TIME AND SPACE COMPLEXITY ANALYSIS OF RSA AND ELGAMAL CRYPTOGRAPHIC ALGORITHMS ON MIXED DATA

M.SC. PROJECT

BY

ADENIYI ABIDEMI EMMANUEL

19PGCD000079

SUPERVISOR

PROFESSOR A. E. OKEYINKA

CO-SUPERVISOR

DR. (MRS.) M. O. ADEBIYI

DEPARTMENT OF COMPUTER SCIENCE,

LANDMARK UNIVERSITY, OMU-ARAN.

MAY, 2021.

TIME AND SPACE COMPLEXITY ANALYSIS OF RSA AND ELGAMAL CRYPTOGRAPHIC ALGORITHMS ON MIXED DATA

ADENIYI ABIDEMI EMMANUEL

(19PGCD000079)

A DISSERTATION SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, COLLEGE OF PURE AND APPLIED SCIENCES, LANDMARK UNIVERSITY, OMU-ARAN, NIGERIA.

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF MASTER OF SCIENCE (MSc.) IN COMPUTER SCIENCE.

MAY, 2021.

DECLARATION

I, **Abidemi Emmanuel ADENIYI** an M.Sc. student in the Department of Computer Science, Landmark University, Omu-Aran, hereby declare that this thesis entitled "***Time and Space Complexity Analysis of Rivest-Shamir-Adleman (RSA) and Elgamal Cryptographic Algorithms on Mixed Data***", submitted by me is based on my original work. Any material(s) obtained from other sources or work done by other persons or institutions have been duly acknowledged.

Abidemi Emmanuel ADENIYI (19PGCD000079)
Student's Full Name and Matriculation Number

_____
Signature & Date

CERTIFICATION

This is to certify that this dissertation has been read and approved as meeting the requirements of the Department of Computer Science, Landmark University, Omu-Aran, Nigeria, for the Award of M.Sc. Degree.

_____	_____
Professor A. E. Okeyinka (Supervisor)	Date
_____	_____
Dr. (Mrs.) M. O. Adebiyi (Co-Supervisor)	Date
_____	_____
Dr. (Mrs.) M. O. Adebiyi (Head of Department)	Date
_____	_____
Professor P. A. Idowu (External Examiner)	Date

DEDICATION

This project is dedicated to the Lord Almighty who made this programme a success, also to my wife Adepeju Adeniyi, my Children, Elizabeth and Emmanuel Adeniyi, and my siblings for their continuous care and support towards me.

ACKNOWLEDGEMENT

My profound gratitude goes to Almighty God the giver of wisdom, for guidance and protection throughout this programme.

I appreciate the efforts of my supervisors Professor A. E. Okeyinka and Dr. (Mrs). M. O. Adebiyi for the time taken to read, correct and proofread this project and also for their parental advice for the success of this project.

I also acknowledge my parents, and family members for the love and support they gave me throughout this programme, May God Bless you. To my siblings, wife and children sister Bola, Brother Victor, Lekan, Ajoke, Adepeju, Elizabeth and Emmanuel Adeniyi I say many thanks to you all.

Also, I will like to appreciate Mr Asani and Dr Gbadamosi for their support. Thanks for your patience with me. I want to appreciate my course mates Jesutofunmi, Pelumi, I say thank you for the friendship bond of good memories. To the School of Postgraduate Landmark University, I say thank you.

Lastly, I want to acknowledge all the lecturers I passed through in the Department of Computer Science and University-Wide Courses Landmark, without them I will not be writing this project, I say a Big Thanks to you all.

ABSTRACT

The complexity study of algorithms, especially computationally intensive ones is of great significance in the field of complexity. Cryptographic algorithms are considered to be computationally intensive because they utilize a substantial number of computational resources, such as CPU memory and processing time. Cryptographic algorithms provide a solution to the security of data transmission whereby ensuring integrity, confidentiality and authentication of any form of data. However, there are still challenges of which cryptographic algorithms are suitable in terms of computation speed and memory usage. Whereas, a good number of research efforts have been put into experimenting on the complexities of the cryptographic algorithm on text, image and audio data, little has been done on video data. In this study, the time and space complexity of RSA and ElGamal cryptographic algorithms on mixed data was carried out. RSA and ElGamal cryptographic algorithms was implemented using C-sharp (C#) programming language to encrypt and decrypt text, image, audio and video dataset. In achieving the objectives of the study, both the implemented algorithms (RSA and ElGamal) are depicted using pseudocodes and flowcharts, while some of the datasets used were sourced from various online repositories. The time complexities of each dataset was obtained using the CPU internal clock while the space usage for each operations on each of the dataset was obtained using the computer internal memory. Tables and graphs was used to carry out the comparative analysis of both algorithms. The time and space complexity of RSA and ElGamal algorithms were experimented on text, image, audio and video dataset. The experimental results revealed that RSA outperformed ElGamal in terms of computational time during encryption of all categories of data. ElGamal outperformed RSA in terms of computational time during

decryption of all categories of data. ElGamal algorithm outperformed RSA in terms of memory usage during encryption of all categories of data while both algorithms used relatively the same amount of space during decryption of all categories of data used. Based on the comparative analysis of the time and space complexity on both RSA and ElGamal algorithms, it was discovered that RSA is a better algorithm when it comes to time complexity, that is, RSA can be said to be a time-efficient algorithm. ElGamal algorithm performed better than RSA in the memory usage aspect, therefore the ElGamal algorithm is said to be a memory-efficient algorithm. Therefore, this study hereby recommend that other measurement metrics may be used to compare both algorithms in future works.

TABLE OF CONTENTS

CHAPTER ONE

1.0. INTRODUCTION

CHAPTER TWO

2.0. REVIEW OF LITERATURE

CHAPTER THREE

3.0. METHODOLOGY

CHAPTER FOUR

4.0. RESULTS AND DISCUSSIONS OF FINDINGS

CHAPTER FIVE

5.0. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Background of the Problem

Data security is the science and study of strategies of securing data from unauthorized disclosure and alteration in computer and communication systems (Mondal *et. al.*, 2020). With the exponential increase in the volume of data communication and transfer, and with the proliferation, diversification and intensity of malicious activities, the need to ensure and sometimes enforce the security of data has become even more urgent and critical (Hong, 2012; Asani *et al.*, 2018; Statistica, 2018). Consequently, research activities in Data security have evolved rapidly and have produced exciting developments in related application fields of computer security such as cryptography. Cryptography is a technique of preventing illegitimate access to information and data (Singanjude and Dalvi, 2020). Cryptography is the act of securing data and information through encryption and decryption. It is always synonymous with transforming plain text (ordinary text, also referred to as simple text) into cipher text (a method called encryption), then back again as plain text (known as decryption) (Aldullah, 2017). Current cryptography is concerned with the following goals:

a) secrecy (the data cannot be understood by any person for whom it was not deliberate)
b) Integrity (the data cannot be changed in storage or transfer between source and proposed receiver without the amendment being noticed)

c) Non-repudiation (the composer of the content cannot dispute, at a later point, his intention to create or distribute the information)

d) Authentication (the transmitter and the recipient can affirm the presence of each other and the source of the data).

The primary application of cryptography is encryption; it helps data to ensure its confidentiality by producing incomprehensible output (Mondal *et al.*, 2020). Various cryptographic methods are used to ensure data security, each approach having unique encryption and decryption processes that only enables approved individuals to read the encrypted message. Figure 1 displays the process of encryption and decryption; a plaintext, P is encrypted through a cypher process E, with an input Key, K, to generate ciphertext, C as the output. This relation can be denoted as $C = E(K, P)$. The inverse cypher process of decryption D converts the cyphertext C back to plaintext. This relation is depicted as $D(K, C)$.

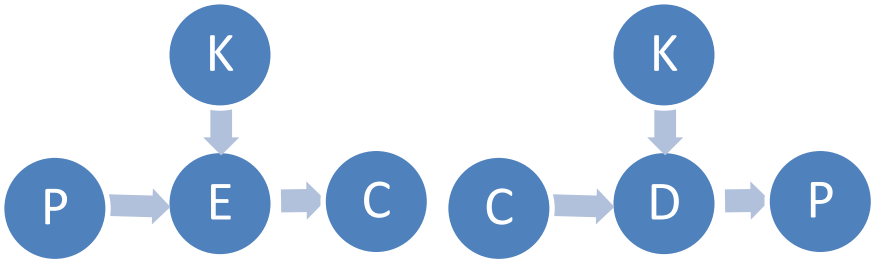

Figure. 1. Encryption and Decryption Process

Cryptography can be categorized as symmetric or asymmetric key. Symmetric cryptography uses single key encryptions. The same key is used for both encryption and decryption. The strength of symmetric key encryption depends on the secrecy of encryption

and decryption keys. Asymmetric cryptography on the other hand demands different keys to be used for the encryption and decryption process. Asymmetric encryption uses hidden (private) and distributed (public) keys to solve computationally complex problems. Some popular Symmetric cryptography includes the Data Encryption Standard (DES), Advanced Encryption Standard (AES). Some Asymmetric techniques include the Diffie Hellman Algorithm, Rivest–Shamir–Adleman (RSA), Elgamal among others (Suguna, Dhanakoti, and Manjupriya, 2016). The RSA and ElGamal are classic asymmetric cryptographic algorithms that provide security for data. RSA has dynamic keys, which can vary depending on the key generation (Shah, *et. al.*, 2018). The El-Gamal algorithm is a public-key cryptosystem that was developed based on the discrete logarithm problem. The algorithm provides both encryption and signature algorithms (Rashmi and Shiv, 2012).

The intricacy of an algorithm is the measure that evaluates the number of resources, such as time, space, energy and so on, that the algorithm requires. It is a measure of how 'good' the algorithm is at solving the problem. It can also be described as the efficiency of the algorithm in terms of the amount of data the algorithm must process (Aljawarneh, and Yassein, 2017). Typically, the complexity of an algorithm is a function that maps the input length/size to the number of main stages (time complexity) or specific storage position (space complexity) (Kumar *et. al.*, 2016). Complexity can be described as the highest limit of primitive operations a function can perform. There are two main complexity metrics for the reliability of an algorithm, Time and Space Complexity (Kumar *et. al.*, 2016). The time complexity of an algorithm is a function that defines the length of time the algorithm takes when it comes to the amount of input to the algorithm. Time may represent the amount of memory accesses that have been made, the numbers of measurements between integers,

the number of times any inner iterations are run, and the amount of real time the algorithm would take. The use of time complexity makes it easy to estimate the running time of a program. Performing an accurate calculation of a program's operation time is a very labour-intensive process. It depends on the compiler and the type of computer or speed of the processor. Best-case, worst-case and average-case are the three ways of measuring time complexity. The space complexity of an algorithm is a function that defines the quantity of memory (space) an algorithm uses in terms of the amount of input to the algorithm. Space complexity includes all the variables, both global and local. Some algorithms are more efficient than others, so having metrics for comparing their efficiency will be necessary; therefore, this study aims to determine the time and space complexity of RSA and Elgamal cryptographic algorithms on mixed data.

1.2 Statement of the Problem

Cryptographic algorithms are known to be computationally intensive; they consume a substantial quantity of computing resources such as CPU time, and memory. Hence, a good number of research efforts have been put into experimenting on the complexity of cryptographic algorithms on text, image, and audio data without much effort on video data. Also previous research dwells more on time complexity of cryptographic algorithms for text, image and audio dataset with limited file size without putting into consideration the memory used by the algorithm (Dindayal and Dilip, 2018; Kyaw, Kyaw and Nya, 2019; Zarni *et. al.,* 2019). The amount of resources (time and space) to be used by cryptographic algorithms when it comes to encrypting and decrypting mixed data cannot be predetermined as the cryptographic algorithms speed need to be measured in terms of either fast, slow or moderate scenario. Additional, literature on the complexity of mixed data is

scarce. Determining the time and space complexities of cryptographic techniques helps to improve infrastructure design, decision making, and allocation of resources. A study on the complexities of mixed data vis-à-vis time and space has therefore become imperative. Hence, this study is an attempt to consider video data in addition to text, audio, and image data. Furthermore, this study has considered the performance of the RSA and ElGamal cryptographic algorithms on each of the mixed data.

1.3 Justification for the study

The cryptographic algorithm can be categorized into two basic forms, which are symmetric and asymmetric algorithms. But asymmetric algorithms are associated with complexity challenges due to the two keys used for encryption and decryption. Therefore, there is a need to study these algorithms in terms of their time and space performance. Analysis of Cryptographic algorithms are an important part of computational complexity theory, which provides theoretical estimation of the required resources of an algorithm to solve a specific computational problem. The running time of a cryptographic algorithm is stated as a function relating the input length to the number of steps (time complexity) or storage locations (space complexity), thus, makes it crucial to study the performance of the cryptographic algorithms in terms of time and space consumption. This research implements two popular asymmetric cryptographic algorithms to determine their time and space complexity on mixed data. The outcome of this study is intended to assist the expertise in the field of computational complexity to identify optimal cryptographic algorithms to solve a computational problem.

1.4 Aim and Objectives

This study aims to determine the space and time complexity of RSA and Elgamal cryptographic algorithms on mixed data. The specific objectives are to:

1. Implement both the RSA and Elgamal algorithms in encrypting and decrypting mixed data using C-sharp (C#) programing language.
2. Compute the time and space complexities of the two algorithms using CPU internet clock and computer primary memory.
3. Determine the behaviour of the algorithms on the mixed data.

1.5 Research Methodology

The RSA and Elgamal are classic cryptographic techniques, known for their computational abilities. The objective is to encrypt and decrypt mixed data using both the RSA and Elgamal and compare their performances based on their time and space complexities. In achieving the first objective, both the RSA and Elgamal algorithms are depicted using pseudocodes and flowcharts while the algorithms are implemented in C# programming environment. The second objective was achieved using the CPU internal clock to compute the encryption time and decryption time of both RSA and ElGamal algorithms while Computer internal memory was used to compute the encryption space usage and decryption space usage of both RSA and ElGamal algorithms. The third objective was achieved using various tables and graphs to analyze and compare the time and space usage of both RSA and ElGamal algorithm on text, image, audio and video dataset. Some datasets shall be sourced from online repositories such as Lipsum, datahub and Kaggle, others shall be generated randomly using a random number generator function. The study chose C#

programming language to implement both algorithms because C# is particularly strong at building windows desktop applications.

1.6 Scope of the Study

This study is limited only to the analysis of time and space complexities of the RSA and ElGamal cryptographic algorithms.

1.7 Significance of the Study

In the real world, users are constrained by the primary storage of the devices on which they want to execute their programs. This is where space complexity comes into play, so we never want to run a process or procedure that takes up more space than the machine has available at any given time. On the other hand, we don't want our procedure to be so lengthy that our processes get congested and get slower. Thus, this study identifies the suitable algorithms for obtaining optimal time and space complexity in a real world application and also compares their performance on text, image, audio and video datasets. Also, this will help to make a precise decision when it comes to the choice of algorithm to be used in solving a computational problem.

1.8 Organization of Dissertation

This project is organized into five chapters. The remaining part of the dissertation is as follows: The second chapter covers the review of literature covering already existing research on the RSA and ElGamal cryptographic algorithms. The third chapter covers the research methodology, the design of the RSA cryptographic algorithm, the design of the ElGamal cryptographic algorithm, the terminology of time and space of an algorithm. The

fourth chapter presents a discussion of the results obtained. The fifth, which is the final chapter discusses the summary of the findings, conclusions and recommendations.

CHAPTER TWO

REVIEW OF LITERATURE

2.1 Conceptual Issues

A large volume of data is accessed daily and transferred from one person to another for real-life applications. When the information is transmitted from the sender to the recipient, it can be eavesdropped by the attacker and thus presents a continuous threat to the secrecy or security of the results. The famous technique to maintain confidentiality of such a large amount of data is cryptography (Preeti *et. al.*, 2018).

Cryptography is an important part of the new field of information technology, making the digital world a safer environment. Cryptography is a method that makes knowledge unreadable to an unauthorized individual (Priyadarshini, *et. al.,* 2015). However, offering confidentiality of legitimate customers. Various cryptographic algorithms can be used. Ideally, a consumer requires a low-cost, high-performance cryptographic algorithm. In fact, however, there is no such algorithm as a one-stop solution. There are some algorithms with a cost-effective trade-off (Haitner and Vadhan, 2017). For instance, a banking firm wants the highest degree of security at a significant expense, and a game software that sends a player template for analytics does not care much about privacy, but needs to be fast and cost-effective.

The Cryptographic method uses an algorithm and a hidden (key) value. The key can be the same with all encryption-decryption procedures or it can be different for each type of encrypted message used (Viney and Sandeep, 2016). Based on the type of key utilized,

cryptography methods can be divided into two distinct categories: symmetrical (secret) and asymmetrical (public) key encryption.

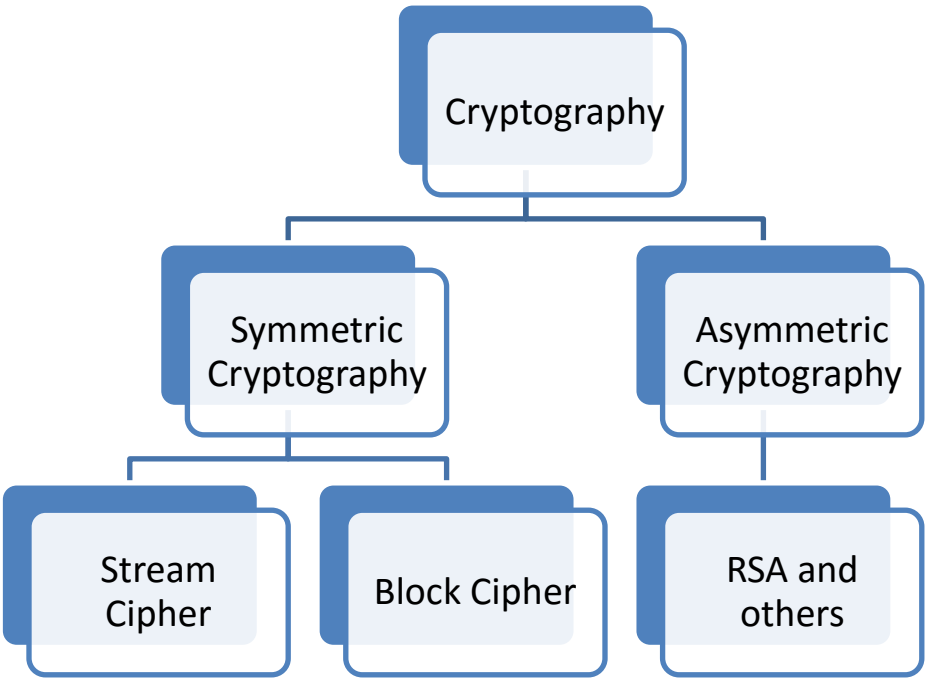

Figure 2.1: Classification of Cryptography

A system which handles encryption and decryption is labeled a cryptosystem. The difficulty of the encryption method lies on the algorithm used for encryption, the program used and the key used in the algorithm for encrypting or decrypting data (Abdullah, 2017).

Based on the number of keys required, encryption techniques may be categorized as asymmetrical algorithms (public key) and symmetrical algorithms (secret key). In Symmetric data cryptography or hidden key encryption, the same key is used by transmitter and recipient. Data Encryption Standard (DES), 3DES, and Advanced Encryption Standard (AES) are instances of a symmetric key encryption technique (Suguna, Dhanakoti and Manjupriya, 2016). Asymmetric key encryption uses two separate keys (public and private keys) for encoding and decoding. The public key is used for encryption and the private key

is used for decryption purposes. Instances of asymmetric key algorithms are Rivest-Shamir-Adelman (RSA) and Elliptic Curve Cryptosystem (ECC).

2.1.1 Symmetric Cryptography

Symmetric cryptography is known as secret key encryption which uses the same key for encryption and decryption. The symmetrical key (secret key) encryption requires a similar key to encrypt and decode a document. Encryption and decryption keys are kept confidential and are accessible only to registered senders and receivers who wish to connect. The strength of symmetric key cryptography depends on the privacy of senders and recipients keys. Symmetric key encryption algorithms can be grouped into a block and stream cipher based on document bit clustering (Mushtaq *et. al.*, 2017; Gupta, Saluja, and Tiwari, 2018). In a block cypher, a set of texts of a fixed size (a block) is encoded at once and sent to the recipient. In addition, the block cipher can be further separated into binary and non-binary block ciphers depending on the final effects of the code, keys and cipher text. The message bit size for the binary block cypher is 64, 128, 192, and 256.

