IJE TRANSACTIONS B: Applications Vol. 39, No. 05, (May 2026) 1238-1253

International Journal of Engineering

Journal Homepage: www.ije.ir

# Achieving Enhanced Efficiency and Security in IoT: A Hybrid Encryption Approach with ECC and Blowfish

M. Baghaei Jezehei[a], S. A. Olamaei*[a], A. Broumandnia[b]

[a] *Department of Electrical Engineering, ST.C., Islamic Azad University, Tehran, Iran*
[b] *Department of Computer Engineering, ST.C., Islamic Azad University, Tehran, Iran*

*PAPER INFO*

Paper history:
Received 10 February 2025
Received in revised form 06 June 2025
Accepted 15 July 2025

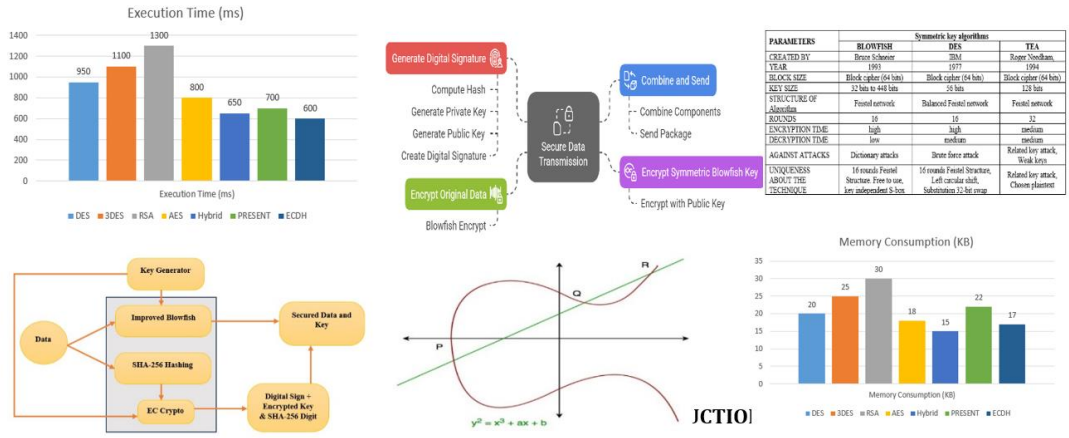
*Keywords:*
*Internet of Things*
*Hybrid Encryption*
*Elliptic Curve Cryptography*
*Blowfish Algorithm*
*Data Security*

*ABSTRACT*

The rapid proliferation of the Internet of Things (IoT) has heightened the urgency to address security vulnerabilities inherent in resource-constrained connected devices. Protecting sensitive user information underscores the necessity of securing personal data. Unauthorized access threats require advanced encryption solutions to enhance security and optimize system performance. This study proposes a novel hybrid encryption framework that uniquely combines asymmetric Elliptic Curve Cryptography (ECC) with a symmetric Blowfish algorithm. Unlike conventional methods such as RSA, AES, or standalone ECC, our approach integrates these techniques innovatively to achieve superior efficiency and security. Furthermore, a new adaptive encryption protocol is developed, optimized for data volume and device capabilities. Blowfish efficiently encrypts large data (over 64 KB) while maintaining speed. Additionally, ECC employs smaller key sizes, enhancing security without significant processing overhead. This research used data from various IoT devices like sensors, wearable gadgets, smart home systems, traffic, and industrial tools, tested both in labs and real-world settings. The data included various types of sensitive information, with each dataset averaging over 50 KB. Furthermore, all experiments were implemented on a system with a dual-core Intel processor at 2.5 GHz, ensuring a consistent performance evaluation. Compared to traditional cryptographic solutions such as RSA, AES, and standalone ECC implementations, our approach demonstrates a significant improvement in execution time—greater than 15%—and substantially enhances overall efficiency. This research shows that our hybrid approach improves IoT security while saving processing power and energy, helping future device performance.

**doi**: *10.5829/ije.2026.39.05b.16*

**Graphical Abstract**




*Corresponding Author Institutional Email: Olamaei@iau.ac.ir (S. A. Olamei)

Please cite this article as follows: Baghaei Jezehei M, Olamei SA, Broumandnia A, Achieving Enhanced Efficiency and Security in IoT: A Hybrid Encryption Approach with ECC and Blowfish. International Journal of Engineering, Transactions B: Applications. 2026;39(05):1238-53.

## 1. INTRODUCTION

The Internet of Things (IoT) connects many devices that collect and share data, emphasizing the need for strong security measures. Due to resource constraints, traditional cryptographic methods often fall short. This paper reviews lightweight protocols, including symmetric and asymmetric encryption, and block and stream ciphers, designed for IoT. While IoT enables real-time data sharing in sectors like smart cities, agriculture, and healthcare, it also introduces security challenges from cyberattacks.

The integration of networks and IoT enhances remote control but increases security risks, especially without proper safeguards. Effective authentication is vital; however, passwords are often inadequate for low-resource devices. A new lightweight authentication scheme is proposed to counter threats like replay and man-in-the-middle attacks. The paper also discusses challenges such as high costs, data management, and bandwidth overload affecting QoS. Its main contribution is a novel, resource-efficient authentication method tailored for low-resource environments, setting it apart from existing solutions (1, 2).

Smart healthcare systems that utilize ICT and IoT analyze real-time data to facilitate proactive care, particularly in the prediction of heart diseases. While these systems are highly effective, IoT devices are vulnerable to security threats often caused by poor configurations (3).

In wireless sensor network security, lightweight encryption algorithms were tested on the Atmel ATmega 128L microcontroller and Riverbed Modeler. The results showed that RECTANGLE was the fastest and most energy-efficient, while Camellia offered higher security. These findings highlight the importance of SPN-based algorithms for protecting sensor networks, especially in military and healthcare applications (4).

Vehicular Ad Hoc Networks (VANETs) facilitate communication between vehicles without fixed infrastructure. The GHRP protocol enhances security with source keys validated via one-way hashing, protecting against black hole attacks. Simulations show GHRP outperforms SAODV in packet delivery, with lower overhead and latency. A new protocol, SGHRP, further improves security and routing, demonstrating VANET insights can advance IoT security in smart homes and healthcare (5).

Networks on Chip (NoCs) boost inter-core communication. The ScRN algorithm manages traffic effectively, reducing packet latency by 38%, increasing throughput by 20%, and lowering energy consumption by 10%. As chip complexity rises, SoC architectures benefit from ScRN's improved latency and efficiency over older methods (6).

Securing data in IoT and cloud storage is challenging. Using NoC with Attribute-Based Encryption (ABE) enhances secure communication and access control through Blinding Value Encrypting Keys (BEKs) linked to user attributes. The system combines symmetric encryption, a polynomial tree, and authentication to ensure confidentiality and resist attacks like collusion and DoS. A lightweight authentication reduces resource depletion. Experiments show high efficiency and low overhead, with potential for improved privacy via multi-authority ABE (7).

IoT services are classified into low-level, data-oriented, and application-based categories, each with unique security risks requiring tailored solutions. All IoT devices must address the 'Big Three' security issues: confidentiality, integrity, and authentication, along with non-repudiation. Table ١ outlines these challenges.

This study highlights rising security and privacy concerns in IoT. It advocates for lightweight cryptography, analyzing algorithms like AES, DES, 3DES, and Blowfish. Tests on a Raspberry Pi 3B+ show Blowfish offers the best throughput despite DES and 3DES being faster. The research emphasizes the importance of secure IoT communication and calls for further work on optimizing power and memory for practical applications (8).

Present study emphasizes the importance of selecting appropriate encryption methods for resource-limited IoT devices. It explores neuromorphic computing and an adaptive routing protocol that reduces delays and power consumption. Additionally, a data-oriented RPL algorithm enhanced with Binary Gray Wolf Optimization and fuzzy logic improves network longevity and security. A comprehensive framework integrating cloud computing, blockchain, and advanced encryption techniques demonstrates superior data security and efficiency, addressing IoT challenges holistically (9-11).

Furthermore, this study compares the performance of RSA and ECC algorithms in IoT devices. Results show that ECC is more efficient in resource-constrained environments, with advantages in memory usage, energy consumption, and execution time, while RSA performs better in signature verification and encryption (12).

**TABLE 1.** Threats in IoT

| Threats | Minimum Level | Data Related | App Level |
|---|---|---|---|
| Replication of Nodes | ✓ | | |
| Tag Cloning | | ✓ | |
| Eavesdropping | | ✓ | |
| Packet Injection | ✓ | ✓ | ✓ |
| DoS Attack | ✓ | | ✓ |
| Camouflage | ✓ | | |

To enhance IoT security without compromising performance, a novel encryption approach combines two Blowfish algorithms with ECC, resulting in reduced memory usage, faster processing, and lower complexity. This scheme, validated through literature review and testing, integrates Improved Elliptic Curve Cryptography (IECC) with deep learning (LSTM) for malware detection. It enhances security by accurately identifying threats and ensuring secure data transmission, achieving 95% accuracy and effectively addressing IoT vulnerabilities.

