International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

A Comparative Performance Analysis of ElGamal and ECC-AES Hybrid Cryptosystems for Secure Communication

Gerardo S. Carlos Jr.[1], Renz Joshua D. Elpedang[2]

[1,2]Computer Science Program,
Cite Department
Northeastern Mindanao State University
Tandag City, Philippines
[1]gerardocarlos0824@gmail.com, [2]elpedang.rj@gmail.com

Abstract

In the age of mobile communication, the Internet of Things (IoT), and blockchain technologies, secure and efficient data encryption is important. This study presents an experimental comparison between two cryptographic methods: ElGamal (1024-bit) and a hybrid encryption model combining Elliptic Curve Cryptography (ECC, 256-bit) with Advanced Encryption Standard (AES). The aim is to evaluate and compare their performance regarding key generation time, encryption and decryption speed, and ciphertext size to determine their suitability for real-world applications.

The experimental results show that ECC + AES significantly outperforms ElGamal in all tested metrics. ElGamal recorded a key generation time of 0.258 seconds, encryption time of 0.0093 seconds, and decryption time of 0.0049 seconds, producing a ciphertext size of around 256 bytes. In contrast, ECC + AES achieved key generation times of 0.0008 seconds (sender) and 0.0001 seconds (receiver), with encryption and decryption times of 0.0008 seconds and 0.00005 seconds, respectively. It also produced a much smaller ciphertext of around 32 bytes while supporting actual text messages like "CARLOS," compared to ElGamal's number-only input.

These findings demonstrate that while both encryption methods provide strong security, ECC combined with AES is much more efficient and practical for environments requiring fast processing, smaller data sizes, and real-time communication, such as secure messaging, IoT networks, and low-power devices. This study provides concrete data supporting adopting ECC-based hybrid systems over traditional methods like ElGamal for modern cryptographic applications.

KEYWORDS: ElGamal, Elliptic Curve Cryptography (ECC), Advanced Encryption Standard (AES), hybrid cryptosystem, key generation time, encryption speed, ciphertext size, secure communication, DLP, ECDLP, real-time encryption, IoT security.

1. INTRODUCTION

In today's digital landscape, the demand for secure, fast, and efficient encryption systems is greater than ever, especially with the explosive growth of mobile communication, Internet of Things (IoT) devices, and blockchain applications [5], [7]. This study compares two cryptographic approaches, ElGamal and ECC combined with AES, to determine which method is more suitable for secure, real-time applications that need low computational resources and fast processing.

Previous researchers have extensively explored the development of hybrid cryptosystems that combine symmetric and asymmetric encryption techniques to strengthen security. For instance, Burgos et al. [2] proposed integrating RSA, ElGamal, and chaos-based algorithms with Schnorr authentication to enhance security in digital transactions. Arboleda et al. [1] examined a hybrid model combining RSA, DSA, ElGamal, and AES to improve the robustness of encryption systems. These studies demonstrate that merging different cryptographic methods can significantly increase protection against data breaches [1], [2], [6].

However, most of these works concentrate on algorithm design and theoretical security. There is a noticeable lack of experimental studies that compare the real-world performance of such algorithms, particularly in terms of speed, resource efficiency, and ciphertext size [4], [7]. Aiming presents a gap in the literature that this study seeks to address.

The significance of this research lies in its practical application. By evaluating how ElGamal and ECC-AES perform under real conditions, this study provides valuable insights for developers and security professionals. The findings will help guide the selection of encryption systems that are not only secure but also efficient for modern applications that demand speed, compact data size, and low power usage.

