GMR
The Original

Safe Crypt: A Comprehensive Solution for Modern Cryptographic Security

Divya Singhal[1], Ankit Verma[2*], Amit Kumar Gupta[2], Rabi Narayan Panda[2], Vipin Kumar[2], Shashank Bhardwaj[2], Akash Rajak[3]

[1]Department of Computer Science, Noida Institute of Engineering & Technology, Greater Noida, India divyasinghal021@gmail.com
[2*]Department of Computer Applications, Krishna Institute of Engineering & Technology (KIET), Ghaziabad, Delhi-NCR, Uttar Pradesh ankit.mca4u@gmail.com
[3]Department of Computer Science, Krishna Institute of Engineering & Technology (KIET), Ghaziabad, Delhi-NCR, Uttar Pradesh, akash.rajak@kiet.edu
*Corresponding author: Ankit Verma, Department of Computer Applications, Krishna Institute of Engineering & Technology (KIET), Ghaziabad, Delhi-NCR, Uttar Pradesh ankit.mca4u@gmail.com

Abstract: Big data analytics uses information from several sources, such as Internet of Things devices and smart meters, to find patterns and spur innovation. The attack surface for hackers is expanded by the connectivity of IoT devices. Data security solutions are essential for protecting sensitive data from theft, hacking, and breaches. Choosing the right cryptographic algorithms is crucial in the field of information security to provide a strong defense against unauthorized access and data breaches. As a crucial countermeasure to encryption, cryptanalysis fosters innovation and fortifies the system's defenses against possible intrusions. The purpose of this study is to assess and contrast the two-hybrid approaches, AES-RSA and ECC-AES. The metrics used to evaluate these approaches include data kinds, input dimensions, key lengths, and encryption times. An important criterion is computational efficiency, which is evaluated by comparing the encryption and decryption times of the two methods. The efficient utilization of cryptographic strengths is confirmed by comparing the produced key sizes for AES-RSA and ECC-AES to their theoretical values. Memory profiling methods are used to analyze memory utilization by elucidating patterns of use during operation.

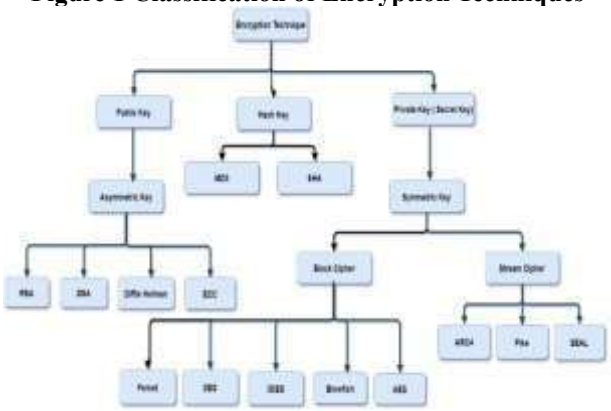
Keywords: Data Security, Cryptography, AES-RSA, ECC- AES, Blowfish, Cryptosystem/

1. Introduction

Recently, Internet of Things (IoT) devices have been utilized across several industries, including retail, manufacturing, healthcare, education, and transportation. The development of an Internet of Things application involves sensors and intelligent devices utilized for the real-time collection and analysis of environmental data [1]. The increasing prevalence of IoT devices, together with the rapid escalation in data creation and collecting, has rendered the safeguarding of data security and integrity increasingly difficult. These devices are frequently vulnerable to attackers because of their restricted memory and processing capabilities. Issues related to data aggregation and profiling stem from the collection and examination of various datasets [2, 3]. The security of IoT devices is becoming critical due to the rise in cybercrimes aimed at these devices [4]. Numerous conventional encryption techniques can safeguard data; nevertheless, their efficacy is contingent upon the secure retention of the encryption key, preventing access by possible adversaries. Given the extensive use of digital platforms and the rapid advancement of technology, the implementation of robust encryption methods and secure protocols is imperative. In the digital realm, cryptography provides essential tools and methodologies for information validation. The primary function of cryptography is to convert plaintext into ciphertext, making it incomprehensible to unauthorized parties. This change is accomplished by several mathematical procedures and cryptography methods [5]. Cryptography serves as a fundamental building block for ensuring data confidentiality and integrity within intrusion detection system (IDS) systems [6, 7]. This ensures that if unauthorized individuals gain access to the data, they are unable to decipher its content without the decryption key. However, a significant challenge arises if an attacker manages to gain access to the hidden key and intercepts encrypted messages within the network. The cryptographic techniques either symmetric or asymmetric each offer distinct security strengths while facing specific challenges [8]. In symmetric key cryptography, a private key (Secret key) is used for both encryption & decryption. The sender and receiver will have the same key [9]. Whereas in asymmetric cryptography two keys are used. One key (public key) with the sender is used for encryption and the other key (private key) is used for decryption by the receiver [10]. The categorization of different encryption techniques is shown in Figure 1. Symmetric key algorithms like Advanced Encryption Standard (AES) are designed to perform fast transformations on blocks of data or streams as they require fewer mathematical operations such as substitutions, transposition [11], and XOR. Also, they have the same small key sizes. However, it suffers from a key distribution problem (KDP) where it is difficult to secure the secret key from potential

interceptors. Securely distributing encryption keys over insecure channels is challenging. Asymmetric key algorithms such as Rivest Shamir Adleman (RSA), and Elliptic Curve Cryptography (ECC) solve KDP by using a public key for encryption and a private key for decryption. RSA encrypts the AES symmetric key while leveraging AES to perform bulk data encryption. In ECC, the shared secret calculated after ECC mathematical calculations is used as a symmetric AES key. The performance of these algorithms is evaluated based on their ability to secure data, the time taken to encrypt data, and the throughput required by the algorithm [12, 13].

