Fall 2024 2024-12-06

Final Project Report

Indronil Bhattacharjee *CS 380/525: Intro to Cryptography*

Cryptography Algorithms Implementation: RSA and ElGamal

1 Introduction

Cryptography plays a pivotal role in securing digital communications. Understanding the performance characteristics of cryptographic algorithms like RSA and ElGamal is crucial for their effective application. This report details a comprehensive performance evaluation of both algorithms, offering guidance on their optimal use cases based on empirical data. This proejct evaluates and compares the performance of RSA and ElGamal cryptographic algorithms across different key sizes—512-bit, 1024-bit, and 2048-bit. The study measures encryption time, decryption time, and total operational time to determine the scalability and efficiency of each algorithm. The objective is to provide insights into which algorithm may be more suitable for various security applications, depending on the balance of encryption and decryption needs.

2 Overview

This project involves the implementation of two fundamental public-key cryptography algorithms: RSA and ElGamal. The implementation is carried out in C++, utilizing the Number Theory Library (NTL) to handle large integers and modular arithmetic, which are crucial for the encryption and decryption processes. The project aims to provide a practical understanding of cryptographic principles, encryption efficiency, and secure data handling. This final report expands upon the initial progress report, documenting the implementation, testing, and analysis of the RSA, Digital Signature Algorithm (DSA), and ElGamal cryptographic algorithms. The focus is particularly on comparing the performance and applicability of DSA and ElGamal in various cryptographic scenarios.

3 Project Objectives

- **Implement RSA and ElGamal algorithms** to understand their operations involving key generation, encryption, and decryption.
- **Measure and compare the performance** of both algorithms in terms of execution times for encryption and decryption.
- **Securely manage data** by reading plaintext from and writing ciphertext to files, ensuring data integrity throughout the process.

4 Development Tools and Environment

- **Compiler**: GNU GCC Compiler
- **IDE**: Any preferred IDE that supports C++ (e.g., Visual Studio Code, CLion, or simply a text editor with a command line terminal)

- **Libraries**: NTL (Number Theory Library) for handling large number operations, and GMP (GNU Multiple Precision Arithmetic Library) as it is often used by NTL for better performance.
- **Analysis Metrics**: Encryption time, decryption time, total time
- **Key Sizes Tested**: 512-bit, 1024-bit, 2048-bit.

5 Implementation Details

5.1 File Structure

- **RSA.h / RSA.cpp**: These files contain the class and methods for RSA algorithm operations.
- **ElGamal.h / ElGamal.cpp**: These files handle the ElGamal algorithm functionalities.
- **FileIO.h / FileIO.cpp**: Responsible for file operations, including reading input messages and writing encrypted or decrypted messages.
- **main.cpp**: Drives the program, orchestrating the encryption and decryption processes, handling file operations, and timing the encryption and decryption operations for performance analysis.

5.2 Key Features

- **Modular Design:** Each cryptographic algorithm is encapsulated in its own class, allowing for easy expansion or modification.
- **File-Based Input/Output:** To simulate real-world applications, all messages are read from and written to files, separating the data handling from the core cryptographic operations.
- **Performance Measurement:** Execution times for each operation (encryption and decryption) are measured and output to the console, providing insights into the efficiency of the implemented algorithms.

5.3 Test Procedure

The algorithms were tested across multiple message sizes ranging from 96 bits to 1022 bits for each key size. Performance was assessed in a controlled environment to ensure consistent and accurate measurement of encryption and decryption times.

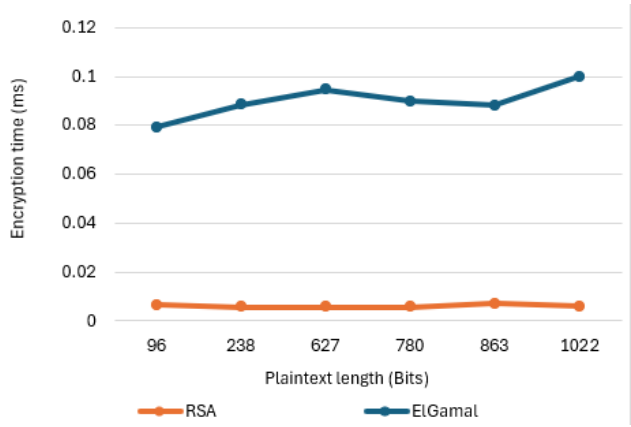
6 Results and Analysis

(a) Results from keylength of 512 bits:

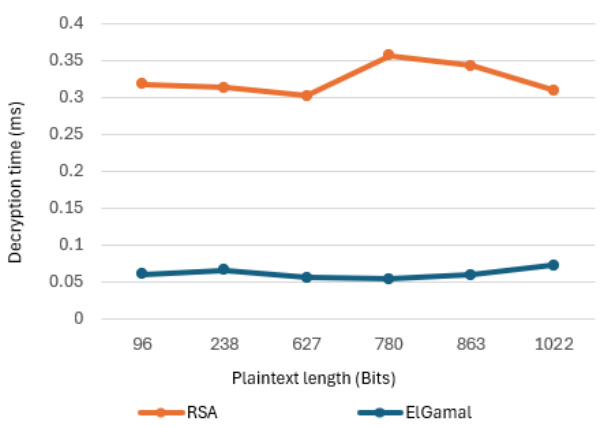
For a key length of 512 bits, RSA shows very consistent encryption times across varying message sizes, but its decryption times tend to increase slightly with larger messages, impacting total times minimally. While ElGamal's encryption times are generally higher than RSA's, its decryption times are consistently lower, resulting in competitive total times.

Table 1: Performance Comparison for RSA and ElGamal (512-bit Key)

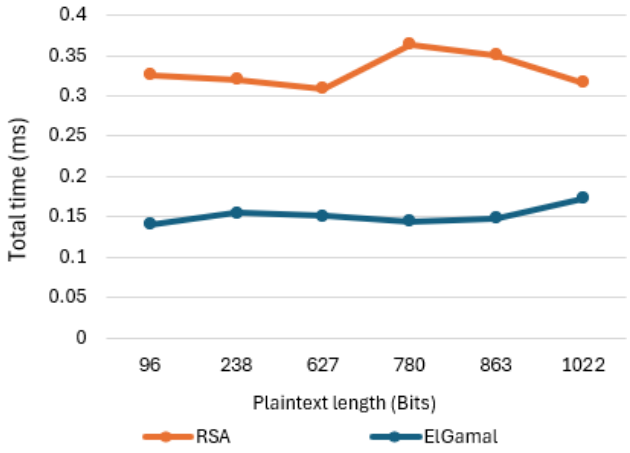
Message Bits	RSA			ElGamal		
	Encryption (ms)	Decryption (ms)	Total Time (ms)	Encryption (ms)	Decryption (ms)	Total Time (ms)
96	0.0067	0.3188	0.3255	0.0791	0.0616	0.1407
238	0.0057	0.3143	0.3200	0.0885	0.0664	0.1548
627	0.0057	0.3030	0.3088	0.0947	0.0562	0.1509
780	0.0057	0.3575	0.3633	0.0897	0.0550	0.1447
863	0.0070	0.3434	0.3504	0.0882	0.0605	0.1487
1022	0.0059	0.3107	0.3167	0.0998	0.0734	0.1733



(a) Plaintext length vs Encryption Time



(b) Plaintext length vs Decryption Time



(c) Plaintext length vs Total Time

Figure 1: RSA and ElGamal Encryption, Decryption and Total Time Comparison for 512-bit Keys

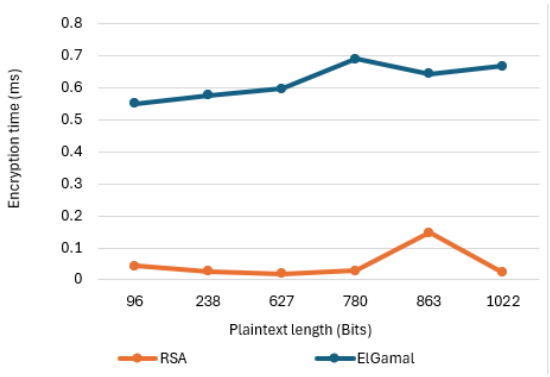
(b) Results from keylength of 1024 bits:

RSA shows relatively stable encryption times that are significantly lower compared to