2. REVIEW OF RELATED LITERATURE

Numerous studies have focused on developing hybrid cryptographic systems in response to the growing demand for secure and efficient encryption systems in the digital age. These systems combine symmetric encryption, known for its speed, with asymmetric encryption, which is valued for secure key exchange. Arboleda, Dellosa, and colleagues (2019) proposed a hybrid model integrating RSA, DSA, ElGamal, and AES to enhance the robustness of encryption, while Burgos, Dimapilis, Arboleda, and others (2024) introduced a system that incorporates RSA, ElGamal, and chaos-based cryptography alongside Schnorr signatures to strengthen message authentication and entropy. These studies emphasize the theoretical benefits of combining multiple algorithms to increase security. Similarly, Garcia and Arboleda (2017) developed a chaos-based hybrid system using AES, RSA, and ElGamal, and Castro, Arboleda, and Corpuz (2019) combined AES with the Merkle-Hellman Knapsack algorithm to further enhance cryptographic strength. However, these works focus on algorithm design and lack empirical evaluation in real-time contexts. Chowdhary, Patel, Kathrotia, Attique, Perumal, and Ijaz (2020) examined hybrid models involving ECC, Hill Cipher, and AES, primarily for image encryption, analyzing encryption time and entropy metrics. Yet, their findings do not extend to practical text-based or real-time communications. Elliptic Curve Cryptography (ECC) has emerged as a modern solution due to its ability to offer strong security with smaller key sizes and faster performance, making it ideal for mobile and low-resource environments (Silva-García, Flores-Carapia, & Cardona-López 2024). Despite these advancements, El-Dalahmeh, El-Dalahmeh, Razzaque, and Li (2024) highlight a gap in the literature—a lack of direct

experimental comparisons between traditional algorithms like ElGamal and modern ECC-based systems. This study seeks to address that gap by evaluating the real-world performance of both systems using key generation time, encryption/decryption speed, and ciphertext size as comparative benchmarks.

3. METHOD

Figure 1 illustrates the general research flow, presenting the methodology as a model diagram. This figure outlines the sequential steps in conducting the study and constructing the cryptographic model.

Experimental Approach

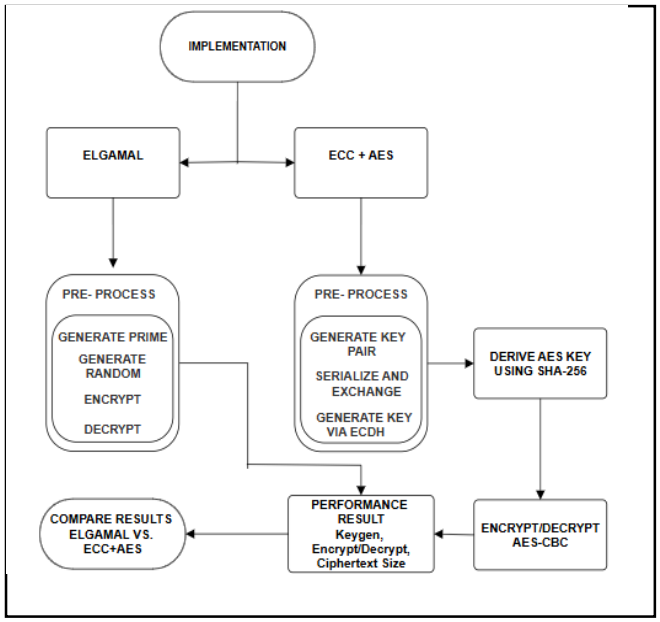

Figure 1. General Approach

This chapter outlines the methodology, tools, and procedures implemented in the development of the system. It provides comprehensive explanations, visual aids, and the experimental setup important for evaluating and comparing the performance of the encryption models.

ElGamal

ElGamal is a classic encryption method that tests how well it performs compared to modern systems like ECC combined with AES. It uses a large prime number and complex math to turn a number-based message into two encrypted values. While secure, it only supports numbers, runs slower, and produces larger encrypted data. The study measures how long it takes to generate keys, encrypt, and decrypt messages, along with how big the encrypted output is. ElGamal is the traditional benchmark for determining if newer encryption methods offer better speed and efficiency.

Key Generation	• Generate a large prime number p • Generator g • Select a private key x • Calculate public key: $y = g^x \bmod p$

Encryption Process	• Input plaintext message m, where m < p • Random number k • Compute: $c_1 = g^k \bmod p$ $c_2 = m\, y^k \bmod p$
Decryption Process	$m = c_2 \cdot (c_1^x)^{-1} \mod p$ $c_1^x \mod p$ is the same as $g^{kx} \mod p$ $(c_1^x)^{-1} \mod p$ is the modular inverse of $c_1^x \mod p$

Table 1. Elgamal Encryption and Decryption Techniques

ECC (Elliptic Curve Cryptography)

In this study, ECC combined with AES is used as a modern and practical way to secure messages. ECC (Elliptic Curve Cryptography) is responsible for safely sharing a secret key between two users using small, fast, and secure key pairs. Once the shared key is created, AES (Advanced Encryption Standard) encrypts real text messages quickly. The process starts with generating keys for each user, exchanging them securely using ECC's ECDH method, and then turning the shared secret into a 256-bit AES key using SHA-256. This method is fast, supports actual messages, and keeps files small—perfect for phones, apps, and smart devices.

