frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 08 July 2025
DOI 10.3389/fcomp.2025.1630222

OPEN ACCESS

EDITED BY
Pravas Ranjan Bal,
Birla Institute of Technology, Mesra, India

REVIEWED BY
Sanjith Bharatharajan Nair,
University of Nizwa, Oman
Suvendra Kumar Jayasingh,
Biju Patnaik University of Technology, India

*CORRESPONDENCE
Yashmin Banu
✉ yashmin.banu@giet.edu
Biplab Kumar Rath
✉ biplab.rath@giet.edu
Debasis Gountia
✉ dgountia@cs.iitr.ac.in

RECEIVED 17 May 2025
ACCEPTED 06 June 2025
PUBLISHED 08 July 2025

CITATION
Banu Y, Rath BK and Gountia D (2025) Analyzing cryptographic algorithm efficiency with in graph-based encryption models. *Front. Comput. Sci.* 7:1630222. doi: 10.3389/fcomp.2025.1630222

Analyzing cryptographic algorithm efficiency with in graph-based encryption models

Yashmin Banu[1]*, Biplab Kumar Rath[1]* and Debasis Gountia[2]*

[1]Department of Mathematics, GIET University, Gunupur, Odisha, India, [2]School of Computer Sciences, OUTR, Bhubaneswar, Odisha, India

This research paper investigates the efficiency of cryptographic algorithms within graph-based encryption models such as star graph, focusing on their computational performance and security robustness. In this study, we analyze the performance of RSA and ElGamal cryptographic algorithms by evaluating time and space complexity across various file types, including text, image, audio, and data of different sizes. The encryption process is modeled using graph structures such as the Star graph, along with other well-known algorithms like A*, Dijkstra, Bellman-Ford, and Floyd-Warshall for comparative analysis and performance benchmarking. Consequently, this research conducts a comparative analysis of RSA and ElGamal cryptographic algorithms by applying them to mixed data, including binary, text, and image files. The CPU's internal clock was employed to record the execution time of encryption and decryption operations, facilitating the assessment of time complexity for both algorithms. The CPU's internal memory was employed to monitor and record memory usage during the encryption and decryption operations performed on mixed datasets. Accordingly, the evaluation of the encryption algorithms was conducted using criteria such as encryption time, decryption time, and throughput to determine their relative performance. In evaluating cryptographic approaches, factors such as response time, confidentiality, bandwidth, and integrity are considered. Experimental results indicate that RSA demonstrates superior time efficiency and resource utilization, whereas the ElGamal algorithm exhibits greater memory efficiency and resourcefulness. This study evaluates RSA and ElGamal encryption on text, image, audio, and data files of varying sizes using graph-based models. The Star graph algorithm is adopted for its simplicity and low computational cost, and its performance is compared against A*, Dijkstra, and Bellman-Ford algorithms. Results show that the Star model offers near-optimal paths with significantly reduced processing time, demonstrating high confidence in efficiency for lightweight encryption tasks. We have added computational performance, logical confidence, and optimality conditions of the proposed Star-based encryption model. The Star algorithm, integrated with RSA/ElGamal encryption, is benchmarked against classical pathfinding algorithms like A*, Dijkstra, and Bellman-Ford, commonly used for routing and shortest-path computations. Computational performance of (i) the worst-case time complexity of star algorithm (proposed) is $O(b^d)$ where b is the branching factor and d is the depth and space complexity is $O(E + V)$ where E is the number of edges and V is the number of nodes, high (central node access) traversal efficiency, excellent (centralized graph encoding) suitability for graph-based encryption, very high structural simplicity, (ii) the worst case time complexity of A* Algorithm is $O(b^d)$ where b is the branching factor and d is the depth and space complexity is $O(E + V)$ where E is the number of edges and V is the number of nodes, high (with good heuristic) traversal efficiency, good (needs proper graph abstraction) suitability for graph-based encryption,

moderate structural simplicity, (iii) the average case time complexity and best case time complexity of Dijkstra's Algorithm is $O((V + E) \log V)$ and the worst case time complexity is $O(\left(V^2\right) logV)$ and space complexity is $O(V)$ where V is the number of vertices, moderate to high traversal efficiency, fair (efficient in weighted graphs) suitability for graph-based encryption, moderate structural simplicity, and (iv) Bellman-Ford Algorithm is $O(V*E)$ time complexity and $O(V)$ space complexity where V is the number of vertices and E is the number of edges, low traversal efficiency, limited (computationally expensive) suitability for graph-based encryption, moderate structural simplicity.

KEYWORDS

cryptography, star graph, asymmetric encryption, RSA, ElGamal, complexity

1 Introduction

Data security involves implementing strategies to safeguard digital information against unauthorized disclosure and modification across computing and communication infrastructures. Given the exponential growth in data communication and transfer volumes, coupled with the escalating frequency and sophistication of cyber threats, ensuring robust data security has become more critical than ever. This urgency is underscored by the increasing number of data breaches and cyberattacks targeting sensitive information across various sectors (Li et al., 2022). Consequently, research in data security has rapidly advanced, leading to significant developments in related fields such as cryptography. These advancements have resulted in the creation of sophisticated encryption techniques, including quantum-resistant algorithms and homomorphic encryption, which enhance the protection of sensitive information across various applications. Cryptography is the practice of securing information by converting readable data (plaintext) into an unreadable format (ciphertext) through encryption, and then reverting it back to its original form via decryption. This process ensures that only authorized parties can access the original information (Adeniyi et al., 2022). Cryptography is the discipline dedicated to safeguarding information by transforming it into an unreadable format for unauthorized individuals during storage and transmission. This process, known as encryption, converts plaintext into ciphertext, ensuring that only authorized parties can revert it back to its original form through decryption. By employing mathematical techniques, cryptography ensures data confidentiality, integrity, and authenticity, making it a fundamental component of modern information security systems (Panda and Nag, 2015). Plaintext, also known as clear text, refers to data in its original, readable form that has not been encrypted. It serves as the fundamental input for encryption processes, where it is transformed into ciphertext to protect its confidentiality. Understanding and managing plaintext is crucial in information security, as it pertains to ensuring data confidentiality, integrity, and authentication (Singh et al., 2022).

Our focus is on evaluating the performance of algorithm through the utilization of graph theory and algebraic concepts (West, 2001). In graph theory, a star graph is a type of tree graph characterized by a central node connected directly to all other nodes, which are known as leaves. This structure is a specific case of a complete bipartite graph, denoted as $K_{1,n}$ where "1" represents the central node and "n" denotes the number of leaf nodes. The star graph S_{n+1} on n+1 vertices can be represented as the corona product $K_1 \odot \overline{K_n}$ where K_1 is a complete graph on one vertex, and K_n is a complete graph on n vertices. Similarly, the corona graph of a cycle C_n with K_1, denoted as $C_n \odot K_1$ is a graph on $2n$ vertices obtained by attaching n pendant edges to the cycle C_n. Each vertex of the cycle is connected to a new pendant vertex, enhancing the cycle's connectivity.

Several recent studies have focused on evaluating the performance of RSA and ElGamal cryptographic algorithms, particularly concerning their time and space complexities when applied to mixed data. This section provides a comprehensive summary of these studies, analyzing the methodologies employed and their relevance to the current research. The authors conducted a comparative analysis of RSA and ElGamal cryptographic algorithms, focusing on their energy efficiency and impact on network longevity.