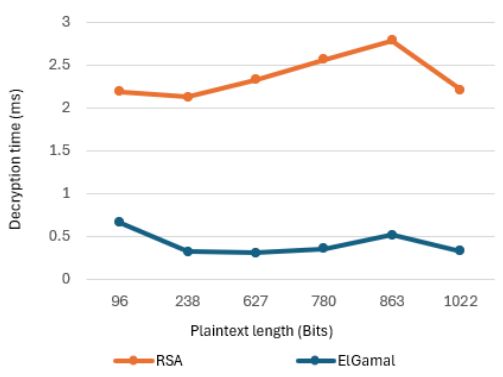
ElGamal. RSA decryption times are substantially higher, leading to a higher overall total time compared to ElGamal in most cases. The consistency in RSA encryption times regardless of message size highlights its efficiency for the encryption phase. However, decryption times increase with the complexity and size of the message. ElGamal demonstrates higher encryption times compared to RSA but compensates with significantly lower decryption times.

Table 2: Performance Comparison for RSA and ElGamal (1024-bit Key)

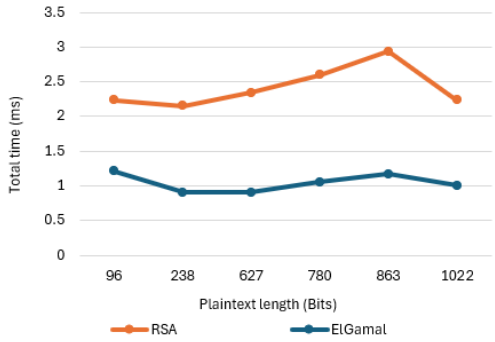
Message Bits	**RSA**			**ElGamal**		
	Encryption (ms)	**Decryption (ms)**	**Total Time (ms)**	**Encryption (ms)**	**Decryption (ms)**	**Total Time (ms)**
96	0.0433	2.1896	2.2329	0.5519	0.6619	1.2138
238	0.0264	2.1247	2.1511	0.5774	0.3242	0.9016
627	0.0198	2.3260	2.3459	0.5972	0.3098	0.9070
780	0.0287	2.5677	2.5965	0.6899	0.3621	1.0520
863	0.1467	2.7891	2.9357	0.6450	0.5224	1.1673
1022	0.0220	2.2131	2.2351	0.6689	0.3351	1.0041



(a) Plaintext length vs Encryption Time



(b) Plaintext length vs Decryption Time

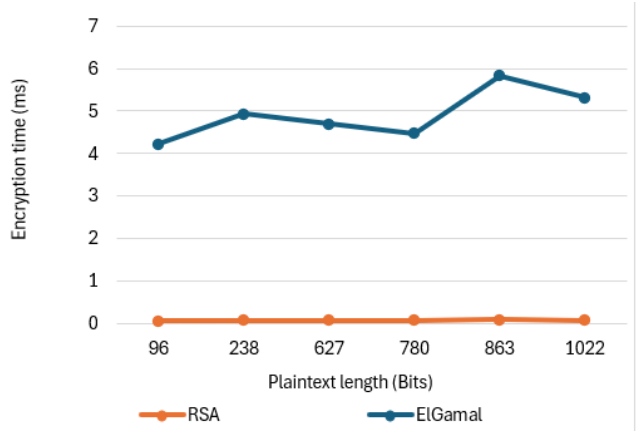


(c) Plaintext length vs Total Time

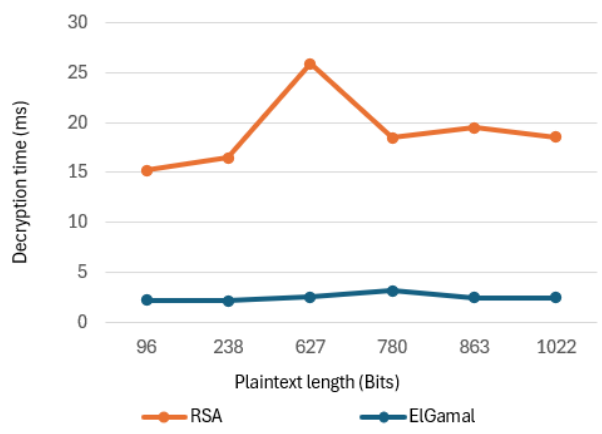
Figure 2: RSA and ElGamal Encryption, Decryption and Total Time Comparison for 1024-bit Keys