Figure 1 Classification of Encryption Techniques



Research Objectives

This research paper aims to explore the domain of cryptographic techniques, with a particular focus on the evaluation and comparison of two hybrid approaches: AES-RSA and ECC-AES. Our research objectives are as follows: (i) Analyze the computation speed.

(ii) Assess the key sizes and memory usage.

(iii) Conduct a comparative study on the performance of proposed techniques.

This section offers an overview of the paper's structural organization. The comprehensive literature review is discussed in Section 2, which is followed by an exploration of research, examining two hybrid encryption methods: AES-RSA and ECCAES. Section 4 describes the datasets, method applied, experimental results with performance metrics. In Section 5, a detailed discussion is done with the analysis of these results.

2. Literature Review

Nowadays, communication paradigms are advanced which brings new challenges related to security and privacy of personal data collected with the help of smart devices. The environment having constraints related to resources will face serious problems at the time of running complex encryption algorithms. A technique is proposed by [14], 2024 for the protection of wireless sensor network channels. With the use of Elliptic-Curve Diffie-Hellman technology, the secure key exchange was done. For the elimination of flaws and enhancing the security of Bluetooth for data transmission between devices of three algorithms i.e. AES, RSA, and Twofish. The message is initially encrypted with AES, followed by Twofish, with a 128-bit key. The first 128-bit key is ultimately encrypted using RSA with a 1024-bit key for safe transmission over the air [15].

Reference [16] propose an efficient hybrid cryptosystem for securing embedded systems. It extends the polynomial modular multiplication of ECC to calculate the polynomial arithmetic operations of the AES on an Altera FPGA to achieve high-level control, accelerated performance, and low power consumption. Reference [17] Propose hybrid algorithms that combine both symmetric and asymmetric cryptographic methods, including Twofish, AES, RSA, and ElGamal. The evaluation of these algorithms is carried out using a JAVA program implementation. The results indicate that the hybrid algorithm AES & RSA provides significantly high security. Reference [18] proposed an algorithm that encrypts the data using an AES key which is encrypted using the RSA algorithm and then nonce is added to the cipher text.

Storing passwords in plaintext exposes risks like attackers, eavesdroppers, and spyware. To prevent such exposure AES encryption can be used to encrypt files, RSA encryption to encrypt the AES password, and Hash-Based Message Authentication Code (HMAC) encryption to secure the data transmission [19]. These encryption techniques collectively ensure secure data transfer and help minimize the possibility of unauthorized access or interception during communication between clients and servers. Simplified Data Encryption Standard (SDES) encryption is performed on unstructured files to protect against data loss. Also, to control the data size, security, and time overheads during clustering the Huffman compression technique is used [20]. [21] Designs a malware detection and prevention approach using a deep LSTM approach and an improved ECC algorithm for secure data transmission among IoT devices.

Due to IoT's dynamic and ever-changing environment, protection against attacks and threats is important. To protect sensitive data against cyber-attacks such as brute force attacks, eavesdropping, and noise attacks integer wavelet transformation (IWT), and RSA algorithm show prominent results encrypting data in a shorter period [22]. [23] uses homomorphic encryption, binding, and permutations to secure privacy-preserving protocols. Often, Cryptography performs well on resource-rich devices such as smartphones and PCs however it performs badly on resource-constrained

IoT devices such as RFID tags and sensors. Reference [24] proposed that an increase in the S-box size boosts the security level while lowering the resource requirements

3. Proposed approaches

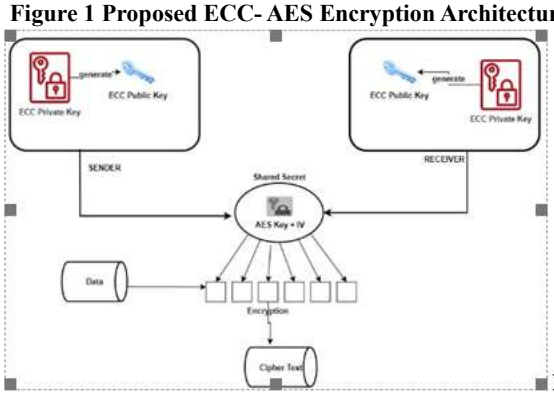
The integration of symmetric and asymmetric algorithms helps to overcome security limitations. Hybrid cryptosystems enhance the performance and add more security levels to the data compared to algorithms that are implemented individually [25]. AES's block cipher design and support for various key lengths such as 128 bits, 196 bits, or 256 bits each makes a round of 10,12, or 14 for encryption making it resistant to brute-force attacks, but its security hinges on proper implementation and key management. RSA's security relies on the difficulty of factoring large numbers, enabling secure key exchange and digital signatures, yet it requires periodic adjustment of key lengths as computational power grows [26]. RSA encryption takes 1 round with two different keys. DES, once prominent, fell out of favor due to its small key length, making it vulnerable to attacks; its successor 3-DES improved security but introduced performance issues. Two such notable methods are AES-RSA and ECC-AES, which combine asymmetric and symmetric encryption paradigms to fortify data protection. These hybrid solutions, while seeking to leverage the benefits of their component methodologies, need careful examination to verify their effectiveness. The performance assessment of these approaches is assessed based on data kinds, input size, key size, and encryption duration to guarantee data security, ensuring access is restricted to authorized users only [27, 28].