After two users securely exchange a shared secret using ECC (Elliptic Curve Cryptography) and the ECDH key exchange method, that shared secret must be turned into a usable AES key. To do this, the system uses a cryptographic hash function called SHA-256.

SHA-256 transforms the shared secret into a fixed-length, 256-bit (32-byte) output. This output is then used as the AES key for encrypting and decrypting messages.

In simple terms:

Shared Secret → SHA-256 → AES Key (256-bit)

This step ensures the AES key is secure, consistent in size, and suitable for strong encryption.

Elliptic Curve Diffie–Hellman (ECDH) key exchange and AES encryption

Generate Key Pair	**P = d · G** d = private scalar G = base point P = public point on the elliptic curve

Serialize and Exchange	Send public keys between users
Generate Shared Key via ECDH	$K_{shared} = d_{A.}P_B = d_{B.}P_A$
Derive AES Key Using SHA-256	Hash the shared secret to create a 256-bit AES key $K_{shared} = SHA256(K_{shared})$
Encryption	$C_{i.} = E_{K.}(P_i \oplus C_{i-1.})$ $C_{0.}$ = IV (Initialization Vector)
Decryption	$P_{i.} = D_{K.}(C_i) \oplus C_{i-1.})$

Table 2. Elgamal Encryption and Decryption Techniques

Performance Metrics

Metric	**Description**	**Formula Example**
Key Generation Time	Time to compute public/private keys	The time between start_time and key generation end
Encryption/ Decryption Time	Time to secure/recover the message	encryption_time = end_time - start_time
Ciphertext Size	Size of encrypted data	len(ciphertext)
ElGamal Ciphertext		Ciphertext Size≈2×size(p)
ECC + AES Ciphertext		Size=IV (16 bytes)+Padded message size

Table 3. Performance Metrics

Key Generation Time

Key generation time is the initial delay before encryption can begin. ElGamal involves selecting a large prime p, a generator g, and computing $y = g^x \bmod p$. For ECC, it generates a private scalar and a public key point on the elliptic curve. Faster key generation is important in systems where sessions are frequently created, such as chat apps, VPNs, or IoT devices, allowing secure communications to start with minimal delay.

Encryption/Decryption Time

Encryption and decryption determine how quickly data can be secured and accessed. In real-time systems like secure messaging, delays in encryption or decryption can influence user experience. ElGamal uses modular exponentiation, which is computationally intensive, while ECC + AES benefits from fast symmetric AES encryption. Systems with lower encryption/decryption times are preferred in mobile apps, live data transfers, or streaming environments where performance is crucial.

Ciphertext Size

Ciphertext size influence how much data needs to be stored or transmitted after encryption. Larger ciphertext increases transmission time and storage costs. ElGamal typically generates large ciphertexts (two values per message), while ECC + AES produces compact ciphertext due to efficient symmetric encryption. The smaller ciphertext is crucial for performance in bandwidth-limited environments like IoT, mobile networks, and cloud storage. Optimizing ciphertext size improves efficiency without compromising security.

4. RESULTS AND DISCUSSIONS

In this section, the results of the study and the findings will be discussed. Tables and figures will be presented to support the discussions stated

Model Accuracy Test

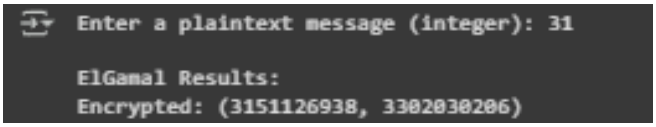

Image 1. Elgamal Encryption Result

The user entered the number 31 as the message they wanted to encrypt. Using the ElGamal encryption method, the system converted that number into a pair of encrypted values (3151126938, 3302030286). These two numbers make up the ciphertext or the hidden version of the message. The first number is based on a random key and the system's public setup, while the second combines that with the actual message. To turn the ciphertext back into 31, the system would need the matching private key to decrypt it.