The remainder of this paper is organized as follows. Section 2 presents an overview of the background of the proposed work including a survey of already published techniques regarding their security concerns. Section 3 elaborates the proposed framework of algorithm. We then discuss the implementation of the proposed work with result in Section 4. Discussion presented in Section 5 and finally, Section 6 concludes the paper along with future scope.

2 Related work

Kayalvizhi et al. (2010), utilizing a cluster-based wireless sensor network topology within the NS2 simulation environment, they assessed how each algorithm influences power consumption and overall network lifespan. The study revealed that RSA tends to consume 14.5% less power compared to ElGamal, thereby potentially extending the operational duration was limited to 10 sensor nodes of the wireless sensor networks. Researchers have implemented various cryptographic algorithms in Java to enhance cloud data security, focusing on both symmetric techniques (AES, DES, and Blowfish) and asymmetric methods (RSA). Performance evaluations revealed that Blowfish demonstrated the fastest processing time, followed by AES and DES, with RSA being the slowest. In terms of memory usage, Blowfish, AES, and 3DES consumed similar amounts, whereas RSA utilized approximately twice as much memory as the symmetric algorithms. These findings

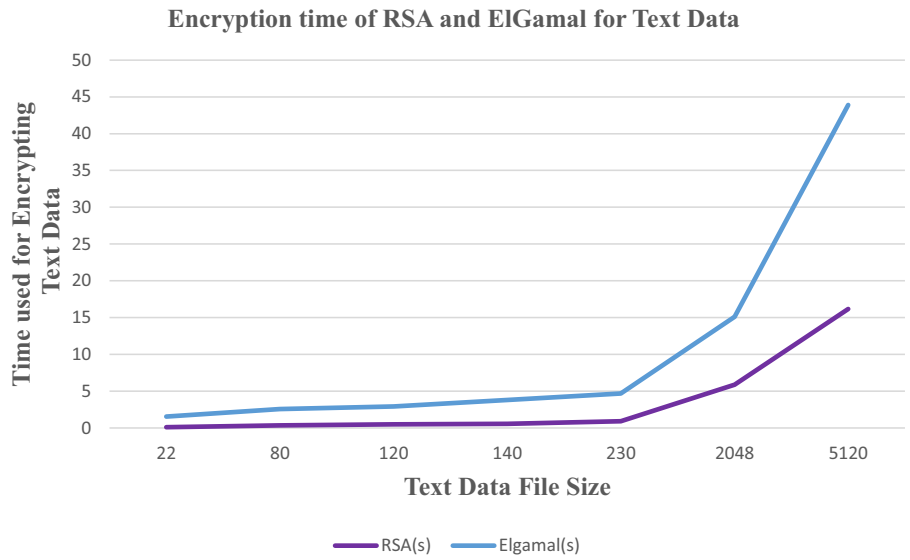

FIGURE 1
Encryption time analysis for RSA and ElGamal cryptographic algorithm for text dataset.

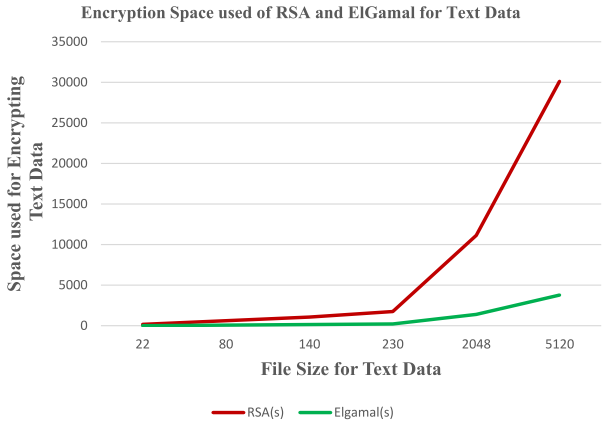

FIGURE 2
Analysis of memory used for RSA and ElGamal for text dataset during the encryption process.

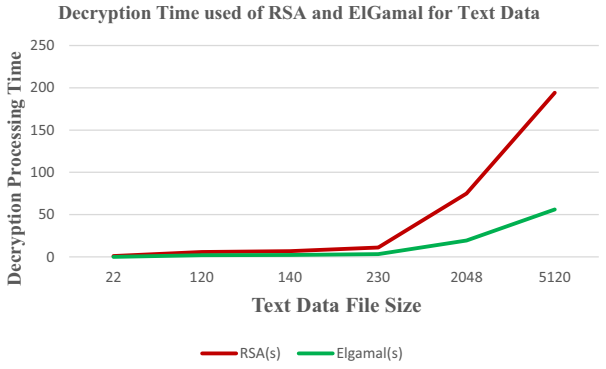

FIGURE 3
Decryption time analysis for RSA and ElGamal cryptographic algorithms for text dataset.

suggest that Blowfish offers superior speed and efficient memory usage, making it suitable for applications where performance is critical. AES provides a balance between speed and security, while RSA, despite its higher resource consumption, remains valuable for scenarios requiring robust asymmetric encryption (Arora et al., 2013). In Boni et al. (2015), the authors proposed an innovative approach to enhance the Diffie-Hellman key exchange algorithm by introducing the Multiplicative Key Exchange (MKE) technique. This method simplifies the key generation process, reducing computational complexity. Their findings indicate that MKE outperforms the traditional Diffie-Hellman algorithm in terms of execution time, requiring fewer computations. This approach is particularly beneficial in scenarios where rapid key generation is essential, such as in less complex systems with minimal setup requirements. In their study, the authors analyzed the performance of RSA and ElGamal algorithms concerning computing speeds for securing, ensuring confidentiality, and authenticating text data. They employed the computer's internal clock to measure and compare the execution times of both algorithms across various text data inputs, aiming to determine which method is more computationally effective. The implementation was tested using text files of different sizes. The results indicated that RSA is more computationally efficient than ElGamal, leading to better performance. However, a limitation of this research is the use of text data with limited character sizes (Okeyinka, 2015). In another study, the authors evaluated the performance of LUC, ElGamal, and RSA algorithms in text encryption. Each algorithm was implemented across various text samples to measure encryption and decryption times. The findings indicated that RSA exhibited superior performance in encryption speed, while

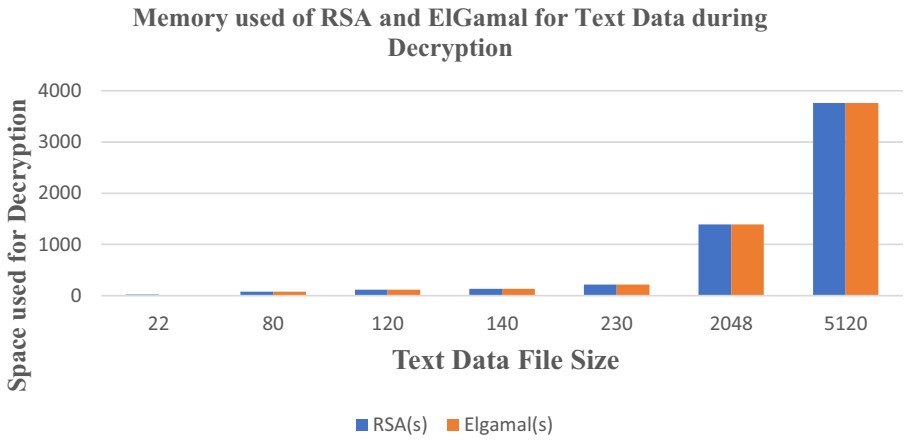

FIGURE 4
Analysis of memory used for RSA and ElGamal for text data set during the decryption process.