## 2. RELATED WORKS

Unlike previous approaches that focus solely on traditional encryption methods, the proposed hybrid approach introduces a novel combination of ECC and Blowfish to significantly improve both security and efficiency in resource-constrained IoT environments.

Mahandru and Kaur (13) introduced a new method for securing data in cloud computing utilizes various cryptographic techniques to tackle existing security challenges. Despite the advantages of online data storage, ensuring data security is crucial. This approach features a hybrid encryption scheme that combines the Blowfish algorithm and the AES with the Elliptic Curve Diffie-Hellman (ECDH) for secure key exchange. It addresses security concerns in public, private, and hybrid cloud models, enhancing speed and efficiency for better data security and resource management. However, its application in IoT environments, where resource limitations and scalability are key factors, requires further investigation.

Bhagwatrao and Lakshmanan (14) reported that to address these security challenges, a new solution aims to enhance remote communication by integrating Deep Q-Network (DQN) and Graph Convolutional Network (GCN) blocks. DQN is utilized to determine the optimal actions for maximizing rewards, while GCN analyzes graph data to bridge various encryption and hashing methods. This system, named RHSSAO, demonstrates significant improvements in terms of adaptability, speed, and energy efficiency. However, additional comparisons with more contemporary IoT-specific solutions are necessary to validate its utility in dynamic and heterogeneous networks.

The objective of the study conducted by Prashant et al. (15) is to ensure the secure transmission of data through AES encryption. Safely transferring information across different working devices has become quite an intimidating challenge. In this context, algorithms such as DES, RSA (Rivest-Shamir-Adleman) and AES are crucial. This paper introduces a mechanism for secure data transfer, whereby security is enhanced by AES encryption. Overall, RHSSAO represents a significant advancement in remote communication, contributing to the secure transmission of data within the broader IoT security framework. More attention should be given to the challenges of data transfer in IoT environments, considering the limited resources and dynamic nature of these networks. Although this mechanism strengthens data security for IoT devices, it fails to address issues related to limited memory and processing power common in IoT environments.

Raghad et al. (16) proposed the security of data transmission in the digital age is crucial, yet it faces risks such as hacking. Effective encryption methods, like symmetric encryption, are essential for protection. The Blowfish algorithm is a widely used symmetric cipher but has limitations, including complex calculations and fixed S-Boxes, complicating its use with complex data. This paper reviews improvements to the Blowfish algorithm, focusing on execution time and security based on around 40 studies. Key enhancements include using a 128-bit block cipher, reducing S-Boxes (Substitution Boxes), and implementing parallel processing. While some improvements increased speed and security, they also raised memory usage. The research emphasizes the need to optimize symmetric encryption algorithms for resource-constrained IoT devices, aiming to reduce computational complexity while maintaining security.

Kapalova et al. (17) proposed a lightweight algorithm for better protection in computing environments. The algorithm employs the Feistel network and permutation architecture in a 16-byte block-encryption scheme to add complexity to the mechanism. While this algorithm shows promise in achieving faster encryption, its dependency on private keys introduces vulnerabilities that are exacerbated in IoT systems requiring dynamic key management.

Trik and Boukani (18) employed a range of encryption techniques to enhance the security measures for storage systems in IoT and fog computing. The research team developed an asymmetric system that uses encryption founded on the AES algorithm in conjunction with asymmetric key transfer methods to ensure security and confidentiality in data exchange. These findings underscore the importance of balancing security and efficiency, especially given the latency-sensitive nature of IoT environments.

Obaidat et al. (19) provided a 2-stage encryption solution to assure the protection of stored information, which entails dividing the primary data into two segments, with each segment being encrypted using a shared key generated according to the chaos theory model. While this method demonstrates reduced processing times, its scalability for real-time IoT applications remains unclear.

Agbelusi and Matthew (20) also provided an in-depth examination of various symmetric encryption algorithms, specifically focusing on their performance in terms of run time and memory utilization. For IoT devices, where resource availability is constrained,

algorithms like Blowfish and DES should be further evaluated under IoT-specific scenarios.

The PRESENT encryption algorithm is a lightweight block cipher designed for securing data in low-resource systems and Internet of Things (IoT) devices. It employs a substitution-permutation network (SPN) structure and utilizes short keys, making it highly suitable for devices with limited computational power. Known for being fast, efficient, and resistant to certain cryptographic attacks, PRESENT's design aims to provide robust security in resource-constrained environments. However, as examined by Lo et al. (21), power analysis attacks— a form of side-channel attack— reveal vulnerabilities in the algorithm. The study indicates that specific components, such as the "Add Round Key" and "S-Box" modules, are susceptible to correlation analysis, which can potentially reveal some key bits. This highlights that, despite its efficiency and low resource requirements, the security of PRESENT against side-channel attacks is limited, and additional precautions may be necessary to ensure complete security in practical applications.

Khezri et al. (22), Khalafi and Boob (23) analyzed the leading encryption solutions relevant to Internet of Things (IoT) systems and distributed computing frameworks. These studies emphasize the pressing requirement for algorithms and solutions that not only ensure security but also maintain service quality parameters, thereby reducing the adverse effects of encryption methods on services offered within IoT environments. Further empirical research is necessary to validate the efficacy of these solutions in diverse IoT applications.

In contrast to the aforementioned methods, our proposed hybrid encryption framework combines ECC and Blowfish in an integrated manner that specifically targets resource constraints and operational efficiency in IoT environments, representing a significant step forward in providing lightweight yet highly secure encryption.

Khezri et al. (24) proposed a novel KGRD system for secure key management in IoT using MQTT, leveraging a TPM for cryptographic operations to enhance key distribution and renewal. It combines asymmetric and symmetric cryptography to improve security and efficiency. Unlike blockchain-based solutions, our method uses hybrid encryption, balancing security and resource consumption without requiring high computational power.

Ding et al. (25) also proposed a novel approach for real-time personalized route selection in urban traffic management. The method addresses issues caused by a rapid increase in vehicle ownership and congestion resulting from events and accidents. Limited road network capacity worsens travel conditions, leading to wasted time and resources. Since expanding infrastructure is difficult, the proposed solution employs a Nash equilibrium established through mutual information swapping and self-adaptive learning. Simulation results demonstrate that this algorithm outperforms existing methods in dynamic, personalized route optimization.

## 3. BLOWFISH AND FEISTEL NETWORK

In this section, we examine the Blowfish algorithm, a widely used symmetric encryption method based on a Feistel network. Developed by Bruce Schneier in 1993, Blowfish features a variable key length ranging from 32 to 448 bits, providing flexibility to balance security and performance. It is known for its high speed and ease of implementation, supporting a block size of 64 bits and making it suitable for a variety of applications. However, like many symmetric algorithms, Blowfish can be vulnerable to certain attacks, such as dictionary attacks, particularly if weak keys are used or key management practices are inadequate.

To facilitate a comprehensive comparison, Table 2 evaluates three prominent symmetric key algorithms: Blowfish, DES, and TEA. This analysis considers criteria such as year of development, block size, key length, internal structure, encryption and decryption times, and potential attack vectors. This comparison highlights the strengths and weaknesses of each algorithm, aiding in the selection of appropriate cryptographic solutions for IoT applications and beyond.

### 3. 1. Key Components

1. Block Size: 64 bits.
2. Key Size: Variable (32 to 448 bits).
3. Rounds: 16 rounds of processing.

**3. 2. Feistel Network Structure** The Blowfish cipher operates using a Feistel network design, which splits the input data into two halves and processes them through multiple iterations.

**3. 2. 1. Block Division** The plaintext P (64 bits) is divided into two halves:

$$P=L_0||R_0 \quad (1)$$

Where:

- $L_0$ (left half): 32 bits
- $R_0$ (right half): 32 bits

**3. 2. 2. Round Process** The encryption process consists of 16 rounds. For each round I (from 0 to 15), the transformation is defined as:

$$L_{i+1}=R_i \quad (2)$$

$$R_{i+1}=L_i \oplus F(R_i) \oplus K[i] \quad (3)$$

Where F is a function applied to the right half Ri, and K[i] is the sub-key for round i.

