Modern Encryption Algorithms Comparative Study: From Symmetric to Asymmetric Systems

Ziyi Weng *

Baradene College of the Sacred Heart, Auckland, New Zealand

* Corresponding Author Email: zw.23296@baradene.school.nz

Abstract. This paper compares different ways of keeping private information secure in modern society, which are named encryption methods. It covers two main types: symmetric and asymmetric. For symmetric encryption, DES and AES are considered. The paper compares how fast and how safe they are. For asymmetric encryption, RSA and ECC are evaluated. It compares their mathematical basis and therefore causes different performances on devices with limited resources and other application situations. The study explains DES's historical role and weaknesses, why AES is widely used for its security and speed, and the trade-offs of RSA and ECC in terms of key length, processing needs, and compatibility; it also analyses some potential applications, including Internet of Things, financial systems, and government data storage, showing how different algorithms fit different situations. Through the comparison between various cryptographies, it suggests that the choice of cryptography should depend on the specific use case. Future work should focus on responding to quantum computing threats.

Keywords: Cryptography; Symmetric Encryption; Asymmetric Encryption; RSA; ECC.

1. Introduction

In modern society, technology is becoming more vital in people's daily lives. Technologies are needed for sending messages, making payments online, and even running entire governments and industries. Therefore, protecting information exchanged from attackers has become one of the most important challenges. Cryptography is a key solution to this problem.

Two fundamental forms of cryptography are commonly recognized. Symmetric-key cryptography, in particular, refers to the use of the same secret key for both encryption and decryption processes. The examples of encryption standards include DES and AES. Symmetric-key cryptography is fast and efficient, which makes it good for dealing with a large volume of data. However, the key distribution problem needs to be considered.

In asymmetric or public-key cryptography, a pair of keys is employed: the public key is applied to encryption while the private key enables decryption. Typical algorithms include RSA and Elliptic Curve Cryptography (ECC). This approach usually requires more computing power.

Four cryptographies are discussed and compared. By showing their weaknesses and strengths, different cryptographies are suggested to be applied to different scenarios. For example, to satisfy the demand of the Internet of Things, lightweight algorithms such as ECC could be used. To ensure the security of financial transactions and government information, strong algorithms such as AES and ECC can be used.

2. Symmetrical-key Cryptography

Symmetric-key cryptography works by applying a single shared key for both the encryption and decryption of information. This characterization requires the shared secret key is securely exchanged between communication parties between the secret communications can begin. This challenge is known as distribution problem which is significantly challenging. Despite this limitation, symmetric algorithms are particularly effective for processing large datasets. Their high computational efficiency and speed make them well-suited for securing substantial amounts of information.

2.1. Data Encryption Standard

As an early symmetric-key method, the Data Encryption Standard (DES) held a central place in the evolution of cryptographic techniques. It was chosen as the U.S. federal standard in 1977 and became one of the most commonly used symmetric ciphers for many years. DES is a block cipher that works on chunks of data that are 64 bits long. Even though it uses a 64-bit key, 8 of those bits are Friused for error checking, so the real key size is 56 bits. The algorithm is based on the Feistel structure, allowing the same algorithm to be applied for both encryption and decryption.

2.1.1 Encryption and Decryption

As DES only can encrypt pain text with a length of 64-bit, any plaintext longer than 64 bits needs to be broken into 64-bit blocks. Firstly, the initial 64-bit plaintext block is subjected to a fixed permutation and rearranged into a new order. Next, the data undergoes 16 rounds of processing. The permuted block is split into two halves of 32 bits each: the left segment, denoted as L_0, and the right segment, denoted as R_0. For each of the 16 rounds (i=1 to 16), the steps are carried out according to the Feistel structure, as described in equations (1) and (2)

$$L_i = R_{i-1} \tag{1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{2}$$

The right segment R_{i-1} is taken to be processed with round subkey K_i as formula (3).

$$f(R_{i-1}, K_i) = P(S_{box}(E(R_{i-1}) \oplus K_i)) \tag{3}$$

The symbol E refers to expansion operation. At this stage, the input R_{i-1} is transformed into a 48-bit sequence by duplicating and reordering certain bits based on a fixed expansion table. The resulting 48-bit block is then combined with the round subkey K_i (also 48 bits) through an XOR operation. S refers to the substitution function. The 48-bit result is partitioned into eight segments of 6 bits each, with every segment processed by a different Substitution Box, which replaces the 6-bit entry with a corresponding 4-bit value based on a non-linear function. P refers to the permutation function. The resulting eight 4-bit outputs are combined into a 32-bit block, which is then permuted by a fixed P-box.