3.1. AES-RSA Hybrid Approach

For enhanced data security, a noticeable issue comes into the picture i.e. the hybrid AES-RSA technique. The advantages of RSA asymmetric encryption are integrated with the efficiency of AES symmetric encryption to form an effective defense to handle modern information transmission threats. Data security and secure transmission are managed with the help of effective key management. By combining the security of RSA with the speed of AES, this hybrid approach provides a strong defense against cyberattacks and unwanted access to data.

Modeling of AES- RSA algorithm 1. Generate RSA Key Pair:
i. Generate a new RSA private key and corresponding public key.
2. Encrypt Data
i. Generate a random initialization vector (IV). ii. Create a new AES cipher with the 256-bit symmetric key and IV.
iii. Create an AES encryptor with the cipher.
iv. Encrypt the data using the AES encryptor with the symmetric key.

Figure 1 Proposed ECC- AES Encryption Architecture



Du

4. 3.1.1. Key Generation and Exchange

The AES-RSA approach employs a dynamic key generation and exchange process that blends the strengths of both AES and RSA encryption techniques. Initially, as shown in Figure 2. RSA creates a pair of public keys 'P_{uk}' and private key P_{vk}', often ranging from 1024 to 4096 bits, establishing a secure channel for key exchange as represented in Eq. (1).

$$\text{Generate RSA Key pair} : (e, d, N) \qquad \text{Public Key} : (e, N)$$
$$\text{Private Key} : (d, N) \qquad (1)$$

Here, e' is the public key exponent, d' is the private key exponent, and 'N' is the modulus. The public key exponent 'e' is an integer prime that is used during the encryption process.

Simultaneously, AES generates a symmetric encryption key (K_{AES}), typically 128, 192, or 256 bits in size, to safeguard data [29, 30]. During data transmission, AES encrypts the actual message with the symmetric key in fixed-size blocks as shown in Eq. (2), which ensures efficiency and speed.

$$C = \text{AES}_{\text{Encrypt}}(\text{Message}, K_{\text{AES}}) \qquad (2)$$

Here, C' represents the ciphertext obtained by encrypting the plaintext message using the key K_{AES}.

Subsequently, K_{AES} itself is encoded using the receiver's RSA public key $'\mathrm{RSA-Pub_k}'$, transforming it into a secure format that only the receiver's private key $'\mathrm{RSA-Pvt_k}'$ can decode.

$$\mathrm{C_{AES}} = \mathrm{RSA_{Encrypt}}\,(\mathrm{K_{AES}}, \mathrm{e}, \mathrm{N}) \quad (3)$$

$$\mathrm{C_{\,Cipher}} = \mathrm{C}||\mathrm{C_{AES}} \quad (4)$$

Here, C_{AES} as in Eq. (3), represents the ciphertext obtained by encrypting $\mathrm{K_{AES}}$ using $\mathrm{RSA-Pub_k}$. This combined ciphertext $\mathrm{C_{\,Cipher}}$ in Eq. (4) is shared with the recipient. On receiving, $\mathrm{RSA-Pvt_k}$ is used to decrypt $\mathrm{K_{AES}}$. This decrypted AES key then allows the recipient to unlock the encrypted message, revealing the original plaintext as in Equation (5, 6).

$$\mathrm{K_{AES}} = \mathrm{RSA_{Decrypt}}\,(\mathrm{C_{AES}}, \mathrm{d}, \mathrm{N}) \quad (5)$$

Where, $\mathrm{K_{AES}}$ represents the decrypted symmetric key.

$$\mathrm{Message} = \mathrm{AES_{Decrypt}}(\mathrm{C}, \mathrm{K_{AES}}) \quad (6)$$

Where, Message represents original plaintext.

RSA's security ensures the secure transfer of the symmetric key, eliminating the need for a shared secret between communicating parties. This addresses concerns of eavesdropping and secure key exchange. Furthermore, utilizing RSA only for key exchange and relying on AES for data encryption minimizes the computational overhead associated with asymmetric encryption, leading to more efficient overall performance. While RSA provides a high level of security, its computational complexity can slow down operations. Thus, AES-RSA strikes a harmonious balance by employing RSA for secure key exchange and AES for rapid and secure data encryption.