**TABLE 2.** Analysis of Symmetric Algorithms

| PARAMETERS | Symmertic key algorithms | | |
|---|---|---|---|
| | **BLOWFISH** | **DES** | **TEA** |
| CREATED BY | Bruce Schneier | IBM | Roger Needham |
| YEAR | 1993 | 1977 | 1994 |
| BLOCK SIZE | Block cipher (64 bits) | Block cipher (64 bits) | Block cipher (64 bits) |
| KEY SIZE | 32 bits to 448 bits | 56 bits | 128 bits |
| STRUCTURE OF ALGORITHM | Feistel network | Balanced Feistel network | Feistel network |
| ROUNDS | 16 | 16 | 32 |
| ENCRYPTION TIME | High | High | Medium |
| DECRYPTION TIME | Low | Medium | Medium |
| AGAINST ATTACKS | Dictionary attacks | Brute force attack | Related key attack, Weak keys |
| UNIQUENESS ABOUT THE TECHNIQUE | 16 rounds Feistel Structure. Free to use, key independent S-box | 16 rounds Feistel Structure, Left circular shift, Substitution 32-bit swap | Related key attack, Chosen plaintext |

**3. 2. 3. Function F** Function F is defined on a 32-bit input and consists of several steps:

**3. 2. 3. 1. Input Splitting** The input R is divided into four 8-bit segments:

$$R= [b_1, b_2, b_3, b_4] \quad (4)$$

**3. 2. 3. 2. S-Boxes Lookup** Each 8-bit part bi indexes one of four S-boxes

$T_1, T_2, T_3, T_4$: (5)

$$\begin{aligned} y_1&=T_1[b_1]\\ y_2&=T_2[b_2]\\ y_3&=T_3[b_3]\\ y_4&=T_4[b_4] \end{aligned} \quad (6)$$

**3. 2. 3. 3. Combining Outputs** As a precaution, it is necessary to analyze whether the modifications made to function F enhance speed without compromising security against analytical attacks, such as differential or linear cryptanalysis. This analysis was conducted during the redesign of the function to ensure it is less vulnerable to such attacks.

$$F(R)=(y_1+y_2) \bmod 256 \oplus y_3+y_4 \quad (7)$$

Where:

- The addition $y_1+y_2$ is taken modulo 256.
- The result of the modulo operation is XORed with $y_3$
- Finally, $y_4$ is added to the total.

**3. 3. Subkey Generation** To perform encryption, the Blowfish algorithm uses a key schedule to generate 18 subkeys and initializes four S-boxes.

**3. 3. 1. P-array Initialization** The P-array is initialized with specific constants:

$$\begin{aligned} &P[0] =0x243F6A88\\ &P[1] =0x85A308D3\\ &P[2] =0x13198A2E\\ &P[3] =0x03707344\\ &\vdots\\ &P[17] =0xA409B34C \end{aligned} \quad (8)$$

**3. 3. 2. S-box Initialization** Each of the four S-boxes is initialized with fixed values. Example Initialization:

$$\begin{aligned} &S_0[0] =0xD1310BA6\\ &S_0[1] =0x98DFB5AC\\ &S_0[2] =0x2FFD72DB\\ &S_0[3] =0xD01ADFB7\\ &\vdots\\ &S0[255] =0xCA62C1D6 \end{aligned} \quad (9)$$

**3. 3. 3. Key Scheduling Steps**

1. XOR Key into P-array: Each entry of the P-array is XORed with successive bytes of the key.

Suppose the key is K with T bytes (T=length of the key in bytes):

$$P[i]=P[i] \oplus Key\ [i \bmod T]\ (i \text{ from } 0 \text{ to } 17) \quad (10)$$

2. Encrypt Zero Input: A block of zero is used to fill the P-array:

temp=0x00000000||0x00000000

For each I (from 0 to 17), calculate:

[i]=Blowfish Encrypt (temp) (11)

3. Fill S-boxes: Use the outputs from step 2 to populate the S-boxes sequentially.
   Filling an S-box Entry Example: Using outputs sequentially:

S[j]=Blowfish Encrypt(temp) (j fills from the outputs)

**3. 4. Encryption Process** The encryption process can be defined with the following steps:

1. Start with halves $L_0$ and $R_0$ derived from plaintext P.
2. Repeat the function transformation for 16 rounds:

$$R_{i+1}=L_i \oplus F(R_i) \oplus K[i] \quad (12)$$

3. Upon completion of all rounds, the final output is constructed by swapping the halves:

$$\text{Ciphertext}=R_{16}||L_{16} \quad (13)$$

**3. 5. Decryption Process** Decryption mirrors the encryption process but applies the keys in reverse order:

1. Split the ciphertext C:

$$C=L_{16}||R_{16} \quad (14)$$

2. Perform rounds in reverse order ii from 15 down to 0:

$$R_i=L_{i+1} \quad (15)$$

$$L_i=R_i+1 \oplus F(L_{i+1}) \oplus K[15-i] \quad (16)$$

3. The output after processing through all rounds is:

$$\text{Plaintext}=R_0||L_0 \quad (17)$$

**3. 6. Security Considerations**

1. **Diffusion:** Each round significantly spreads the influence of the input bits over the output bits. A change in one input bit will result in a significant change in output bits.
2. **Confusion:** Due to the S-boxes and the function FF, the relationship between the key and ciphertext becomes highly complex.
3. **Resistance to Attacks:** Blowfish ensures strong resistance against various attack types, including differential and linear cryptanalysis.

The modifications to function F have been carefully analyzed to ensure that while speed is improved, security against differential and linear cryptanalysis is not compromised. The relationship between input changes and output results has been preserved to maintain the robustness of the encryption process.

**3. 7. Summary of Key Mathematical Operations in Blowfish** Here's a summary of the key mathematical operations used in the Blowfish algorithm:

- Round Transformation:
- The transformation steps in each round are defined as follows:

$$L_{i+1}=R_i \quad (18)$$

$$R_{i+1}=L_i \oplus F(R_i) \oplus K[i] \quad (19)$$

- Function F:

$$F(R)=(T_1[b_1]+T_2[b_2]) \bmod 256 \oplus T_3[b_3]+T_4[b_4] \quad (20)$$

In this equation:

- T1, T2, T3, and T4 are the four S-boxes that correspond to the 8-bit input segments R.
- The mod 256 operation is applied because the sum of T1[b1] and T2[b2] may exceed the range of 0 to 255, necessitating assurance that the result remains within the 8-bit range.
- Since y3 and y4 are directly sourced from the S-boxes and are naturally within the range of 0 to 255, they do not require the mod 256 operation. This design helps ensure that the values are used correctly while maintaining computational complexity.
- This use of mod 256 does not alter the standard functionality of Blowfish, as it continues to preserve the distribution pattern and information processing, ensuring the algorithm's security is not compromised.
- Key Scheduling:

$$P[i]=P[i] \oplus \text{Key}[i \bmod T] \quad (21)$$

Here:

- P[i] is used as an input for generating subkeys.
- Key is the main key that assists in generating and expanding the keys.

**Blowfish Security:** The security of the Blowfish algorithm arises from its complex key schedule and Feistel structure. This design makes it resistant to brute-force attacks and cryptanalysis. Key expansion ensures that changes in the key produce significant changes in the key array (K-array) and S-boxes (substitution-boxes), while the Feistel network and round function ensure that each bit of the plaintext influences many bits of the ciphertext.

## 4. NOVELTY AND COMPARISON WITH RELATED METHODS

**4. 1. Introduction to the Proposed Method** In this study, we introduce a novel hybrid encryption framework that combines Elliptic Curve Cryptography (ECC) with the Blowfish algorithm to address the dual challenges of security and efficiency in IoT environments. The primary aim of this approach is to leverage the strengths of both asymmetric and symmetric cryptography, ensuring a high level of security while maintaining low computational overhead suitable for resource-

constrained devices. ECC facilitates secure and efficient key exchange with minimal energy consumption, while Blowfish provides fast symmetric encryption for data transmission. By integrating these techniques, our framework offers a lightweight yet robust encryption solution specifically tailored for the dynamic and resource-limited context of IoT networks.

**4. 2. Key Innovations and Contributions** This study presents several significant innovations that enhance the efficiency and security of cryptographic solutions for IoT environments. The core advancement is an optimized Blowfish algorithm featuring a modified F function that enables parallel processing of addition operations, reducing processing time by up to sixteen times without compromising security. By integrating ECC with this improved Blowfish, the framework provides a high-security, resource-efficient encryption scheme suitable for constrained IoT devices. The design emphasizes minimizing energy consumption and memory usage, making it lightweight yet robust. Overall, the work advances the state of the art in hybrid cryptography for IoT, offering practical algorithmic improvements and a scalable solution to meet the evolving security demands of modern networks.

**4. 3. Comparison with Existing Methods** In this section, we compare our proposed hybrid encryption scheme with traditional cryptographic techniques such as RSA, AES, standard ECC, and standalone Blowfish, focusing on four critical aspects: security level, speed and computational efficiency, resource consumption (memory and energy), and suitability for IoT environments.