The resulting 32-bit block is then combined with the left segment to produce the new right segment. Simultaneously, $R_{i-1}$is taken as L_i. Since the Feistel structure is inherently reversible, decryption follows the same steps as encryption, except that the 16 subkeys are used in the opposite order $(K_{16}, K_{15}, K_{14}, \dots, K_1)$.

2.2. Advanced Encryption Standard

AES is currently the most commonly accepted method for symmetric encryption, which is based on the Rijndael algorithm, developed by Rijmen and Daemen in 1998. AES functions as a block cipher, which means it processes data in fixed-size blocks of 128 bits. The algorithm supports three different key lengths: 128, 192, and 256 bits. The number of rounds performed during encryption depends on the chosen key size, which is 10 rounds for a 128-bit key, 12 rounds for a 192-bit key, and 14 rounds for a 256-bit key, specifically.

2.2.1 Encryption and Decryption

In contrast to DES, AES is not based on the Feistel framework; rather, it employs a substitution–permutation network that manipulates data arranged in a 4×4 byte matrix. The original cipher key undergoes expansion through the AES key schedule, producing a series of round keys. During the initial stage, the plaintext block (16 bytes) is mapped into the state matrix and combined with the first-round key using the AddRoundKey operation. Each main round then applies four transformations in sequence: byte substitution (SubBytes), row shifting (ShiftRows), column mixing (MixColumns), and key addition (AddRoundKey). Each byte of the state matrix is replaced according to the S-box, a predefined table that introduces non-linear substitution.

The AES S-box is generated through a mathematical process: each byte is first transformed to its multiplicative inverse within the finite field GF (2^8), followed by an affine mapping, introducing confusion into the cipher. In the ShiftRows step, the last three rows of the state matrix are cyclically shifted by different offsets, which helps distribute changes across the block and contributes to diffusion; in the MixColumns stage, every column undergoes a linear transformation within the same finite field. This further spreads the influence of each input byte over multiple output bytes, strengthening diffusion; the AddRoundKey operation then merges the state matrix with the corresponding round key using a bitwise XOR, incorporating the key into the encryption. In the final round, the same sequence of operations is applied, except that MixColumns is skipped.

Decryption is performed by applying the inverse of each transformation in reverse order. This involves the inverse final round, followed by AddRoundKey, InverseShiftRows, InverseSubBytes, and a sequence of inverse main rounds (which include AddRoundKey and InverseMixColumns), ultimately concluding with the inverse of the initial AddRoundKey step.

2.3. Properties

The primary weakness of DES is its 56-bit key size. The small key size of 56-bit makes applying brute-force to crack DES practical. Attackers can decrypt the data by systematically testing all possibilities. In modern society, due to the improvement of computing power, even though the key space for a 56-bit key 2^{56} is theoretically large, brute-force is increasingly easier to crack them.

The weakness of DES due to its small size has been proven historically. For instance, in 1977, the first DES challenge was completed in just 96 days. In a subsequent third challenge, the same encryption was compromised in just 56 hours. Furthermore, the Electronic Frontier Foundation highlighted DES's vulnerabilities by breaking a DES key in approximately 22 hours using a purpose-built hardware device. As a result, DES is considered insecure and unsuitable for modern cryptographic applications.

Considering the length, AES is extremely secure compared to DES. With a 128-bit key, the number of possible keys is 2^{128}, making a brute-force attack computationally infeasible with current technology. There are no practical cryptographic attacks against AES that have been discovered so far [1].

DES was designed for efficient implementation in mid-1970s hardware. In modern software, it is significantly slower than contemporary algorithms like AES. AES is developed to achieve high performance and speed in both software applications and hardware devices. Its computational simplicity and parallelizable structure allow for very high throughput, making it significantly faster than DES. It performs well even in resource-constrained environments like smart cards and mobile devices. Therefore, DES is considered inefficient and not suitable for high-speed data processing by today's standards compared to AES.

3. Asymmetrical-key Cryptography

Asymmetrical-key cryptography, or public-key cryptography, relies on a pair of mathematically related keys. The public key may be shared openly, whereas the private key need to be kept secret. Information encrypted with the public key can only be decrypted using the matching private key. This system effectively resolves the key distribution challenge that is a major concern in symmetric cryptography.

