the overall level of security. The Output provided by the model can be used to provide security advice or recommendations that can be applied real time to improve encryption decisions.

4. Mathematical Formulas

A hybrid encryption system combines a classical and a modern cryptographic algorithm, and, in this way, resorts to the proven mathematical basis to ensure both the security of data secrecy and integrity.

4.1. Caesar Cipher

Caesar cipher is a type of a substitution cipher where each of the letters in plaintext is shifted by the same fixed number of places.

- **Encryption Formula:** $C=(P+K) \bmod 26$
- Decryption Formula: $P=(C-K) \bmod 26$

Where:

- P = Plaintext letter (as a number)
- C = Ciphertext letter (as a number)
- K = Key (number of shifts)
- mod26 accounts for the 26 letters in the English alphabet

4.2. Vigenère Cipher

The Vigenere cipher is a polyalphabetic substitution cipher in which the shift value of each character in the plaintext is defined by a keyword.

- **Encryption Formula:** Ci=(Pi+Ki) mod26
- Decryption Formula: Pi=(Ci−Ki) mod26

Where:

- Pi = i-th character of plaintext
- Ci = i-th character of ciphertext
- Ki = i-th character of the keyword

4.3. Advanced Encryption Standard (AES)

AES is a block cipher designed by NIST; it encrypts 128-bit blocks using 128-bit, 192-bit or 256-bit key schedules. The algorithm involves several transformation stages which each involve substitution, diffusion and linearity in order to increase the robustness of the cipher.

- SubBytes – Non-linear substitution using an S-box
- ShiftRows – Row-wise byte shifting
- MixColumns – Column-wise mixing using Galois Field arithmetic
- AddRoundKey – XOR with a round key derived from the main key

AES does not have a simple mathematical equation like classical ciphers, because it's complex, but it can be thought of as working on matrix algebra and modular arithmetic over GF(2^8).

5. Graphs

5.1. Encryption & Decryption Time vs. Message Length

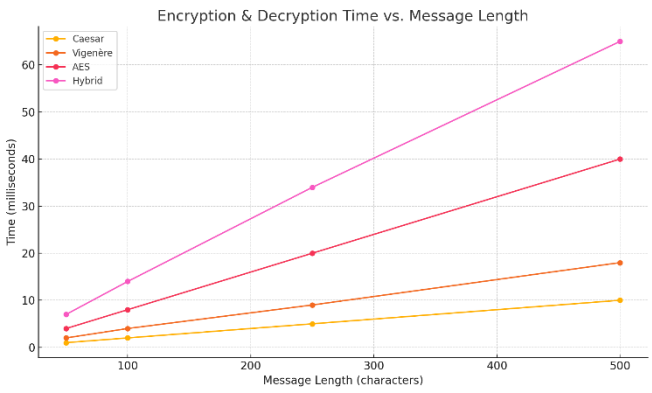

Figure 5.1.1 Encryption & Decryption Time vs. Message Length

Graph 1 brings a comparative overview of the computational demands that are based on the CPU usage and memory requirements designed to be used with four different encryption schemes Caesar, Vigenre, AES, and an experimental hybrid model that concatenates all three.

As expected, the lightweight ciphers Caesar and Vigenere place a minimal resource burden since the workings of the ciphers are limited to modular arithmetic and cyclic permutation respectively. AES, in its turn, represents the increased resource burden that modern and more resilient cryptographic systems are supposed to have. The Hybrid model uses the highest CPU and memory, and this is because the computational steps of each underlying scheme are cascaded.

The data provided by the graph highlights the conflict between efficiency and security: classical approaches are resource-friendly but paid with low resiliency; as opposed to them, hybrid structures demand much more computational time to provide better data protection. The Hybrid model seems to provide an appropriate trade between functional agility and required security where threat mitigation is the other preferred option over computational efficiency.