Image 2. ECC + AES Encryption Result

The user typed "CARLOS" as the secret message they wanted to send to Zoren. Behind the scenes, the system used a secure method that combines two technologies: ECC to safely create a shared key between the sender and receiver and AES (256-bit) to encrypt the message using that key.

The encrypted message looks like a long, scrambled text:

XnFQIg2+SJvCfIL2gzwMBIDJ09yyapOTFcjPxNkQGM=
The locked version of "CARLOS" was made unreadable using strong encryption and then converted into a text-friendly format (Base64) to be easily sent. Only Zoren, with the correct key, can unlock and read the message.

A. Experimental Evaluation

ElGamal
Enter a plaintext message (integer): 31 ElGamal Results: Encrypted: (93121833818631615126589996591618847213409068733319616752573469024545979087552501058151675989780697101030379161915341051805790010075777006716786942938357503067251657526545854303632010873277502811533140041745213907748948150822553655626197367003149204877239961943356148929570812734701173220507772677608937218295, 73166657106199282855776925859338360330698579736617935232338511786878321514297922761714584761680828950198948528971411873807959812908760485796632923451313792411583174890230732577537838697267880025498309525576828103401069625795851046496579849081232978241726812575224358455491738514985526060776364478930246858 51) Decrypted: 31 Keygen Time: 0.258849 seconds Encryption Time: 0.009306 seconds Decryption Time: 0.004974 seconds Ciphertext Size: 2 bytes

Table 4. ElGamal Performance Results

ECC + AES (256-bit)
Enter a secret message to send to Zoren: CARLOS ECC + AES (256-bit) Secure Communication Results: Encrypted (base64): XnFQIg2+SJGvCfiL2gzrWBIDjD0yyapOTFcjPxNkqGM= Decrypted: CARLOS Keygen Time: 0.000797 sec (ian), 0.000116 sec (zoren) Shared Secret Time: 0.000487 sec Encryption Time: 0.000779 sec Decryption Time: 0.000048 sec

Ciphertext Size: 32 bytes

Table 5. ElGamal Performance Results

B. Analysis and Report

Compare the performance of ElGamal encryption and ECC-based key exchange + symmetric encryption in terms of key generation time, encryption/decryption time, and ciphertext size. (You can present your data in a tabular format to support your discussion)

	ElGamal (1024-bit)	**ECC + AES (256-bit)**
Key Generation	0.26 seconds (slow)	0.0008 sec (Ian) 0.0001 sec (Zoren) (very fast
Encryption Time	0.0093 seconds	0.0008 seconds (much faster)
Decryption Time	0.0050 seconds	0.00005 seconds (super fast)
Ciphertext Size	Around 256 bytes	Around 32 bytes
Decrypted Result	31 (integer)	CARLOS (string)

Table 6. Analysis Comparison

Key Generation: ECC creates keys much faster than ElGamal. ElGamal takes 0.26 seconds, while ECC takes just a fraction of a millisecond. This is important for quickly generating keys (e.g., chatting apps).

Encryption/Decryption Speed: ECC (with AES) is much faster than ElGamal when encrypting and decrypting messages. Enables ECC to be more suitable for real-time use, such as messaging or online transactions.

Ciphertext Size: The encrypted output (ciphertext) from ElGamal is large (256 bytes), while ECC with AES gives a small, encrypted message (32 bytes). Smaller ciphertext is better for saving storage and sending data over the internet.

Message Support: ElGamal only works on numbers, ex., 31, while ECC + AES can handle actual text messages like "CARLOS."

ECC + AES is faster, uses less space, and can handle real messages.
ElGamal is slower, creates a bigger ciphertext, and is limited to encrypting numbers.
ECC is more modern and practical for secure communication.