5. 3.2. ECC-AES Hybrid Approach

ECC is the foundation of numerous cryptographic primitives such as key agreement protocols, digital signatures & zero-knowledge proofs, and is widely deployed for secure communication. Fast ECC relies on heavily optimized modular arithmetic operations such as point addition, and scalar multiplication which are often tailored to specific platform architectures [31]. ECC is appealing because of its unmatched performance and security properties. An m-bit secure elliptic curve key can be encoded in 2n + x bits, with x being a small number. This means that for the typical 128-bit security level, a key can be encoded in only 32 bytes. Hence, ECC has been harnessed as the main asymmetric cryptography for key exchange, and the strength of AES for symmetric encryption [32]. By intertwining these two influential cryptographic methods, ECC-AES offers a compelling solution that addresses the challenges of secure data transmission in resource-constrained environments [33].

Modeling of ECC- AES Algorithm:

1. Key Generation:
a. Generate ECC Key Pair:
i. Select an elliptic curve 'Ec'.
ii. Select a base point 'G' on the curve.
iii. Generate a random private key 'd'.
iv. Compute the public key 'Q' from 'd' and 'G'.
b. Generate AES Key:
i. Generate a random AES key '' of size 128, 192, or 256 bits.
2. Key Exchange:
a. ECC Key Exchange:
i. Sender and receiver exchange their ECC public keys '' and Q_B'.
ii. Shared Secret Calculation: Calculate shared secret 'S'.
3. Data Encryption:
a. AES Encryption:
i. Break the data into fixed-size blocks `B1, B2, Bn`.
ii. For each block `Bi`, Apply AES encryption using key '' to get ciphertext `C`.

This process forms the bedrock of the cryptographic framework, facilitating secure key exchange and subsequent data encryption. The ECC key pair consists of two components, the ECC private key 'd' and the public key 'Q'. Private key 'd' is a randomly generated secret value that should not be shared with anyone [34]. Public Key 'Q' is generated with the careful selection of an elliptic curve (E_c) and a base point (G) on it. The ECC public key is derived from the ECC private key and is computed using elliptic curve mathematics as given in Eq. (7). This property ensures the security of the ECC algorithm.

$$Q = d.G \quad (7)$$

Where '.' Denotes scalar multiplication.

5.1.1. Key Generation and Exchange

In ECC-based cryptographic systems, when two parties want to establish a shared key, they use their respective private keys to generate their public keys. These public keys are then exchanged between the parties, allowing them to compute a shared secret 'S' that can be used as a symmetric key for encoding or other cryptographic operations.

Suppose, Party A has a private key 'dA' and a public key $'Q_A = d_A.G'$ and Party B has a private key 'dB' and a public key $'Q_B = d_B.G'$, The shared secret (S) is computed by Party A and Party B as in Eq. (8),

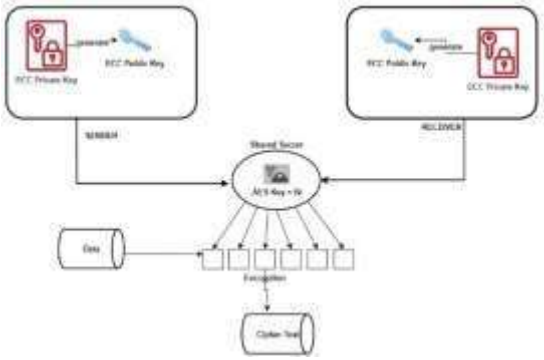
$$S = d_A.Q_B \quad S = d_B.Q_A \tag{8}$$

This shared secret is then used as the symmetric AES key for data encryption and decryption as given in Eq. (9).

$$K_{AES} = Hash(S) \quad C = AES_{Encrypt}(Data, K_{AES}, IV) \tag{9}$$

Here, C is the ciphertext resulting from encrypting the plaintext data using the derived AES k ey and IV.

Figure 1 Proposed ECC- AES Encryption Architecture



During encryption, the sender uses the shared secret 'K_t' to encrypt the plaintext data, dividing it into fixed-size blocks as shown in Figure 3. The ciphertext 'C', is then transmitted to the receiver. To decrypt the data, the receiver uses the same 'K_t' key obtained from the ECC key exchange. AES symmetric decryption is applied to each encrypted block of data, resulting in the original plaintext.

6. Experimental results and discussion

The experimental evaluation of cryptographic techniques for data security was conducted on a Windows 10 OS with an Intel

Core i7 processor, 16 GB of RAM, and an SSD. Python programming language version 3.10.9 and relevant libraries such as PyCryptodome [35] were employed for implementing the cryptographic algorithms and measuring encryption and decryption

times. The Pandas library was used for data handling and analysis. Our experiment was conducted on a smart meter's energy consumption dataset available online on Kaggle [36]. The implementation was based on the distinct key sizes, memory usage, encryption time, and decryption time.

6.1. Performance Analysis of Proposed Approaches

The comprehensive performance analysis of the evaluated cryptographic techniques was based on three crucial criteria: time efficiency, memory consumption, and key size. Time efficiency sheds light on the swiftness of encryption and decryption operations, indicating the responsiveness of a technique in dynamic scenarios. Memory consumption unveils the resource demands of each technique, influencing its feasibility in resource-constrained environments. Key size analysis highlights the intricacies of data protection, as larger key sizes are often associated with higher security levels.