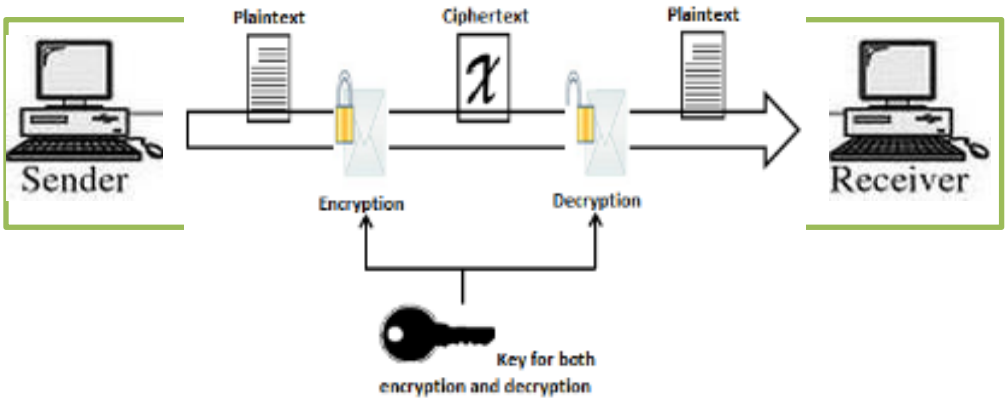

Figure 2.2: Overview of Symmetric Encryption Process

There are quite number of symmetric algorithms like Blowfish, DES, Triple DES, AES among others. Each algorithm uses a different method of encrypting and decrypting data,

each of which often encrypts and decrypts set data sizes as blocks and fixed key sizes (Stallings, 2014; Panda, 2017). These algorithms can only permit the Common letter, special characters and numerical values to be entered as plaintext.

A. Data Encryption Standard Symmetric Algorithm

Data Encryption Standard (DES) was the very first cryptography protocols to be established by the National Institute of Standards and Technology (NIST). It was created in 1974 by the IBM group and accepted as a global practice in 1997 (Rabah, 2005). DES is a common system for the protection of confidential and unlabeled data (Suo *et. al*., 2012). Over the last years, DES has been very common in industrial, armed forces and other domains (Rabah, 2005).

Originally, DES uses 64 bits as input block, a 56-bit key and the outstanding 8 bits for odd parity checks. Numerous threats and techniques have found the vulnerabilities of DES from the time, which rendered it an unsafe cypher block (Oduyiga, 2018). The decryption of the cypher data is done with the reversed steps of the encryption mechanism with the same key. Using identical key for the encoding and the decoding mechanisms gives the attackers of the cypher data a big opportunity to attack the encryption system, which forms an important weakness of this algorithm.

B. Triple Data Encryption Standard (3DES) Symmetric Algorithm

Brute-force assaults became more possible given the abundance of rising computing resources. The initial 56-bit key size of the DES cypher was defective in cryptography, 3DES is an upgrade to the DES which was produced in 1998; it is 64-bit block length, expanding the block length to 192 bits. This encryption technique is similar to the previous DES but applied 3 times, so 3DES is 3 times comparatively slow compared to DES, low power consumption and throughput performance but more stable (Thambiraja, 2012).

The message in 3DES is encoded with the first key, decoded with the second key, and encoded with the third key. Decryption is done in reversal and also in the encryption procedure. This algorithm defines the three primary sorting steps of the package (Singh, 2013):

Step 1: The chosen level consists of three mutually independent keys (K1 to K2 to K3 to K1). Gives a 3 x 56 = 168 bit keyspace.

Step 2: Using two mutually exclusive keys and a final key that is the same as the first key (K1 name K2 and K3 = K1). This offers a main area of 2×56 = 112 bits.

Step 3: Is a key package that uses the same parameters (K1 = K2 = K3). It's the same DES algorithm.

C. Blowfish Symmetric Algorithm

This recently designed encryption method is a regular symmetric block cypher. It should be easy to encrypt data with a 32-bit processor at a clock speed of 18 cycles per byte. A

small memory capacity of 5K less can be used. This norm has a basic framework that is easy to enforce and use. The strength of the norm could also be easily measured. The length of the Blowfish standard key is variable and can be 448 bit long (Singh and Singh, 2013). This offers the operator of this method greater stability, but the operator should recognize speed concerns when implementing larger key length values with a Block size of 64bits at defaults.

Blowfish is a 64-bit block cypher that can be taken from 32-bit to 448-bit variable-length keys; 128-bit standard bit. Built to substitute the DES cryptographic algorithms, so blowfish is a better cryptographic algorithm than other cryptographic algorithms described earlier. Blowfish is license-free and easily accessible for all applications (Rahim, 2017). Usually, data encryption in a blowfish algorithm is achieved with a 16-round. Each round includes two parts: dependent variable substitution cipher and key-and data-dependent substitution; all procedures are XORs and additions to 32-bit words; the only operation introduced is fourth ordered array data using S-boxes for each round.

D. Advanced Encryption Standard (AES) Symmetric Algorithm

AES is planned to be a modern encryption standard endorsed by NIST to succeed DES. AES is a block cypher named Rijndael, created in 1997 by Joan and Rijmen. It has a separate key size of 128, 192, or 256 bits; the standard is 256. The cryptography method has varying circles based on key duration, it requires 10 sequences for key size 128, 12 rounds for key size 192 and 14 rounds for key size 256. In the AES cryptography algorithm; the sequence phase is done recursively several times depending on the key length. The decryption algorithm method is the same as the encryption algorithm phase, but with each transition stage inversely.

The AES algorithm has been developed to solve the shortcomings of the Data Encryption Standard. It is a regular block cypher with a symmetric key solution (Talirongan, 2018). It offers a superior encryption option than 3DES with an increased level of protection and improved performance quality. AES is based on the theory of architecture identified as the substitution-the permutation framework. And AES is running on a 4x4 column-major byte order matrix. The key size used by the AES cypher determines the number of transformation round repeats. For 128-bit keys, 10 replicate cycles were required, 12 replicate cycles were needed for 192-bit keys and 14 replicate cycles were needed for 256-bit keys. Each round has specific operations, such as SubBytes, ShiftRows, MixColumns and AddRoundKey (Lee, *et. al.,* 2018).

2.1.2 Asymmetric Cryptography

Asymmetric cryptography is an encryption category where a hidden key can be split into two sections, a public key and a private key. With asymmetrical encryption, its secret key can only decrypt a message encrypted with a public key, and inversely (Mavroeidis, 2018). Asymmetric encryption fixes the issue of sharing without encrypted transmission by encouraging parties to share their public keys and encrypt data utilizing mathematical equations so that the message cannot be decrypted by eaves-dropper.

Asymmetric encryption has been created to solve the challenges of key exchange. The private key of asymmetric is never shared; it is kept secret and is used only by its owner while the public key is made available to anyone who wants to use it, this is the main strength of asymmetric encryption that makes it to be more secure. RSA, Elgamal, Diffie-Hellman, DSA among others are major asymmetric algorithms used for data security.

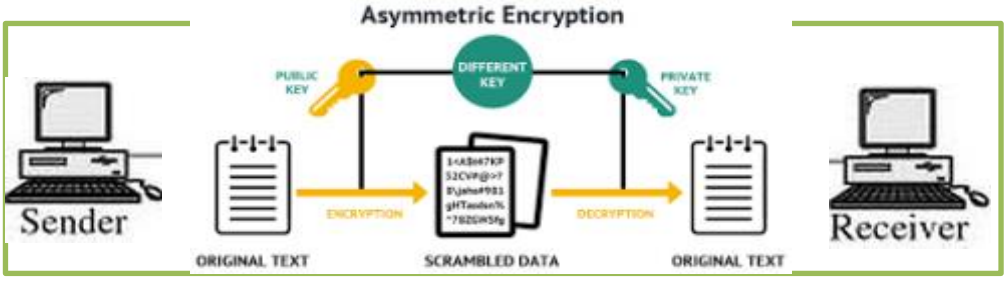

Figure 2.3: Overview of Asymmetric Encryption process.

The encryption process of asymmetric is as follows:

1 The sender of a message uses the intended recipient's public key, which is freely available, to encrypt a message.

2 The recipient decrypts the message using his or her private key. Only the private key associated with the public key that encrypted it can be used to decrypt the message.

2.2 RSA Cryptographic Algorithm

In 1977 Rivest, Shamir and Adelman of the Massachusetts Institute of Technology proposed the RSA algorithm which is an asymmetric cryptographic algorithm (Singh, 2013). It is one of the best techniques of public-key cryptosystems for key swap and encryption of blocks of data. It requires two keys, one is the public key for encryption and the other is the private key to decrypt a message. The key size of RSA is 1024 to 4096 bits. It uses two prime numbers to generate a public-key and a private key. The sender encrypts the message using the receiver’s public key and on the decryption side, the message is decrypted using the receiver’s private key, because of this attribute the Rivest, Shamir and Adleman have been the widely used algorithm (Kyaw, Kyaw and Nay, 2019).

The RSA algorithm involves three steps which include Key generation, encryption and decryption. The RSA key was made as strong as possible to increase ciphertext robustness. A key that is difficult to crack and requires a lot of time certainly has higher robustness. Two initial variables that are very important in building RSA keys are prime number p and prime number q. RSA algorithm, choosing prime number p and prime number q can affect the key and also the ciphertext.

2.3 Elgamal Cryptographic Algorithm

The ElGamal encryption method is an asymmetric key encryption technique for public-key cryptography based on the Diffie–Hellman key exchange. Taher Elgamal presented it in 1985. ElGamal cryptography is often used in unlimited GNU Privacy Shield, latest models of PGP, and other cryptography. Digital Signature Algorithm (DSA) is a modified version of the ElGamal signature scheme that should not be mistaken with ElGamal cryptography (Rashmi and Shiv, 2012). Encryption and decryption for ElGamal was established on the complexity of the discrete algorithm problem directed to increase numbers of big powers. However, it is much harder to do the reverse calculation of the discrete logarithm (Rashmi and Shiv, 2012).

ElGamal cryptography is probabilistic, which means that a specific plaintext can be encoded to several potential ciphertexts, with the effect that the standard ElGamal encryption generates a 2:1 extension in length from plaintext to ciphertext. Encoding under ElGamal needs two exponentiations; thus, these exponentiations are independent of the text and can be calculated long in advance if required. Decryption involves one

exponentiation and one reverse category calculation, which can conveniently be merged into one exponentiation.

2.4 The Elliptic Curve Cryptographic Algorithm

The Elliptic Curve Cryptography (ECC) is an asymmetric cryptographic technique based on an elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys (Johnson *et. al.,* 2001; Mahmood *et. al.,* 2018). It offered a new level of security for Public Key Cryptographic (PKC) which provides both encryption and digital signature services. ECC certificate allows the Key size to stay compact while offering a stronger degree of protection. Its key generation process is distinct from RSA and Elgamal, though focusing on the use of a shared encryption key and a private decryption key. It was designed for devices with limited computing power and/or memory, such as smart cards and PDAs (Kessler, 2010).

2.5 Diffie–Hellman Cryptographic Algorithm

The Diffie-Hellman algorithm was one of the earliest known asymmetric key implementations that are typically used to swap keys. While symmetric key algorithms are fast and reliable, the exchanging of keys is often a concern. The user have to work out a way to get a secret key to all the networks that the Diffie-Hellman algorithm is helping with (Costello *et. al.,* 2016). The Diffie-Hellman algorithm would be used to set up a stable communications network. The device uses this mechanism to share a private key. This

secret key is then used to make symmetrical encryption of the two schemes. The Diffie-Hellman key exchange, also known as the exponential public key, is a digital encoding system that uses digits lifted to particular powers to create decryption keys centered on elements that are never explicitly distributed, making the role of a possible code breaker numerically daunting (Suganya and Ramya, 2014). The Diffie–Hellman public key system helps two people who have no previous knowledge of each other to mutually create a mutual private key over an unprotected communication medium (Boni *et. al.*, 2015). This key will then be used to encode future communications using a symmetric key cipher.

The system was first reported by Whitfield Diffie and Martin Hellman in 1976, whereas it had been independently conceived several years ago by Malcolm J. Williamson, the British Signals Intelligence Service, but remained secret (Kaur and Kaur, 2017). In 2002, Hellman proposed that the technique be renamed Diffie–Hellman–Merkle Key Exchange in acknowledgment of Ralph Merkle's contribution to the development of public-key encryption (Biswas, 2012; Suganya and Ramya, 2014). Although Diffie–Hellman key agreement itself is an anonymous (non-authenticated) key-agreement protocol, it provides the basis for a variety of authenticated protocols and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cypher suite). The method was followed shortly afterwards by RSA and the implementation of public-key cryptography using asymmetric algorithms. Diffie–Hellman establishes a shared secret that can be used for secret communications by exchanging data over a public network (Kaur and Kaur, 2017; Vasundhara, 2017).

To introduce Diffie-Hellman, the two end-users Alice and Bob, when talking over a medium they consider to be personal, settle on positive integer values p and q, so that p is

a prime number and q is a p generator. Generator q is a variable that, when lifted to positive full-number powers less than p, never generates the same outcome for any two such whole values. The p value can be high, but the q value is typically small.

2.6 Review of Methodological Approach

Recently, several studies have examined the performance evaluation of RSA and Elgamal cryptographic algorithms on text data, audio and image data. This section gives a detailed summary of studies conducted concerning the time and space complexity of RSA and Elgamal cryptographic algorithms. The methods applied are reviewed based on their relatedness to this study.

Kayalvizhi, Vijayalakshmi and Vaidehi (2010) worked on the performance and comparison of RSA and ElGamal cryptography algorithms by assessing their power productivity and network lifespan. The researcher used a group-based wireless network topology situation with NS2 to investigate the performance of the collection. The information was scrambled at the foundation node and the ciphertext was sent to the target node through the cluster heads. The power consumption of RSA and ElGamal algorithms was analyzed and showed that RSA uses less resources and thus improves the life of the network relative to ElGamal. From the findings, the RSA algorithm has improved support for wireless communications and absorbs 14.5 per cent less power than the ElGamal algorithm. The study was limited to 10 sensor nodes of the wireless network.

Shashi and Rajan (2011) worked on performance comparison for proposed approach of symmetric and asymmetric encryption algorithms, i.e. AES, DES and RSA, in terms of processing time, memory consumption and processing bytes of various file sizes. The results of the analyses revealed that, among other things, the DES algorithm provides better performance of encryption time, AES had the lowest memory consumption and RSA algorithm produced the minimum output disk. The study was limited to text data only.

Farah *et. al.,* (2012) worked on the performance evaluation of asymmetric encryption. The study presented the evaluation of RSA, Elgamal and Pallier in terms of encryption and decryption processing time, throughput, encrypted data size and decrypted data size. The study used 68KB, 105KB, 124KB and 235KB text file sizes for the analysis. The result showed that RSA performs better in terms of encryption time, Elgamal performs better in terms of decryption time. It was observed also that RSA is faster in the encryption process than all those in terms of its throughput. The RSA needs a minimum volume of storage capacity. RSA's average result is higher than Elgamal and Paillier in terms of the criteria employed. The study is limited to the four text file size given above.

Arora *et. al.*, (2013), suggested the introduction of cryptographic algorithms in Java programming language to create a safe cloud data by using diverse features to make a distinction between symmetric and asymmetric techniques such as AES, DES, Blowfish, and RSA. They reported that AES used the minimum time to execute cloud data. Blowfish used less memory consumption. DES has invested the least amount of time in cryptography. RSA has spent the most time in cryptography and the highest memory capacity.

Ankush, *et.al.,* (2014), discussed the security and time-consuming analysis of RSA and Elgamal algorithms. The study implements RSA and Elgamal Algorithms on JCrypt Tool 1.0.0 to determine the security level of the two algorithms and time consumption for encryption and decryption. The result obtained from the plaintext shows that the Elgamal algorithm is more secure compared to the RSA algorithm because it generates more complex ciphertext, meanwhile, RSA performs better in terms of encryption and decryption time. The study was limited to the use of characters entered by users.

Shaify and Meenakshi (2014), worked on the performance evaluation of different symmetric encryption algorithms DES, 3DES, AES for different parameters and data types like text files and image to analyze their encryption and decryption time, throughput and memory utilization. They reported that AES consume less time than other algorithms in terms of encryption and decryption time as the number of rounds is comparatively less in the case of AES while 3DES has more encryption and decryption time as to apply the algorithm three times. The throughput varies inversely to the encryption or decryption time, thus AES has more and 3DES has less throughput than the other algorithms. AES takes more memory while DES utilizes less memory than the other symmetric key encryption algorithms.

Tin and Su (2014), worked on the performance comparison of RSA and Elgamal for audio protection, based on the period of operation. They used an audio (.mp3) file type with varying file sizes but the same data format. The work was projected to apply confidentiality, secrecy in the real data communication environment. The performance analysis of the two algorithms used shows clearly that the RSA algorithm is significantly faster than the Elgamal algorithm for encryption and decryption time of audio data. The

work was limited to the single audio file format (.mp3) and only the execution time was analyzed.

Boni *et. al.,* (2015), suggested a novel methodology to enhance the Diffie-Hellman algorithm, thereby involving complicated calculations that increase the computational complexity when producing shared keys called the Multipliers Key Exchange Technique. They reported that the Multiplicative Key Exchange is better than the Diffie-Hellman algorithm in terms of execution time, thus need few computations compared with the Diffie-Hellman algorithm the new method is used when keys are to be generated frequently and faster instead of protection where systems are not complicated or have a reduced setup.

Okeyinka (2015) worked on RSA and ElGamal Algorithms computing speeds performance for securing, confidentiality and authentication of Text Data. The researcher used a computer internal clock for both RSA and Elgamal to compare and determine the execution times of each input text data and which one of the two methods is more computationally effective. The implementation of the work was checked with text details in different sizes. The result showed that RSA is more computationally efficient than Elgamal which makes it to perform better than Elgamal. However, the limitation of this research work is that text data was used with limited character size.

Kumar *et. al.,* (2016), proposed an evaluation of the efficiency of some cryptographic methods DES, AES, and Blowfish. They reported that Blowfish produces better performance in several block cypher forms with minimal weaknesses. AES demonstrated low efficiency compared to other methods. In particular, symmetric encryption techniques are quicker than the asymmetric techniques, but there was only one fragile fact where it

exchanged the key with other parties interested in the procedure. Asymmetric encryption has the power that two separate keys are used, but more computation time is taken than symmetric encryption.

Muneer *et. al.,* (2017) presented a systematic analysis of Symmetric key and Asymmetric key cryptographic protocols that improved information security in a cloud storage environment. AES, DES, 3DES and Blowfish were explored for symmetric key encryption and RSA, DSA, Diffie-Hellman and Elliptic Curve for asymmetric cryptography. They reported that the performance of the different algorithms is influenced by the discrepancy factor. They observed that it is crucial to provide effective, reliable and high-security algorithms that are appropriate for all huge datasets and that pace and protection are the most essential in evaluating the performance of any cryptographic algorithms.

Andysah, Elviwani, and Boni (2018) studied comparative analysis of RSA and Elgamal public key. The work focused on the bit length of the public key which is the strength of the two cryptographic algorithms. The RSA used the factorization process to determine the value of the two large prime numbers as the degree of difficulty while Elgamal uses discrete logarithms for its calculations. They used key generation techniques to compare which of the two algorithms performs better during the encryption and decryption process. At the process of main production, RSA generates six variables (P, Q, N, F, E, D). Parameters 'N' and 'E' are keys used for encoding, and 'N' and 'D' are keys used for decryption. At the process of main generation, ElGamal generates four variables (P, G, X, Y). During the encryption scheme, parameters "P," "G" and "Y" are used, while parameters "P" and "X" are used during the decryption process. In the main generation, RSA and ElGamal have essentially the same duration. It doesn't take long to produce a key for a quantity that isn't

that high. RSA and ElGamal would take longer to produce 2048 bit keys since the equation results in a must-have compact phrase. The work was limited to the key generation time of RSA and Elgamal cryptographic algorithm.