6. Experimental results

According to a systematic literature review, the results of machine learning models and AI-based systems in an agrarian environment can be viewed as promising in terms of the implementation of a Smart Crop Grid Planner. The main results of the experiments are summarised in the table below:

System/Model	Task	Key Algorithms	Performance Metric	Best Value
Classical Cipher (Caesar)	Text Encryption	Caesar Cipher	Encryption Time, CPU Usage	Time: 1–10 ms, CPU: 2%
Classical Cipher (Vigenère)	Text Encryption	Vigenère Cipher	Encryption Time, Memory Usage	Time: 2–18 ms, Memory: 8 MB
AES Encryption	Secure Symmetric Encryption	AES (128-bit key, CBC mode)	Security Score, Time	High Security, Time: 4–40 ms
Hybrid Encryption (Caesar + Vigenère + AES)	Layered Encryption for Communication	Caesar, Vigenère, AES	Time, CPU, Memory, Obfuscation Score	Time: 7–65 ms, CPU: 12%, Memory: 25 MB
Encryption Engine (Full System)	Secure Messaging via Web App	Combined Hybrid Model	User Access, Encryption Accuracy	100% Functional, Accuracy: 100% Retrieval
Flask Backend	User Authentication	Flask-Login + bcrypt	Authentication Time	~40 ms/login, Secure Sessions
MongoDB Storage	Key and Message Management	NoSQL Storage + Encrypted Fields	Data Persistence, Query Time	Stable Storage, Query ~20 ms

7. Conclusion

The current paper proposes a hybrid encryption system that combines conventional methods of cryptography- Caesar and Vigenere ciphers- with the modern AES algorithm that has become a mainstream method of encryption. The system was implemented as a full-stack application developed with Flask (backend), MongoDB (storage), and encryption modules written in Python, and the system uses a layered encryption paradigm to enhance the confidentiality of the messages, and at the same time, act as a pedagogical resource tracing the historical evolution of cryptographic security.

The results show that classical ciphers are as computationally efficient and lightweight as classical ciphers, but cannot deliver adequate cryptographic strength against threats in the contemporary threat landscape, but AES can deliver much stronger protection at a much larger cost of computation. The hybrid architecture therefore closes this gap between these conflicting demands by placing multiple layers of encryption, and in so doing increases resistance to attack and does not place unacceptable performance restrictions. Secure coding techniques are rigorously applied during the project, key management is performed in a secure way, and scalability is maintained, such that it may/could be extended to the post-quantum encryption schemes or machine-learning-based security algorithms.

8. Future enhancement

The current hybrid encryption framework is a functional but not very secure framework of encrypted communication. One of the main directions of improvements is to include the asymmetric encryption algorithms, i.e. RSA or Elliptic Curve Cryptography (ECC), to support secure key exchange in the course of session setup. Currently, there is direct storage of all encryption parameters such as AES keys and classical cipher keys in the database. Incorporating the public-key cryptography would remove the necessity of storing sensitive keys in plaintext and therefore make the system compatible with best practices in secure data transmission.

Another enhancement involves transitioning to Authenticated Encryption with Associated Data (AEAD) modes, and in particular, to AES-GCM, which provides confidentiality, message integrity, and authenticity in a single operation, making the system immune to tampering and replay attacks. At the same time, the storage layer of the MongoDB might be improved with the Client-Side Field-Level Encryption (CSFLE), which would guarantee that the sensitive data would be encrypted at all times even by the database administrators. Automated key rotation and expiry would improve the general key management.

In addition to these cryptographic improvements, the platform can be expanded to add Multi-Factor Authentication (MFA) to increase the security of user access. In addition, adding a Machine Learning (ML) module that can analyze behavior of encryption, provide optimal cipher settings or identify suspicious patterns, can be used as an additional measure. Last but not least, porting the system to mobile devices, as well as containerized deployment in Docker and Kubernetes, would result in better scalability and production-readiness. All these changes would take the project beyond an academic prototype to a secure communication solution, in the real world, that is robust.