4.1.1. Computational Overhead

Our analysis commences with a comprehensive comparison of various cryptographic techniques, including Fernet, ARC4, AES, Blowfish, AES-RSA, and ECC-AES-based encryption and decryption times, representing the operational efficiency of each method as shown in Table 1. The line chart in Figures 4 & 5 further accentuates these differences, providing a succinct overview of their relative efficiencies. In most cases, the execution time increased due to the increase in key size.

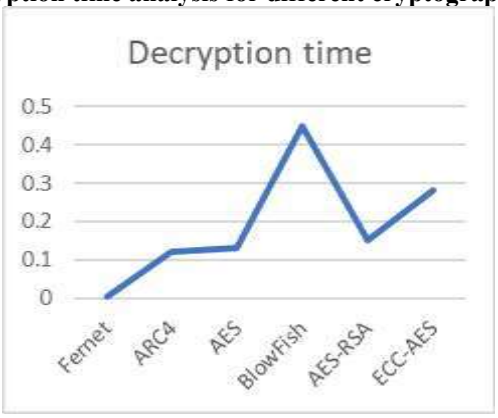
Table 1 Performance comparison of different cryptographic techniques

Cryptographic Technique	Encryption Time	Decryption time
Fernet	**36.69**	**0.004**
ARC4	**19.6**	**0.12**

AES	**8.27**	**0.13**
Blowfish	**3.91**	**0.45**
AES-RSA	**7.76**	**0.15**
ECC-AES	**1.67**	**0.28**

Figure 2 Encryption Time analysis for different cryptographic techniques

Figure 3 Decryption time analysis for different cryptographic techniques



Among the proposed tested algorithms ECC-AES exhibited the fastest encryption but it takes a longer time for decryption as compared to AES-RSA. The encryption process in AES-RSA involves modular exponentiation, which can be computationally intensive, especially when using large RSA keys. This tends to result in slower encryption times. However, RSA decryption, which involves modular exponentiation with the private key, is typically faster than encryption because the private key's components are smaller. In contrast, ECC-AES encryption is faster due to the use of symmetric AES encryption after ECC key establishment, and its decryption speed is also relatively quick since it employs the same symmetric key for both encryption and decryption.

4.1.2. Memory Usage Analysis

Memory consumption is a critical aspect to consider when evaluating cryptographic techniques. This memory requirement depends on the number of operations to be done by the algorithm, key size used, initialization vectors used, and the type of operations. The memory used impacts the cost of the system and can be calculated using equation 10. The memory required should be as small as possible [37]. Rigorous memory profiling was achieved via the integration of the memory profiler tool [38]. The %memit magic command facilitated memory usage analysis across encryption, decryption, and key generation processes. Subsequent records of peak memory utilization and incremental memory changes elucidated critical insights into resource consumption for both algorithms.

$$Memory\ Consumption = \frac{Final\ Memory\ Usage - Initial\ Memory\ Usage}{Number\ of\ Iterations} \quad \textbf{(10)}$$

Figure 4 Memory Analysis of AES-RSA

```
peak memory: 172.62 MiB, increment: 0.00 MiB
peak memory: 172.62 MiB, increment: 0.00 MiB
peak memory: 172.62 MiB, increment: 0.00 MiB
peak memory: 172.62 MiB, increment: 0.00 MiB
```

For the AES-RSA approach, as in Figure 6, our analysis reveals consistent memory utilization throughout the operations. The peak memory remains stable at 172.62 MiB, with no observable increment. This suggests that AES-RSA maintains a low memory overhead, making it well-suited for applications where memory resources are limited.
In contrast, Figure 7, exhibits slightly higher memory usage compared to AES-RSA. The peak memory utilization is recorded at 224.41 MiB, also showing a consistent pattern across multiple cycles. Although ECC-AES demonstrates slightly increased memory consumption, the overall impact remains within acceptable bounds for many practical scenarios.

Figure 7 Memory Analysis of ECC- AES

```
peak memory: 224.41 MiB, increment: 0.00 MiB
peak memory: 224.41 MiB, increment: 0.00 MiB
peak memory: 224.41 MiB, increment: 0.00 MiB
peak memory: 224.41 MiB, increment: 0.00 MiB
peak memory: 226.12 MiB, increment: 0.00 MiB
peak memory: 226.12 MiB, increment: 0.00 MiB
peak memory: 226.12 MiB, increment: 0.00 MiB
peak memory: 226.12 MiB, increment: 0.00 MiB
```