**4. 3. 1. Security Level** RSA and standard ECC provide robust security based on difficult mathematical problems. However, RSA generally requires larger key sizes, which can increase computational load. AES is known for high security in symmetric encryption but does not inherently facilitate secure key exchange. Our hybrid approach leverages ECC's strong security for lightweight key exchange and Blowfish's efficient symmetric encryption, offering a comprehensive security framework optimized for IoT devices.

**4. 3. 2. Speed and Computational Efficiency** Although RSA and AES are often faster than standard ECC in processing large data, RSA's encryption and decryption demands are still significant, especially for resource-limited devices. ECC achieves comparable security with smaller keys, but its computations remain relatively intensive. Standalone Blowfish, renowned for its speed, lacks secure key exchange capabilities. Our method enhances Blowfish with an optimized F function that enables parallel execution of addition operations, resulting in encryption and decryption speeds up to times faster than traditional implementations. This makes the scheme highly suitable for real-time IoT applications.

**4. 3. 3. Resource Consumption (Memory and Energy)** RSA's large key sizes and ECC's complex mathematical operations can impose heavy demands on device memory and energy, limiting practicality in IoT environments. AES, while more efficient, still consumes considerable resources. Blowfish, known for its lightweight design, excels in fast processing but does not address secure key management alone. By combining ECC for secure, low-energy key exchange with an optimized Blowfish for fast data encryption, our approach significantly reduces both memory usage and power consumption, making it ideal for constrained devices.

**4. 3. 4. Suitability for IoT Environments** Traditional encryption algorithms like RSA are often too resource-intensive for many IoT devices. AES is more practical but may still pose challenges in highly constrained scenarios. ECC offers a good compromise but can be demanding for extremely limited hardware. Blowfish's speed is advantageous, but without a secure key exchange method, it's insufficient alone. Our hybrid scheme provides a balanced solution: ECC enables secure and efficient key distribution with minimal energy, while the optimized Blowfish provides fast, secure data encryption. Overall, this makes our method highly suitable for IoT networks, where security, efficiency, and resource economy are paramount.

**4. ۴. Summary of the Comparative Analysis** Our proposed hybrid encryption scheme stands out among existing solutions due to its unique combination of ECC and an optimized Blowfish algorithm, specifically designed for the constraints of IoT devices. Unlike traditional methods that often sacrifice speed for security or vice versa, our approach achieves a balanced enhancement in both areas. It significantly reduces processing time and energy consumption while maintaining robust security. This makes it the most effective and practical solution for resource-limited IoT environments, setting a new benchmark in the field of lightweight cryptography.

## 5. METHODS

This involves establishing a comprehensive security structure that incorporates various safeguarding methods related to encryption. In this context, symmetric and asymmetric encryption techniques are integrated with an advanced signature mechanism to ensure the integrity of information using the SHA-256 (Secure Hash Algorithm 256-bit) hashing algorithm. All these factors lead to methods that seem to provide strong security for

computing environments. A critical parameter in cryptography is the speed of calculations, which directly affects network performance. One of the weaknesses of the asymmetric algorithm is that it takes more time to generate a key than the symmetric algorithm. Therefore, they are slower than symmetric key techniques like Blowfish. Thus, encrypting the original message for transmission, the symmetric key approach, due to its rapid processing capabilities, is considered the optimal choice. The main task of the Blowfish algorithm is to increase security in communication. Figure 1 illustrates the protection of network data using the SHA-256 hashing algorithm. Then, by using the elliptic curve encryption algorithm, we can improve security and create a digital signature as well as generate a private key in the network. Finally, to improve the two main parameters, i.e., speed and security, we use a symmetric algorithm and an advanced algorithm to encrypt the primary data.

**5. 1. Digital Signature** With the proposed solution, the hash function is combined with the digital signature to protect the integrity of the data. If an attacker attempts to access the data transmitted over the network using a fake digital signature, as shown in the literature (24, 25), they will not gain authorized access to the message according to the creation structure.

Thus, this digital signature is instrumental in fortifying the IoT environment. The various stages focused on maintaining data integrity through hash codes and digital signatures are illustrated below in the flowchart of the proposed solution, as presented in Figure 2.

A user should utilize Figure 2 to protect information from contamination and alteration.

**5. 2. Data Encryption Module** Though the provision for encrypted data is prioritized in production environments, the core aim is to realize efficient, secure encryption using a low execution time. The encryption of information on the network adopts an improved variant of the Blowfish algorithm to achieve this goal. The Blowfish algorithm supports key lengths starting from 32 bits and can accommodate key sizes up to 448 bits. However, more practically, it effectively operates with key lengths from 32 bits to 384 bits for most applications. This inconsistency arises because while the theoretical maximum is 448 bits, performance and efficiency considerations limit practical usage to 384 bits.

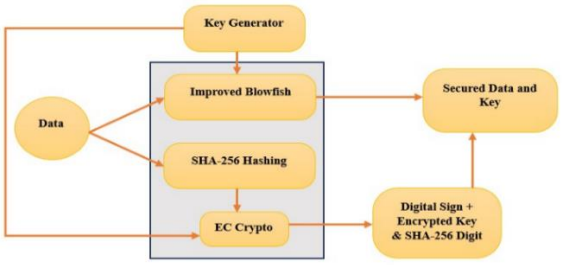



**Figure 1.** Structure of The Proposed Method

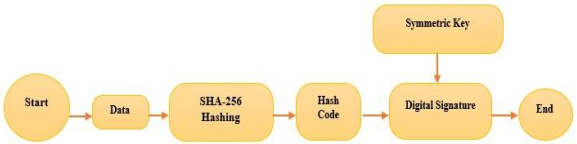



**Figure 2.** Digital Signature Flowchart

Furthermore, the generation of subkeys necessitates handling several keys within this range, which are created before any encryption or decryption of data. Blowfish distinguishes itself from other algorithms with higher speed, better security, and lower memory consumption. It uses four input boxes, each containing 256 keys with a maximum length of 32 bits. The first byte of the 32-bit input is fed into the first S-box, while the second byte accesses the second S-box, with subsequent inputs selecting S-boxes accordingly. The decryption process closely mirrors the encryption routine but applies subkeys in reverse. Each round revolves around the F module, which combines addition and shifting functions through modular arithmetic (26, 27). This paper aims to enhance execution speed by modifying the F module of the Blowfish algorithm. To optimize the algorithm's time complexity, the value of F is calculated using the standard Blowfish algorithm as described in Equations 22 and 23.

$$G(XL)=(X+Y) \bmod 2^{32} \quad (22)$$

$$F(XL)=G(XL) \oplus Z \quad (23)$$

Smaller values of F can be obtained with minimal integrity degradation, as shown in Equations 24 and 25:

$$Z=(U+V) \bmod 2^{32} \quad (24)$$

$$F'(XL)=G(XL) \oplus Z \quad (25)$$

Finally, the application of this modification allows the simultaneous execution of two addition operations represented as follows: The process of parallelization streamlines two addition operations:

$$G(X_L)=(X+Y) \bmod 2^{32} \text{ and } A=(U+V) \bmod 2^{32} \quad (26)$$

These operations run simultaneously, improving efficiency by consolidating tasks into one complete operation, resulting in a 16-fold reduction in execution time for encryption and decryption rounds. However, the algorithm's security relies on the strength of its keys. The pseudo-code for the revised Blowfish algorithm uses an adapted F function. The 64-bit input is divided into two 32-bit segments: left (L) and right (R). The left segment is processed with an XOR operation using an initial value P, generating an output for the F function. This output is then XORed again. The revised F function optimizes encryption, and decryption reverses these steps with keys applied in reverse order.

**5. 3. Secure Techniques for Symmetric Keys** In this section, the paper discusses the use of asymmetric encryption based on elliptic curves to overcome challenges related to transmitting private keys in the Blowfish algorithm. ECC (Elliptic Curve Cryptography) leverages the mathematical complexity of elliptic curves over finite groups to facilitate secure key distribution (28, 29). This approach provides high security with a 164-bit key and is more efficient than previous methods due to lower power consumption. It offers the highest level of confidentiality among asymmetric encryption techniques. Notably, by generating a private key through elliptic curve-based encryption and combining it with the Blowfish algorithm, both energy and memory usage are minimized without compromising security, which remains comparable to a 164-bit key, but with better overall performance.