LUC demonstrated enhanced efficiency during decryption. The scope of the study was confined to encrypting secret messages in textual format (Sari et al., 2020). Ni et al. (2021) collaboratively developed novel encryption algorithms utilizing specific graph structures, notably corona graphs and bipartite graphs, to enhance secure message transmission. In Desai et al. (2022) and Behera and Gountia (2024), the authors conducted a comprehensive analysis of several asymmetric public key cryptosystems, focusing on performance-based criteria and metrics. Their research entailed an in-depth comparative examination of RSA, ElGamal, and ECC-ElGamal algorithms. The study aimed to derive clear conclusions regarding the performance requirements of these algorithms. Notably, the research highlighted that Elliptic Curve Cryptography (ECC) offers significant advantages in terms of smaller key sizes and higher computational efficiency, making it well-suited for modern devices with limited processing capabilities, such as smart cards and IoT devices. In Parenreng and Wahid (2022), the study proposed a hybrid cryptographic approach that utilizes the ElGamal encryption model for secure symmetric key distribution, coupled with the Advanced Encryption Standard (AES) algorithm for encrypting message content. This combination leverages the strengths of both asymmetric and symmetric encryption techniques to enhance email security. The implementation was integrated into an email system, effectively encrypting messages and data transmitted via email. The primary objective was to address prevalent email security challenges, particularly the risk of data leakage during email transmission. In this research article (Ali et al., 2024), the authors collectively developed novel encryption algorithms utilizing specific graph structures—namely corona graphs, star graphs, and complete bipartite graphs—to enhance secure message transmission.

3 Proposed algorithm

This study evaluates the performance of RSA and ElGamal cryptographic algorithms by measuring their encryption and decryption times, as well as memory usage on text data.

RSA Algorithm (Pseudocode):

Plaintext:
Input: Message M, Key size (k), Two large primes p and q
Output: Ciphertext C
Key Generation:

1. Choose primes p, q
2. Compute modulus: n ← p × q
3. Compute $\phi(n) \leftarrow (p-1) \times (q-1)$
4. Choose public exponent e such that gcd (e, $\phi(n)) = 1$
5. Compute private key
   d ← $e^{-1}(mod\ \phi(n))$
   Encryption:
6. C ← $M^e(mod\ n)$
   Decryption:
7. M ← $C^d(mod\ n)$

ElGamal Algorithm (Pseudocode):
Input: Message M, Prime p, Generator g, Private key x
Output: Ciphertext ($C1, C_2$)
Key Generation:

1. Choose large prime p and generator g ϵ $Z_p{}^*$.
2. Select private key x ϵ [1, p-2]
3. Compute h ← $g^x\ mod\ p$
   Public key = (p, g, h)
   Encryption:
4. Select random k ϵ [1, p-2]
5. C1 ← $g^k\ mod\ p$
6. C_2 ← $M.h^k\ mod\ p$
   Ciphertext = ($C1, C_2$)
   Decryption:
7. $s \leftarrow C_1{}^x\ mod\ p$
8. s^{-1} ← modular inverse of s mod p.
9. M ← $C_2\ .s^{-1}\ mod\ p$.

Dataset justification & computational considerations:
Data sources: realistic test files (text, images, and audio clips) and randomly generated structured data were used. File sizes: 22, 50, 55, 60, 90, 120, 200, 2,048, and 5,120 KB.

Execution platform: windows 10 (64-bit), Intel i7 2.23 GHz, 8 GB RAM.

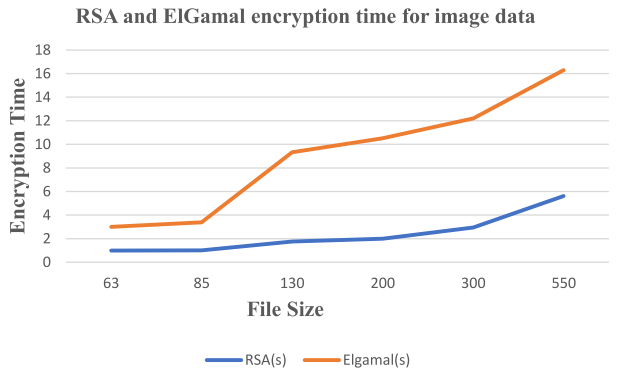

FIGURE 5
RSA and ElGamal encryption time analysis for image data.

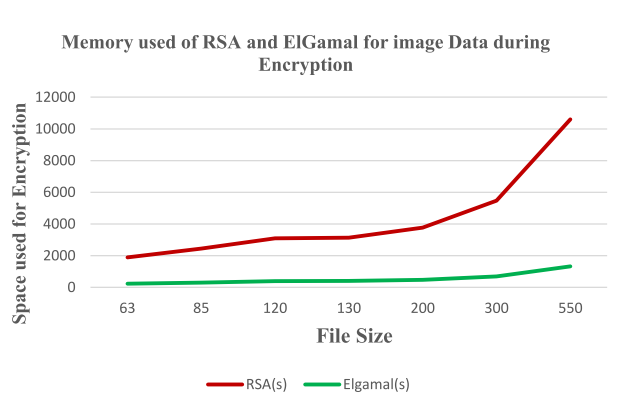

FIGURE 6
RSA and ElGamal space used for encrypting image data.

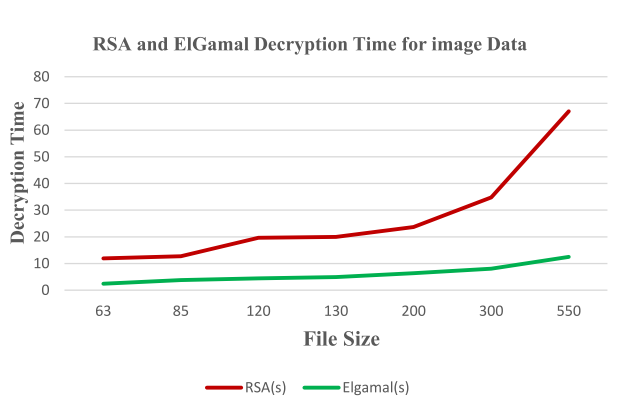

FIGURE 7
RSA and ElGamal space decryption time analysis for image data.

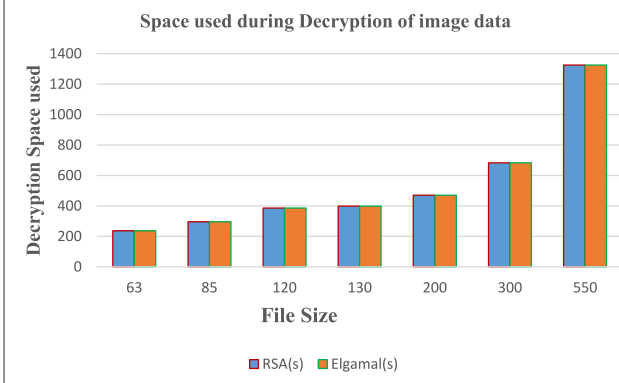

FIGURE 8
Space used by RSA and ElGamal during decryption of image data.