Dindayal and Dilip (2018) the authors presented in their paper, the performance study and analysis of RSA and Elliptic Curve cryptography. The study suggests the supremacy between the two algorithms. The study used different key sizes for RSA and ECC to analyze the execution time and the security level. 1024, 2048 and 3072-bit key sizes were used for RSA while 160, 224 and 256-bit key sizes were used for ECC. The result of the experiment shows that ECC is better than RSA and best suits memory-constrained devices, as an ECC-based cryptosystem requires fewer resources than an RSA-based cryptosystem. The performance analysis in terms of encryption and decryption time and operational efficiency and security shows that ECC outperforms RSA. The study was limited to the use of certain key size for the analysis.

Nureni and Sayyidina (2018), evaluated the comparative analysis of encryption algorithms. AES, RSA and DES encryption algorithms were implemented on audio and video files of different sizes to determine the encryption and decryption time. The result showed that AES is best in terms of encryption and decryption time when compared with the performances of RSA and DES under the same condition. The study was limited to certain metrics for comparative analysis.

Kyaw, Kyaw and Nay (2019) discussed the encryption and decryption time performance analysis of RSA and Elgamal public-key cryptosystems. The researcher encrypted the plaintext (text, image and audio) file with a public key for RSA and Elgamal and show the

comparison of encryption time for the two algorithms. The result shows that RSA is about 4 times faster than Elgamal during the encryption process and also RSA is faster than Elgamal during the decryption process. The result contradicted the previously reviewed works that Elgamal is faster than RSA during the Decryption process. However, the comparison is only based on encryption and decryption time for Text, Image and Audio data.

Omar *et. al.* (2019) used the Elgamal algorithm to encrypt and decrypt speech signal process transmitted to a particular recipient. The Elgamal algorithm is a special type of public-key technology used in this article for speech signals encryption/decryption. The security of this cryptosystem is based on the difficulty of calculating the discrete logs modulus of a large prime. Elgamal was applied on speech signals to perform secrecy, confidentiality and security of important information. The performance analysis of the presented cryptosystem in terms of different encryption and decryption speech quality measures indicates a satisfactory level of security while maintaining a good quality of the restored speech signal as compared with the original signal.

Zarni *et. al,* (2019) used ElGamal encryption and the RSA algorithm to secure mail-message before storing mails to the mail server. These algorithms consume a considerable amount of time and resources of memory, CPU time, and computation time to encrypt and decrypt data, therefore the researcher finds the performance comparison of these algorithms in terms of encryption time, decryption time, and memory usage over variable file sizes. The result showed that ElGamal and RSA have relatively similar time in generating the key. The encryption time of ElGamal is greater than RSA encryption time. The decryption time of ElGamal is a little less than the RSA algorithm. RSA requires the

least amount of storage space for encrypted files. It was concluded that RSA performs better in terms of encryption time, ElGamal performs better in terms of decryption time. In the use of RSA and Elgamal for an email message, it was observed that large scale file won't be able to be sent because of the longer time it will take to encrypt; therefore the work was based on the limited file size of the mail message.

Putri, Erna and Muhammad (2020), examined the comparative study of LUC, Elgamal and RSA algorithms in Encoding texts. The study implements each algorithms using several different texts to determine the encryption and decryption time. The result showed that the RSA algorithm performed better in the encryption process time of text file while the LUC algorithm performed better in decryption process time. The work was limited to encrypting the secret message in text form.

2.7 Gap Identified in the literature

Conscious awareness of previously published detailed literature survey about the performance analysis of RSA, Elgamal cryptographic algorithms in terms of Encryption time, decryption time, storage usage of both algorithms on Text data, Audio data, images data, and the use of key sizes; it was found that there are several works on the comparative analysis of RSA and Elgamal cryptographic algorithms. This current research work in contrast to the one reviewed so far is concerned with the time and space complexity of RSA and Elgamal on mixed data. From existing literature, most of the studies focused mainly on text data, images, audio, and mail-message with limited file size. Some of the reviewed study compared each of the algorithms (RSA and Elgamal) with either ECC or

Paillier. Some of the studies only analyze the encryption and decryption time of RSA and Elgamal. The study extends the Performance analysis of RSA and Elgamal using mixed data to improve infrastructure design, decision making, and allocation of resources on a computer system.

CHAPTER THREE

METHODOLOGY

3.1 Research Design

This study implements RSA and Elgamal cryptographic algorithms to obtain Encryption time, Decryption time, and the memory usage of both algorithms using mixed data. The data used were extracted from various data repositories such as (lipsum, datahub, kaggle and random text generator). Figure 3.1 displays the framework that was used in this study. The framework shows the phases involved in the development of the models which includes the loading of mixed data for processing encryption and decryption with RSA and Elgamal cryptographic algorithm for further analysis like time and space complexities.

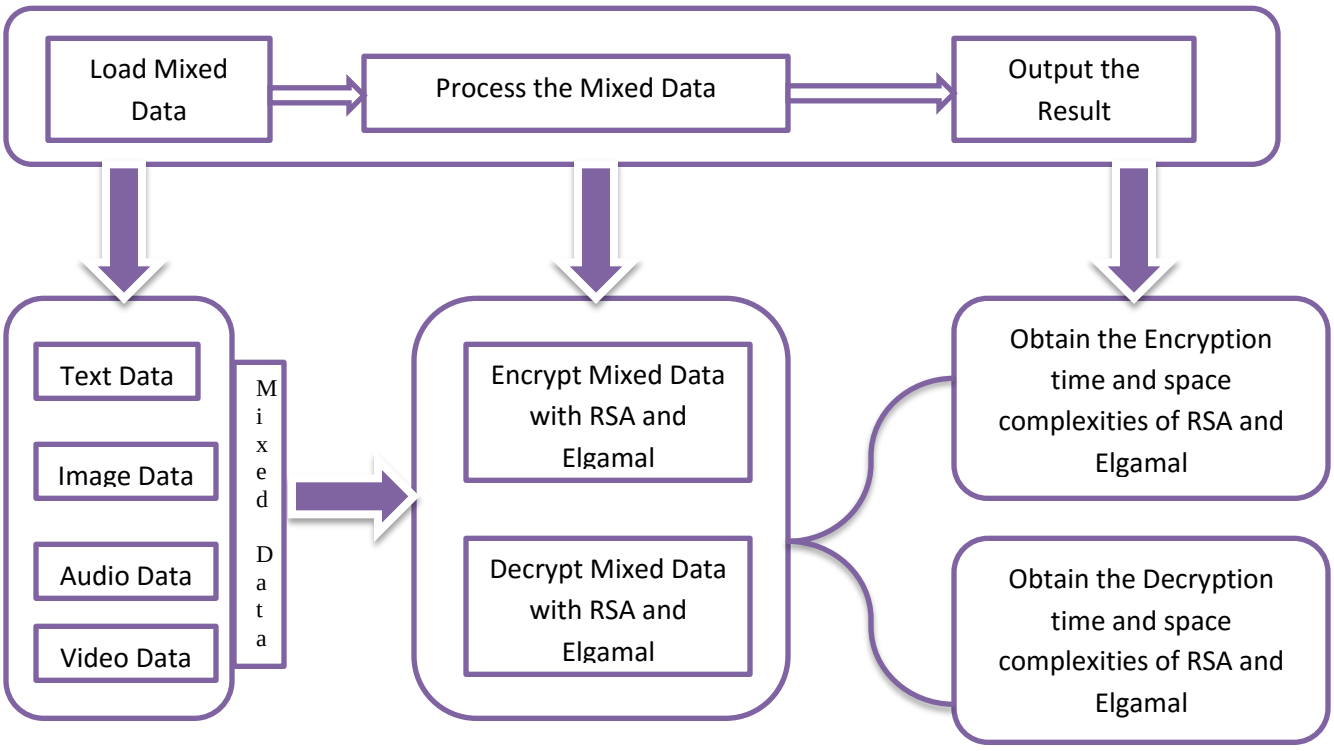

Figure 3.1: Conceptual Framework of the study

Figure 3.1 shows the hierarchical input-process-output (HIPO) of the study. This display the stages involve to obtain the desired result of this research. The mixed data which includes (text, image, audio and video) are the input tools in the study. The second stage of the framework is the process tool which includes RSA and Elgamal cryptographic algorithms to scramble the input data into unreadable content. The third stage is the result stage, this display the complexities of each of the algorithms used. The time and memory usage of RSA and Elgamal are obtained and their performance was compared to determine which of the algorithms performs better on mixed data. Table 1 display the methodology table and each activity that was performed on each objective.

Table 1: Research Methodology Table

S/N	Objective	Methodology
1	Implement both the RSA and Elgamal algorithms in encrypting and decrypting mixed data using C-sharp (C#) programing language	Psuedocode and Flowcharts were used to depict the RSA and ElGamal cryptographic algorithms. C# programming language was used to code the RSA and ElGamal algorithms to obtain the desired results. Various online repositories such as lipsum, datahub, kaggle and random text generator were consulted to obtain mixed dataset.
2	Compute the time and space complexities of the two algorithms using CPU internet clock and computer primary memory.	CPU internal clock was used to obtain the computational time during encryption and decryption process of RSA and ElGamal algorithms for all categories of dataset used. Computer primary memory was used to compute the space usage during encryption and decryption process of RSA and ElGamal

		algorithms for all categories of dataset used.
3	Determine the behaviour of the algorithms on the mixed data.	The data obtained from objective 2 was analyzed using Tables and Graphs.

3.2 Research Design Layout

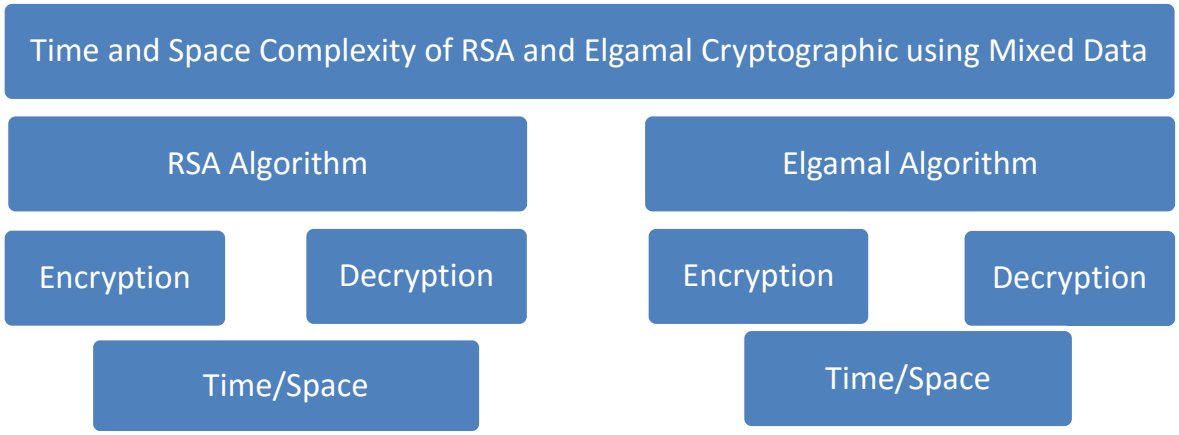

Figure 3.2: Research Design Layout:

Figure 3.2 display the process of how the dataset was loaded into the RSA and ElGamal implemented program and display the computational time and memory utilized during encrypting and decrypting the mixed data.

Methodology Used to Implement RSA and ElGamal Algorithms

1. Implement both the RSA and Elgamal algorithms in encrypting and decrypting mixed data using C-sharp (C#) programing language

Programming Language used: C#

The method used: RSA/Elgamal Cryptographic Algorithms with Mixed Data.

Performance Evaluation: Encryption time, Decryption time, Encryption Memory and Decryption Memory.

3.2.1 RSA Cryptographic Algorithm

Security of the RSA is based on the difficulty of factoring large integers. The encryption and decryption processes of the RSA algorithm require modular exponentiation. This section will outline how encryption, decryption, and key generation for RSA are performed in theory and some actual implementations and illustrations. Figure 3.3 display the design of flowchart for RSA cryptographic algorithm which is used to address objective one of this study.

$$Select\ Random\ prime\ number = p, q\ such\ that\ \ p \neq q$$

$$n = p\,x\,q$$

$$\lambda(N) = (p-1)(q-1)$$

$$(\lambda(N), e) = 1 < e < \lambda(N)$$

$$d \leftarrow e\,x\,d \bmod \lambda(N) = 1$$

$$Encryption\ \mathrm{Pr}ocess \leftarrow C = M^{e} \bmod\ n$$

$$Decryption\ \mathrm{Pr}ocess \leftarrow M\ = C^{d} \bmod\ n$$

3.2.1.1 Key Generation

The key generation process is detailed below;

1. Get two integers, $p\ and\ q$ from the user.

2. Check if $p\ and\ q$ are prime.

    2.1 If prime, continue the process, else exit the code.

3. Calculate $(p-1)*(q-1)$ and name it $\phi(n)$.

4. Calculate $n = p*q$.

5. Get an input e to act as a private key, under the condition that $1 < e < \phi(n)$ and $gcd(e, \phi\ (n)) = 1$. (gcd-greatest common divisor)

6. Compute the value of d such that $1\ <\ d\ <\phi(n)$ and $e.d\ \equiv\ 1\ (mod\ \phi(n))$.

NOTE: The public key is (n, e) and the private key is (n, d).

The values of p, q and $\phi(n)$ are private. 'e' is the public or encryption exponent. 'd' is the private or decryption exponent.

3.2.1.2 Encryption and decryption

Given the message to be M and Cipher C

i. Encryption is done using the public key (e, n)

ii. $C = M^{e} \bmod n$

iii. Decryption is done using the private key (d, n)

iv. $M = C^{d}\ mod\ n$.

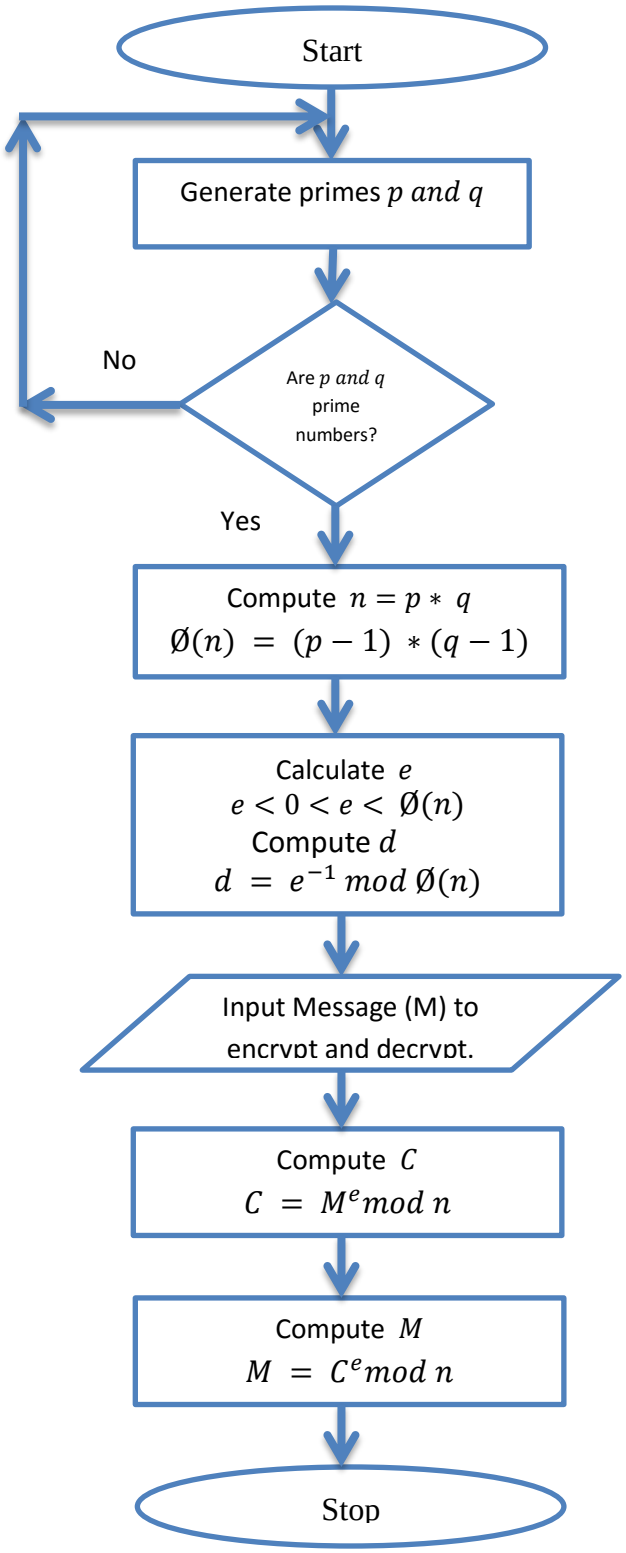

Figure 3.3: Flowchart of RSA Cryptographic Algorithm

3.2.1.3 Illustration of RSA Cryptographic Algorithm

For example, Sender A (the source) requests to encrypt and send a letter to Recipient B (the receiver). Receiver B has to generate a public key and private key for encryption and decryption processes respectively.

Step 1: choose two prime number p and q. Let's take $p = 3$ and $q = 11$

Step 2: compute the value of n and Ø. It is given as, $n = p\ x\ q$ and $Ø = (p-1)\ x\ (q-1)$. Here in the example, $n = 3\ x\ 11 = 33$ and $Ø = (3-1)\ x\ (11-1) = 2\ x\ 10 = 20$.

Step 3: Find the value of e (public key). Choose e, such that e should be co-prime. Co-prime means it should not multiply by factors of Ø and also not divide by Ø. Example factor of Ø are, $20 = 5x4 = 5x2x2$ so e should not multiply by $5\ and\ 2$ and should not divide by 20. So primes are $3,7,11,17,19..,as\ 3\ and\ 11$ are taken to choose e as 7. Therefore, $e = 7$.

Step 4: compute the value of d (private key). The condition is given as $gcd(Ø, x) = Øx + ey = 1$ where y is the value of d.

Step 5: do the encryption and decryption

Encryption is given as, $c = t^e\ mod\ n$

Decryption is given as $t = c^d\ mod\ n$

Suppose $t = 2$, so

Encryption is $c = 2^7\ mod\ 33 = 29$

Decryption is $t = 29^3 \, mod \, 33 = 2$

Therefore in the final, $p = 3, q = 11, \emptyset = 20, n = 3, e = 7 \, and \, d = 3$

3.2.2 Elgamal Cryptographic Algorithm

The ElGamal algorithm which was designed by Dr Taher ElGamal is a public-key cryptosystem. It depends on the one-way function which means that the encryption and decryption are done in separate functions. Figure 3.4 display the flow diagram of ElGamal algorithm to achieve objective one.