Figure 3 illustrates an elliptic curve based on the standard form of the fundamental Equation 27:

$$y^2= x^3+ax+b \quad (27)$$

The plot clearly displays the x and y axes alongside the curve, enabling precise identification and analysis of points on it. The standard form of this elliptic curve is widely used in cryptography, including key exchange protocols, digital signatures, and encryption algorithms. A key point to note is that including the xy term in Equation 28:

$$y^2+xy-(x^3+ax+b) = 0 \quad (28)$$

indicates a more general and flexible form of elliptic curve. This form allows for broader mathematical analysis and transformations in cryptographic applications. Equation 28 can be converted into the standard form of Equation 27 through coordinate transformations. However, for simplicity and common cryptographic use, the standard form $y^2=x^3+ax+b$ is most frequently employed.

It offers the highest level of confidentiality among asymmetric encryption methods, while consuming little energy and requiring minimal memory resources. In summary, a significant feature of this encryption technique is its operation within finite fields. The logarithm of a discrete elliptic curve problem is known to be an NP-hard (Non-deterministic Polynomial-time hard) problem, and elliptic curves are widely used in cryptography based on this. The equation in Figure 3 describes an elliptic curve (30).

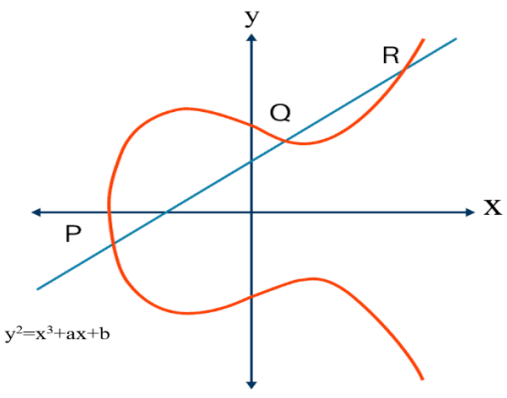



**Figure 3.** Elliptic Curve

This is where the essential features of elliptic curves come into play. Specifically, when we know two points on the curve, we can establish a rule for finding a third point by adding the two given points. The finite group formed by the points associated with an elliptic curve follows a specific addition rule. However, adding two points is not uniquely defined since they do not intersect at a single point. When defining an elliptic curve, we incorporate this distinguished point into the addition process. O is not an actual point on the curve; however, it possesses the characteristics of one, meaning that adding any point to O yields the original point. Within the framework of an elliptic curve, the total number of points is explicitly defined, taking into account both finite points and those regarded as infinite.

**5. 4. Fixed Curve Selection** As stated in Equation 27, elliptic curve cryptography (ECC) operates over a finite field GF(p), where p is a large prime number that defines the field. This allows cryptographic operations to be performed on a set of integers less than p.

In this context, m refers to the bit length of the key. Specifically, m indicates the number of bits used to represent the key in binary form. For example, a 256-bit key means m=256. This parameter is crucial because it directly influences the security level of the system and the computational effort required for cryptographic operations.

To ensure the mathematical validity of the elliptic curve, the condition of Equation 2٩ must hold:

$$4a^3 + 27b^2 \neq 0 \quad (29)$$

This guarantees that the curve is *nonsingular*, meaning it has no self-intersections or singular points, which could compromise security.

For cryptographic applications, standardized curves are selected following guidelines from organizations such as NIST or SECG. These curves have been thoroughly analyzed for vulnerabilities and are widely used due to their proven security.

In the process, a random integer k is chosen within the range: $1\leq k<p$ Selecting k uniformly at random within this interval ensures unpredictability, which is critical for security. The public key is then generated using:

$$Q=kG \quad (30)$$

Where G is a fixed point on the curve with a large prime order, ensuring the discrete logarithm problem remains hard and providing security against various attacks. The value m, representing the key length in bits, directly affects the effort needed to compute k from Q=kG. Based

on optimized algorithms, the number of operations required to compute kG is approximately

$$\text{Number of operations} \approx 3 * \left(\frac{m}{3}\right) = m \quad (31)$$

The phrase 'about 3(m/3) operations' indicates that the number of operations needed to compute kG is approximately equal to the key length m. This estimate is derived from optimized algorithms, which approximate the required computational effort as proportional to the size of the key.

This balance allows ECC to deliver high security with relatively small key sizes compared to other methods like RSA, while maintaining computational efficiency.

### 5. 5. ECC Implementation in Encryption Processes

Consider a scenario involving four users—Anna, Ben, Clara, and Daniel—who have mutually selected a non-confidential elliptic curve and a specific fixed point on that curve, referred to as G. To send a message to Ben, Anna must first acquire Ben's public key. Once she obtains it, she can use Ben's public key to encrypt the message and send the resulting ciphertext to him. In the ECC algorithm, Ben's public key corresponds to a curve point Q on the agreed-upon elliptic curve shared by Anna and Ben. Additionally, Ben has selected a secret integer H, which enables him to derive his public key. For securing the message, Anna begins by creating a random integer k. To ensure that k is generated securely, it is crucial that k be randomly selected from a valid range (between 1 and the order of the elliptic curve G) to prevent vulnerabilities associated with weak values of k. This validation ensures that k does not lead to predictable outcomes that could be exploited by an attacker. Following this, she identifies the poin T on the elliptic curve. Anna then computes the ciphertext, where N denotes the plaintext. Finally, she forwards the pair (T, C) to Ben as the encrypted payload.

$$Q=H\cdot G \quad (32)$$

$$T=kG \quad (33)$$

$$C=N+k\cdot Q \quad (34)$$

$$V=k.Q \quad (35)$$

The secret key H allows him to retrieve the original message corresponding to the ciphertext C. He computes V and finds it concerning C. Anna first performs the multiplication by H and then applies the result to the decryption process. Likewise, Ben can determine the same value for other reasons.

$$N=C-V=C-(kQ+N-kHG) \quad (36)$$

$$HAj=H\cdot Aj\cdot G=Aj\cdot H\cdot G=Aj\cdot Q \quad (37)$$

A strong assumption underpins the security of the method, the challenge of calculating k in relation to G and kG is significant. This is because the operations involved in this method are performed within the finite field. Thus, this method can leverage efficient algorithms for scaled multiplication to enhance the efficiency of elliptic curve-based encryption.

Therefore, this technique generally follows these steps:

- **Phase 1:** Before applying encryption, the original content undergoes hash code generation using the SHA-256 hashing algorithm.
- **Phase 2**: The resulting hash output is encoded and stored alongside the digital certificate and private encryption key.
- **Phase 3**: The private key of the Blowfish algorithm is encrypted using elliptic curve encryption.
- **Phase 4**: Before the data is transmitted, the original information is encrypted using a symmetric method, specifically the Blowfish algorithm, which employs the XOR function. Subsequently, the transmission data is combined with this encrypted data string before being sent out.
- **Phase 5**: The receiving side performs the reverse process. Here, decryption is executed, and the received data can be accessed using the private key.
- **Phase 6**: The secret key of the Blowfish algorithm is employed to recover the original data. After this, analysis and authentication take place using the hashing function (i.e., the digital signature).

### 5. 6. Pseudo Code for Hybrid Encryption Algorithm

**Input:**

- Original Data (M)
- Symmetric Blowfish Key (K)
- Elliptic Curve (EC)
- Generator Point (G)

**Output:**

- Encrypted Data (C)
- Digital Signature (DS)
- Encrypted Key (encrypted Key)

In Figure 4, the Secure Data Transmission Process illustrates the overall workflow involved.

**Step-by-Step Explanations of the Pseudo Code:**

**Inputs:**

- **Original Data (M):** The data intended for encryption.
- **Symmetric Blowfish Key (K):** The key used for encrypting and decrypting the data with the Blowfish algorithm.
- **Elliptic Curve (EC):** The mathematical structure used for generating public and private keys in asymmetric encryption.
- **Generator Point (G):** A fixed point on the elliptic curve used for key generation.

**Outputs:**

- **Encrypted Data (C):** The data obtained after encryption.

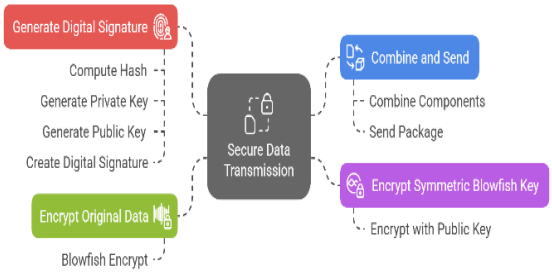



**Figure 4.** Secure Data Transmission Process

- **Digital Signature (DS):** The signature generated to ensure data integrity and authenticity.
- **Encrypted Key (encrypted Key):** The symmetric key that is securely transmitted.

**Detailed Steps:**

**Step 1: Generate Digital Signature**

- **1.1:** The hash value (H) of the original data (M) is computed using the SHA-256 algorithm, creating a unique representation of the original data.
- **1.2:** A private key for the elliptic curve is generated, which will be used for signing and encrypting.
- **1.3:** The public key is derived from the private key, used for encrypting the symmetric key.
- **1.4:** A digital signature (DS) is created for the hash value (H) using the private key, ensuring that the original data is verifiable by the holder of the private key.