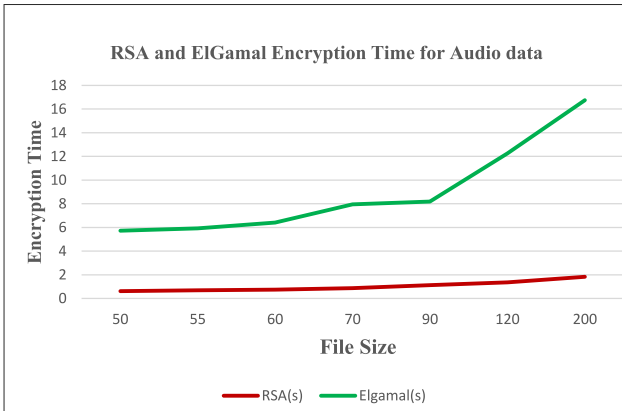

FIGURE 9
Encryption time of RSA and ElGamal algorithm for audio data.

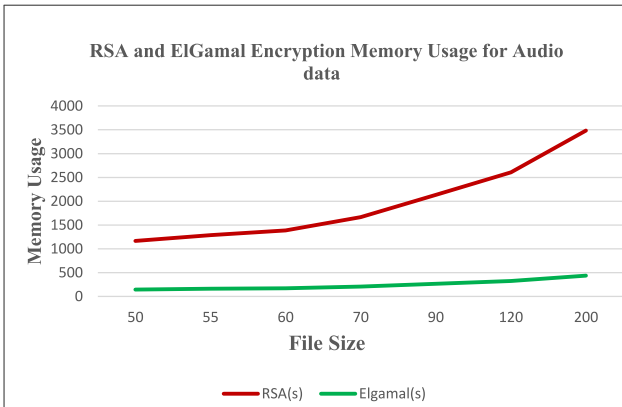

FIGURE 10
Memory usage of RSA and ElGamal algorithm during encryption of audio data.

Language and Mode: RSA and ElGamal implemented in C#, using CBC mode with key sizes of 64-bit and 128-bit.

Rounds: 10 rounds of encryption/decryption per block.

Measurement: CPU clock used for timing. Execution times and memory usage recorded and plotted (see Figures 1–12).

Relevance of modulo arithmetic: modulo operations (mod n or mod p) play a critical role in ensuring bounded number systems and protecting against overflow during exponentiation. The choice of p, n, and their bit-length directly influences:

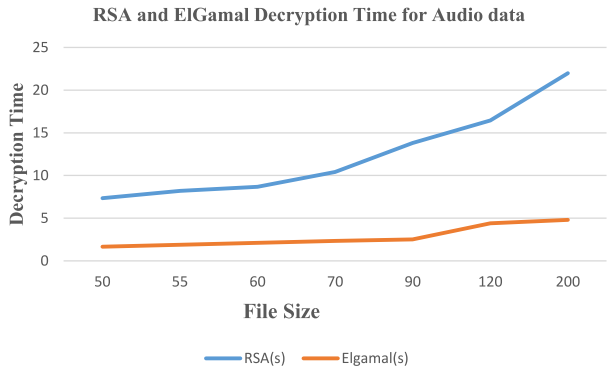

FIGURE 11
Decryption time of RSA and ElGamal algorithm for audio data.

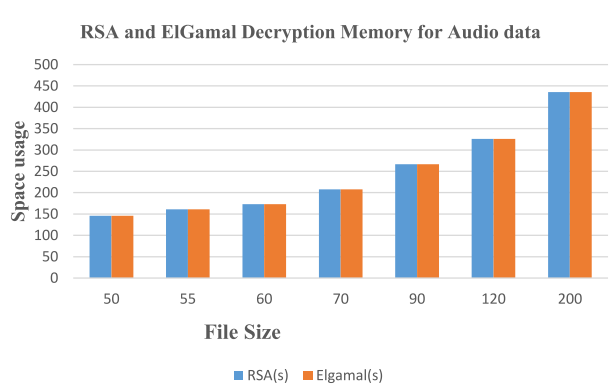

FIGURE 12
Decryption memory usage of audio data using RSA and ElGamal algorithm.

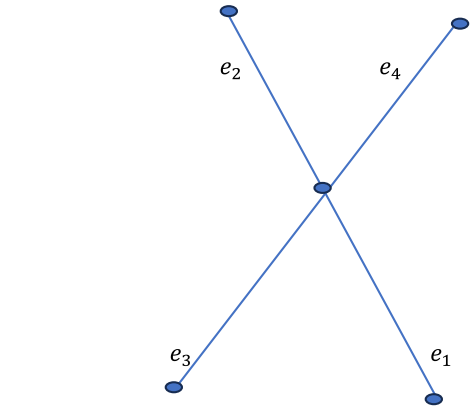

FIGURE 13
Star graph.

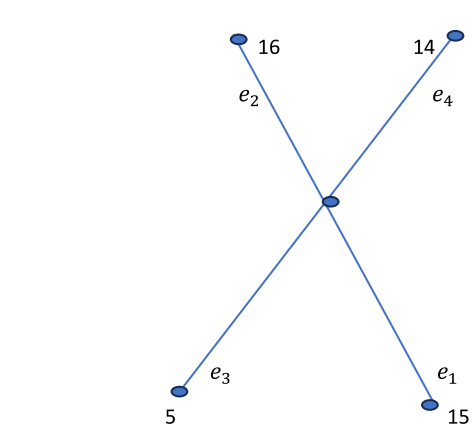

FIGURE 14
Star graph with numeric value.

- Execution time (due to large number operations).
- Memory use (due to ciphertext expansion).
- Security strength (based on size of p/n).

Let us assume the encryption time (C1), decryption time (C2), and memory usage (C3) for a 22 KB text file using RSA and ElGamal with 128-bit keys, highlighting the impact of modulo arithmetic operations. The number of blocks (B) is estimated as 22 KB/16 bytes (CBC block size) ≈ 1,375 blocks.

RSA: each block requires one modular exponentiation $c = (M^e mod\ n)$ for encryption and one $M = (c^d mod\ n)$ for decryption, with time complexity $O\left(\log^2 n\right)$ *and* $O\left(\log^3 n\right)$. For 1,375 blocks, C1 ≈ , $1375 * k_1 * \log^3 (2^{128}) \approx 0.1082$ s, where $k_1 \approx 3.6 * 10^{-8}$ *seconds per modular* reflecting efficient implementation. Memory usage (C3) is higher due to larger buffers for n and temporary values.

ElGamal: encryption requires two modular exponentiations per block ($g^k\ mod\ p, y^k * M\ mod\ p$), doubling the computational cost, so $C1 \approx 1,375 * 2 * k_4 * \log^3 (2^{128}) \approx 1.55$ seconds where $k_4 \approx 2.8 * 10^{-7}$ *s*. Decryption involves one exponentiation and one inverse, increasing C2. Lower memory usage (C3) results from optimized storage of p and ciphertexts.

In this section, while RSA key generation is a standard cryptographic step, in this study it is primarily used to support the performance evaluation of encryption and decryption processes. Therefore, detailed key generation steps are intentionally abstracted to maintain focus on time and space complexity analysis. However, for completeness, key length (64-bit and 128-bit) and their role in computational cost are considered in performance graphs. Key generation complexity can be added in future work for deeper cryptographic analysis.