4.1.3. Key size analysis

The analysis indicates the length of the cryptographic keys used for encryption and decryption. The calculated key sizes were compared against theoretical sizes to validate the cryptographic strengths. In the AES-RSA approach, the key size is composed of two components: the AES key size and the RSA key size. The length of the AES key depends on the level of security needed and might be either 128 or 192 or 256 bits. In RSA algorithms, the key size is 1024 or more bits for the asymmetric encryption in the hybrid approach. The integration of AES secure asymmetric key exchange and efficient symmetric encryption.
The integration of Elliptic Curve Cryptography with AES enhances the powers of AES with the key exchange strength of Elliptic Curve Cryptography. In Elliptic Curve Cryptography, key size is between 256 and 384 bits, which is helpful in secure key exchange. The AES key size can also be 128, 192, or 256 bits like the AES-RSA process. The integration of Elliptic Curve Cryptography and AES results in a small or strong key size to ensure improved security and storage to increase transmission performance. Sometimes, indications of inconsistencies recommend safety thresholds. In this research work, it is found that the theoretical key sizes of 256 bits in AES, 2048 bits in RSA, and 256 bits for ECC-AES are critical, which influence the security and cryptographic resilience of these algorithms. The key size ensures that RSA is strong against factorization attacks, ECC-AES uses elliptical curve cryptography for the generation of secure private keys and AES-RSA uses symmetric keys for encryption. This evaluation is helpful in the demonstration of algorithms to meet security standards and rebound the contemporary cryptographic challenges.

4.2 Comparison Metrics

It is crucial to select a suitable cryptographic algorithm. It is shown in Tables 2 & Table 3, a thorough comparison of cryptographic techniques. Comparative metrics are taken to evaluate key factors to identify memory use, performance, security, computation efficiency, and key size. Each result lights upon a comprehensive assessment of the algorithms in the practical scenarios**.**

Table 2. Comparative Analysis of Different Cryptographic Techniques (Part 1)

Technique	Security	Key Size	Block Size	Encrypt Time	Decrypt Time
Fernet	High	256 bits	128 bits	36.69 ms	0.004 ms
ARC4	Medium	128 bits	N/A	7.69 ms	0.08 ms
AES	High	256 bits	128 bits	8.27 ms	0.13 ms
Blowfish	Medium	128 bits	64 bits	3.91 ms	0.45 ms
AESRSA	High	256 bits	128 bits	7.76 ms	0.15 ms

ECCAES	**High**	**256 bits**	**128 bits**	**1.67 ms**	**0.28 ms**

Table 3. Comparative Analysis of Different Cryptographic Techniques (Part 2)

Techni que	Mem ory	Key Manag ement	Mode of Opera tion	Resista nce to Attacks	Vulner ability History
Fernet	**Low**	**Simple**	**CBC**	**Secure**	**Limite d**
ARC4	**Low**	**Simple**	**Strea m Ciphe r**	**Vulner able**	**Vulner abilitie s Known**
AES	**Low**	**Compl ex**	**CBC**	**Secure**	**Resista nt**
Blowfis h	**Low**	**Simple**	**ECB**	**Vulner able**	**Vulner abilitie s Known**
AESRSA	**Low**	**Compl ex**	**CBC**	**Secure**	**Resista nt**
ECCAES	**Low**	**Compl ex**	**CBC**	**Secure**	**Resista nt**

The results of the use of encryption technology and data security have several implications. Two cryptographic technologies,
i.e., AES-RSA and ECC-AES are analyzed for the efficacy and applicability in various scenarios. ECC-AES provides quick encryption and decryption, which is helpful in various applications to be used as safeguard data in real-time. The integrated approach of symmetric and asymmetric encryption, AES-RSA, provides a flexible method for the achievement of a security and efficiency balance. These methods have the potential to greatly enhance data security in areas like communication networks and the IoT 5. Conclusion and future research directions
In this research, cyber threats are explored by IoT systems and presents two advanced encryption techniques i.e. AES-RSA and ECC-AES. In this study four performance metrics are evaluated on the Kaggle smart meter dataset: key size, memory consumption, encryption time, and decryption time. In the proposed method, assessment of these parameters and benchmark them beside leading algorithms in the field. It was found that ECC-AES is suitable for applications where time is one of the critical key factors whereas the key generation is also quicker than the RSA. AES-RSA is also a balanced approach that leverages the strength of AES and RSA both for secure data transmission. ECC is more efficient than RSA in terms of efficient key exchange due to requirements of smaller key sizes to achieve equivalent security levels leading to reduced computational demands. AES facilitates rapid and secure encryption of actual data payloads. The RSA required substantial key size and processing requirements which create difficulties in high performance or resource-limited settings. ECC-AES achieved the fastest encryption time at 1.67 ms among all the methods assessed which is five times quicker than AES-RSA. The decryption time of ECC-AES is 0.28 ms, which is approximately 70 times faster than other algorithms and twice as fast as AES-RSA. In comparison with AES-RSA, ECC-AES does require more memory due to its hybrid cryptography design. The research has focused on brute attacks. This algorithm can be further extended to test and predict behavior for other cyberattacks. The hybrid approach in a cryptosystem can be made more powerful by increasing the number of iterations or rounds in the encryption algorithm.