3.2.2.1 Key Generation

The key generation module has the following procedures

i. Generate a large random prime number (p)
ii. Choose a generator number (a)
iii. Choose an integer (x) less than $(p - 2)$, as the secret number.
iv. Compute (d) where $d = \boldsymbol{a^x} \, mod \, p$
v. The private key is given as (x) and the public key as (p, a, d)

3.2.2.2 Encryption and decryption

Represent the plaintext as an integer m where: $0 < m < p - 1$

Encryption is done using the public key (p, a, d)

i. Choose an integer k such that: $1 < k < p - 2$
ii. Compute $y, y = a^k \, mod \, p$
iii. Compute $z, z = (d^k * m) \, mod \, p$

iv. The ciphertext is given as $C = (y, z)$

Decryption is done using the private key (x)

i. The receiver obtains the ciphertext $C = (y, z)$

ii. Compute (r) as follows: $r = y^{p-1-x} mod\ p$

iii. Recover the plain text as follows: $m = (r * z)\ mod\ p$

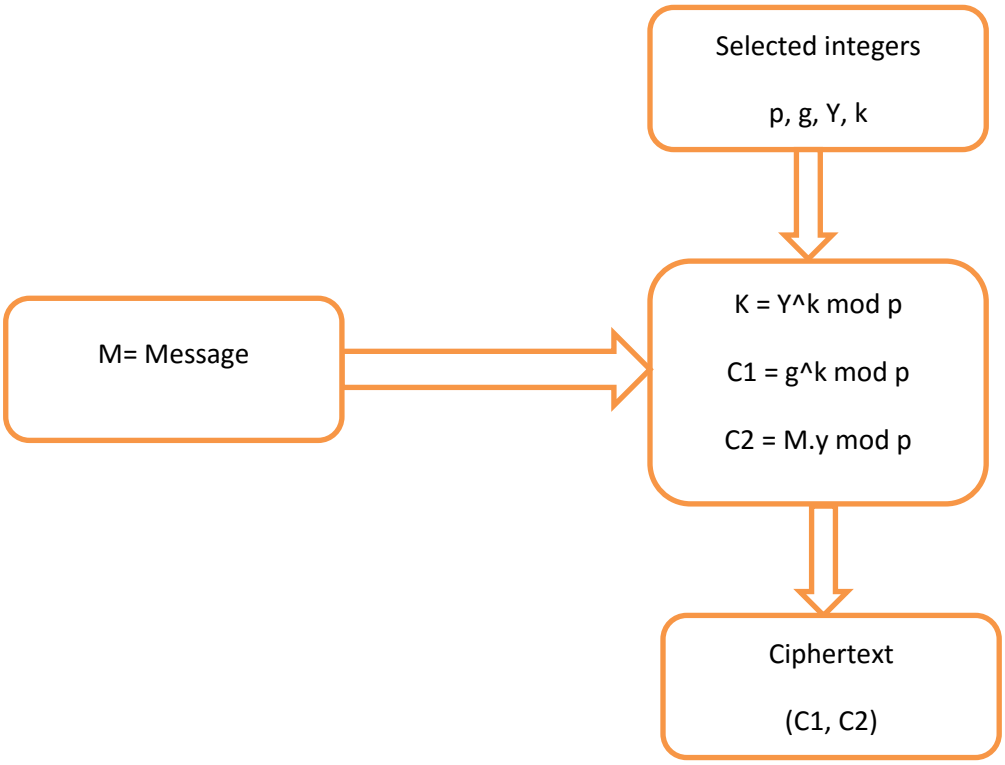

Figure 3.4: Flow Diagram of Elgamal Algorithm

3.2.2.3 Illustration of the ElGamal Cryptographic Algorithm

For example, sender A (the source) desires to encrypt and send a message to recipient B (the receiver). Receiver B has to generate a public key and private key for encryption and

decryption processes respectively. The encryption and decryption process is performed as follows to achieve objective one.

Sender A encrypts the message, given Receiver B public key $(p, a, d)\ is\ (83, 5, 66)$

- Given message $M = 3$
- Sender A chooses random $k = 29$, and then computes a stream key: $k = 66^{29}$ mod 83 = 22
- $y = 5^{29}\ mod\ 83 = 67$
- $z = (22 * 3) mod\ 83 = 66$
- $C(y, z)\ =\ (67, 66)$

To decrypt the message, receiver B receives the cypher text and decrypts the message making use of his private key given as (37)

- $C = (67,66)$
- Receiver B recovers the message key $r\ = 67^{37}\ mod\ 83 =\ 22$
- Compute the inverse of $r, r\_inverse\ = 1/22$
- $M = (1/22) * 66\ mod\ 83 =\ 3$
- $M\ =\ 3$

Methodology Used to Compute Time and Space Complexities of both Algorithms

2. Compute the time and space complexities of the two algorithms using CPU internet clock and computer primary memory.

3.2.3 Analysis of Time and Space Cryptographic Algorithm

The time complexity of an algorithm is commonly expressed through asymptotic notations: Big O which is denoted as $O(n)$, Big Theta denoted as Θ(n) and Big Omega demoted as Ω(n). The analysis of an algorithm for time complexity begins by trying to count the number of elementary operations that must be performed when the algorithm is executed. The elementary operations are addition, subtraction, multiplication, division and comparison. The space complexity is the amount of space (or memory) the algorithm takes to run as a function of its input-length, n. The spatial complexity includes both auxiliary space and input space. Many algorithms have inputs, e.g. an array, which may differ in size. In such instances, the complexity of the space will depend on the input size and therefore cannot be less than $O(n)$ for a size n input. Complexity for fixed-size inputs should be a constant $O(1)$.

If for a given size the complexity is taken as the maximum complexity over all inputs of that size, then the complexity is called the **worst-case complexity.** In the case of time complexity, it is the maximum number of operations needed to solve the given problem using the algorithm on an input of a specified size. If the complexity is taken as the average complexity over all inputs of a given size, then the complexity is called the **expected complexity.** The expected complexity of an algorithm is usually more difficult to ascertain than the worst-case complexity. Table 2 displays some common terminology used to describe the time complexity algorithms to achieve objective two of this study. For example, an algorithm is said to have logarithmic complexity if it has a time complexity of algorithms. For example, an algorithm is said to have logarithmic complexity if it has time complexity O (log n).

Table 2: Terminology of Complexity of Algorithms

Complexity	Terminology
O $(\log n)$	Logarithmic complexity
O (n)	Linear complexity
O (n log n)	n log n complexity
O (n^a)	Polynomial complexity
O (b^n) **, where b > 1**	Exponential complexity

Methodology Used to determine the behavior of both Algorithms

3. Determine the behaviour of the algorithms on the mixed data.

3.3 Data Collection

The mixed dataset used in this study were collected from various online repositories such as lipsum, datahub, kaggle and random text generator. The figure 3.4 display the various categories of dataset used, which each result obtained through encryption and decryption was analyzed with tables and graphs to achieve objective three.

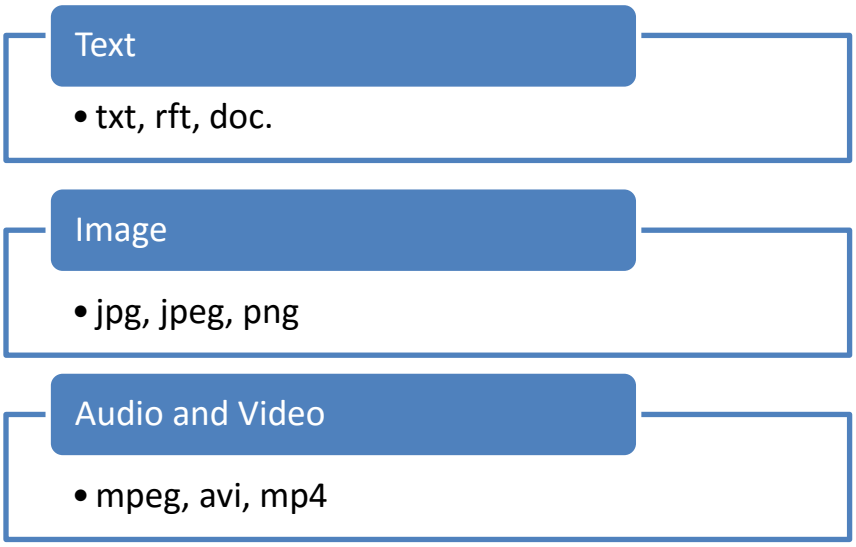

Figure 3.4: Categories of Data

3.4 Data Analysis Algorithm

The following are steps carried out in RSA and ElGamal cryptographic algorithms to determine their behaviours on text, image, audio and video files.

1. Mixed data (text, image, audio and video) of different sizes are supplied into the implemented RSA and ElGamal algorithms.

2. When each data have been supplied into the executed program, the time and space functions are initiated.

3. The contents of each mixed data are read, and implemented while RSA and ElGamal Algorithms are initiated.

4. After execution, the time and space of execution for each mixed data file are produced for each of the algorithms and analyzed using tables and figures.

3.5 Research Instruments/Tools

The following development tools were used to implement the RSA and ElGamal cryptographic algorithms.

a. C-Sharp (C#) Programming Language of Visual Studio 2015.

The C# programming Language is a high-level language, which is simple, general-purpose and known as an object-oriented language. The language which was developed by Microsoft and runs on the dot net framework is used to develop various applications including web applications, mobile applications, and desktop applications among others.

b. Cryptographic algorithms.

The cryptographic algorithms are known to be the most frequently used privacy protection method for data encryption, authentication, and digital signatures. There are various classes of cryptographic algorithms which include Hash functions, symmetric-key algorithms and asymmetric-key algorithms. This study uses two popular asymmetric algorithms (RSA and ElGamal).

c. Cryptographic Algorithm Efficiency Functions

The algorithm efficiency functions deal with the Time complexity and Space complexity functions of the cryptographic algorithms. The time and space functions were used in this study to determine the behaviour of the asymmetric cryptographic algorithms on mixed data.

Time complexity is a function describing the amount of time that an algorithm takes in terms of the amount of input data while space complexity is a function describing the amount of memory used.

3.6 Mixed Data Analysis

This study used mixed dataset which is defined as text, image, audio and video data.

a. Text Data

Text data generally comprises records that can describe expressions, phrases or even columns of a free-flowing document. The intrinsic unstructured (no perfectly ordered data columns!) and noisy design of text data make it more difficult for machine learning approaches to operate exclusively on actual textual information.

b. Image Data

Image data is quite widely used to depict visual or pictures data. Usually, this data must be transformed into a raster format (and possibly a vector) to be used analytically for the GIS. Image data is usually stored in a range of industry-standard proprietary de facto formats.

c. Audio Data

These are analogue sound waves that are saved in digital format on the computer system. Sounds produced on laptop exist as digital media stored as audio clips. The sound input through the microphone is transformed to digital for storage and handling. Digital audio is

decomposed to thousands of samples per second. Each sample of voice is kept as binary code.

d. Video Data

Video data typically exists as continuous analogue signals. For the device to interpret this video data, analogue signals must be transferred to a non-continuous digital format. Video data can be stored in a digital format as a sequence of bits on a hard drive or in memory space.

3.7 Implementation

The system was implemented on a Windows Operating System using the C-Sharp (C#) programming language. All Development, Testing, and Design were implemented on a Windows 10 of Intel core i5 processing power of 3.40 GHz CPU with 6 GB Ram.

A Hardware Requirements

a. Personal computer with Intel Core i3 or higher processor recommended.

b. 4 GB of RAM or higher recommended.

c. The processing speed of 2.4 GHz or higher recommended.

d. Hard disk memory of 500GB or higher recommended.

B Software Requirements

a. Operating System: Windows 7 or macOS El Capitan or higher.

b. Microsoft Visual Studio 12.

CHAPTER FOUR

RESULTS AND DISCUSSION OF FINDINGS

4.1 Experimental Interfaces

Figures 4.1 to 4.3 display the experimental interfaces used to achieve the implementation of RSA and ElGamal algorithms of this study. Figure 4.1 shows how text dataset is loaded into the RSA and ElGamal algorithms implemented in C# programming language. Figure 4.2 shows the image dataset and how it was encrypt and decrypt to achieve objective one. Figure 4.3 shows how audio and video dataset is loaded into the C# programming environment.

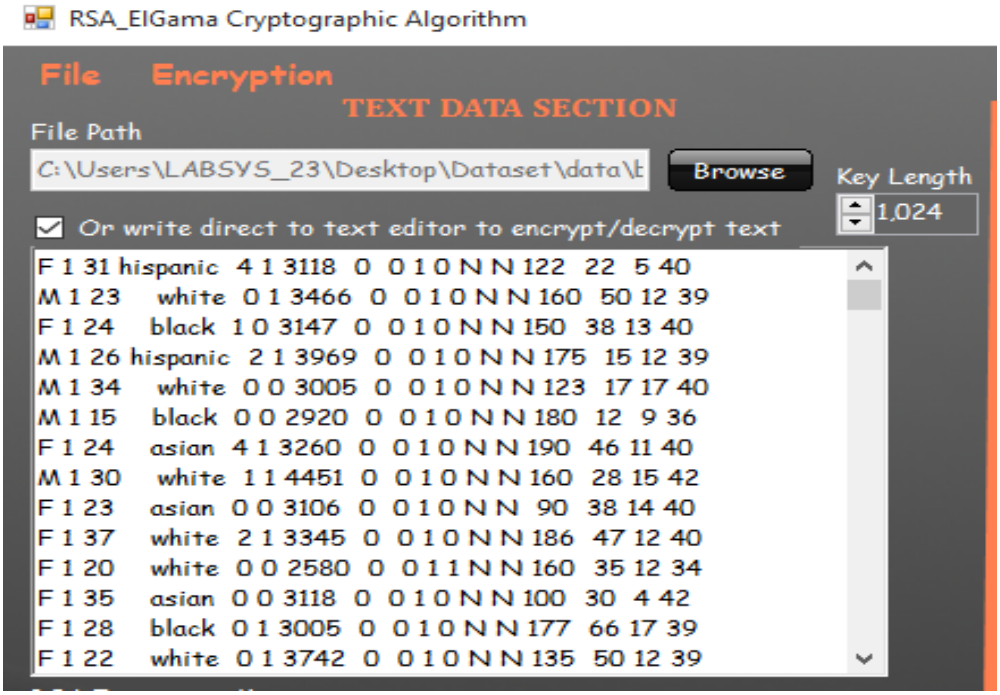

Figure 4.1: Experimental Interface of Birth and Birth rate dataset of text data.

The dataset in figure 4.1 is processed to perform encryption and decryption using RSA and ElGamal algorithms which was implemented in C# programming language. The displayed birth rate dataset will be converted to cipher unreadable text after the encryption button is being pressed.

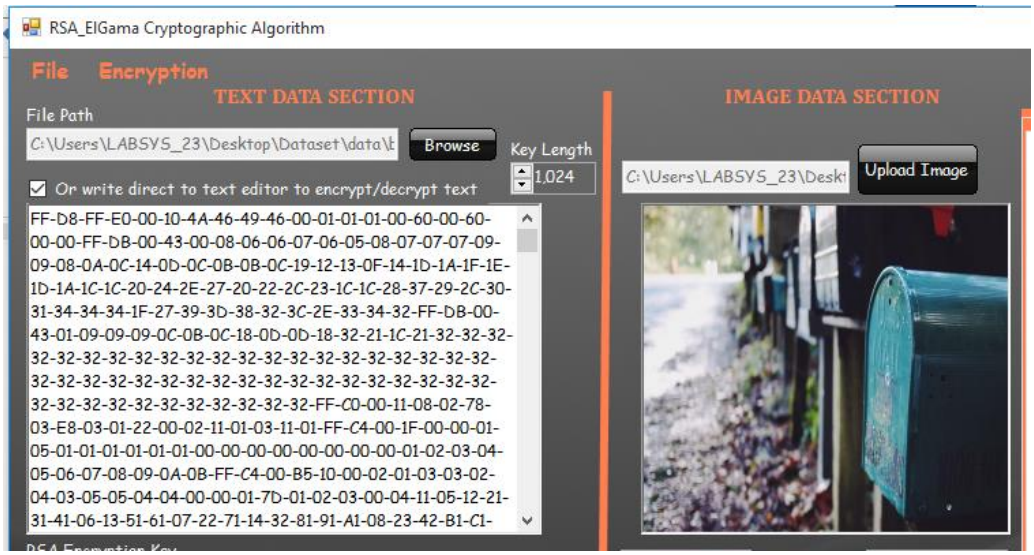

Figure 4.2: Interface of Image data and its corresponding hexadecimal form.

The image dataset is converted to hexadecimal form and the hexadecimal form is encrypted using RSA and ElGamal cryptographic algorithms to achieve the objective two (2). During which the encryption time, decryption time and memory usage for both algorithms are obtained for further analysis.

Figure 4.3: Interface of Audio and Video dataset.

The audio and video data are uploaded and converted to its hexadecimal form. The hexadecimal form is encrypted using RSA and ElGamal algorithms to obtain the encryption time, decryption time and memory usage for both algorithms to achieve the computation of time and space of both algorithms.

4.2 Experimental Results

To determine the behavior of both algorithms on mixed data, RSA and Elgamal Cryptographic algorithms were implemented in C-sharp programming language on mixed data (text, image, audio and video). The experimental results of each dataset is indicated using tables and figures. The time taken to encrypt and decrypt each dataset is given in seconds (s), while space (memory) used to encrypt and decrypt each dataset are given in

kilobytes (KB). Table 4.1 displays the time and space usage during encryption and decryption process on text dataset using RSA and ElGamal cryptographic algorithms.

A. Results for RSA and ElGamal on Text Data

Table 4.1 Tabular representation of Text Data encryption for RSA and ElGamal algorithms.

S/N	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	623.5	77.93
3	120	0.4835	2.92	925.85	115.71
4	140	0.5664	3.80	1054.83	131.84
5	230	0.9315	4.67	1740.99	217.62
6	2048	5.8852	15.12	11133.64	1391.70
7	5120	16.1733	43.90	30116.30	3764.52

The above table 4.1 shows the data values of encryption time and space obtained from the text data of different file sizes in kilobytes (KB). The encryption time of both RSA and ElGamal algorithms are captured and recorded in seconds (s) while the memory usages for both algorithms are captured and recorded in kilobytes (kb). Figure 4.3 displays the graphical analysis of encryption time of RSA and ElGamal cryptographic algorithms on text data.

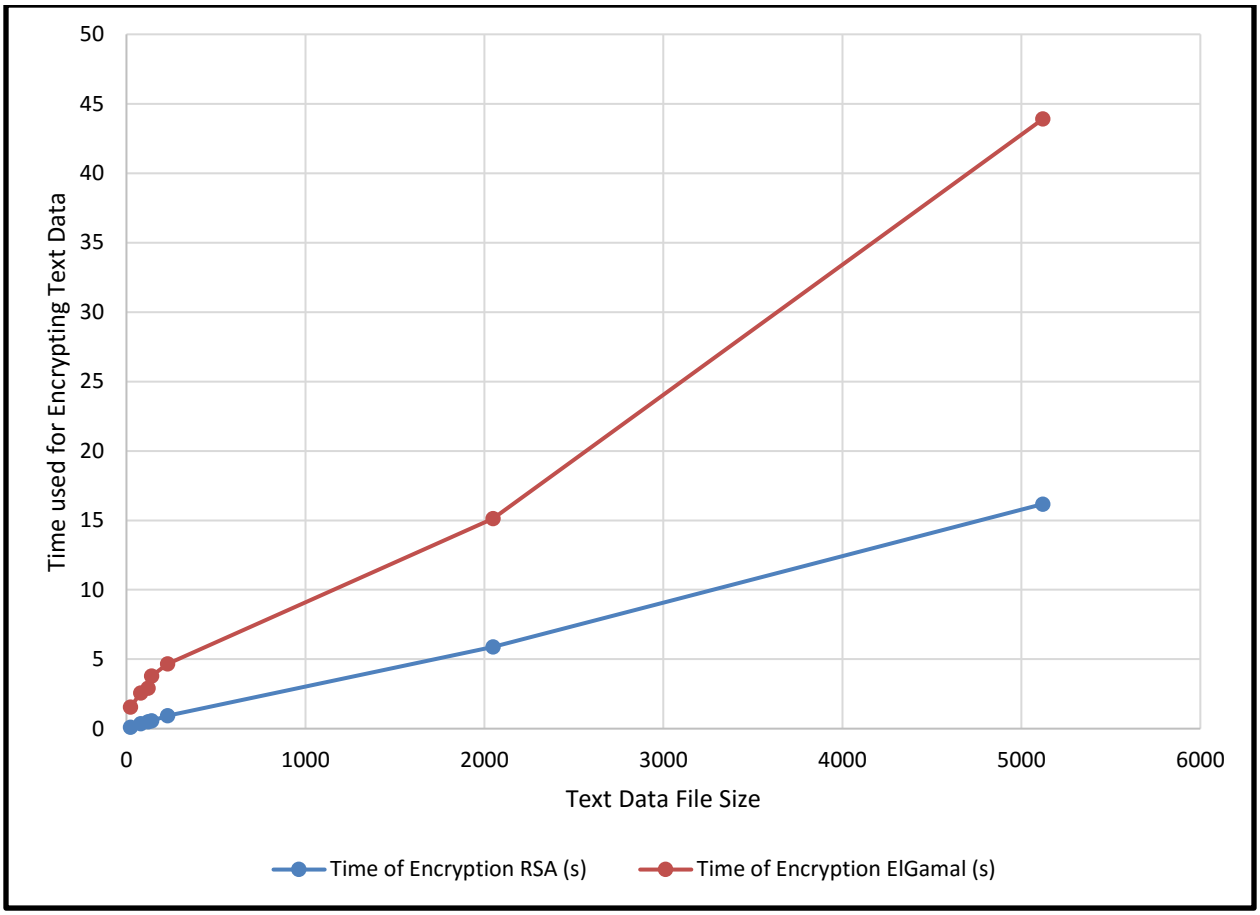

Figure 4.3: Encryption Time Analysis for RSA and ElGamal cryptographic algorithms for text dataset.