Performance evaluation metrics: this study assesses the efficiency of cryptographic algorithms using the following metrics:

Encryption time: the duration required by the algorithm to encrypt text datasets. This is measured using the system's internal clock.

Decryption time: the time taken by the algorithm to decrypt text datasets, also recorded via the system's internal clock.

Encryption memory usage: the amount of system memory consumed during the encryption process of text data.

Decryption memory usage: the memory utilized during the decryption process of text data.

CPU internal clock: utilized to accurately measure the encryption and decryption times across various data categories.

TABLE 1 Tabular representation of text data encryption for RSA and ElGamal algorithms.

Serial No.	File size (KB)	Time of encryption		Space of encryption	
		RSA(s)	ElGamal(s)	RSA (kb)	ElGamal (kb)
1	22	0.1082	1.55	169.82	0.1650
2	80	0.3545	2.57	623.50	77.9300
3	120	0.4835	2.92	925.85	115.7100
4	140	0.5664	3.80	1,054.83	131.8400
5	230	0.9315	4.67	1,740.99	217.6200
6	2,048	5.8852	15.12	11,133.64	1,391.7000
7	5,120	16.1733	43.90	30,116.30	3,764.5200

TABLE 2 Tabular representation of text data decryption for RSA and ElGamal algorithms.

Serial No.	File size (KB)	Time of decryption		Space of decryption	
		RSA(s)	ElGamal(s)	RSA (kb)	ElGamal (kb)
1	22	1.0756	0.0802	21.22	0.1650
2	80	3.9254	1.6674	77.93	77.93
3	120	5.7463	1.9284	115.71	115.71
4	140	6.8078	2.2112	131.84	131.84
5	230	11.1189	3.2596	217.62	217.62
6	2,048	74.9069	19.3083	1,391.69	1,391.69
7	5,120	194.2630	56.1964	3,764.52	3,764.52

CPU internal memory: employed to determine the memory consumption of both algorithms during the encryption and decryption processes for all data types.

4 Results

Example 1: consider the word "OPEN." To encrypt this word using a specific scheme, we begin by converting each alphabetic character to its corresponding numeric value. Assuming a simple substitution where $A = 1$, $B = 2$,..., $Z = 26$, the conversions are as follows:

O P E N

15 16 5 14

Thus, the word "OPEN" is represented numerically as 15 16 5 14. The length of the message, denoted as K, is 4. Consider a star graph S_5, which can be represented as the corona product $S_5 = K_1 \odot \overline{K_4}$. In this structure, the central vertex of K_1 is connected to each vertex of the complete graph K_4, forming a star-like configuration. This graph comprises five vertices in total, corresponding to the length of the message. As illustrated in Figures 13, 14 the edges connecting the central node to the peripheral nodes are labeled sequentially as e_1, e_2, e_3, e_4.

4.1 Application of RSA algorithm in star graph $K_1 \odot \overline{K_n}$

Now RSA algorithm begins: select two prime numbers, $p = 3$ and $q = 11$. Calculate $n = pq = 3*11 = 33$. Calculate $\phi(n) = (p - 1)(q - 1) = (3 - 1)(11 - 1) = 2*10 = 20$. Select e such that e is relatively prime to $\phi(n)$. So, we select $e = 7$ determine d such that

$$\begin{aligned} de &\equiv 1 \left(mod \varnothing (n) \right) \\ \Rightarrow 7d &\equiv 1 \left(mod\ 20 \right) \\ \Rightarrow 7 * 3 &\equiv 1 \left(mod\ 20 \right) \\ \Rightarrow 21 &\equiv 1 \left(mod\ 20 \right) \end{aligned}$$

where, d is private key.

Here, Public key PU $(e, n) = 7, 33$, Private key PR $(d, n) = 3, 33$.

Assign the vertex of graph as $\beta_1 = 15$, $\beta_2 = 16$, $\beta_3 = 5$, $\beta_4 = 14$.

Then find $\delta_i = \beta_i^e \left(mod\ n \right)$.

$$\begin{aligned} \delta_1 &= (15)^7\ mod\ 33 = 27 \\ \delta_2 &= (16)^7\ mod\ 33 = 25 \\ \delta_3 &= (5)^7\ mod\ 33 = 14 \\ \delta_4 &= (14)^7\ mod\ 33 = 20 \end{aligned}$$

Subtract k to each δ_i give γ_i. So, $\gamma_1 = \delta_1 - k$, $\gamma_2 = \delta_2 - k$, $\gamma_3 = \delta_3 - k$, $\gamma_4 = \delta_4 - k$.

We will get, $\gamma_1 = 23$, $\gamma_2 = 21$, $\gamma_3 = 10$, $\gamma_4 = 16$.

Convert each γ_i value to character letter as $W = 23$, $U = 21$, $J = 10$, $P = 16$.

So, sender send message "WUJP" to the receiver.

Receiver after getting message, convert to numeric value as 23, 21, 10, 16.

TABLE 3 Image data encryption for RSA and ElGamal algorithms.

Serial No	File size (KB)	Time of encryption		Space of encryption	
		RSA(s)	ElGamal(s)	RSA (kb)	ElGamal (kb)
1	63	0.9896	2.9947	1,890.32	236.29
2	85	1.0023	3.3907	2,439.15	295.41
3	120	1.6205	8.7705	3,088.11	385.73
4	130	1.7495	9.3232	3,129.38	399.20
5	200	1.9853	10.5232	3,764.52	470.56
6	300	2.9534	12.2056	5,470.91	683.85
7	550	5.6149	16.2851	10,597.82	1,324.71

TABLE 4 Image data decryption for RSA and ElGamal algorithms.

Serial No.	File size (KB)	Time of decryption		Space of decryption	
		RSA(s)	ElGamal(s)	RSA (kb)	ElGamal (kb)
1	63	11.8935	2.4517	236.29	236.29
2	85	12.6888	3.8033	295.41	295.41
3	120	19.6372	4.3965	385.73	385.73
4	130	19.9276	4.9207	399.20	399.20
5	200	23.6912	6.3696	470.56	470.56
6	300	34.7945	8.0873	683.85	683.85
7	550	67.0517	12.4493	1,324.71	1,324.71

TABLE 5 Audio data encryption for RSA and ElGamal algorithms.

Serial No	File size (KB)	Time of encryption		Space of encryption	
		RSA(s)	ElGamal(s)	RSA (kb)	ElGamal (kb)
1	50	0.6186	5.7240	1,167.72	145.96
2	55	0.6806	5.9135	1,289.66	161.21
3	60	0.7383	6.4193	1,384.90	173.10
4	70	0.8740	7.9503	1,663.56	207.92
5	90	1.1263	8.1892	2,131.86	266.48
6	120	1.3651	12.2567	2,606.37	325.79
7	200	1.8295	16.7535	3,483.51	435.55

TABLE 6 Audio data decryption for RSA and ElGamal algorithms.

Serial No.	File size (KB)	Time of decryption		Space of decryption	
		RSA(s)	ElGamal(s)	RSA (kb)	ElGamal (kb)
1	50	7.3565	1.6570	145.96	145.96
2	55	8.2104	1.8803	161.21	161.21
3	60	8.6874	2.1033	173.10	173.10
4	70	10.4017	2.3383	207.92	207.92
5	90	13.7977	2.5158	266.48	266.48
6	120	16.4544	4.4145	325.79	325.79
7	200	21.9815	4.7963	435.55	435.55

TABLE 7 Statistical analysis such as mean, standard deviation, performance metrics of encryption/decryption times (50 runs, AES-CBC 128-bit, RSA/ElGamal 2,048-bit, 10 rounds, star graph).