C. Mathematical Core Comparison

Feature	**ElGamal Encryption**	**ECC (Elliptic Curve Cryptography**

Underlying Math	Modular arithmetic using large prime numbers	Point multiplication on elliptic curves
Security Principle	Discrete Logarithm Problem (DLP)	Elliptic Curve Discrete Logarithm Problem (ECDLP)
Key Formula Example	$y = g^x \bmod p$	**P = d · G**
Direction of Operation	Multiply: easy Reverse (log): very hard	Multiply points: easy Reverse (discrete log): very hard
Input Format	Works only with numeric plaintext	Used for key exchange (text encrypted later via symmetric cipher like AES)
Output (Ciphertext) Size	Typically large, due to numeric pair output	Small, thanks to compact key exchange and AES's efficient block encryption
Processing Speed	Slower due to large key sizes and modular exponentiation	Faster with more minor keys and lower computational cost
Use Case	Primarily for encrypting small numbers in theoretical or educational settings	Common in real-world secure systems: messaging, IoT, cryptocurrencies, TLS
Encryption Role	Encrypts message directly	Establishes secure key; message is encrypted using AES (symmetric encryption)

Table 7. Mathematical Core Comparison

ElGamal encryption relies on modular arithmetic, which uses big prime numbers and a principle known as the discrete logarithm problem. Multiplying numbers together is simple but extremely difficult to back up and determine the original numbers. To illustrate, ElGamal employs a mathematical formula such as g^x mod p, where even if you know the output, it's nearly impossible to determine the value of x without the secret key. Pertains to ensuring the message is protected. But ElGamal can work only with numbers, typically resulting in large, encrypted output.

However, ECC (Elliptic Curve Cryptography) employs another type of math. Rather than manipulating powers and large numbers, ECC manipulates points on a specific curve. The principle is the same: it is simple to travel forward by multiplying a point repeatedly but extremely difficult to reverse and determine how many times it was multiplied. Commonly referred to as the elliptic curve discrete logarithm problem. ECC is strong as it offers the same security level as ElGamal but with more minor keys, quicker processing, and smaller encrypted data. ECC is usually applied to exchange keys securely; subsequently, a speedy algorithm such as AES is utilized to encrypt real messages such as text.

D. Comparison of Security Assumptions

DLP (Discrete Logarithm Problem) and **ECDLP (Elliptic Curve Discrete Logarithm Problem)**.

Aspect	**DLP (Discrete Logarithm Problem)**
Computational Difficulty	Hard to reverse g^x mod p finding x is very difficult for large p
Key Size for Equivalent Security	Requires larger keys for strong security (e.g., 2048-bit)
Resource Efficiency	Slower encryption/decryption uses more CPU power, memory, and bandwidth
Quantum Resistance	Not quantum-safe – Shor's algorithm can break it easily
Example Algorithm	ElGamal, DH, DSA

Table 8. DLP Assumptions

Aspect	**ECDLP (Elliptic Curve Discrete Logarithm Problem)**.
Computational Difficulty	Much harder than DLP for the same key size; reversing k × G = P is tough
Key Size for Equivalent Security	Provides the same security with much smaller keys (e.g., 256-bit ECC ≈ 3072-bit RSA/DLP)
Resource Efficiency	Faster, uses less CPU, memory, and energy — great for mobile and IoT devices
Quantum Resistance	Also, it is not quantum-safe, but it takes longer to break due to the smaller key size; it is still vulnerable.
Example Algorithm	ECC (ECDH, ECDSA), used in modern secure systems

Table 9. ECDLP Assumptions

DLP is like solving a puzzle using huge numbers. The bigger the number, the harder it gets.
ECDLP is solving a puzzle with points on a curved graph. It's way more complicated than DLP, even with smaller numbers.
So, ECC is more efficient just as secure, but with shorter keys and less resource usage.
However, both are breakable by quantum computers using Shor's algorithm in the future, so neither is future-proof, though ECC might resist for a bit longer due to efficiency.

5. CONCLUSION

This study compared the performance of two cryptographic systems—ElGamal and ECC combined with AES, to determine which approach is better suited for secure communication in today's digital landscape. We measured both models' key generation time, encryption and decryption speed, and ciphertext size through experimental evaluation.

The results showed that the ECC + AES hybrid cryptosystem significantly outperforms ElGamal in all tested areas. ECC offers the same level of security as traditional algorithms like ElGamal but with much smaller keys, which leads to faster processing, lower memory usage, and reduced power consumption.

When combined with AES, ECC becomes a powerful tool for encrypting real-world text messages efficiently and securely.