Table 3: Performance Comparison for RSA and ElGamal (2048-bit Key)

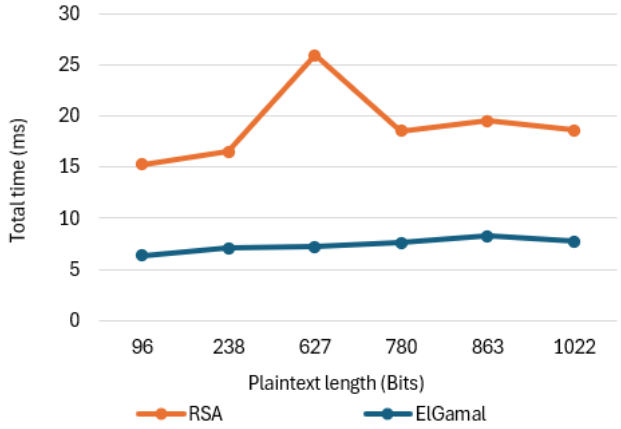
Message Bits	RSA			ElGamal		
	Encryption (ms)	Decryption (ms)	Total Time (ms)	Encryption (ms)	Decryption (ms)	Total Time (ms)
96	0.0661	15.2067	15.2727	4.2314	2.1511	6.3825
238	0.0745	16.4261	16.5006	4.9262	2.1481	7.0743
627	0.0801	25.8577	25.9378	4.6996	2.5072	7.2068
780	0.0785	18.4658	18.5443	4.4823	3.1457	7.6279
863	0.0887	19.4485	19.5373	5.8272	2.4133	8.2404
1022	0.0798	18.5031	18.5830	5.3246	2.4079	7.7325



(a) Plaintext length vs Encryption Time



(b) Plaintext length vs Decryption Time



(c) Plaintext length vs Total Time

Figure 3: RSA and ElGamal Encryption, Decryption and Total Time Comparison for 2048-bit Keys

(c) Results from keylength of 2048 bits:

RSA encryption times are consistently low across all message sizes, showing the algo-

rithm's efficiency in the encryption phase despite the large key size. RSA fecryption times are significantly higher than encryption times, increasing notably with larger message sizes, indicating considerable computational demands during decryption. ElGamal displays higher encryption times compared to RSA, but consistently lower decryption times, suggesting a more balanced performance. The total times are generally lower than RSA, particularly in scenarios with larger messages.