3.1. RSA

The Rivest-Shamir-Adleman (RSA) algorithm was introduced in 1978 and has since become a commonly adopted encryption technology. Its strength comes from the computational challenge of decomposing a large number into its two prime factors. While the encryption can be done by anyone using the public key, recovering the original message requires the knowledge of the private key, which means only the holder of the private key can efficiently decrypt the message.

3.1.1 Key Generation

The RSA algorithm operates in three fundamental phases: key generation, encryption, and decryption. To create a public-private key pair, a user begins by choosing two large prime numbers, denoted as p and q.

Then compute modulus n by formula (4).

$$n = p * q \tag{4}$$

n will be used in both encryption and decryption, and its size determines the overall key length. Next, Euler's totient function is computed as formula (5).

$$\varphi(n) = (p-1)(q-1) \tag{5}$$

An integer e is then chosen to serve as the public exponent with the following restraints: e is greater than 1 and smaller than $\varphi(n)$, and e shares no common factor with $\varphi(n)$ computed previously. The number d, which is used to form the private number pair, is determined by formula (6).

$$d * e \equiv 1 \ (mod\ \varphi(n)) \tag{6}$$

This process completes the key generation stage. $\{e, n\}$ is used to represent the public key, and the private key is $\{d, n\}$.

3.1.2 Encryption and Decryption

To encrypt a plaintext message M, the public key pair $\{e, n\}$ is used by the formula (7).

$$C = M^e \ mod\ n \tag{7}$$

Here, *C* denotes the ciphertext. To retrieve the original message, the ciphertext *C* is decrypted using the private key $\{d, n\}$, as shown in formula (8).

$$M = C^d \ mod\ n \tag{8}$$

3.2. ECC

Elliptic Curve Cryptography (ECC), which emerged in the mid-1980s, is another form of public-key cryptography. Its security is based on the mathematical challenge of computing discrete logarithms over elliptic curves, a problem known as the Elliptic Curve Discrete Logarithm Problem (ECDLP).

3.2.1 Key Generations

ECC operations rely on the math of an elliptic curve set within a finite field. All parties agree on a set of public parameters: the elliptic curve equation (e.g., $y^2 = x^3 + ax + b$), a chosen finite field, and a publicly known base point G on the curve. Each user generates a private key by selecting a random integer d. The corresponding public key is calculated as shown in formula (9).

$$Q = d * G \tag{9}$$

This operation involves adding point G to itself d times, using special rules for point addition on an elliptic curve.

3.2.2 Encryption and Decryption

ECC is often used within a specific encryption scheme to encrypt data. Elliptic Curve Integrated Encryption Scheme (ECIES), for instance, is a popular method.

To encrypt a message M, which must first be mapped to a point on the curve, the formulas (10) and (11) will be used.

$$C_1 = k * G \tag{10}$$

$$C_2 = M + k * Q \tag{11}$$

k is the integer that is randomly selected. The ciphertext is the pair of points (C_1, C_2).

To recover the original message, the recipient employs their private key d, computing $d * C_1$ by the formula (12) and formula (13), resulting in $d * C_1 = k * Q$.

$$C_1 = k * G \tag{12}$$

$$d * C_1 = d * (k * G) = k * (d * G) = k * Q \tag{13}$$

The result $k * Q$ is subtracted from C_2. Thus, as formula (14), the original message point M is decrypted.

$$C_2 - k * Q = (M + k * Q) - k * Q = M \tag{14}$$

3.3. Properties

RSA and ECC are two main types of algorithms. They work based on different math ideas, which makes them very different in how well they perform, how efficient they are, and how they handle keys. RSA was first introduced in a famous 1978 paper. Its security comes from how hard it is to determine the original prime factors of a large composite number [2]. In contrast, ECC emerged later and employs a different principle, drawing its strength from the challenge of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) [3]. The biggest difference between them is how the size of the key relates to how strong the security is.

ECC can achieve equivalent security to RSA while requiring much shorter keys. For example, a 256-bit ECC key is typically regarded as offering a level of protection similar to that of an RSA key with about 3072 bits [4][5]. Performance analyses consistently demonstrate that the smaller key and certificate sizes of ECC lead to reduced storage and bandwidth requirements, as well as faster computational speeds, especially for private key operations such as digital signature generation; some studies even focus on how ECC outperforms RSA specifically in these resource-limited contexts [6].