Data type	File size (KB)	RSA (mean ± SD, s)		ElGamal (mean ± SD, s)		RSA	ElGamal
		Encryption	Decryption	Encryption	Decryption	Thr (KB/s)	Thr (KB/s)
Text	22	0.1082 ± 0.002	1.0756 ± 0.08	1.55 ± 0.031	0.0802 ± 0.014	203.3	14.2
Text	80	0.3545 ± 0.0047	3.9254 ± 0.17	2.57 ± 0.113	1.6674 ± 0.032	341.9	14.2
Text	120	0.4835 ± 0.0113	5.7463 ± 0.26	2.92 ± 0.169	1.9284 ± 0.049	212.4	14.2
Text	140	0.5664 ± 0.0132	6.8078 ± 0.30	3.80 ± 0.197	2.2112 ± 0.057	212.8	14.2
Text	230	0.9315 ± 0.0216	11.1189 ± 0.50	4.67 ± 0.324	3.2596 ± 0.093	212.9	14.2
Text	2,048	5.8852 ± 0.193	74.9069 ± 4.4	15.12 ± 2.88	19.3083 ± 0.83	212.5	14.2
Text	5,120	16.1733 ± 0.482	194.2630 ± 11.1	43.90 ± 7.20	56.1964 ± 2.06	212.4	14.2
Image	63	0.9896 ± 0.0030	11.8935 ± 0.11	2.9947 ± 0.089	2.4517 ± 0.021	417.2	14.1
Image	85	1.0023 ± 0.0041	12.6888 ± 0.15	3.3907 ± 0.120	3.8033 ± 0.028	417.7	14.1
Image	120	1.6205 ± 0.0058	19.6372 ± 0.21	8.7705 ± 0.170	4.3965 ± 0.039	416.7	14.1
Image	130	1.7495 ± 0.0062	19.9276 ± 0.23	9.3232 ± 0.184	4.9207 ± 0.043	416.7	14.1
Image	200	1.9853 ± 0.0096	23.6912 ± 0.35	10.5232 ± 0.284	6.3696 ± 0.066	416.7	14.1
Image	300	2.9534 ± 0.0144	34.7945 ± 0.53	12.2056 ± 0.426	8.0873 ± 0.098	416.7	14.1
Image	550	5.6149 ± 0.0264	67.0517 ± 0.97	16.2851 ± 0.780	12.4493 ± 0.180	416.7	14.1
Audio	50	0.6186 ± 0.0032	7.3565 ± 0.12	5.7240 ± 0.093	1.6570 ± 0.022	312.5	10.8
Audio	55	0.6806 ± 0.0035	8.2104 ± 0.13	5.9135 ± 0.102	1.8803 ± 0.024	312.5	10.8
Audio	60	0.7383 ± 0.0038	8.6874 ± 0.14	6.4193 ± 0.112	2.1033 ± 0.026	312.5	10.8
Audio	70	0.8740 ± 0.0045	10.4017 ± 0.16	7.9503 ± 0.130	2.3383 ± 0.030	312.5	10.7
Audio	90	1.1263 ± 0.0058	13.7977 ± 0.21	8.1892 ± 0.167	2.5158 ± 0.039	312.5	10.8
Audio	120	1.3651 ± 0.0077	16.4544 ± 0.28	12.2567 ± 0.224	4.4145 ± 0.052	312.5	10.7
Audio	200	1.8295 ± 0.0128	21.9815 ± 0.46	16.7535 ± 0.372	4.7963 ± 0.086	312.5	10.8

He then add length of message 4 to the numeric value. We will get, $23 + 4 = 27$, $21 + 4 = 25$, $10 + 4 = 14$, $16 + 4 = 20$. Assign the value as δ_i. So, $\delta_1 = 27$, $\delta_2 = 25$, $\delta_3 = 14$, $\delta_4 = 20$.

Now receiver use private key d = 3 and find $\alpha_i = {\delta_i}^d \ (mod\ n)$. So,

$\alpha_1 = {\delta_1}^d \ (mod\ n) = 15$,

$\alpha_2 = {\delta_2}^d \ (mod\ n) = 16$,

$\alpha_3 = {\delta_3}^d \ (mod\ n) = 5$,

$\alpha_4 = {\delta_4}^d \ (mod\ n) = 14$. Convert the α_i value to alphabetic character. $\alpha_1 = \mathrm{O}, \alpha_2 = P, \alpha_3 = E, \alpha_4 = \mathrm{N}$.

Finally, receiver receives message "OPEN."

4.2 Application of ElGamal algorithm in star graphs $K_1 \odot \overline{K_n}$

- Select a large prime number $p = 11$ and a generator $g = 2$ of the multiplicative group ${Z_p}^*$. Choose a private key $x = 3$ such that $1 \leq x \leq p - 2$ and $gcd\ (x, p) = 1$. Compute $h = g^x \ mod\ p = 2^3 \ mod\ 11 = 8$.
- The public key is the tuple $(p, g, h) = (11, 2, 8)$ and the private key is $x = 3$.

Encryption:

To encrypt a message "OPEN," we assign $M_1 = 0 = 15$, $M_2 = P = 16, M_3 = E = 5, M_4 = N = 14$.

Select a random integer $k = 4$ such that $1 \leq k \leq p - 2$ & $gcd\ (k, p) = 1$.

Find $C_1 = g^k \ mod\ p = 2^4 \ mod\ 11 = 5$,

$C_2 = M_1.h^k \ mod\ p = 15.8^4 \ mod\ 11 = 5$,

$C_3 = M_2.h^k \ mod\ p = 16.8^4 \ mod\ 11 = 9$,

$C_4 = M_3.h^k \ mod\ p = 5.8^4 \ mod\ 11 = 9$,

$C_5 = M_4.h^k \ mod\ p = 14.8^4 \ mod\ 11 = 1$.

Convert the numeric value of C_1, C_2, C_3, C_4, and C_5 as alphabetic character as "EEIIA".

Sender send message "EEIIA" to the receiver.

Decryption:

Receiver now convert message to numeric value as $C_1 = 5$, $C_2 = 5$, $C_3 = 9$, $C_4 = 9$, $C_5 = 1$. Compue $M_1 = C_2.(C_1{}^x)^{-1} \bmod P = 15$. Similarly $M_2 = C_3.(C_1{}^x)^{-1} \ mod\ P = 16$, $M_3 = C_4.(C_1{}^x)^{-1} \ mod\ P = 5$, $M_4 = C_5.(C_1{}^x)^{-1} \bmod P = 14$.

Now receiver convert numeric value to alphabetic character as "OPEN." Hence receives original message "OPEN."

In this study, the RSA and ElGamal cryptographic algorithms were implemented using the C# programming language to assess their performance across diverse data types, including mixed data (text, image, audio) (Adeniyi et al., 2023; Gountia et al., 2025). The

TABLE 8 Single-round estimates for all data types text, image and audio files.