The graphical interface in figure 4.3 shows the computational time analysis of RSA and ElGamal algorithms. The line in colour red was used to denotes the encryption time of ElGamal while the colour blue denote the encryption time of RSA. The ElGamal line grows wider as the file size gets larger. This shows that Elgamal consumes more CPU time during the encryption of text data while RSA consumes less time. Figure 4.4 shows the encryption space used for RSA and ElGamal for text dataset.

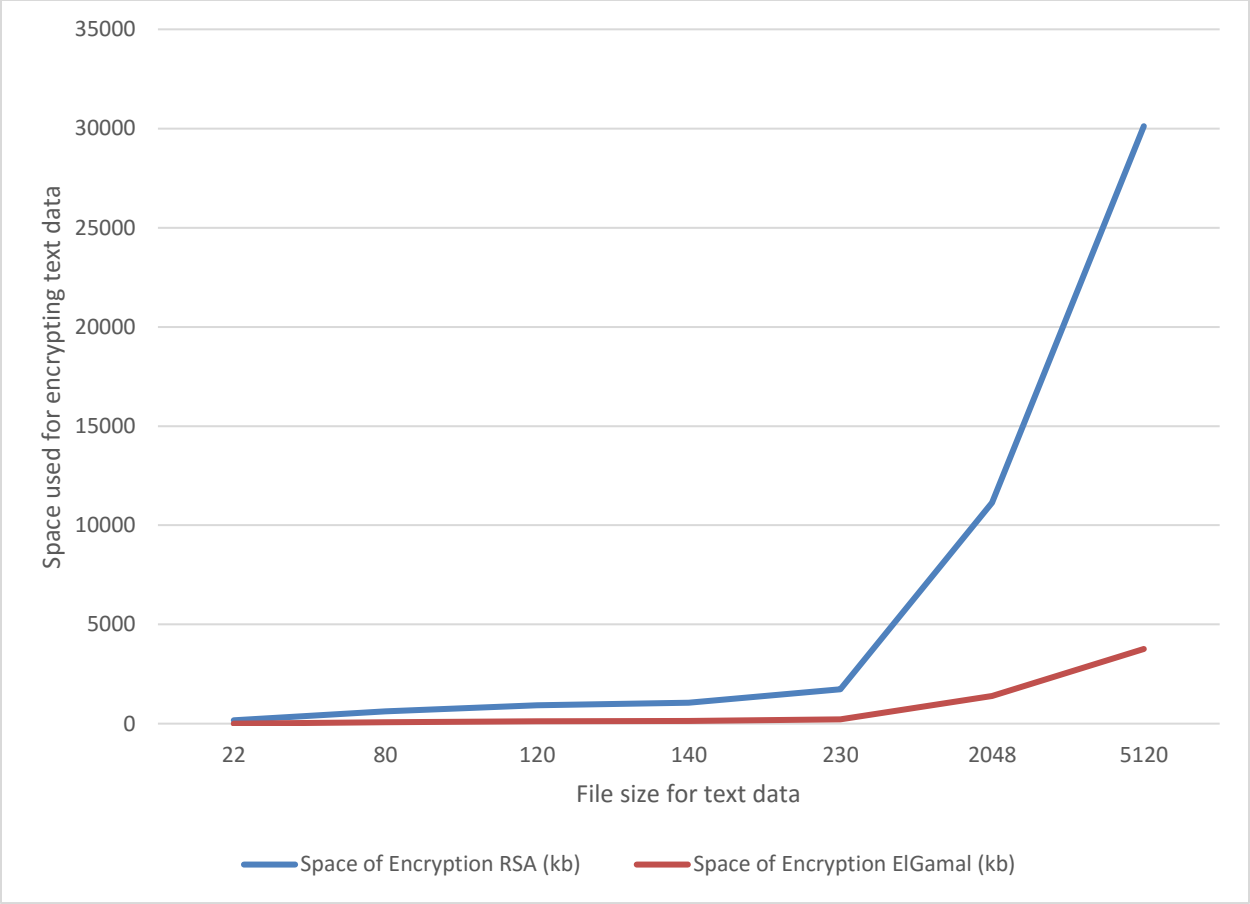

Figure 4.4: Analysis of Memory Used for RSA and ElGamal for Text Dataset during Encryption Process.

The colour blue line which was used to denote the RSA algorithm consumes more space during the encryption process of text dataset while the red line which denotes the ElGamal algorithm uses the least amount of memory during encryption of text dataset. Table 4.2 shows the tabular representation of text dataset values of RSA and ElGamal algorithms during the decryption process.

Table 4.2 Tabular representation of Text Data decryption for RSA and ElGamal algorithms.

S/N	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	2048	74.9069	19.3083	1391.69	1391.70
7	5120	194.2630	56.1963	3764.52	3764.52

The values in the above table 4.2 describe the decryption time and memory usage of RSA and ElGamal cryptographic algorithms on the text dataset. From the table, various file sizes were used to experiment with the implemented algorithms. The decryption time of the text data was captured and recorded in seconds (s) while the decryption space was captured and recorded in kilobytes (kb). Figure 4.5 shows the graphical interface of decryption time of RSA and ElGamal algorithms for text dataset.

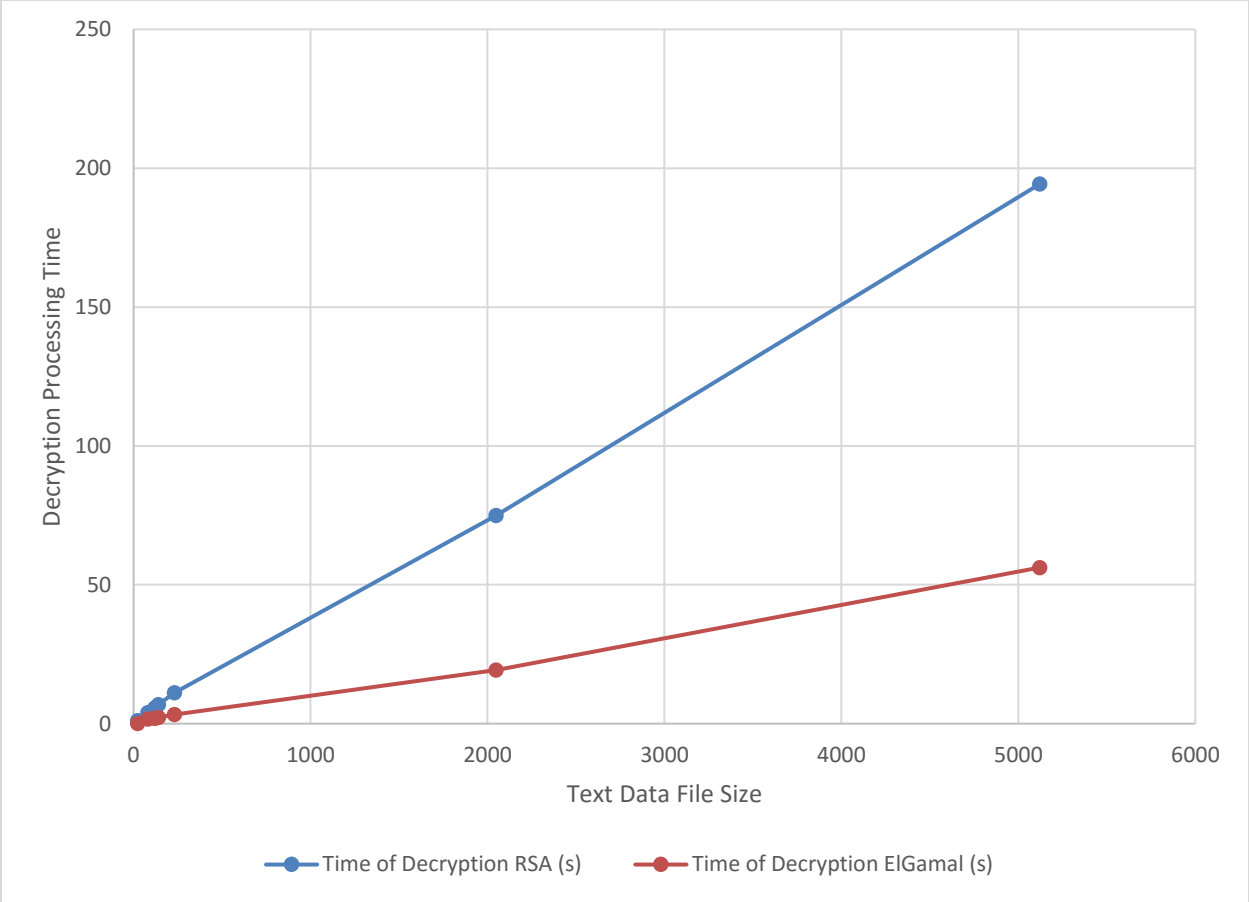

Figure 4.5: Decryption Time Analysis of RSA and ElGamal Algorithms for Text Dataset.

The computational time of decryption for text data of RSA and ElGamal algorithms with blue and red colour lines respectively shows that RSA consumes more CPU time during decryption of text data while the ElGamal algorithm consumes least CPU time during decryption of text data. Figure 4.6 describes the memory usage of RSA and ElGamal for text data during the decryption process.

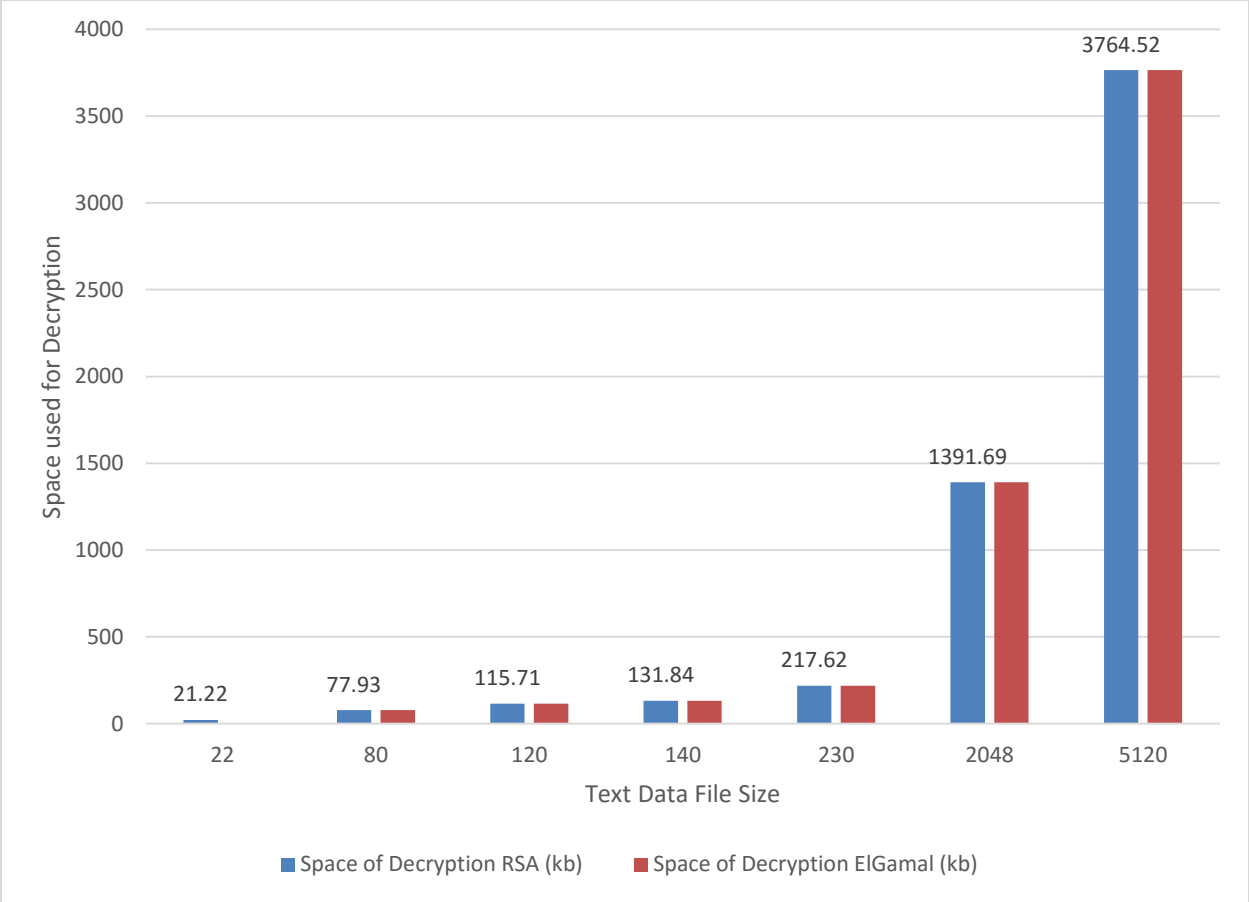

Figure 4.6: Memory Usage of RSA and ElGamal during Decryption of Text Dataset

Figure 4.6 shows that both RSA and ElGamal cryptographic algorithms consumed relatively the same amount of space during the decryption of text data.

4.2.1 Discussion for RSA and ElGamal on Text Data

During the analysis of the time of encryption for RSA and ElGamal algorithms for Text dataset, Table 4.1 and Figure 4.3, showed that the RSA algorithm uses less time during encrypting text data while the ElGamal algorithm consumes more time during encrypting text data. Figure 4.4 uses more space (memory) during encrypting text data while the

ElGamal algorithm consumes less or relatively small space in encrypting text data compared to the RSA algorithm.

From Table 4.2 and Figure 4.5, the time and space used for both algorithms during the decryption process was analyzed and showed that the RSA algorithm uses more time to decrypt any given text data of different file sizes while ElGamal uses the less or minimal time to decrypt text data of different file sizes. Figure 4.6, showed the memory usage of both algorithms during the decryption process. It was discovered that both algorithms consume the same amount of space (memory) during the decryption process of Text Data. Table 4.3 describe the tabular representation of image data.

B. Results for RSA and ElGamal on Image Data

Table 4.3 Tabular representation of Image Data encryption for RSA and ElGamal algorithms.

S/N	File Size (KB)	Time of Encryption		Space of Encryption	
		RSA (s)	ElGamal (s)	RSA (kb)	ElGamal (kb)
1	63	0.9896	2.9947	1890.32	236.29
2	85	1.0023	3.3907	2439.15	295.41
3	120	1.6205	8.7705	3088.11	385.73
4	130	1.7495	9.3232	3129.38	399.20
5	200	1.9853	10.5232	3764.52	470.56
6	300	2.9534	12.2056	5470.91	683.85
7	550	5.6149	16.2851	10597.82	1324.71

Table 4.3 displays the data obtained from encrypting image data using RSA and ElGamal cryptographic algorithms. The third column of the table shows the encryption time of both algorithms in seconds (s) while the forth column shows the space usage during the encryption process of image data of both algorithms. The figure 4.7 described the graphical representation of encryption time of image data using RSA and ElGamal.

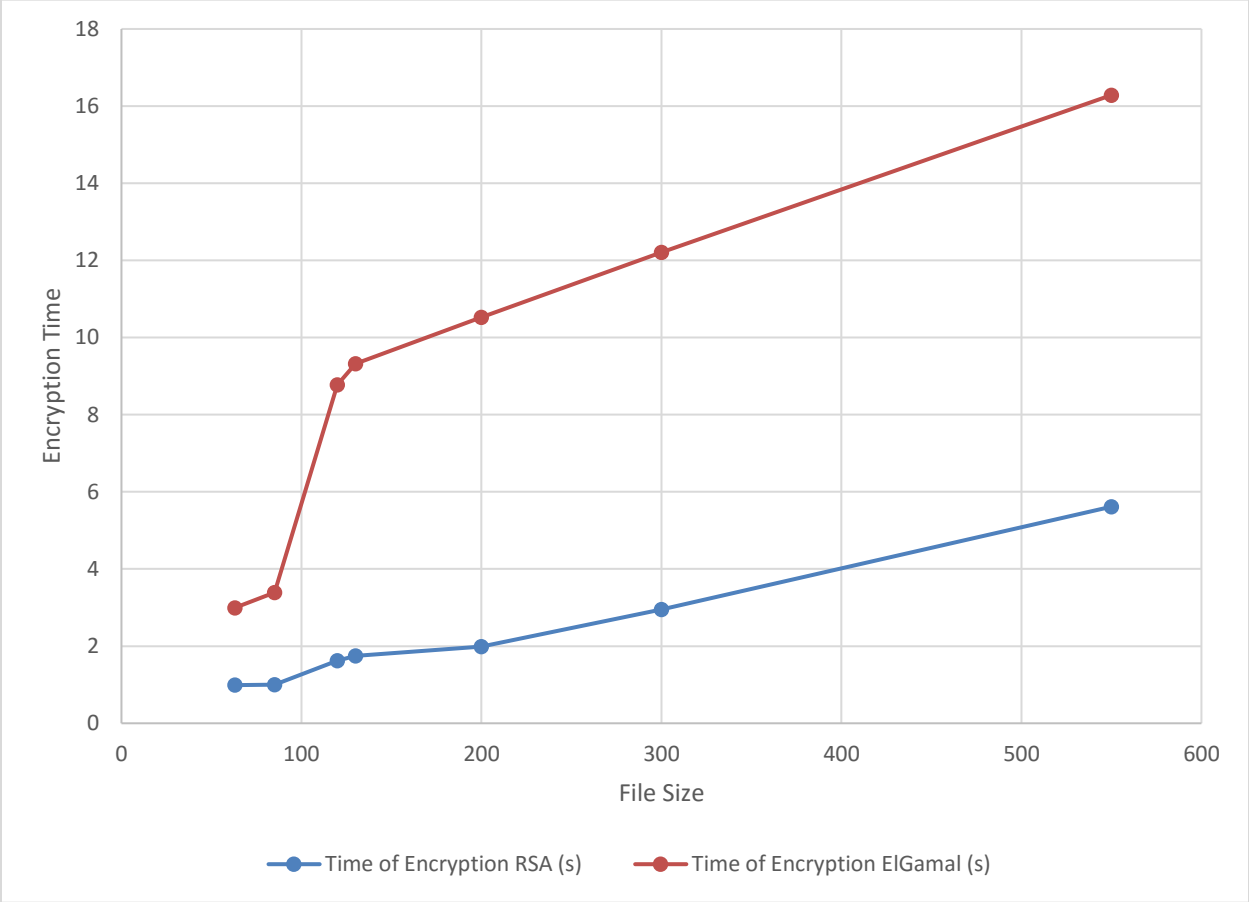

Figure 4.7: RSA and ElGamal Encryption Time Analysis for Image Data

The red line which denotes the ElGamal encryption time shows that the ElGamal algorithm consumes more computational time to encrypt image data compared to RSA in blue colour which uses less computational time during encryption of image data. Figure 4.8 described the space usage during encryption of image data using RSA and ElGamal algorithm.