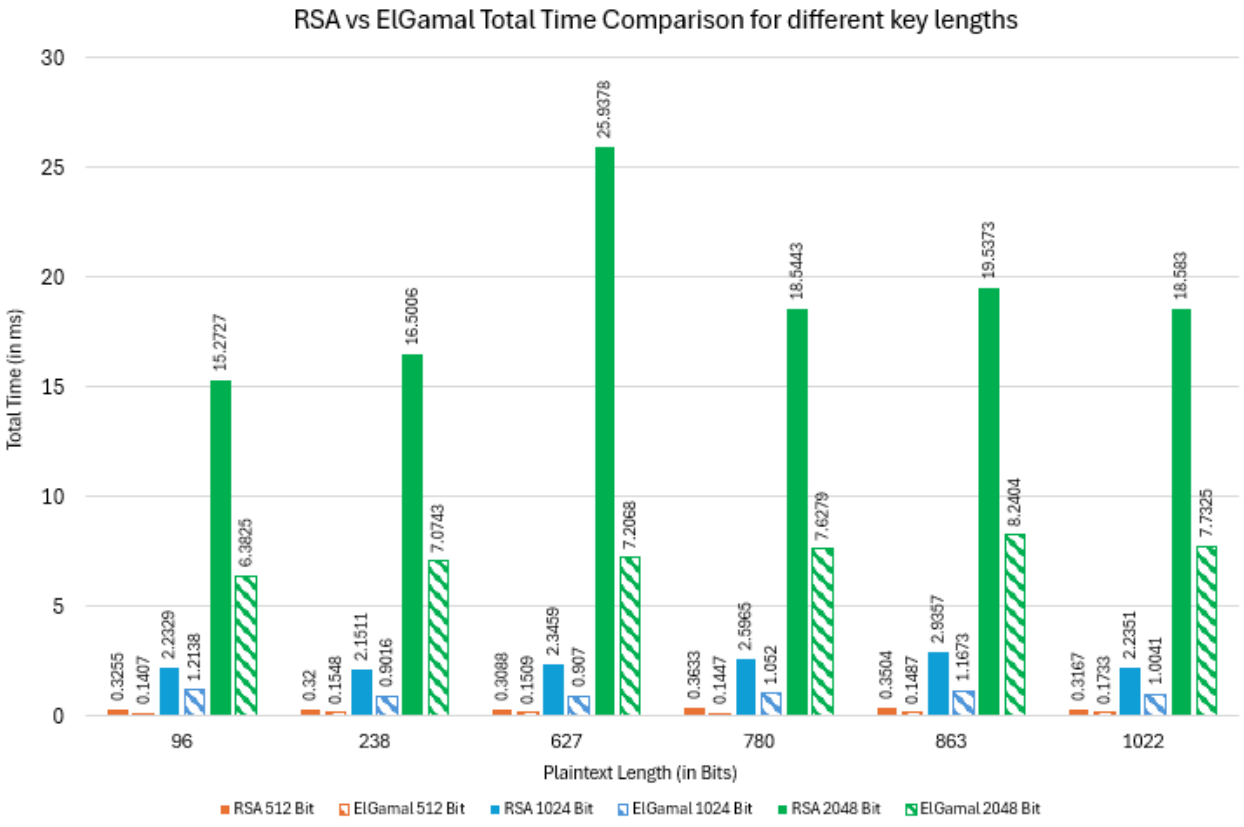


Figure 4: RSA vs ElGamal Time Comparison for Different Key Lengths

(d) Overall Performance Comparison:

Overall Trends:

- RSA Total Time:

  RSA consistently exhibits higher total times as the key size increases. The decryption process dominates RSA's total time, contributing significantly to the increase as the key size grows.

- ElGamal Total Time:

  ElGamal demonstrates more stable total times, with less variation across different message sizes and key lengths. Its total time remains lower than RSA's in most cases for larger key sizes (1024-bit and 2048-bit).

Comparison by Key Size:

- 512-bit Key:

  RSA has marginally higher total times compared to ElGamal, indicating its decryption time is less of a bottleneck at smaller key sizes. ElGamal performs more consistently, with minimal variations in total time across message sizes.

- 1024-bit Key:

  RSA's total times increase notably compared to the 512-bit key, reflecting the growing computational demand of the decryption process. ElGamal retains its efficiency, consistently showing lower total times than RSA, especially for larger message sizes.

- 2048-bit Key:

  The gap between RSA and ElGamal widens significantly at this key size, with RSA's total time increasing steeply due to decryption inefficiencies. ElGamal continues to deliver lower total times, highlighting its scalability and efficiency at higher security levels.

Message Bit Size Impact:

- Both algorithms demonstrate minimal sensitivity to the message bit size for encryption times.
- RSA's decryption process becomes significantly slower as the key size increases, contributing to the sharp rise in total times.
- ElGamal remains relatively unaffected by message size variations, showing stable decryption and total times across different message bit sizes.

Implications for Algorithm Selection

- RSA Suitable for scenarios where encryption speed is prioritized over decryption, especially for smaller key sizes (512-bit). As the key size increases, RSA may become unsuitable for applications requiring frequent or real-time decryption due to its steep decryption time increases.
- ElGamal:

  Ideal for systems requiring balanced encryption and decryption times, especially at larger key sizes. Its stable performance across varying key sizes makes it more versatile and scalable for high-security applications. of libraries for handling complex arithmetic and the

7 Conclusion

This project serves as a practical exploration of RSA and ElGamal encryption algorithms, emphasizing the use nditemizeRSA: importance of performance in cryptographic implementations. The analysis indicates that RSA is suitable for applications where high-speed encryption is critical, and decryption can be deferred or is less frequent. ElGamal, however,

is recommended for applications that require efficient decryption without compromising on encryption, making it ideal for systems with high-security needs where both operations are equally important