Data type	File size (KB)	RSA (s)				ELGAMAL (s)			
		10 round		Single round		10 round		Single round	
		Enc	Dec	Enc	Dec	Enc	Dec	Enc	Dec
Text	22	0.1082	1.0756	0.0108	0.1075	1.55	0.0802	0.155	0.00802
Text	80	0.3545	3.9254	0.0354	0.3925	2.57	1.6674	0.257	0.16674
Text	120	0.4835	5.7463	0.0483	0.5746	2.92	1.9284	0.292	0.19284
Text	140	0.5664	6.8078	0.0566	0.6807	3.80	2.2112	0.380	0.22112
Text	230	0.9315	11.1189	0.0931	1.1119	4.67	3.2596	0.467	0.32596
Text	2,048	5.8852	74.9069	0.5885	7.4909	15.12	19.3083	1.512	1.93083
Text	5,120	16.1733	194.263	1.6173	19.4263	43.90	56.1964	4.390	5.61964
Image	63	0.9896	11.8935	0.0989	1.18935	2.9947	2.4517	0.29947	0.24517
Image	85	1.0023	12.6888	0.1002	1.26888	3.3907	3.8033	0.33907	0.38033
Image	120	1.6205	19.6372	0.1620	1.96372	8.7705	4.3965	0.87705	0.43965
Image	130	1.7495	19.9276	0.1749	1.99276	9.3232	4.9207	0.93232	0.49207
Image	200	1.9853	23.6912	0.1985	2.36912	10.5232	6.3696	1.05232	0.63696
Image	300	2.9534	34.7945	0.2953	3.47945	12.2056	8.0873	1.22056	0.80873
Image	550	5.6149	67.0517	0.5614	6.70517	16.2851	12.4493	1.62851	1.24493
Audio	50	0.6186	7.3565	0.0618	0.73565	5.7240	1.6570	0.57240	0.16570
Audio	55	0.6806	8.2104	0.0680	0.82104	5.9135	1.8803	0.59135	0.18803
Audio	60	0.7383	8.6874	0.0738	0.86874	6.4193	2.1033	0.64193	0.21033
Audio	70	0.8740	10.4017	0.0874	1.04017	7.9503	2.3383	0.79503	0.23383
Audio	90	1.1263	13.7977	0.1126	1.37977	8.1892	2.5158	0.81892	0.25158
Audio	120	1.3651	16.4544	0.1365	1.64544	12.2567	4,4145	1.22567	0.44145
Audio	200	1.8295	21.9815	0.1829	2.19815	16.7535	4.7963	1.67535	0.47963

TABLE 9 Comparison of encryption/decryption times for AES-256, ECC, RSA, and ElGamal (single-round, star graph, 128-bit AES-CBC, 2,048-bit RSA/ElGamal/ECC).

Data type	File size (KB)	AES-256(s)		ECC(s)		RSA(s)		ELGAMAL(s)	
		Enc	Dec	Enc	Dec	Enc	Dec	Enc	Dec
Text	22	0.0010	0.0010	0.015	0.015	0.0108	0.1075	0.155	0.00802
Text	5,120	0.230	0.230	3.50	3.50	1.6173	19.4263	4.390	5.61964
Image	550	0.050	0.050	0.375	0.375	0.5614	6.70517	1.62851	1.24493
Audio	200	0.018	0.018	0.137	0.137	0.1829	2.19815	1.67535	0.47963

evaluation focused on key performance metrics such as encryption time, decryption time, and memory usage. These metrics were systematically recorded and presented in tabular formats shown in Tables 1–9 with encryption and decryption times measured in seconds (s) and memory usage detailed in kilobytes (KB). To provide a clearer comparative analysis, Figures 1–12 is a graphical representations corresponding to each dataset were also generated, illustrating the time efficiency and memory consumption of both algorithms (Arhin et al., 2023; Utama Siahaan et al., 2018).

Figure 1 results indicate that the RSA algorithm consumes less time during text data encryption compared to the ElGamal algorithm whereas Figure 2 shows that the RSA algorithm consumes more CPU internal memory while encrypting text data than the ElGamal algorithm.

Figure 3 shows that RSA algorithm consumes more CPU time during the decryption of text data while ElGamal consumes less CPU time during the decryption of text data. Figure 4 shows that both algorithms consume an equal volume of CPU internal memory to decrypt text data.

Figure 5 shows that RSA algorithm consumes less CPU time during the encryption of image data while ElGamal consumes more CPU time during the decryption of image data. Figure 6 shows that ElGamal consumes less memory during image data encryption than the RSA algorithm.

Figure 7 shows that RSA algorithm consumes more CPU time during the encryption of image data while ElGamal consumes less CPU time during the decryption of image data. Figure 8 shows that both algorithms consume an equal volume of CPU internal memory to decrypt text data.

Figure 9 shows that RSA algorithm consumes less CPU time during the encryption of audio data while ElGamal consumes more CPU time during the encryption of audio data. Figure 10 shows that ElGamal consumes less memory during audio data encryption than the RSA algorithm.

Figure 11 shows that RSA algorithm consumes more CPU time during the decryption of audio data while ElGamal consumes less CPU time during the decryption of audio data. Figure 12 shows that both algorithms consume an equal volume of CPU internal memory to decrypt audio data.

Table 1 presents the encryption time and memory usage for the text dataset using RSA and ElGamal cryptographic algorithms. The execution times were measured using the computer's CPU internal clock.

Table 2 shows the decryption time and memory usage of RSA and ElGamal cryptographic algorithms on the test dataset.

Table 3 shows the encryption time and memory usage of RSA and ElGamal cryptographic algorithms on the image dataset.

Table 4 shows the decryption time and memory usage of RSA and ElGamal cryptographic algorithms on the image dataset.

Table 5 shows the encryption time and memory usage of RSA and ElGamal cryptographic algorithms on the audio dataset.

Table 6 shows the decryption time and memory usage of RSA and ElGamal cryptographic algorithms on the audio dataset.

Table 7 explain a statistical analysis of 50 runs per file size, reporting mean encryption/decryption times and standard deviations. For example, RSA's mean encryption time for 22 KB text is 0.1082 s (SD: 0.002 s), and ElGamal's is 1.55 s (SD: 0.03 s). Table 7 also define throughput as file size divided by encryption time (KB/s). For example, RSA's throughput for 22 KB text is 203.3 KB/s (22/0.1082), while ElGamal's is 14.2 KB/s (22/1.55).

Table 8 extrapolate single-round times (e.g., RSA: ∼0.0108 s for 22 KB text) and discuss applications: RSA's speed suits secure messaging (e.g., 50 KB audio), while ElGamal's security fits constrained devices.

Table 9 comparison with AES-256 (symmetric) and ECC (asymmetric). AES-256 encrypts faster (e.g., ∼0.05 s for 550 KB image) but requires secure key exchange, where RSA/ElGamal excel, ECC is faster than ElGamal but less standard.

5 Discussion

In this study, two prominent asymmetric cryptographic algorithms—RSA and ElGamal—were implemented using the C# programming language to evaluate their performance across various data types, including text, image and audio files (Yousif, 2023).

5.1 Experimental setup

Hardware configuration: the simulations were conducted on a laptop equipped with a Windows 10 64-bit operating system, an Intel i7 processor running at 2.23 GHz, and 8 GB of RAM. Test Data: Randomly generated files of varying sizes—22 KB, 80 KB, 120 KB, 140 KB, 230 KB, 2048 KB, and 5120 KB—were used as input datasets. Implementation Details: Both RSA and ElGamal algorithms were implemented in C#, utilizing the Cipher Block Chaining (CBC) mode with key sizes of 64 bits and 128 bits. Each data block underwent 10 rounds of encryption and decryption, with execution times recorded for each run using the system's CPU internal clock. Encryption time: RSA consistently demonstrated faster encryption times across all data categories compared to ElGamal.