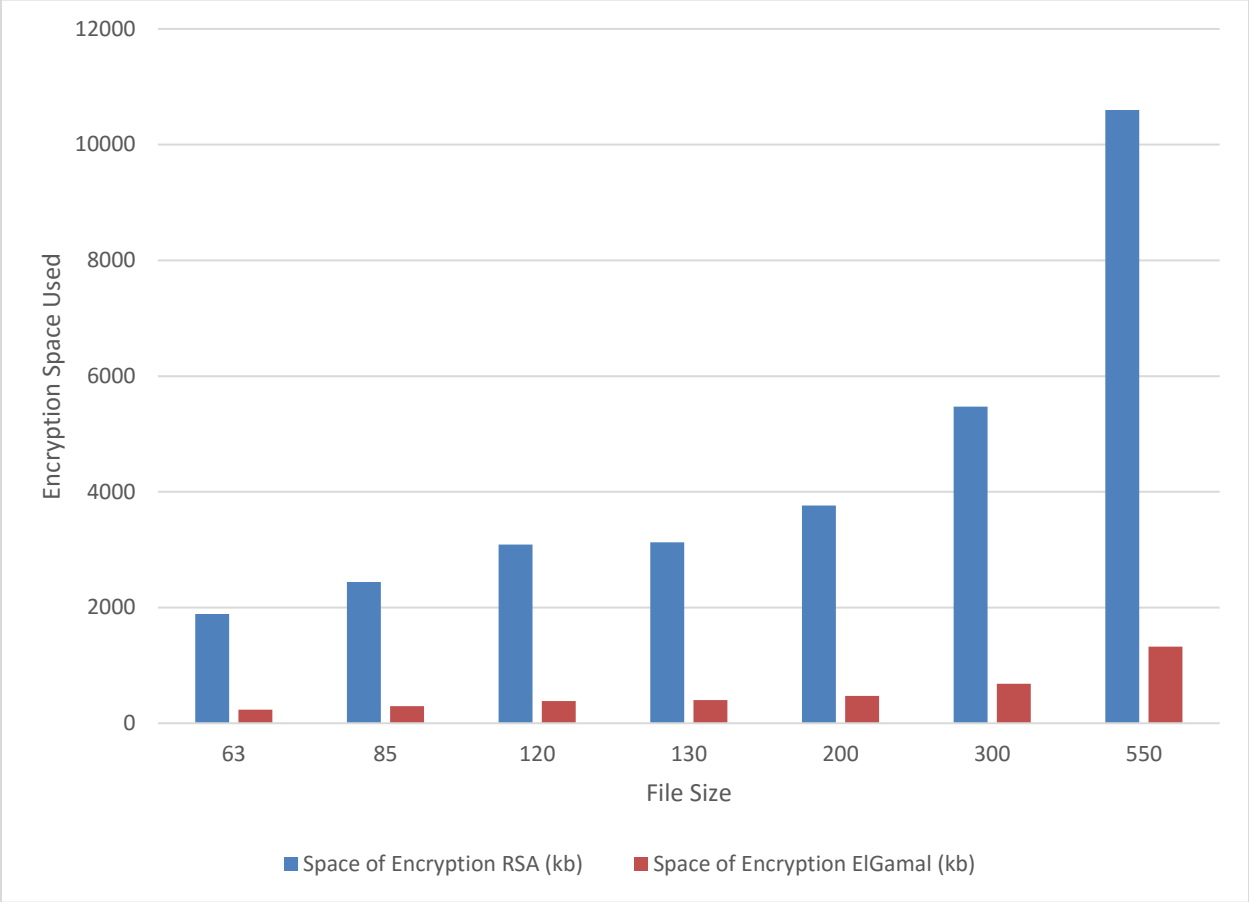

Figure 4.8: RSA and ElGamal Space used for encrypting Image data.

From the figure 4.8, the blue line which is used to denote RSA grows taller than the red line (ElGamal). This implies that the RSA algorithm consumed more space than ElGamal during the encryption of image data. Table 4.4 shows that time and space values generated during the decryption process using RSA and ElGamal algorithms.

Table 4.4 Tabular representation of Image Data decryption for RSA and ElGamal algorithms.

S/N	File Size (KB)	Decryption Time		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	386.00	386.00
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	1324.71	1324.71

The generated data obtained from the image dataset decryption process was captured and tabulated. The decryption time and space were labelled in seconds (s) and kilobytes (kb) respectively for both RSA and ElGamal cryptographic algorithms. Figure 4.9 shows the graph of RSA and ElGamal decryption time for image data.

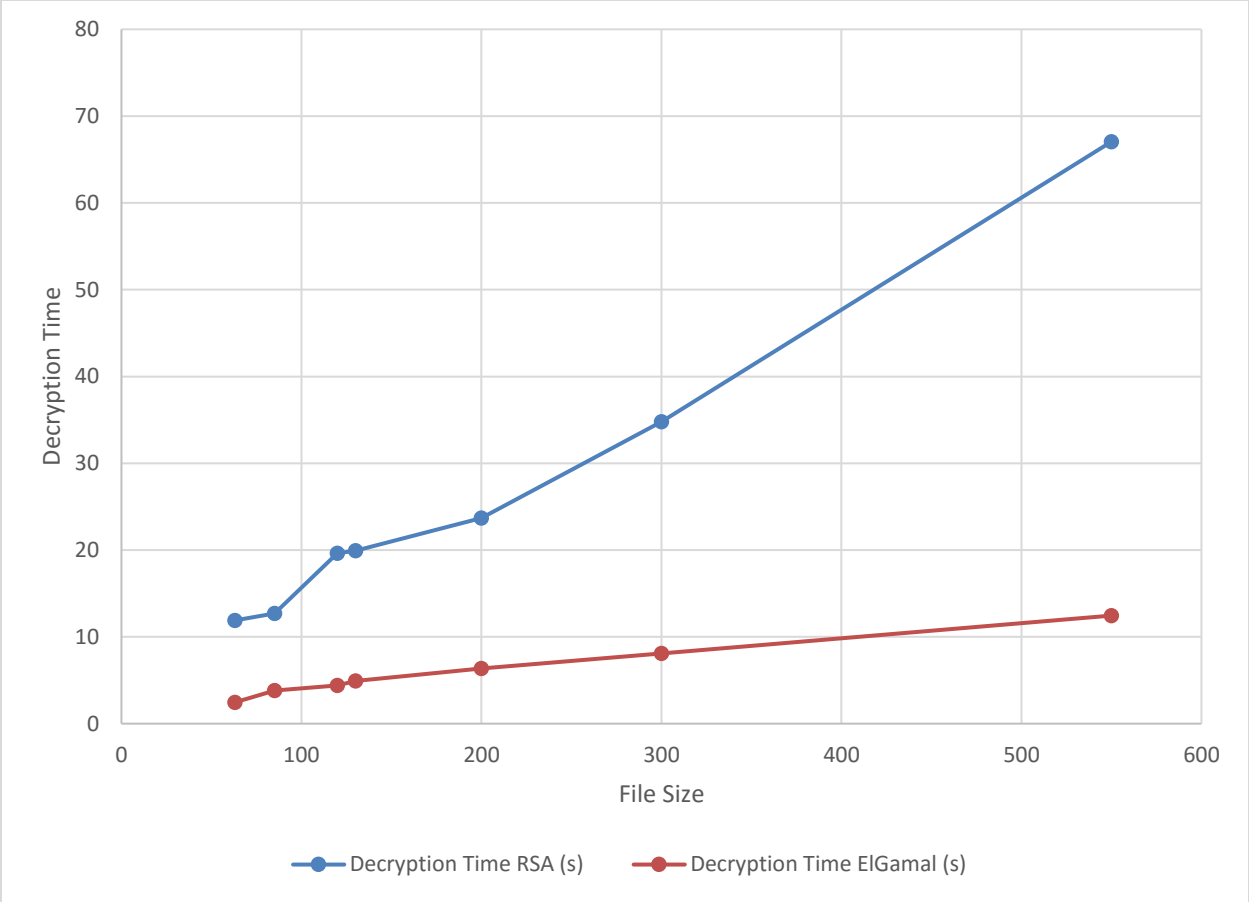

Figure 4.9: RSA and ElGamal Decryption Time for Image data.

The red line which denotes ElGamal algorithm uses less computational time during the decryption of image data while the blue line which denotes RSA uses more time complexity during the decryption of image data. The larger the file size, the wider the space between the two lines. Figure 4.10 shows the graphical interface of space usage during decryption of image data.

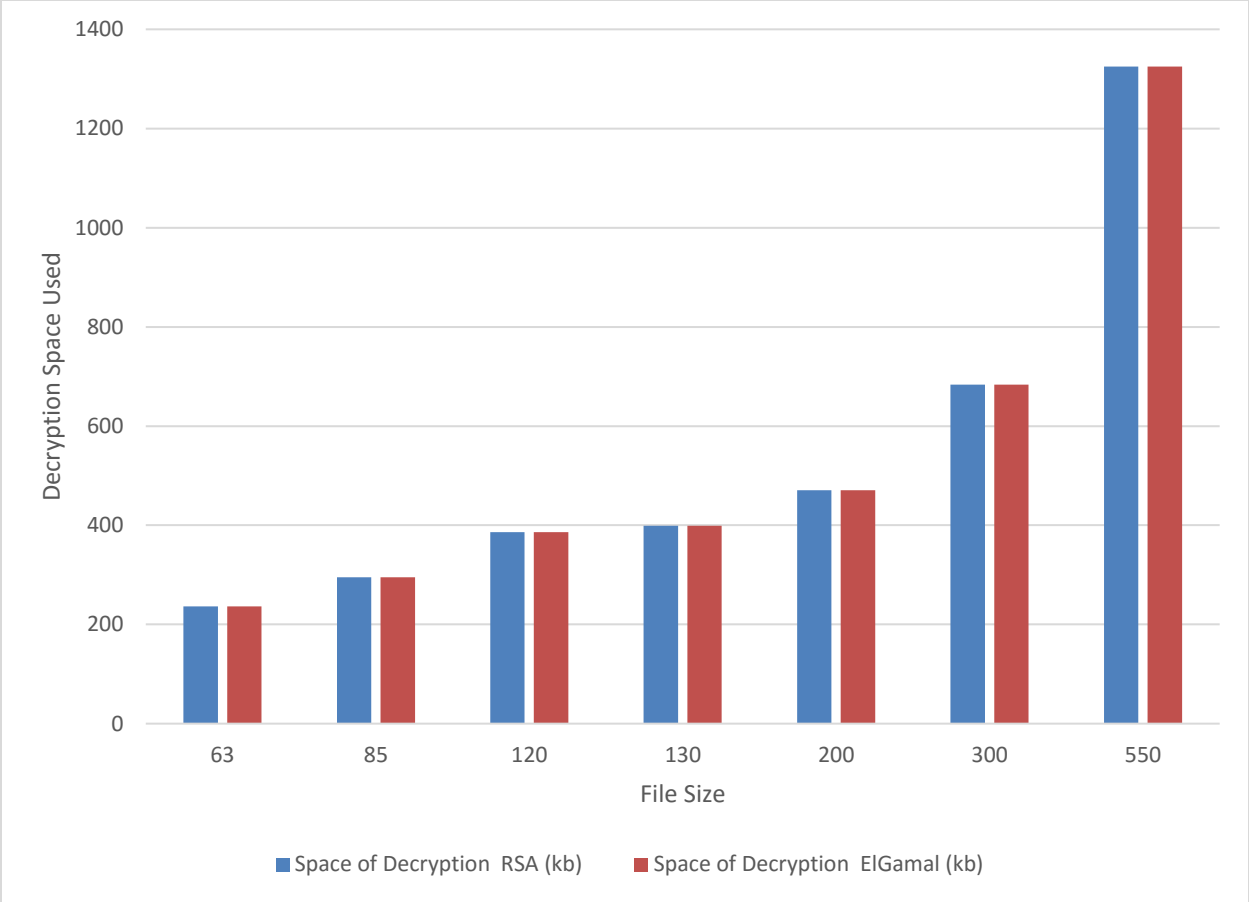

Figure 4.10: Space used by RSA and ElGamal during Decryption of Image Data.

From the decryption of image data, it shows that both RSA and ElGamal use relatively the same amount of space during the decryption of image data of different file size.

4.2.2 Discussion for RSA and ElGamal on Image Dataset

Tables 4.3 to 4.4 and figures 4.7 to 4.10 represents the execution time and memory usage for both RSA and ElGamal algorithms for Image data. The statistical data in table 4.3 and graph in figures 4.7 and 4.8 showed that ElGamal processing time is higher than RSA processing time during encryption of image data, but consumes less space during the

encryption process of image dataset while RSA consumes relatively high memory during the encryption process.

Table 4.4 and figure 4.9 shows that ElGamal uses less processing time for decrypting image data while RSA processing time for decrypting image data is higher. Figure 4.10 depicts that RSA and ElGamal used the same amount of memory for decrypting image data. Table 4.5 shows the statistical data of encryption time and space usage for RSA and ElGamal algorithm on audio data.

C. Results for RSA and ElGamal on Audio Data

Table 4.5: Encryption Time and Space Usage of RSA and ElGamal for Audio Data

S/N	File Size (Kb)	Encryption Time		Memory Usage	
		RSA	ElGamal	RSA	ElGamal
1	50	0.6186	5.7240	1167.72	145.96
2	55	0.6806	5.9135	1289.66	161.21
3	60	0.7383	6.4193	1384.90	173.10
4	70	0.8740	7.9503	1663.56	207.92
5	90	1.1263	8.1892	2131.86	266.48
6	120	1.3651	12.2567	2606.37	325.79
7	200	1.8295	16.7535	3483.51	435.55

From the statistical data of encryption time and space usage on audio data, it was observed that RSA has the least computational time while Elgamal consumes a minimal amount of space during the encryption process for the various file sizes used. The figure 4.11 shows the chart of encryption time for RSA and ElGamal algorithms on audio data.

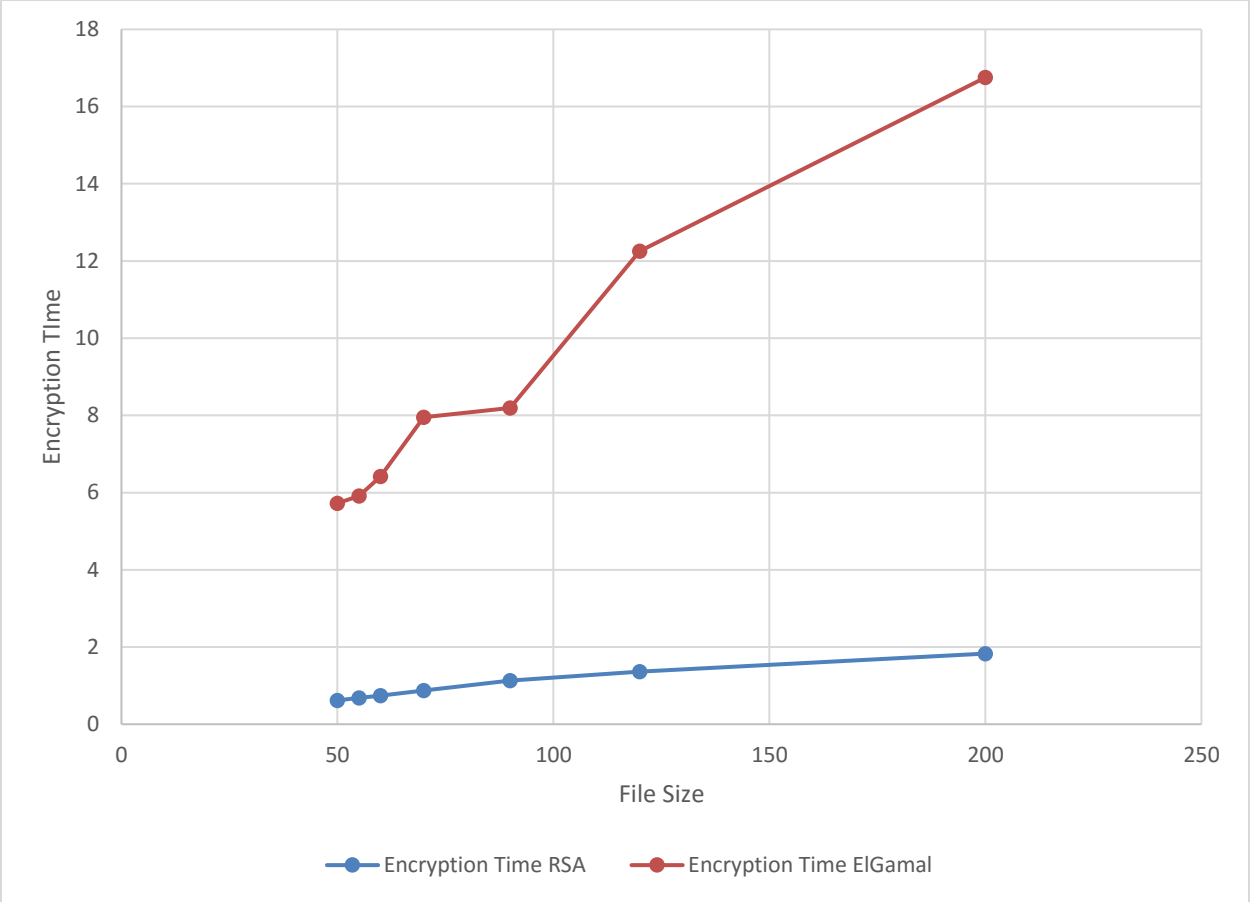

Figure 4.11: Encryption Time of RSA and ElGamal Algorithms for Audio Data.

From the chart above, the blue which denotes the RSA algorithm takes less computational time during the encryption of audio data. This makes the RSA algorithm perform better than ElGamal with longer time of computation during the encryption of audio data. Figure 4.12 shows the chart of memory usage for RSA and ElGamal during the encryption process of audio data.

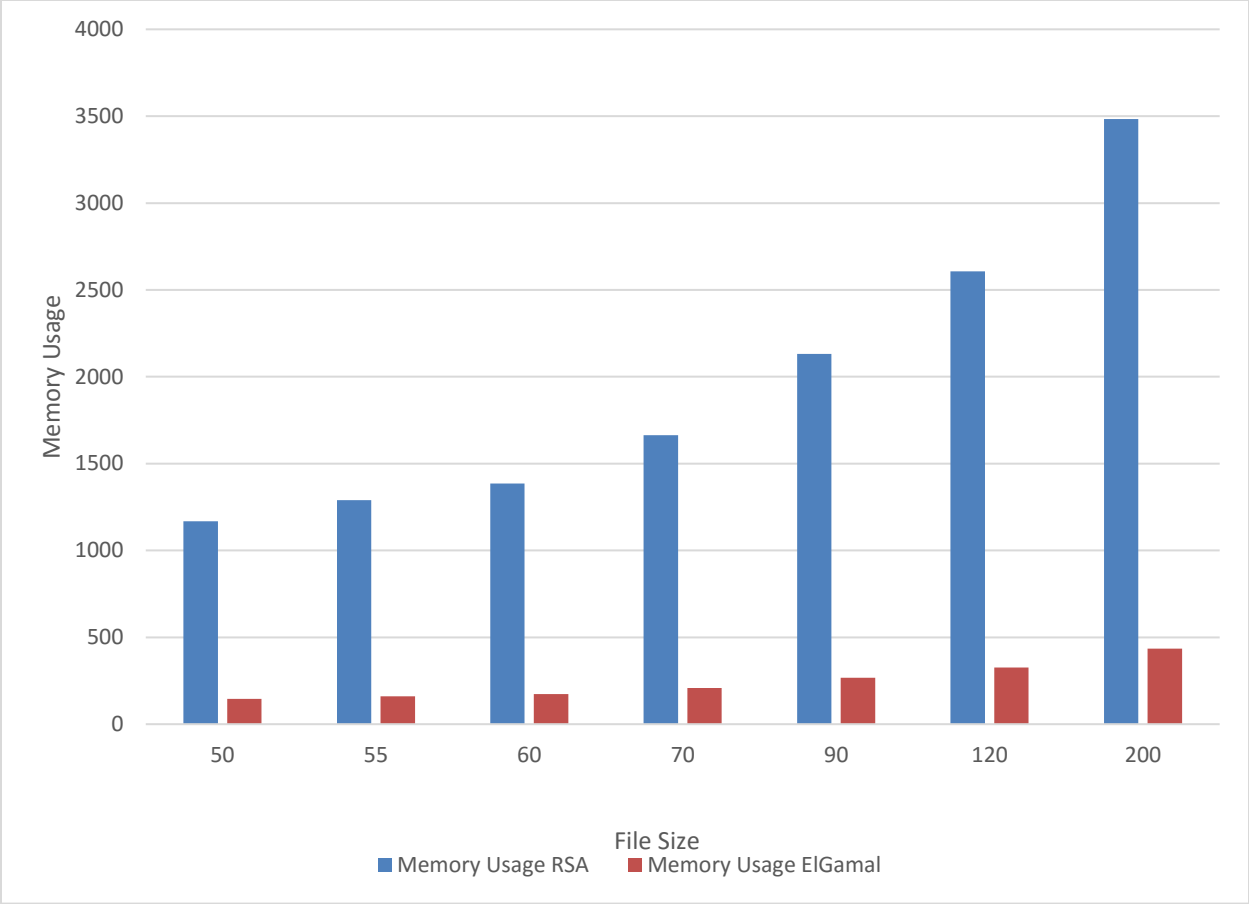

Figure 4.12: Memory Usage of RSA and ElGamal Algorithms during Encryption of Audio data.