Acknowledgement
The authors are grateful to Noida Institute of Engineering & Technology, Greater Noida, India, for their continuous cooperation & support.
Funding Support
This work doesn't support any funding.
Ethical Statement
This study does not contain any studies with human or animal subjects performed by any of the authors.
Conflicts of Interest
The authors declare that they have no conflicts of interest to this work.
Data Availability Statement
The data that support the findings of this study are openly available in Kaggle at https://www.kaggle.com/datasets**.**

References

1. S. Dhar, A. Khare, A. D. Dwivedi, and R. Singh, “Securing IoT devices: A novel approach using blockchain and quantum cryptography,” Internet of Things (Netherlands), vol. 25, no. September 2023, p. 101019, 2024, doi: 10.1016/j.iot.2023.101019.
2. Z. Abedjan, L. Golab, and F. Naumann, “Profiling relational data: a survey,” VLDB J., vol. 24, no. 4, pp. 557–581, 2015, doi: 10.1007/s00778-015-0389-y.
3. L. van Zoonen, “Privacy concerns in smart cities,” Gov. Inf. Q., vol. 33, no. 3, pp. 472–480, Jul. 2016, doi:
4. 10.1016/J.GIQ.2016.06.004.
5. Z. Dewamuni, B. Shanmugam, S. Azam, and S. Thennadil, “Bibliometric Analysis of IoT Lightweight Cryptography,” Inf., vol. 14, no. 12, 2023, doi: 10.3390/info14120635.
6. M. Victor, D. D. W. Praveenraj, R. Sasirekha, A. Alkhayyat, and A. Shakhzoda, “Cryptography: Advances in
7. Secure Communication and Data Protection,” E3S Web Conf., vol. 399, 2023, doi:
8. 10.1051/e3sconf/202339907010.
9. K. Sharma, M. Chawla, and N. Tiwari, “Intrusion Detection System Using Machine Learning Approach: A Review,” Lect. Notes Networks Syst., vol. 473, no. June 2020, pp. 727–734, 2023, doi: 10.1007/978-981-19-28215_61.
10. M. Marwaha, A. Singh, and T. Singh, “Comparative analysis of cryptographic algorithms,” Int. J. Adv. Eng. Technol. E-issn 0976-3945, vol. 4, no. September, pp. 2013–2016, 2018, [Online]. Available: https://www.researchgate.net/publication/327664102_Comparative_Analysis_Of_Cryptographic_Hash_Functions
11. M. Agrawal and P. Mishra, “A comparative survey on symmetric key encryption techniques,” Intern. J. Comput.
12. Sci. Eng., vol. 4, no. 5, pp. 877–882, 2012, [Online]. Available:
13. http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=82397469&site=ehost-live
14. M. N. Alenezi, H. Alabdulrazzaq, and N. Q. Mohammad, “Symmetric encryption algorithms: Review and evaluation study,” Int. J. Commun. Networks Inf. Secur., vol. 12, no. 2, pp. 256–272, 2020.
15. T. Liu, G. Ramachandran, and R. Jurdak, “Post-Quantum Cryptography for Internet of Things: A Survey on Performance and Optimization,” pp. 1–13, 2024, [Online]. Available: http://arxiv.org/abs/2401.17538
16. Z. Yu, “Historical research on classification of Classical Cryptography,” Theor. Nat. Sci., vol. 11, no. 1, pp. 165–171, 2023, doi: 10.54254/2753-8818/11/20230403.
17. M. Vanitha and R. Mangayarkarasi, “Comparative study of different cryptographic algorithms,” Int. J. Pharm. Technol., vol. 8, no. 4, pp. 26433–26438, 2016, doi: 10.4236/jis.2020.113009.
18. M. Mittal, “Performance Evaluation of Cryptographic Algorithms,” Int. J. Comput. Appl., vol. 41, no. 7, pp. 1–6, 2012, doi: 10.5120/5550-7618.
19. C. Silva, V. A. Cunha, J. P. Barraca, and R. L. Aguiar, “Analysis of the Cryptographic Algorithms in IoT Communications,” Inf. Syst. Front., no. February, 2023, doi: 10.1007/s10796-023-10383-9.
20. M. A. Albahar, O. Olawumi, K. Haataja, and P. Toivanen, “Novel Hybrid Encryption Algorithm Based on Aes, RSA, and Twofish for Bluetooth Encryption,” J. Inf. Secur., vol. 09, no. 02, pp. 168–176, 2018, doi:
21. 10.4236/jis.2018.92012.
A. Hafsa, A. Sghaier, M. Machhout, and J. Malek, “A New Security Approach to Support the operations of ECC and AES Algorithms on FPGA,” 19th Int. Conf. Sci. Tech. Autom. Control Comput. Eng. STA 2019, pp. 95–100, 2019, doi: 10.1109/STA.2019.8717302.
22. E. Jintcharadze and M. Iavich, “Hybrid Implementation of Twofish, AES, ElGamal, and RSA Cryptosystems,”
23. 2020 IEEE East-West Des. Test Symp. EWDTS 2020 - Proc., pp. 0–4, 2020, doi:
24. 10.1109/EWDTS50664.2020.9224901.
25. K. R. Saraf and P. Malathi, “Security enhancement of cyber-physical system using modified encryption AESGNRSA technique,” Indones. J. Electr. Eng. Comput. Sci., vol. 33, no. 2, pp. 1177–1185, 2024, doi: 10.11591/ijeecs.v33.i2.pp1177-1185.
26. E. Salim and I. Harba, “www.etasr.com Harba: Secure Data Encryption Through a Combination of AES,” Technol. Appl. Sci. Res., vol. 7, no. 4, pp. 1781–1785, 2017, [Online]. Available: www.etasr.com
27. M. T. Nafis and R. Biswas, “A secure technique for unstructured big data using clustering method,” Int. J. Inf. Technol., vol. 14, no. 3, pp. 1187–1198, 2022, doi: 10.1007/s41870-019-00278-x.
28. R. Aiyshwariya Devi and A. R. Arunachalam, “Enhancement of IoT device security using an Improved Elliptic Curve Cryptography algorithm and malware detection utilizing deep LSTM,” High-Confidence Comput., vol. 3, no. 2, p. 100117, Jun. 2023, doi: 10.1016/J.HCC.2023.100117.
29. S. Du and G. Ye, “IWT and RSA based asymmetric image encryption algorithm,” Alexandria Eng. J., vol. 66, pp. 979–991, 2023, doi: 10.1016/j.aej.2022.10.066.
30. Z. Chokparova, K. Becher, A. Klose, T. Strufe, and L. Urbas, “Cryptographic protocol for privacy-preserving integration of HAZOPs in modular process plants,” Comput. Chem. Eng., vol. 176, p. 108295, Aug. 2023, doi: 10.1016/J.COMPCHEMENG.2023.108295.
31. V. A. Thakor, M. A. Razzaque, A. D. Darji, and A. R. Patel, “A novel 5-bit S-box design for lightweight cryptography algorithms,” J. Inf. Secur. Appl., vol. 73, p. 103444, Mar. 2023, doi: 10.1016/J.JISA.2023.103444.