**Step 2: Encrypt Symmetric Blowfish Key**

- **2.1:** The symmetric key (K) is encrypted using the public elliptic curve key, ensuring the symmetric key's security during transmission.

**Step 3: Encrypt Original Data**

- **3.1:** The original data (M) is encrypted using the Blowfish algorithm with the symmetric key (K). The result of this step is the encrypted data (C).

**Step 4: Combine and Send**

- **4.1:** The encrypted data (C), digital signature (DS), and encrypted key (encrypted Key) are combined into a single package for transmission.
- **4.2:** The Transmit Package is sent to the recipient, allowing them to retrieve the plaintext using the encrypted key and verify the signature.

## 6. ANALYSIS

The suggested approach was executed within the Eclipse development platform utilizing version 7 of the Java Development Kit (JDK). This JDK comprises two principal libraries: the Java Cryptography Architecture (JCA) and the Java Cryptography Extension (JCE). The functionalities of the JCA are seamlessly integrated into the fundamental Java APIs, offering vital cryptographic capabilities. The hardware used for the implementation was a dual-core Intel processor operating at 2.5 GHz, and all tests were conducted on the Windows 7 operating system. To evaluate the proposed method against various symmetric and asymmetric cryptographic techniques, a range of performance metrics, including processing time, data throughput, and memory utilization, were analyzed. Firstly, small-sized data (50 KB) solutions were evaluated to ascertain and analyze the operation using small-sized data. Thereafter, the solutions were analyzed by comparing their performance at sizes of 1024 KB (1 MB) and 2048 KB (2 MB) (15-17).

**6. 1. Analysis Criteria** There are specific metrics that can be used to determine the effectiveness of the solution and its viability. The following criteria have been established for use in this study:

- **Execution Time**: This is the time taken to accomplish all processes that require encryption functions. The execution time is calculated as shown in the following equation:

$$\text{Execution Time } (T_e) = T_f - T_i \quad (3^{\wedge})$$

Where:

- $T_e$ = Execution Time
- $T_i$ = Initial time (recorded at the start of the encryption process)
- $T_f$ = Final time (recorded after the encryption process is completed)

To evaluate the execution duration, the initial moment of the encryption procedure is recorded, and the final moment is then deducted from that.

- **Throughput**: Throughput is defined as the number of tasks processed and encrypted within a specific time frame. The throughput for the proposed solutions is calculated using the following equation:

$$\text{Throughput(TP)} = \frac{\text{Data Size(D)}}{\text{Execution Time(Te)}} \quad (3^{9})$$

- Where:
- TP = Throughput
- D = Data Size (in bytes)
- $T_e$ = Execution Time (in seconds)

In this context, throughput is assessed by measuring the amount of data processed over the time required for execution. The proposed method was evaluated with various data sizes, including small-sized data (50 KB), 1024 KB (1 MB), and 2048 KB (2 MB), to comprehensively analyze its performance.

## 7. RESULTS AND DISCUSSION

The first assessment is conducted over 50 KB of data, as shown in Figure 5. The execution times differ among the algorithms, with RSA taking 800 ms, AES 550 ms, DES 300 ms, 3DES 400 ms, and PRESENT at 800 ms (for

2SH). Overall, the execution times have been reduced, and the encryption tasks are accomplished more quickly than before. This improvement is attributed to the reduced execution time offered by Blowfish, which is recognized as one of the fastest encryption algorithms. This encoding method is renowned for its performance characteristics and demonstrates high efficiency in data processing. Additionally, enhancements in the speed and efficiency of other functionalities were made based on modifications in this area. The primary function of the EC algorithm is to encrypt the key and generate a hash, thereby reducing processing time. Compared to commonly used algorithms, it is significantly faster and consumes less memory.

In Figure 6, the comparison of solutions based on throughput is conducted with the 50 KB data size. The combination of Blowfish and ECC has been evaluated to confirm that it not only enhances speed and efficiency but also increases overall security by leveraging the strengths of both algorithms. However, it is crucial to ensure that this integration does not introduce new vulnerabilities that could compromise the encryption process. This has been tested through various simulations, demonstrating that the combined approach maintains robustness against potential threats. The proposed algorithm shows higher throughput than other solutions, as illustrated in this evaluation. Thus, the optimality rate reaches 100% when compared to RSA, 3DES, DES, AES, and ECDH. The time taken to implement this solution was exceptionally brief.

Given the limited performance observed in the evaluation, it is reasonable to expect this degree of success. In the next experiment, the amount of data being encrypted was identified as a crucial factor impacting the results during execution. Consequently, the size was increased to 1 megabyte to allow a fair comparison among all algorithms for this specified data size. Figure 7 demonstrates that our solution outperformed other algorithms, as indicated by the results presented by Khezri et al. (5).

Specifically, execution times improved by 240 ms, 240 ms, 130 ms, 80 ms, and 330 ms compared to other symmetric and asymmetric algorithms, highlighting the superior speed of the proposed solution. Likewise, the computation time has decreased by more than 731 ms compared to the RSA asymmetric algorithm, further establishing the faster implement ability of our solution. Furthermore, the throughput rate of the solutions was above average due to the reduced execution time, as shown in Figure 7.

At the same time, in the earlier scenario, RSA achieved performance that exceeded it by over 50%. The parallelization and optimization implemented in this segment significantly enhanced both execution time and throughput.

In this final assessment, we evaluate the algorithm using a sample data volume of 2 MB. Figures 9 and 10 clearly illustrate the advantages of this algorithm in comparison to the other two algorithms. It is essential to note that as the data size increases, there is a proportional rise in the time required for encryption. However, even with this increased demand, our algorithm consistently delivered faster operation times than the DES, 3DES, and RSA algorithms.

Specifically, the proposed solution was able to complete encryption tasks 470 ms and 320 ms faster than the two symmetric algorithms (3DES and DES),