The computational speed of RSA is relatively higher than that of ElGamal during the encryption of audio data of different file sizes. The table 4.6 shows the statistical table of computational time and space for RSA and ElGamal decryption process on audio data.

Table 4.6: Decryption Time and Memory Usage of RSA and ElGamal for Audio Data

S/N	File Size (Kb)	Decryption Time		Decryption Memory Usage	
		RSA	ElGamal	RSA	ElGamal
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

From the audio data statistical table, it was observed that the decryption time of RSA is less than that of ElGamal while both algorithms use the same computational speed during decryption of audio data. Figure 4.13 shows the chart of computation time for RSA and ElGamal algorithms on audio data during the decryption process.

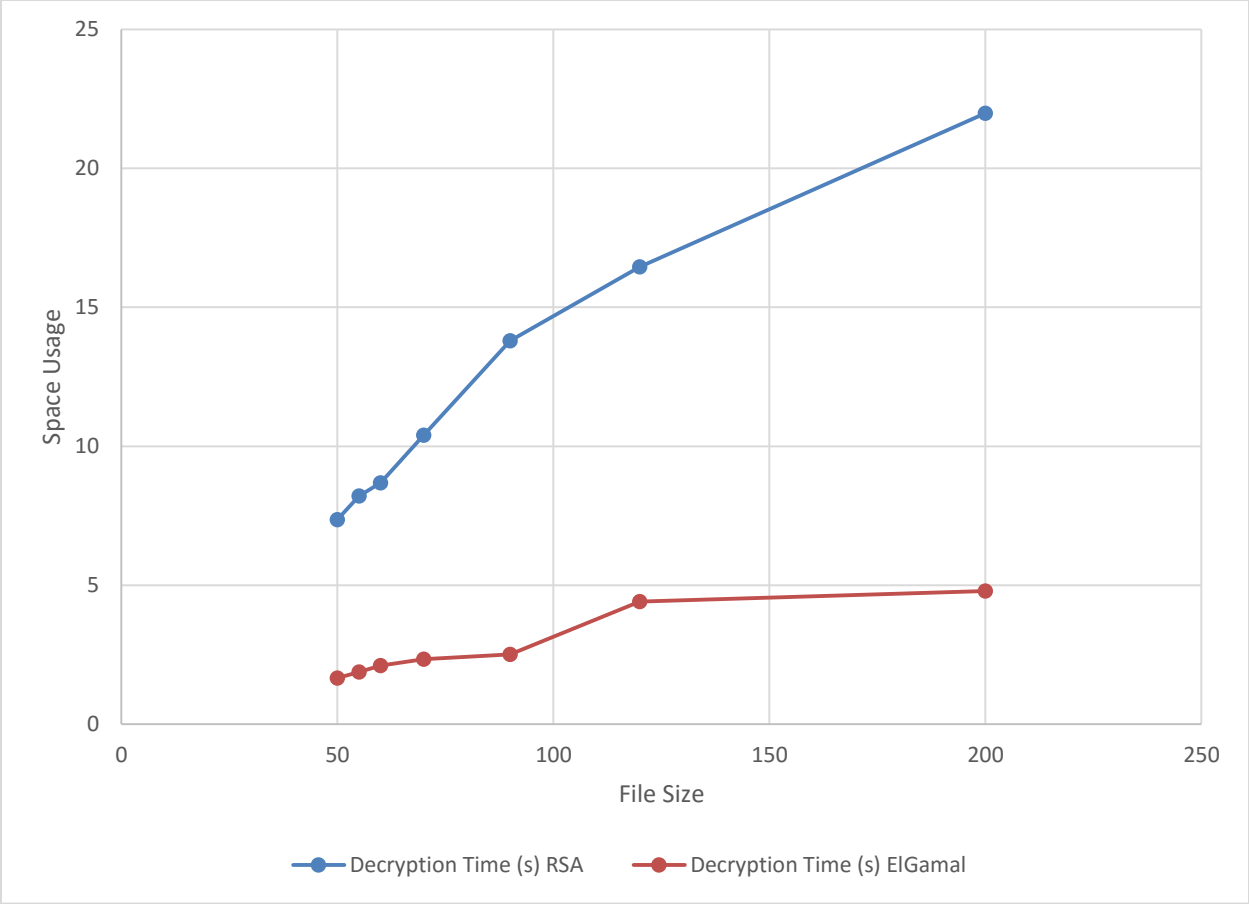

Figure 4.13: Decryption time for RSA and ElGamal Algorithms for Audio Data.

The computation time of RSA is longer than that of the ElGamal algorithm during the decryption of audio data of different file sizes. This shows that ElGamal algorithm performance is better in terms of computation time during the decryption process. Figure 4.14 shows the chart of computation speed for RSA and ElGamal algorithms on audio data of different file sizes.

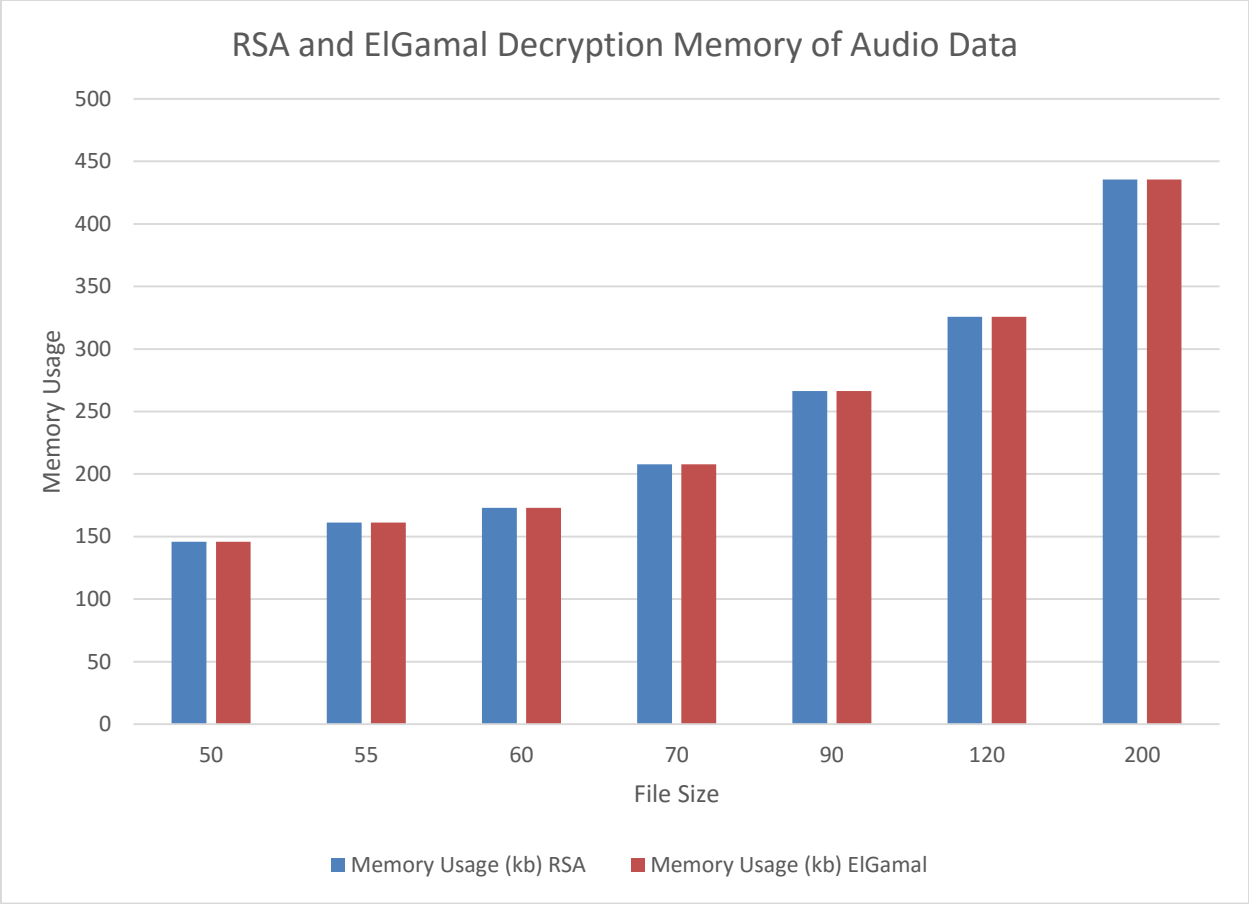

Figure 4.14: Decryption Memory Usage of Audio Data using RSA and ElGamal Algorithms.

The chart in figure 4.14 shows that both RSA and ElGamal algorithms consume the same amount of computation speed during decryption of audio data.

4.2.3 Discussion for RSA and ElGamal on Audio Data

The analysis of result from Table 4.5 and Figure 4.11 showed that the RSA algorithm was faster in terms of encrypting audio data it used a small time in encrypting audio data while ElGamal used higher time processing for encrypting audio data. Figure 4.12 showed that

ElGamal used a small amount of space for encrypting audio data while RSA used a high portion of memory for encrypting audio data.

From Table 4.6 and Figure 4.13, it was observed that ElGamal is faster in decrypting the encrypted files, and it has small time processing for decrypting audio file while RSA has a higher time processing for decrypting audio data. Figure 4.14 displayed that memory usage for decrypting audio data and it was observed that both RSA and ElGamal consume the same amount of memory for decrypting audio data.

D. Results for RSA and ElGamal on Video Data

Table 4.7: Encryption Time and Space Usage for RSA and ElGamal on Video Data

S/N	File Size (Kb)	Video Data Encryption Time (s)		Video Data Memory Usage (Encryption) (kb)	
		RSA	ElGamal	RSA	ElGamal
1	452	5.7833	54.8000	10853.30	1356.66
2	700	8.6895	81.5632	16585.40	2073.17
3	900	9.6173	88.5221	17841.30	2230.15
4	1372	17.3919	161.4096	32559.90	4069.97
5	3072	24.0244	222.1748	44514.91	5564.35
6	7608	52.1230	412.3847	56691.24	16691.01
7	10639	80.9202	749.1150	97942.83	18360.01

The statistical table shows the encryption time for both RSA and ElGamal in column 3, column 4 shows the computational speed while the second column shows the various video file sizes that were used for the encryption process. Figure 4.15 shows the chart of computational time for RSA and ElGamal algorithms on video data during the encryption process.

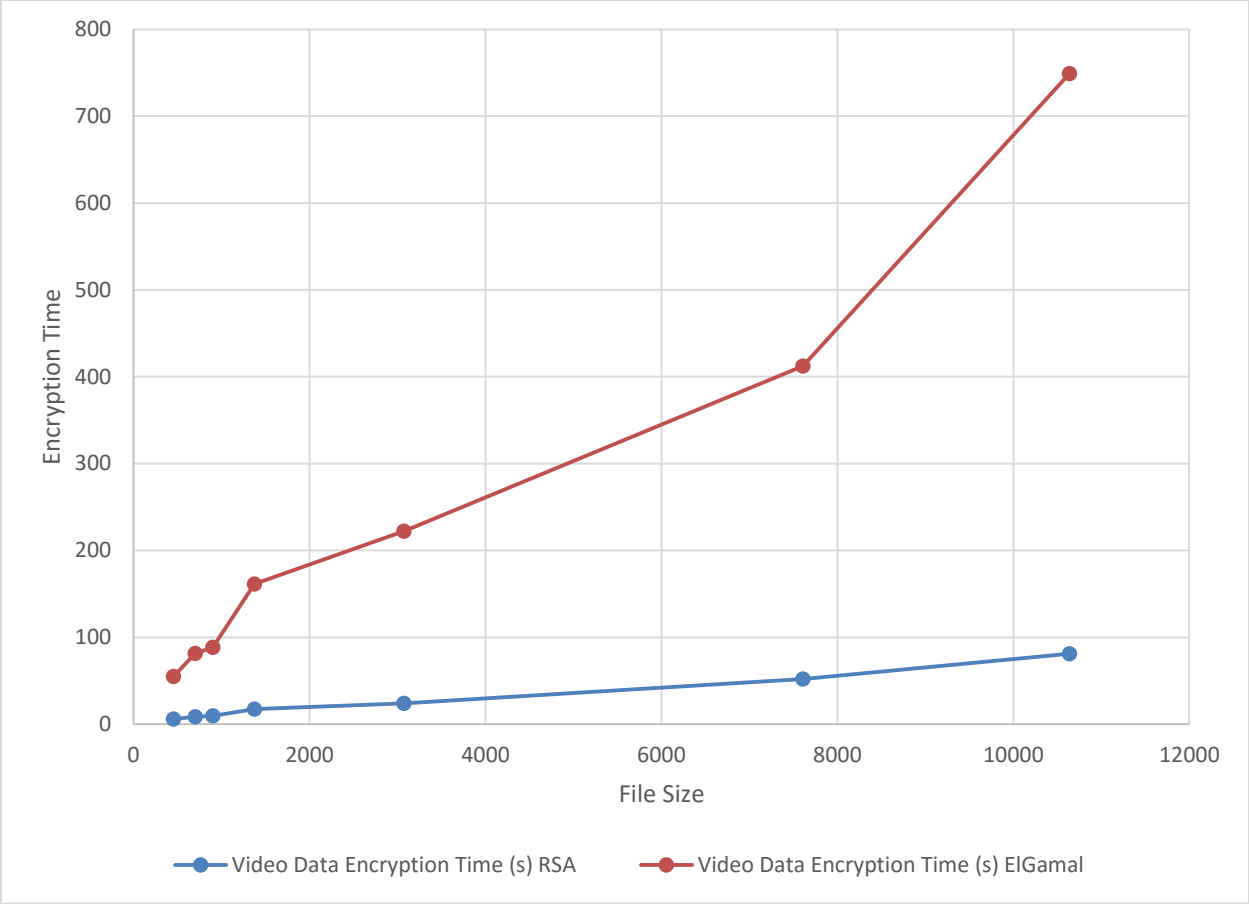

Figure 4.15: Chart of encryption time for RSA and ElGamal algorithm on video.

It is evident from figure 4.15 that the red line which denotes the ElGamal algorithm always takes longer computational time during the encryption process while the blue line which denotes RSA takes less computational time during the encryption process. Figure 4.16 shows the chart of computational speed for RSA and ElGamal on video data during the encryption process.

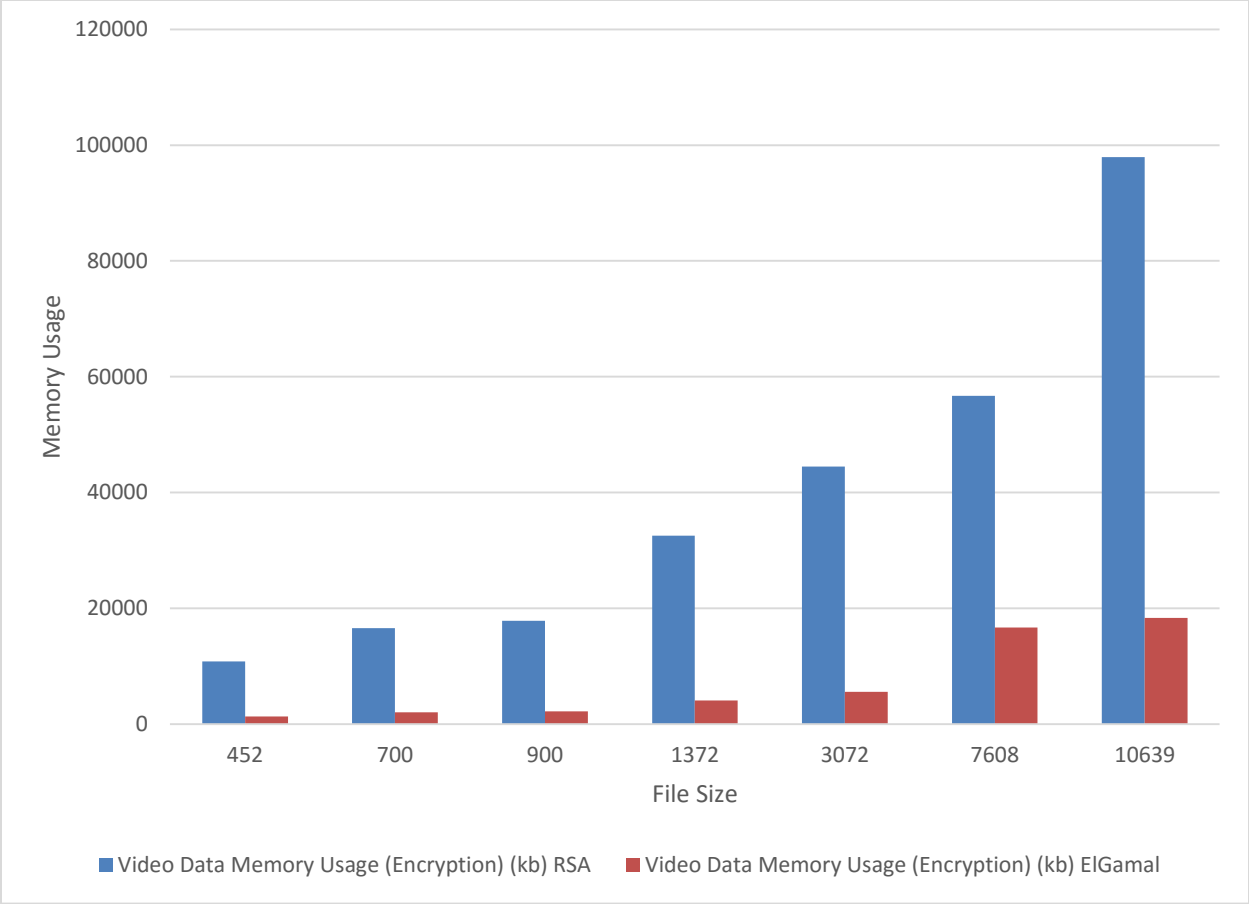

Figure 4.16: Memory usage during encryption of video data with RSA and ElGamal algorithms.

The column bar chart presented in figure 4.16 shows that the blue bar (RSA) has higher computational speed during the encryption of video data. The red bar (ElGamal) has smaller computational speed during encryption of video data. Table 4.8 shows the statistical data obtained from decrypting video data using RSA and ElGamal algorithms.

Table 4.8: Decryption Time and Space Usage of RSA and ElGamal for Video Data

S/N	File Size (Kb)	Video Data Decryption Time (s)		Video Data Memory Usage (Decryption) (kb)	
		RSA	ElGamal	RSA	ElGamal
1	452	69.3627	14.7762	1356.66	1356.66
2	700	105.5035	22.4683	2073.17	2073.17
3	900	114.6094	23.9383	2230.15	2230.15
4	1372	207.9480	45.0798	4069.97	4069.97
5	3072	268.2103	61.0020	5564.35	5564.35
6	7608	394.8201	98.5002	16691.01	16691.01
7	10639	566.3400	133.4137	18360.01	18360.01

The table gives the statistical data of computational time and speed generated during the decryption of video data using RSA and ElGamal algorithms. Figure 4.17 depicts the computation time for the RSA and ElGamal algorithms during video data decryption.