6 Conclusion

In this study, the RSA and ElGamal cryptographic algorithms were implemented using the C# programming language to evaluate their performance across various data types, including text, image, and audio files. The experimental results indicated that the RSA algorithm consistently outperformed ElGamal in terms of encryption time across all data categories. For instance, encrypting a 22 KB text file took ∼0.1082 s with RSA, whereas ElGamal required about 1.55 s. This trend persisted across larger file sizes and different data types, including images and audio files. Conversely, ElGamal demonstrated superior performance in memory efficiency during both encryption and decryption processes. For the same 22 KB text file, RSA used about 169.82 KB of memory during encryption, while ElGamal utilized ∼0.1650 KB. These findings suggest that RSA is more time-efficient, making it suitable for applications where speed is critical. On the other hand, ElGamal's lower memory consumption makes it preferable in environments with limited memory resources. This study contributes to the existing body of knowledge by providing a comprehensive analysis of the time and space complexities of RSA and ElGamal algorithms on mixed data types. Future research could explore additional performance metrics, such as throughput, accuracy, precision, and recall, or consider other cryptographic algorithms to further enhance the understanding of algorithm performance in various contexts (Yu et al., 2016).

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

YB: Validation, Formal analysis, Writing – original draft. BR: Writing – review & editing. DG: Supervision, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Gen AI was used in the creation of this manuscript.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Adeniyi, A. E., Misra, S., Daniel, E., and Bokolo, A. Jr. (2022). Computational complexity of modified blowfish cryptographic algorithm on video data. *Algorithms* 15:373. doi: 10.3390/a15100373

Adeniyi, E. A., Imoize, A. L., Awotunde, J. B., Lee, C., Falola, P., Jimoh, R. G., et al. (2023). Performance analysis of two famous cryptographic algorithms on mixed data. *J. Comput. Sci.* 19, 694–706. doi: 10.3844/jcssp.2023.694.706

Ali, N., Sadiqa, A., Shahzad, M. A., Imran Qureshi, M., Siddiqui, H. M. A., Abdallah, S. A. O., et al. (2024). Secure communication in the digital age: a new paradigm with graph-based encryption algorithms. *Front. Comput. Sci.* 6:1454094. doi: 10.3389/fcomp.2024.1454094

Arhin, P. K. Jr., Asante, M., and Otoo, L. (2023). A comparative study of RSA and ELGAMAL cryptosystems. *Int. J. Comput. Eng.* 4, 33–41. doi: 10.47941/ijce.1291

Arora, R., Parashar, A., and Transforming, C. C. I. (2013). Secure user data in cloud computing using encryption algorithms. *Int. J. Eng. Res. Appl.* 3, 1922–1926.

Behera, R. R., and Gountia, D. (2024). A secure fault detection for digital microfluidic biochips. *Comput. J.* 68, 217–227. doi: 10.1093/comjnl/bxae106

Boni, S., Bhatt, J., and Bhat, S. (2015). Improving the Diffie-Hellman key exchange algorithm by proposing the multiplicative key exchange algorithm. *Int. J. Comput. Appl.* 130, 7–10. doi: 10.5120/ijca2015907170

Desai, A., Parekh, V., and Unadkat, U. N. (2022). "Performance analysis of various asymmetric public-key cryptosystem," in *Pervasive Computing and Social Networking: Proceedings of ICPCSN 2022* (Singapore: Springer Nature Singapore), 437–449. doi: 10.1007/978-981-19-2840-6_34

Gountia, D., Behera, R., Bal, P., and Pati, S. (2025). Trojan detection in digital microfluidic biochips via image classification: a deep-learning based approach. *IEEE Trans. Dependable Secure Comput.* 22, 1—13. doi: 10.1109/TDSC.2025.3568217

Kayalvizhi, R., Vijayalakshmi, M., and Vaidehi, V. (2010). "Energy analysis of RSA and ElGamal algorithms for wireless sensor networks," in *CNSA 2010, Chennai, India, July 23-25, 2010. Proceedings 3* (Cham: Springer Berlin Heidelberg), 172–180. doi: 10.1007/978-3-642-14478-3_18

Li, C. T., Weng, C. Y., Chen, C. L., Lee, C. C., Deng, Y. Y., and Imoize, A. L. (2022). An efficient authenticated key agreement scheme supporting privacy-preservation for internet of drones communications. *Sensors* 22:9534. doi: 10.3390/s22239534

Ni, B., Qazi, R., Ur Rehman, S., and Farid, G. (2021). Some graph-based encryption schemes. *J. Math.* 2021:6614172. doi: 10.1155/2021/6614172

Okeyinka, A. E. (2015). "Computational speeds analysis of RSA and ElGamal algorithms on text data," in *Proceedings of the World Congress on Engineering and Computer Science, Vol. 1*, 21–23. Available online at: https://www.iaeng.org/publication/WCECS2015/WCECS2015_pp115-118.pdf (Accessed April 2, 2025).

Panda, M., and Nag, A. (2015). "Plain text encryption using AES, DES and SALSA20 by java based bouncy castle API on Windows and Linux," in *2015 Second International Conference on Advances in Computing and Communication Engineering* (Dehradun: IEEE), 541–548. doi: 10.1109/ICACCE.2015.130

Parenreng, J. M., and Wahid, A. (2022). The E-mail security system using ElGamal hybrid algorithm and AES (advanced encryption standard) algorithm. *Int. Things Artif. Intell. J.* 2, 1–9. doi: 10.31763/iota.v2i1.510

Sari, P. P., Nababan, E. B., and Zarlis, M. (2020). "Comparative study of luc, ElGamal and RSA algorithms in encoding texts," in *2020 3rd International Conference on Mechanical, Electronics, Computer and Industrial Technology (MECnIT)* (Medan: IEEE), 148–151. doi: 10.1109/MECnIT48290.2020.9166586

Singh, S., Rathore, S., Alfarraj, O., Tolba, A., and Yoon, B. (2022). A framework for privacy-preservation of IoT healthcare data using federated learning and blockchain technology. *Future Gener. Comput. Syst.* 129, 380–388. doi: 10.1016/j.future.2021.11.028

Utama Siahaan, A. P., Elviwani, E., and Oktaviana, B. (2018). "Comparative analysis of RSA and ElGamal cryptographic public-key algorithms," in *Proceedings of the Joint Workshop KO2PI and the 1st International Conference on Advance and Scientific Innovation (ICASI'18)* (Brussels: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 163–172. doi: 10.31227/osf.io/x56df

West, D. B. (2001). *Introduction to Grapheory*, 2nd Edn. London: Pearson.

Yousif, S. F. (2023). Performance comparison between RSA and El-gamal algorithms for speech data encryption and decryption. *Diyala J. Eng. Sci.* 16, 123–137. doi: 10.24237/djes.2023.16112

Yu, Y., Au, M. H., Ateniese, G., Huang, X., Susilo, W., Dai, Y., et al. (2016). Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. *IEEE Trans. Inf. Forensics Secur.* 12, 767–778. doi: 10.1109/TIFS.2016.2615853