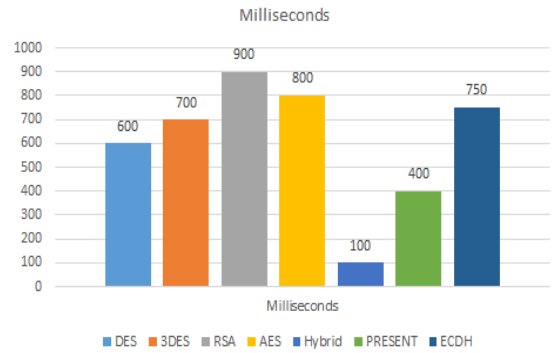



**Figure 5.** Execution time recorded in milliseconds (for a data size of 0.05 megabytes)

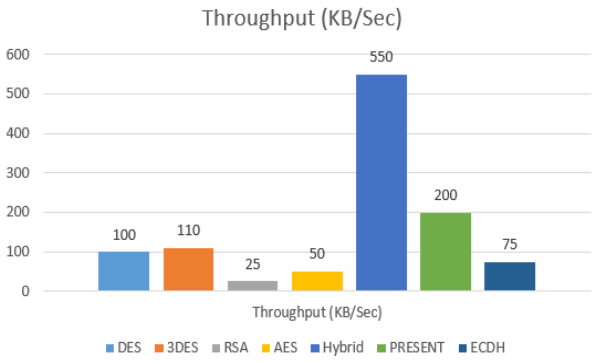



**Figure 6.** Efficiency measured (data size: 0.05 megabytes)

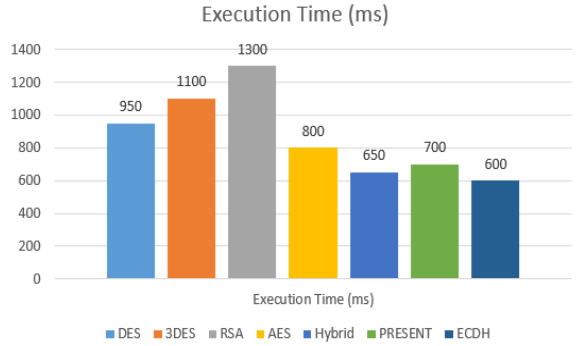



**Figure 7.** Execution duration measured in milliseconds (data size: 1 MB)

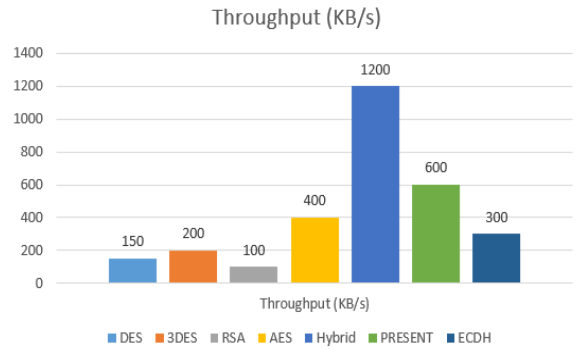



**Figure 8.** Throughput expressed in kilobytes per second (data size: 1 MB)

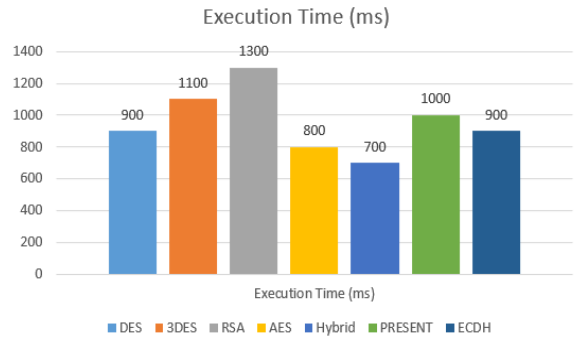



**Figure 9.** Operational time (data size: 2.0 MB)

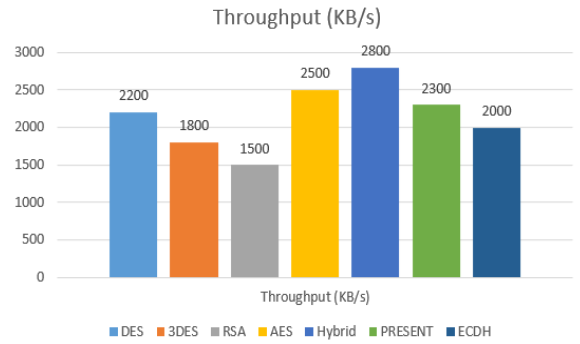



**Figure 10.** Efficiency indicated (data size: 2.0 MB)

respectively. Moreover, throughput demonstrated a significant increase, as depicted in Figures 8 and 9. During this experiment, the asymmetric RSA failed to show any improvement in performance compared to the symmetric AES.

It is important to highlight that the proposed solution encompasses additional features that extend beyond the capabilities of the conventional AES algorithm, including digital signatures and hashing. As a result, an increase in execution time is justified and is not expected to substantially affect the overall efficiency of the IoT infrastructure. Furthermore, the enhanced features of the proposed solution are instrumental in improving security and efficiency, indicating that the minor increase in execution time is acceptable within the broader context of IoT applications. Through these tests, we have demonstrated that the proposed algorithm not only meets but exceeds the performance metrics established by existing encryption methods.

**7. 1. Memory Usage Analysis** Another crucial element to consider is the memory capacity needed by encryption algorithms. The memory consumption induced by the developed technique is analyzed and compared with other commonly used encryption techniques. The outcomes of this evaluation are displayed in Figure 11. For this experiment, we considered a data size equal to 50 KB. Thus, as demonstrated in Figure 11 and Table 2, the lowest amount of memory consumption is associated with the suggested approach. The performance stems from the optimization of the cryptographic core; in this case, stemming from the Blowfish algorithm, achieving superior throughput while minimizing memory usage. Due to its reputation, the main features of the Blowfish algorithm include memory efficiency and speed of operations.

The innovative application of a 164-bit key makes this algorithm one of the most efficient asymmetric options regarding memory consumption. The memory requirements are so low that it can be effectively utilized within older systems and even integrated into the smallest embedded platforms.

In Table 3, the data indicates significant differences in memory requirements, with RSA demanding the most resources at 30.5 MB, while the hybrid approach is the most efficient, utilizing only 9.5 MB. Other algorithms such as 3DES and DES display moderate memory usage, with values of 20.5 MB and 18.0 MB, respectively.

In contrast, AES and ECDH each require 14.5 MB, while PRESENT's memory usage stands at 18.0 MB. This analysis highlights the varying efficiency of each method concerning memory utilization (18).

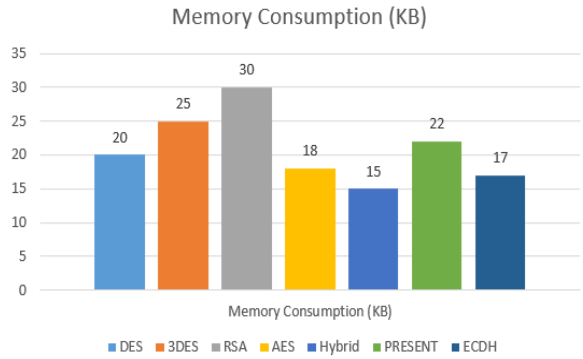



**Figure 11.** Memory usage by the solutions

**TABLE 3.** Memory Utilization Among the Methods Compared

| Procedures for Algorithms | Memory Usage |
|---|---|
| DES | 18.0 |
| 3DES | 20.5 |
| RSA | 30.5 |
| AES | 14.5 |
| Hybrid | 9.5 |
| PRESENT | 18 |
| ECDH | 14.5 |

## 8. CONCLUSION AND FUTURE WORK

This research proposes a robust encryption strategy designed to enhance security in IoT-enabled computing environments, focusing on efficiency, security measures, and data integrity. The use of EC (Elliptic Curve) asymmetric encryption is pivotal for secure key sharing, addressing the limitations of symmetric algorithms. Additionally, SHA-256 digital signatures have proven effective in ensuring data integrity and validating transmitted information.

Compared to standard cryptographic algorithms like AES, DES, 3DES, and RSA, the proposed method demonstrates superior execution time within the IoT framework. ECC enables smaller key sizes without sacrificing security, showing that the proposed solution not only enhances efficiency but also provides comparable or better security, especially in resource-constrained scenarios.

However, the effectiveness of this method has yet to be tested in real-world IoT environments. Future work should include deploying the solution in practical settings to assess its performance under conditions typical of IoT devices, such as limited battery life, processing power, and network bandwidth.

Despite these advantages, the proposed method has limitations. Scalability is a significant concern, particularly due to the need for larger key sizes to maintain security, which may strain low-power IoT devices. Furthermore, using Blowfish encryption necessitates a single key for both encoding and decoding, imposing challenges for scalability with numerous connected devices. Addressing these issues will require optimizing encryption algorithms to efficiently support larger networks.

Additionally, using Blowfish in conjunction with EC encryption may introduce potential drawbacks, such as increased complexity and the challenge of managing multiple encryption schemes, which could affect overall security if not properly implemented. Innovations and further optimizations, including leveraging machine learning and artificial intelligence for threat detection, could enhance the efficiency and applicability of the proposed solution.

Key areas for future research include:

- **Improving scalability for algorithms**: Optimizing encryption algorithms to accommodate a larger number of connected devices.
- **Investigating larger key sizes**: Analyzing the impact of larger key sizes on security and performance.
- **Secure data transmission methods**: Exploring effective methods for secure transmission in dynamic, resource-constrained environments.
- **Optimizing encryption algorithms**: Further research on symmetric and asymmetric algorithms to boost efficiency.

These future endeavors aim to strengthen security and efficiency in IoT infrastructures while paving the way for innovative solutions to meet the diverse challenges ahead. Overall, this research highlights the necessity for comprehensive solutions in IoT security.

## 9. REFERENCES


1. Rana M, Mamun Q, Islam R. Lightweight cryptography in IoT networks: A survey. Future Generation Computer Systems. 2022;129:77-89. 10.1016/j.future.2021.11.011
2. Kumar V, Malik N, Singla J, Jhanjhi N, Amsaad F, Razaque A. Light weight authentication scheme for smart home iot devices. Cryptography. 2022;6(3):37. 10.3390/cryptography6030037
3. Liao Y, Tang Z, Gao K, Trik M. Optimization of resources in intelligent electronic health systems based on Internet of Things to predict heart diseases via artificial neural network. Heliyon. 2024;10(11). j. heliyon. 2024.e32090
4. Panahi U, Bayılmış C. Enabling secure data transmission for wireless sensor networks based IoT applications. Ain Shams Engineering Journal. 2023;14(2):101866. 10.1016/j.asej.2022.101866
5. Khezri E, Zeinali E, Sargolzaey H. SGHRP: Secure Greedy Highway Routing Protocol with authentication and increased privacy in vehicular ad hoc networks. Plos one. 2023;18(4):e0282031. 10.1371/journal.pone.0282031
6. Trik M, Molk AMNG, Ghasemi F, Pouryeganeh P. A hybrid selection strategy based on traffic analysis for improving performance in networks on chip. Journal of Sensors. 2022;2022(1):3112170. 10.1155/2022/3112170
7. Nasiraee H, Ashouri-Talouki M. DoS-Resistant Attribute-Based Encryption in Mobile Cloud Computing with Revocation. International Journal of Engineering Transactions C: Aspects. 2019;32(9):1290-8. 10.5829/ije.2019.32.09c.09
8. Devi RA, Arunachalam A. Enhancement of IoT device security using an Improved Elliptic Curve Cryptography algorithm and malware detection utilizing deep LSTM. High-Confidence Computing. 2023;3(2):100117. 10.1016/j.hcc.2023.100117
9. Trik M, Pour Mozaffari S, Bidgoli AM. Providing an adaptive routing along with a hybrid selection strategy to increase efficiency in NoC-based neuromorphic systems. Computational Intelligence and Neuroscience. 2021;2021(1):8338903. 10.1155/2021/8338903
10. Wang Z, Jin Z, Yang Z, Zhao W, Trik M. Increasing efficiency for routing in internet of things using binary gray wolf optimization and fuzzy logic. Journal of King Saud University-Computer and Information Sciences. 2023;35(9):101732. 10.1016/j.jksuci.2023.101732