While RSA encryption/verification (public key operations) can be computationally faster, its decryption/signing (private key operations) is significantly more intensive than the equivalent operations in ECC. Despite these advantages, RSA's long history has resulted in widespread legacy support and a perception of simpler implementation, a factor that has slowed its replacement in some sectors [7].

4. Discussion

The selection of a cryptographic algorithm in practical applications is not random; it is guided by the specific security requirements rather than being arbitrary. By comparing the properties of different algorithms, optimal use of cryptography can be suggested for certain scenarios. A hybrid approach is always applied to modern security systems.

4.1. The Internet of Things (IoT)

DES represents one of the earliest symmetrical key algorithms used for securing digital information. In 1977, it became the official encryption standard for the United States government and was widely used for many years. “As a block cipher, DES handles data in 64-bit segments. The algorithm is based on the Feistel structure, allowing the same algorithm to be applied for both encryption and decryption.

The Internet of Things (IoT) describes a system of physical devices equipped with sensors, software, and electronic components that are capable of communicating with one another and sharing data over the internet. These devices range from everyday household gadgets to larger-scale systems, including in vehicles, industrial machinery, and even city infrastructure. With sensors, IoT devices can collect real-time information, such as temperature, location, movement, and health data, and by enabling communication and data exchange, real-time information can be integrated to help with decision-making.

Connecting to the internet gives IoT the chance to be hacked, which raises the concern of security. IoT devices of health monitoring, for instance, demand high security due to the sensitivity of patient data and potential impact on user wellbeing. Based on the HIPAA secure bootstrapping, key management, secure firmware updates, and protection of sensitive health data during transmission and storage are required [8].

However, IoT medical devices operate with resource constraints, including limited computational power and low energy capacity. These devices must operate continuously without frequent battery changes while maintaining constant security [9].

Conventional cryptographic algorithms are frequently unsuitable for constrained systems because of their heavy resource demands. Issues such as high latency and excessive power consumption pose significant challenges, particularly in environments with limited computational capacity. In such cases, there is often an unavoidable trade-off between achieving strong security and maintaining acceptable performance. ECC is ideal for this scenario due to its smaller key sizes and lower computational overhead [10]. For symmetric encryption, AES-128 provides adequate security with reasonable resource usage [11]. Lightweight cryptographic algorithms specifically designed for IoT environments should be considered where available [12][13][14].

4.2. Financial Transaction Processing System

Banking systems generally require a significant number of computational resources. In addition, the completion of transaction processing needs high throughput and low latency. Financial transactions require the highest levels of security, including authentication, confidentiality, integrity, and non-repudiation. Legacy system compatibility requirements may limit algorithm choices. Performance cannot be compromised despite high security requirements, and regulatory compliance may require specific cryptographic approaches. RSA remains appropriate for digital signatures due to its broad compatibility and faster verification performance. For bulk encryption, AES-256 provides strong security for sensitive financial data. Organizations should begin planning for post-quantum cryptography migration to address future quantum threats [15].

4.3. Government-Classified Information Storage

Government system must often follow specific cryptographic standards. Long-term storage requirements must be considered, and systems must resist sophisticated state-level attacks. The highest security standards are required for protecting classified information. Cryptographic algorithms must be approved by relevant government agencies. Long-term confidentiality must be maintained, and multi-factor authentication with strong cryptographic proofs is essential. Allied nations' systems may impose additional constraints on the choice of cryptographies. AES-256 for encryption of stored data. ECC with approved curves for key establishment. Implementation should follow FIPS 140-3 validation requirements for cryptographic modules.

5. Conclusion

In conclusion, cryptography is important for modern digital security. Both symmetric and asymmetric algorithms play crucial roles, and each has its own weaknesses and strengths. Historically, the Data Encryption Standard (DES) was a major milestone, but its short key length made it insecure over time. It was eventually replaced by AES, which offers stronger security and improved efficiency. Both RSA and ECC are crucial in asymmetric cryptography. However, ECC stands out by delivering the same level of protection as RSA while requiring much shorter keys, resulting in better performance and efficiency.