32. N. K, F. S, S. B, S. V. S, and P. B, “Secure File Storage on Cloud Using Cryptography,” Int. J. Res. Appl. Sci. Eng. Technol., vol. 10, no. 6, pp. 428–438, 2022, doi: 10.22214/ijraset.2022.43535.
33. M. A. Al-Shabi, “A Survey on Symmetric and Asymmetric Cryptography Algorithms in Information Security,” Int. J. Sci. Res. Publ., vol. 9, no. 3, p. p8779, 2019, doi: 10.29322/ijsrp.9.03.2019.p8779.
34. M. Alam Hossain, M. Biddut Hossain, M. Shafin Uddin, and S. Md Imtiaz, “Performance Analysis of Different Cryptography Algorithms,” Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 6, no. 3, p. 2277, 2016, [Online]. Available: www.ijarcsse.com
35. M. E. Haque, S. Zobaed, M. U. Islam, and F. M. Areef, “Performance Analysis of Cryptographic Algorithms for Selecting Better Utilization on Resource Constraint Devices,” 2018 21st Int. Conf. Comput. Inf. Technol. ICCIT 2018, no. June 2020, 2019, doi: 10.1109/ICCITECHN.2018.8631957.
36. B. M. Vivek Parashar, “Performance Comparison of Various Cryptographic Algorithms Along with Energy Consumption in Wireless Sensor Network,” Int. J. Sci. \& Technol. Res., vol. 9, no. 06, pp. 77–86, 2020.
37. O. G. Abood and S. K. Guirguis, “A Survey on Cryptography Algorithms,” Int. J. Sci. Res. Publ., vol. 8, no. 7, 2018, doi: 10.29322/ijsrp.8.7.2018.p7978.
38. R. De Smet, R. Blancquaert, T. Godden, K. Steenhaut, and A. Braeken, “Armed with Faster Crypto: Optimizing Elliptic Curve Cryptography for ARM Processors,” Sensors, vol. 24, no. 3, pp. 1–16, 2024, doi:
39. 10.3390/s24031030.
A. Tidrea, A. Korodi, and I. Silea, “Elliptic Curve Cryptography Considerations for Securing Automation and SCADA Systems,” Sensors, vol. 23, no. 5, 2023, doi: 10.3390/s23052686.
40. R. Bhagyalakshmi, D. Roopashree, and K. N. Shruthi, “Performance Analysis of Hybrid Cryptography System for High Security and Cloud-Based Storage,” WSEAS Trans. Power Syst., vol. 17, pp. 244–253, 2022, doi:
41. 10.37394/232016.2022.17.25.
A. Chopra, “Comparative Analysis of Key Exchange Algorithms in Cryptography and its Implementation,” IMS Manthan (The J. Innov., vol. 8, no. 2, 2015, doi: 10.18701/imsmanthan.v8i2.5126.
42. https://www.piwheels.org/project/pycryptodome/.
43. Find open datasets and machine learning projects | Kaggle. https://www.kaggle.com/datasets.
44. P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, “A Comprehensive Evaluation of Cryptographic
45. Algorithms: DES, 3DES, AES, RSA and Blowfish,” Procedia Comput. Sci., vol. 78, no. December 2015, pp. 617–624, 2016, doi: 10.1016/j.procs.2016.02.108.
46. https://pypi.org/project/memory-profiler/.