11. Puneeth R, Parthasarathy G. Security and data privacy of medical information in blockchain using lightweight cryptographic system. International Journal of Engineering Transactions B: Applications. 2023;36(5):925-33. 10.5829/ije.2023.36.05b.09
12. Fadia T, Toufik L. Elliptic curves cryptography for lightweight devices in IoT system. Brazilian Journal of Technology. 2024;7(4):e73725-e. 10.38152/bjtv7n4-003
13. Mahandru T KR. A study on the impact of cloud computing on the digital transformation of organizations. International Journal of Computer Applications. 2018;179(22):41-4. 10.5120/ijca2018916422
14. Bhagwatrao G, Lakshmanan R. A Novel Approach to use Deep Dyna Q Learning for Enhancing Selection and Performace of Encryption and Hashing Techniques in Remote Healthcare Environment. International Journal of Engineering Transactions A: Basics. 2024;38(1). 10.5829/ije.2025.38.01a.07
15. Prashant MSH, Amrinder Kaur, Pankaj Yadav. . Comparative analysis of AES and RSA with other encryption techniques for secure communication. International Journal of Scientific Research in Computer Science Engineering and Information Technology. 2024;10(2):565-74. 10.32628/CSEIT2410263
16. Qader RAHA, AL-Wattar AHS. A Review of the Blowfish Algorithm Modifications in Terms of Execution Time and Security. Technium Science Journal. 2022;4(9):89-101. 10.47577/technium. v4i9
17. Kapalova N, Algazy K, Haumen A. DEVELOPMENT OF A NEW LIGHTWEIGHT ENCRYPTION ALGORITHM. Eastern-European Journal of Enterprise Technologies. 2023;123(9). 10.15587/1729-4061.2023.280055
18. Trick M, Boukani B. Placement algorithms and logic on logic (LOL) 3D integration. Journal of mathematics and computer science. 2014;8(2):128-36. 10.22436/jmcs.08.02.04
19. Obaidat M, Brown J, Obeidat S, Rawashdeh M. A hybrid dynamic encryption scheme for multi-factor verification: a novel paradigm for remote authentication. Sensors. 2020;20(15):4212. 10.3390/s20154212
20. Olutola A, Olumuyiwa M. Comparative analysis of encryption algorithms. European Journal of Technology. 2023;7(1):1-9. 10.47672/ejt.1312
21. Lo O, Buchanan WJ, Carson D, editors. Correlation power analysis on the PRESENT block cipher on an embedded device. Proceedings of the 13th International Conference on Availability, Reliability and Security; 2018. 10.1145/3230833.3232801
22. Khezri E, Zeinali E, Sargolzaey H. A Novel Highway Routing Protocol in Vehicular Ad Hoc Networks Using VMaSC-LTE and DBA-MAC Protocols. Wireless communications and mobile computing. 2022;2022(1):1680507. 10.1155/2022/1680507
23. Khalafi M, Boob D, editors. Accelerated primal-dual methods for convex-strongly-concave saddle point problems. International conference on machine learning; 2023: PMLR. 10.48550/arXiv.2209.04604
24. Khezri E, Yahya RO, Hassanzadeh H, Mohaidat M, Ahmadi S, Trik M. DLJSF: data-locality aware job scheduling IoT tasks in fog-cloud computing environments. Results in Engineering. 2024;21:101780. 10.1016/j.rineng.2024.101780
25. Ding X, Yao R, Khezri E. An efficient algorithm for optimal route node sensing in smart tourism Urban traffic based on priority constraints. Wireless Networks. 2024;30(9):7189-206. 10.21203/rs.3.rs-3276051/v1
26. Zhu J, Hu C, Khezri E, Ghazali MMM. Edge intelligence-assisted animation design with large models: a survey. Journal of Cloud Computing. 2024;13(1):48. 10.1186/s13677-024-00601-3
27. Hosseini A, Rahaeifard M, Mojahedi M. Analytical and numerical investigations of the ultrasonic microprobe considering size effects. Mechanics of Advanced Materials and Structures. 2020;27(24):2043-51. 10.1080/15376494.2018.1539890
28. Karabulut E, Gholizadeh F, Akhavan-Tabatabaei R. The value of adaptive menu sizes in peer-to-peer platforms. Transportation Research Part C: Emerging Technologies. 2022;145:103948. 10.2139/ssrn.4073150
29. Golovko G, Kalynovych M. SPECIFICS OF IMPLEMENTATION OF THE ASYMMETRIC ENCRYPTION ALGORITHM ON ELLIPTIC CURVES. Системи управління, навігації та зв'язку Збірник наукових праць. 2023;1(71):84-90. 10.26906/SUNZ.2023.1.084
30. Goyal TK, Sahula V, Kumawat D. Energy efficient lightweight cryptography algorithms for IoT devices. IETE Journal of Research. 2022;68(3):1722-35. 10.1080/03772063.2019.1670103

## Persian Abstract

**چکیده**

رشد سریع اینترنت اشیاء (IoT) اهمیت رسیدگی به آسیب‌پذیری‌های امنیتی در دستگاه‌های متصل با منابع محدود را افزایش داده است. حفاظت از اطلاعات حساس کاربران بر لزوم امن‌سازی داده‌های شخصی تأکید می‌کند. تهدیدهای دسترسی غیرمجاز نیازمند راه‌حل‌های پیشرفته رمزنگاری است تا امنیت را افزایش داده و عملکرد سیستم را بهینه کند. این مطالعه یک چارچوب رمزنگاری هیبریدی نوآورانه را پیشنهاد می‌دهد که به‌طور منحصربه‌فرد، رمزنگاری بر مبنای منحنی‌های بیضوی (ECC) و الگوریتم متقارن Blowfishرا ترکیب می‌کند. برخلاف روش‌های مرسوم مانندRSA ، AESیا ECC مستقل، رویکرد ما این تکنیک‌ها را به روشی نوآورانه ترکیب کرده است تا کارایی و امنیت برتر را فراهم کند. همچنین، یک پروتکل رمزنگاری تطبیقی جدید توسعه یافته است که برای حجم داده و قابلیت‌های دستگاه‌ها بهینه شده است Blowfish .با کارایی بالا، داده‌های بزرگ (بیشتر از ٦٤ کیلوبایت) را در کوتاه‌ترین زمان ممکن رمزگذاری می‌کند. علاوه بر این، ECCبا استفاده از کلیدهای کوچک‌تر، امنیت را بدون بار اضافی در پردازش افزایش می‌دهد. در این پژوهش، داده‌هایی از دستگاه‌های مختلف IoT مانند حسگرها، تجهیزات پوشیدنی، سیستم‌های هوشمند منزل، سیستم‌های ترافیک و ابزارهای صنعتی جمع‌آوری شد و در آزمایشگاه‌ها و محیط‌های واقعی مورد آزمایش قرار گرفتند. این داده‌ها شامل انواع مختلفی از اطلاعات حساس بودند و هر مجموعه داده به طور میانگین بیش از ۵۰ کیلوبایت حجم داشت. همچنین، تمام آزمایش‌ها در سیستم‌هایی با پردازنده دو هسته‌ای اینتل با سرعت ۲.۵ گیگاهرتز انجام شد تا ارزیابی عملکرد یکنواختی صورت گیرد. در مقایسه با راه‌حل‌های رمزنگاری سنتی مانندRSA ، AESو پیاده‌سازی‌های مستقلECC ، رویکرد ما زمان اجرای بیش از ۱۵ درصد سریع‌تر را نشان می‌دهد و کارایی کلی سیستم را به طور چشم‌گیری افزایش می‌دهد. این تحقیق نشان می‌دهد که رویکرد هیبریدی ما، امنیت IoT را بهبود می‌بخشد و در عین حال مصرف توان و انرژی را کاهش می‌دهد، که این امر به بهبود عملکرد دستگاه‌های آینده کمک می‌کند.