The choice of which algorithm to use always depends on the situation. For IoT devices, as the computing power is limited, ECC and AES-128 are more suitable. In banking and financial transactions, speed and very high security are required, so AES-256, together with RSA digital signatures, is commonly used. For government data, security is the most important concern. Therefore,

AES-256 and ECC are always applied. These examples show how the different cryptographies are targeted at different scenarios.

In the future, cryptography will definitely continue to face new challenges. The growth of the Internet of Things, cloud computing, and artificial intelligence will create even greater demands for secure and efficient encryption methods. In the meantime, the development of quantum computers needs to be considered, as it may threaten the security of the cryptographies we are using currently, like RSA. Because of this, researchers are developing post-quantum cryptography. In other words, they are trying to create algorithms that can resist quantum attacks. Besides quantum attack resistance, another important research direction is lightweight cryptography, which can run on devices with very limited battery and computing power.

References

[1] Works CitedBaig, Mirza Hamza Munir, et al. “A Comparative Analysis of AES, RSA, and 3DES Encryption Standards Based on Speed and Performance.” Management Science Advances, vol. 1, no. 1, 19 Nov. 2024, pp. 20–30, https://doi.org/10.31181/msa1120244.

[2] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (Feb. 1978), 120–126. https://doi.org/10.1145/359340.359342

[3] Koblitz, N. (1987). Elliptic curve cryptosystems. Mathematics of Computation, 48(177), 203–209. https://doi.org/10.1090/S0025-5718-1987-0866109-5

[4] Gobi, M & .R, Sridevi & Rahini, R. (2020). A Comparative Study on the Performance and the Security of RSA and ECC Algorithm.

[5] Desai, A. (2020). Comparative Study of RSA and ECC for Secure E-commerce Transactions [Review of Comparative Study of RSA and ECC for Secure E-commerce Transactions]. International Journal of Research in Modern Engineering and Emerging Technology (IJRMEET) , 08(2320-6586).

[6] Bafandehkar, Mohsen & Yasin, Sharifah & Mahmod, Ramlan & Zurina, Mohd Hanapi. (2013). Comparison of ECC and RSA Algorithm in Resource Constrained Devices. 2013 International Conference on IT Convergence and Security, ICITCS 2013. 1-3. 10.1109/ICITCS.2013.6717816.

[7] Mahto, D., & Yadav, D. K. (2017). RSA and ECC: A comparative analysis. International journal of applied engineering research, 12(19), 9053-9061.

[8] Tsantikidou, K.; Sklavos, N. Hardware Limitations of Lightweight Cryptographic Designs for IoT in Healthcare. Cryptography 2022, 6, 45. https://doi.org/10.3390/ cryptography6030045

[9] Bouzidi, M., Gupta, N., Cheikh, F. A., Shalaginov, A., & Derawi, M. (2022). A Novel Architectural Framework on IoT Ecosystem, Security Aspects and Mechanisms: A Comprehensive Survey. IEEE Access, 1–1. https://doi.org/10.1109/access.2022.3207472

[10] Divya, & Setia, U. (2024). The Interplay between RSA Key Length, Security, and Computational Efficiency: A Comprehensive Review. International Journal of Progressive Research in Engineering Management and Science (IJPREMS), 04(2583-1062).

[11] Agarwal, A., & Saini, A. S. (2025). A Comparative Analysis of AES and DES: Security, Efficiency, and Applications [Review of A Comparative Analysis of AES and DES: Security, Efficiency, and Applications]. International Journal of Research Publication and Reviews, 6(2582-7421), 5954–5957.

[12] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. 2015. The SIMON and SPECK lightweight block ciphers. In Proceedings of the 52nd Annual Design Automation Conference (DAC '15). Association for Computing Machinery, New York, NY, USA, Article 175, 1–6. https://doi.org/10.1145/2744769.2747946

[13] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann and L. Uhsadel, "A Survey of Lightweight-Cryptography Implementations," in IEEE Design & Test of Computers, vol. 24, no. 6, pp. 522-533, Nov.-Dec. 2007, doi: 10.1109/MDT.2007.178.

[14] Biryukov, A., & Perrin, L. (2017). State of the Art in Lightweight Symmetric Cryptography. IACR Cryptol. ePrint Arch., 2017, 511.

[15] Himmat Rathore. (2021). Next-generation cryptographic techniques for robust network security. World Journal of Advanced Research and Reviews, 11(3), 496–508. https://doi.org/10.30574/wjarr.2021.11.3.0413