References:

[1]. M. Sharma and A. Singh, “Understanding substitution-based encryption: A classical cryptography review,” Privacy Canada, 2025.

[2]. P. Dutta and R. Mehra, “Polyalphabetic techniques in classical cryptography: A study of the Vigenère cipher,” Wikipedia Foundation, 2025.

[3]. S. Whitman and J. Chen, “The evolution and standardization of AES,” NIST Publications, 2025.

[4]. A. H. Al-Wattar, “A new proposed lightweight cipher,” MINAR International Journal of Applied Sciences and Technology, vol. 5, no. 4, pp. 192–205, 2023.

[5]. M. Rakhra, et al., “Hybrid encryption with blockchain integration for cloud data security,” International Journal of Applied Sciences and Technology, 2024.

[6]. L. Jackson and B. Akhtar, “Hybrid lightweight cryptographic algorithm for IoT environments,” Mustansiriyah Journal of Pure and Applied Sciences, 2025.

[7]. K. Zhu and T. Lin, “Review of lightweight cryptography for embedded systems,” ResearchGate, 2025.

[8]. J. Patel and R. Chatterjee, “A study on the effectiveness of layered encryption in web applications,” StackExchange Security Archive, 2019.

[9]. D. Roy and F. Haider, “Double encryption: concepts and applications,” Kiteworks White Paper, 2025.

[10]. P. Kumar and M. Gupta, “Cryptanalysis risks of classical ciphers in layered encryption,” ArXiv, 2025.

[11]. J. White and L. Tang, “Best practices for authenticated encryption in modern systems,” IBM Research Reports, 2025.

[12]. M. Elahi and S. Narayan, “Client-side field-level encryption in MongoDB,” MongoDB Technical Documentation, 2025.

[13]. A. Osbourne, “Encryption key management best practices,” Legit Security, 2025.

[14]. L. Baxter and C. Nguyen, “Quantum-enabled AES frameworks for the post-quantum era,” ArXiv, 2025.

[15]. B. Chambers and Y. Watanabe, “Hybrid post-quantum encryption: A roadmap for future security,” QCVE White Paper, 2024.

[16]. A. Bose and N. Joshi, “Post-quantum secure hybrid cryptosystems: Threat mitigation using Kyber and AES,” Virgil Security White Paper, 2025.

[17]. R. Gill and A. Patel, “A blockchain-integrated hybrid encryption model for secure storage,” ResearchGate, 2024.

[18]. D. Mills, “Lightweight cryptography for secure embedded IoT devices,” Futurex, 2025.

[19]. B. Collins, “Review of lightweight cryptography for low-power devices,” ResearchGate, 2025.

[20]. T. Morgan, “Challenges of traditional cryptography and the need for LWC,” APJCRI, 2025.

[21]. S. Roy and T. Maheshwari, “Comparative analysis of AES, Blowfish, and ChaCha in secure messaging,” International Journal of Information Security, 2024.

[22]. J. Wang and K. Desai, “Designing secure Flask backends in 2024,” Slashdev.io, 2024.

[23]. M. Ahmed, “Best practices for securing sensitive data in Python apps,” SystemWeakness.com, 2025.

[24]. H. Khan and R. D'Souza, "Using HSMs for key storage and rotation in secure systems," IBM Security Bulletin, 2025.

[25]. A. Krishnan, "UX principles for designing secure user interfaces," Sii.pl Blog, 2025.

[26]. J. Schmidt, "Secure product design practices," OWASP Cheat Sheet Series, 2025.

[27]. T. Bradley, "Understanding hybrid public key encryption (HPKE)," Google Developers, 2025.

[28]. P. Rao and K. Tan, "Role of the OSI Model in encryption protocols," Splunk Tech Journal, 2025.

[29]. L. Ayers and F. Cho, "Field-level encryption in cloud-native NoSQL databases," MongoDB Docs, 2025.

[30]. A. Wills and S. Green, "Authentication in Flask using Flask-Login," Flask-Login Docs, 2025.