ElGamal, while secure, is less practical for modern applications due to its slower key generation, larger ciphertext output, and limitation to numeric input. In contrast, ECC + AES is lightweight, supports text-based communication, and delivers rapid encryption and decryption, making it an ideal choice for mobile devices, IoT systems, secure messaging, and blockchain environments.

This study concludes that ECC + AES is a more efficient and scalable solution for secure communication in resource-constrained and real-time environments. Its performance advantages and growing adoption in modern technologies affirm its role as a preferred cryptographic system for today's fast and secure digital interactions.

6. RECOMMENDATION

Based on the findings of this study, it is recommended that developers, system architects, and security professionals adopt the ECC + AES hybrid cryptosystem for applications that require secure, fast, and efficient communication. Given its shorter key lengths, faster key generation, and smaller ciphertext size, ECC + AES is suitable for mobile devices, IoT environments, low-power systems, and real-time communication platforms such as messaging apps and secure emails.

Organizations relying on traditional cryptographic methods like ElGamal or RSA should consider transitioning to ECC-based systems to improve performance and scalability without compromising security. Moreover, educational institutions and training programs should emphasize ECC and hybrid cryptography in their curriculum to prepare future professionals for real-world security demands.

Further studies are also recommended to explore integrating post-quantum cryptographic techniques with ECC or AES to enhance resistance against potential future threats posed by quantum computing. Finally, performance evaluations of ECC + AES on various hardware platforms (e.g., smartphones, embedded systems, and microcontrollers) help understand its adaptability across different devices and environments.

REFERENCES

1. Arboleda, E. R., & Dellosa, R. (2019). Hybrid Cryptosystem Using RSA, DSA, ElGamal, and AES. International Journal of Scientific & Technology Research, 8(11), 1873–1876.
2. Burgos, J. J. P., Dimapilis, P. B. S., Arboleda, E. R., Bojorcelo, J. M., & De Vila, J. B. A. (2024). CERS Algorithm: An Enhanced Security System Using Hybrid Cryptosystem. Isabela State University Journal of Engineering, Computing, and Technology, 1(1), 1–8.
3. Castro, M. C., Arboleda, E. R., & Corpuz, R. R. (2019). AES and Merkle-Hellman Knapsack Hybrid Cryptosystem. International Journal of Scientific & Technology Research, 8(12), 97–101.
4. Chowdhary, C. L., Patel, P. V., Kathrotia, K. J., Attique, M., Perumal, K., & Ijaz, M. F. (2020). Analytical study of hybrid techniques for image encryption and decryption. Sensors, 20(18), 5162. https://doi.org/10.3390/s20185162

5. El-Dalahmeh, A., El-Dalahmeh, M., Razzaque, M. A., & Li, J. (2024). Cryptographic methods for secured communication in SDN-based VANETs: A performance analysis. Security and Privacy, 7(6), e446. https://doi.org/10.1002/spy2.446
6. Garcia, D. W., & Arboleda, E. R. (2017). Enhanced hybrid algorithm of secure and fast chaos-based AES, RSA, and ElGamal cryptosystems. Indian Journal of Science and Technology, 10(27), 1–5. https://doi.org/10.17485/ijst/2017/v10i27/105001
7. Silva-García, V. M., Flores-Carapia, R., & Cardona-López, M. A. (2024). A hybrid cryptosystem incorporating a new algorithm for improved entropy. Entropy, 26(2), 154. https://doi.org/10.3390/e26020154
8. Dhasade, V., & Panchbhai, A. (2023). Efficient Hybrid Cryptography Algorithm. In 2023 International Conference on Communication, Computing and Internet of Things (IC3IoT), 1–6. IEEE. https://doi.org/10.1109/IC3IoT57137.2023.10337220
9. Wahyuni, S., Surantha, D., & Refianti, R. (2023). Hybrid Cryptosystem Using Polybius and ElGamal Algorithm Over the Gaussian Integers. In 2023 6th International Conference on Computer and Informatics Engineering (IC2IE), 1–6. IEEE. https://doi.org/10.1109/IC2IE58009.2023.10339989
10. Basak, S., Hossain, M. R., & Hasan, K. (2023). A Dynamic Hybrid Cryptosystem Using Chaos and Diffie–Hellman Protocol: An Image Encryption Application. Multimedia Tools and Applications, 82(21), 32475–32497. https://doi.org/10.1007/s11042-023-16191-7