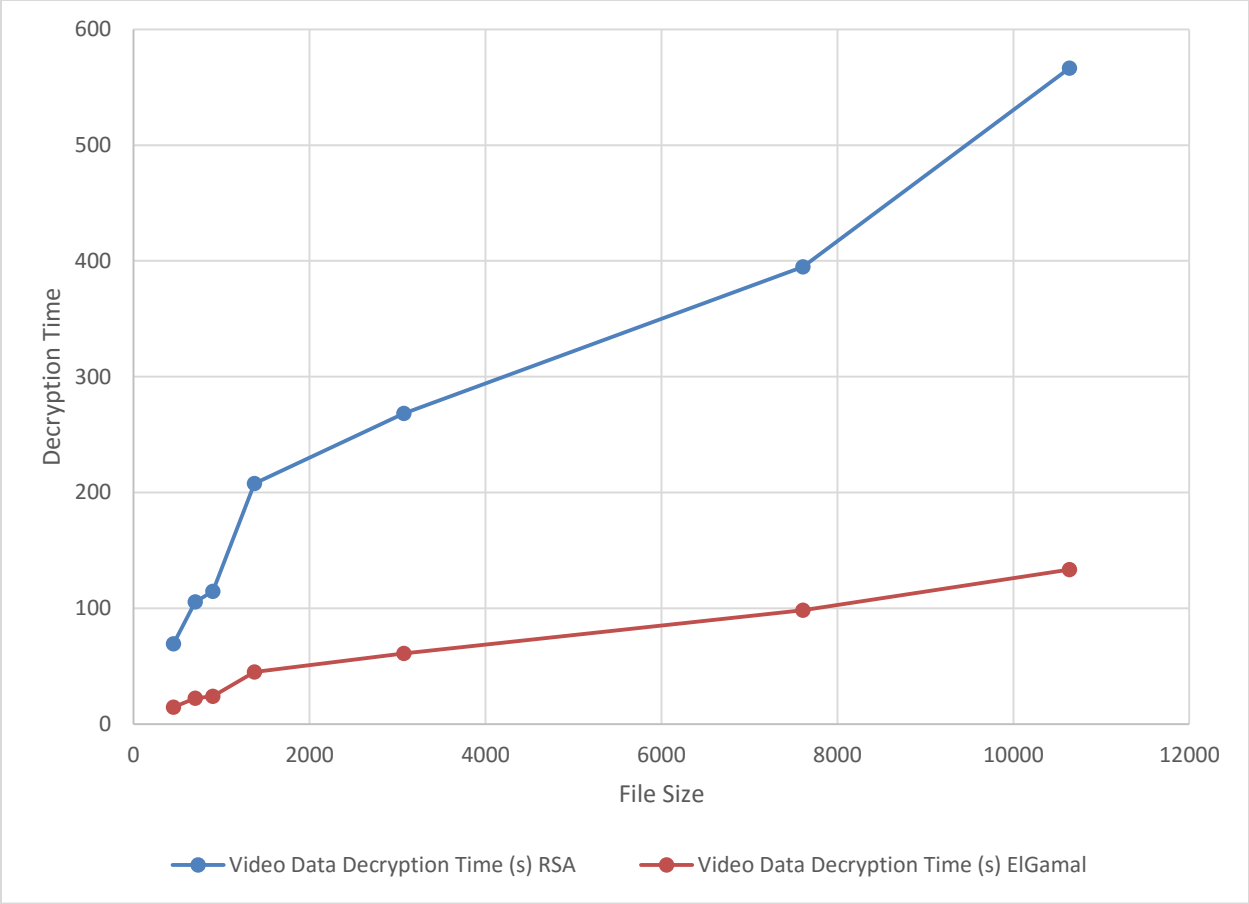

Figure 4.17: RSA and ElGamal Decryption Time obtained from Video Data

The computational time of the RSA algorithm (blue line) is longer during the decryption process of video data of different file sizes. While the ElGamal algorithm (red line) enjoys smaller computational time during the decryption process of video data. Figure 4.18 shows the chart of computational speed for RSA and ElGamal on video data during the decryption process.

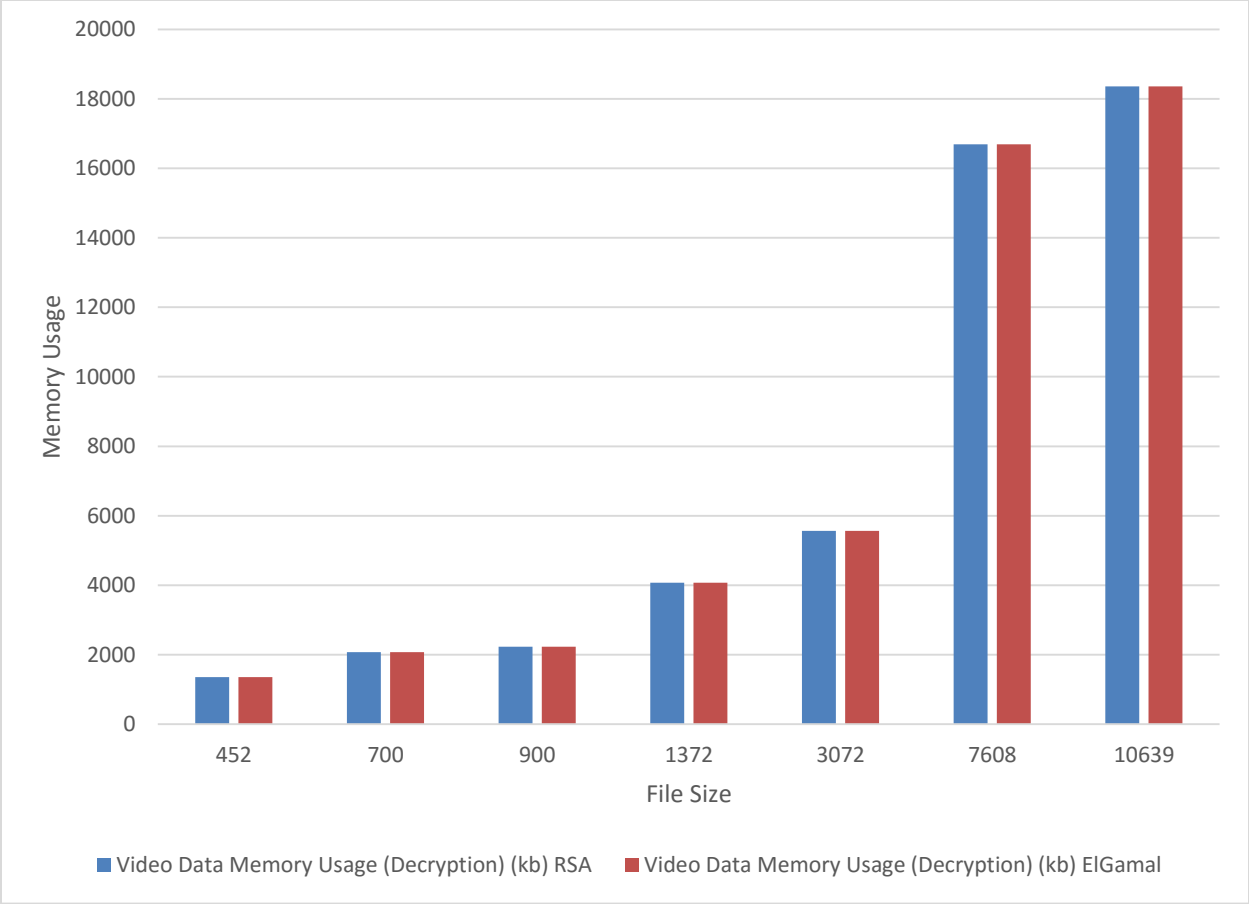

Figure 4.18: RSA and ElGamal Memory Usage from Video data Decryption Process.

The RSA algorithm, which is represented with a blue bar, and the ElGamal algorithm, which is represented with a red bar, have the same computational speed during decryption of video data.

4.2.4 Discussion for RSA and ElGamal on Video Dataset

Table 4.7, Figure 4.15 and Figure 4.16 displayed the time and space data result generated during the encryption of video data and the result was analyzed using graphs. From those tables and figures, RSA used smaller time during the encryption process of video data while ElGamal used higher time processing to encrypt video data. ElGamal used smaller memory

during the encryption process of video data while RSA generated large file which consumed more space during the encryption process.

Table 4.8, figure 4.17 and figure 4.18 displayed the result of time and memory usage during the decryption process. It was observed that ElGamal used lesser time during the decryption process compared to that of RSA. Both RSA and ElGamal used the same amount of memory during the decryption process of video data.

4.3 Discussions of Findings

In an attempt to evaluate the time and space complexity of RSA and ElGamal cryptographic algorithms, mixed data, consisting of text, image, audio and video data of different sizes were used in the computational evaluation to determine the behaviours of the algorithms. During the analysis, it was observed that RSA performed better than ElGamal

The experimental results obtained from all the dataset used (text, image, audio and video) as displayed in the tables and figures showed that the RSA algorithm outperformed the ElGamal algorithm during the encryption time of all categories of the dataset. This is in accordance with the existing works of literature (Farah *et. al.,* 2012; Kyaw, Kyaw and Nay, 2019; Zarni *et. al.,* 2019; Putri, Erna and Muhammad, 2020). Elgamal algorithm outperformed RSA algorithm during decryption time of all categories of data. This is consistent with the existing pieces of literature (Farah *et. al.,* 2012; Andysah, Elviwani, and Boni, 2018; Zarni *et. al.,* 2019). It was observed that the RSA algorithm-generated large file, and consumed more space (memory) during the encryption process of all categories of data. This is consistent with the existing literature (Andysah, Elviwani, and

Boni, 2018). ElGamal algorithm outperformed the RSA algorithm in terms of memory usage. It was also observed that both RSA and ElGamal algorithms consumed similar computational space for decrypting mixed data.

Therefore, this study used time and space performance metrics to evaluate which of the RSA algorithm and the ElGamal algorithm performs better when it comes to mixed data. RSA algorithm have lower computational time during encryption of text, image, audio and video dataset. ElGamal algorithm have lower computational time during the decryption of text, image, audio and video dataset. ElGamal algorithm consume less computational speed during the encryption of text, image, audio and video dataset compare to RSA that consume higher computational speed during the encryption process. Both algorithms (RSA and ElGamal) consume the same amount of computational speed during the decryption of text, image, audio and video dataset. Based on the various experimental results generated it was observed that the RSA algorithm is time-efficient while the ElGamal algorithm is a memory-efficient algorithm for all categories of data.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

In this study, the time and space of asymmetric cryptographic algorithms was studied due to the complexity challenges of two keys being used during encryption and decryption process. In determining the complexity of the cryptographic algorithm, it is important to consider the time complexity and space usage for effective comparative study. The comparative study of a cryptographic algorithm is the evaluation of the performance of the various cryptographic algorithm concerning time and memory usage based on the inputted values. Cryptographic algorithm complexity is a metric that determines the order of the number of operations executed by an algorithm depending on the scale of the data entry. Complexity is a general approximation of the activities to be conducted to execute an algorithm. The complexity of the algorithm is usually expressed by the $O(f)$ notation, also known as the asymptotic notation or the "Big $O\ notation$," where f is a function of the size of the input data. Asymptotic computational complexity $O(f)$ calculates the order of the used resource (CPU time, memory, etc.) by some algorithm represented as a function of the input file sizes.

The time complexity of the cryptographic algorithm is most generally calculated by calculating the number of elementary steps required by every algorithm to complete the execution. This is generally represented using the big $O\ notation$, which indicates the cumulative time the program has to run before it is finished. Space complexity is

characterized as the process of determining a function for predicting how much storage space is needed for the efficient execution of the algorithm. Memory space is commonly considered to be the main memory.

Data security is a major concern involving many areas, including computers and communications. A cryptography algorithm is a way to ensure data secrecy, authentication, integrity, availability and identity of user data. It helps to maintain and provide protection and privacy to user data. RSA and ElGamal algorithms are known as an asymmetric cryptographic algorithm that use both public and private key to encrypt and decrypt user data. RSA is a public-key encryption algorithm developed by Ron Rivest, Adi Shamir and Len Adleman. It is the most common cryptographic key asymmetric algorithm. It offers user information with both privacy and digital signature. It uses prime factors to produce public and private keys based on mathematical facts and to multiply large numbers together. ElGamal is an asymmetric key encryption algorithm based on the Diffie-Helman key exchange as an option to RSA for public-key encryption. ElGamal is also used for the generation of a digital signature algorithm called the ElGamal signature scheme. This study used both RSA and ElGamal cryptographic algorithms to determine the time and space usage on mixed (text, image, audio, and video) data.

From the literature, the performance evaluation of RSA and ElGamal has been compared with text, image and audio data in determining the running time of each of the algorithms and which of the algorithms performs better. The time performance of RSA and ElGamal were compared in the findings. RSA was picked to have performed better during the encryption time while ElGamal was picked to have performed better during the decryption

time (Zarni *et. al.*, 2019). RSA algorithm was concluded to have performed better than the ElGamal algorithm (Tin and Su, 2014; Kyaw, Kyaw and Nay, 2019).

5.2 Conclusion

In this study, RSA and ElGamal cryptographic algorithms were implemented to determine the time and space complexity of both algorithms on mixed (text, image, audio, video) data to achieve objective one. The experimental results showed that the RSA algorithm performs better in time complexity for all categories of data set (text, image, audio, and video) during the encryption process. RSA algorithm is better in terms of time complexity during the decryption of mixed data. ElGamal algorithm performs better in terms of memory consumption for both encryption and decryption process for all the categories of the dataset as the objective two was achieved.

The objectives three of this study was achieved as we were able to determine and analyzed the time and space complexity of both RSA and ElGamal cryptographic algorithms on text, image, audio and video data. Based on the comparative analysis of the time and space complexity of both RSA and ElGamal algorithms, it was discovered that RSA is a better algorithm when it comes to time complexity, that is, RSA can be said to be a time-efficient algorithm. ElGamal algorithm performed better than RSA in the memory usage aspect, therefore the ElGamal algorithm is said to be a memory-efficient algorithm.

5.3 Limitation

The following are some of the limitations of the study:

a. This work was implemented in a single programming language (C-Sharp).

b. Only two measurement metrics were used during implementation, that is, the time and space complexity.

5.4 Recommendation

The following propositions are recommended to further this research:

a. Additional larger data size should be used to implements both algorithms.

b. Complexity analysis should be carried out on other asymmetric algorithms for better decision making.

c. Other measurement metrics can be used to compare both algorithms.

5.5 Contribution to Knowledge

This study provides an addition to the body of knowledge by investigating the performance of selected cryptographic algorithms (RSA and ElGamal) in terms of computer resources usage (time and memory) on a mixed dataset (text, image, audio and video). This seeks to enhance decision making on which algorithms performs better concerning time and memory usage as well as the design of a high impact computer system.

5.6 Future works

As seen from the results obtained, the RSA algorithm is a more efficient algorithm for time complexity while ElGamal is a more efficient algorithm for space complexity. As future work, the two algorithms can be studied with the goal of getting more optimal space and time complexity for RSA and ElGamal respectively.

References

Andysah P., Elviwani, and Boni O. (2018). Comparative Analysis of RSA and ElGamal Cryptographic Public-key Algorithms. *Ina-Rxiv papers* 1(6):1-10.

Ankush S., Jyoti A., Aarti D. and Pratibha S. (2014) Implementation & Analysis of RSA and ElGamal Algorithm. *Asian J. of Adv. Basic Sci*.: 2(3): 125-129.

Asani, E.O, Omotosho, A. and Longe, O. B. (2018). A Real-time Gesture Engineered CAPTCHA. *International Journal of Mechanical Engineering and Technology* 9(12): 618-629.

Choi, J., Shin, Y., and Cho, S. (2018, January). Study on information security sharing system among the industrial IoT service and product provider. *In 2018 International Conference on Information Networking (ICOIN)*:551-555. IEEE.

Dindayal M and Dilip K.Y. (2018). Performance Analysis of RSA and Elliptic Curve Cryptography. *International Journal of Network Security*, 20(4): 625-635.

Farah, S., Javed, Y., Shamim, A., and Nawaz, T. (2012, December). An experimental study on performance evaluation of asymmetric encryption algorithms. *In Recent Advances in Information Science, Proceeding of the 3rd European Conf. of Computer Science*.121-124. EECS-12.

Gonzalez, C., Ben-Asher, N., and Morrison, D. (2017). Dynamics of decision making in cyber defense: Using multi-agent cognitive modeling to understand cyberwar. *In Theory and Models for Cyber Situation Awareness*. 113-127. Springer, Cham.

Haitner, I., and Vadhan, S. (2017). The many entropies in one-way functions. *In Tutorials on the Foundations of Cryptography*. 159-217. Springer, Cham.

Hong, J (2012). The state of phishing attacks. *Communications of the ACM*, 55 (1): 74-81.

Kayalvizhi, R., Vijayalakshmi, M., and Vaidehi, V. (2010, July). Energy analysis of RSA and ELGAMAL algorithms for wireless sensor networks. *In International Conference on Network Security and Applications*. 172-180. Springer, Berlin, Heidelberg.

Kyaw M.T., Kyaw S.H., and Nay A.A. (2019) Time Performance Analysis of RSA and Elgamal Public-Key Cryptosystems. *International Journal of Trend in Scientific Research and Development*. 3(6): 18-29.

Mondal, H. S., Hasan, M. T., Hossain, M. M., Arifin, M. M., and Saha, R. (2020, June). An RSA-Based Efficient Dynamic Secure Algorithm for Ensuring Data Security. *In Proceedings of International Joint Conference on Computational Intelligence*. 643-653. Springer, Singapore.

Okeyinka, A. E. (2015, October). Computational speeds analysis of RSA and ElGamal algorithms on text data. *In Proceedings of the world congress on engineering and computer science*.

Osman, A. M., Dafa-Allah, A., and Elhag, A. A. M. (2017, January). Proposed security model for web based applications and services. *In 2017 International Conference on Communication, Control, Computing and Electronics Engineering*. 1-6. IEEE.

Paverd, A., Tamrakar, S., Nguyen, H. L., Pendyala, P., Nguyen, D. T., Stobert, E., ... and Sadeghi, A. R. (2018). OmniShare: Encrypted Cloud Storage for the Multi-Device Era. *IEEE Internet Computing*. 22(4): 27-36.

Preeti C., Arjun C., and Atul K. (2018, July) “Multimedia Big Data Security”, *International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering*. 112-117

Priyadarshini P., Prashant N., Narayan D. and Meena S. (2015, December). “A Comprehensive Evaluation of Cryptographic Algorithms: DES”. *International Conference on Information Security & Privacy*. 11-12. Nagpur, INDIA.

Rashmi Singh, and Shiv Kumar (2012). Elgamal’s Algorithm in Cryptography. *International Journal of Scientific & Engineering Research*. 3(12): 1-4.

Sann, Z., thi Soe, T., Knin, K. W. M., and Win, Z. M. (2019). Performance comparison of asymmetric cryptography (case study-mail message). *APTIKOM Journal on Computer Science and Information Technologies*. 4(3): 105-111.

Shah, H., Sharma, V., Panchal, D., Patel, S., and Degadwala, S. (2018). A Comparative Study on Digital Encryption Algorithms. *Journal of Scientific & Engineering Research.* 3(2):101-121.

Shashi M. S., and Rajan M., (2011). Comparative Analysis Of Encryption Algorithms for Data Communication, *International Journal of Clothing Science and Technology.* 2(2): 11-31.

Siahaan A. P. (2016). “Genetic Algorithm in Hill Cipher Encryption,” *American International Journal of Research in Science, Technology, Engineering & Mathematics.* 15(1): 84–89.

Siahaan, A. P. U. (2018). Comparative Analysis of RSA and ElGamal Cryptographic Public-key Algorithms. *American International Journal Research in Science, Technology, Engineering & Mathematics. 16(3): 45-51.*

Singanjude, M. D., and Dalvi, R. (2020). Secure and Efficient Application of Manet Using RSA Using Vedic Method Combine With Visual Cryptography and Identity Based Cryptography Technique. *Available at SSRN 3570567*. Accessed 20/09/20.

Statistica, (2018). The number of active e-mail accounts worldwide from 2014 to 2019 (in millions). Available online: *https://www.statista.com/statistics/456519/forecast-number-of-active-email-accounts-worldwide/ Accessed 22/10/20.*

Tin Z. N., and Su W.P. (2014) Performance Analysis of RSA and ElGamal for Audio Security. *International Journal of Scientific Engineering and Technology Research*. 3(11): 2494-2498.

Viney B. P., and Singh, S. (2015, December). A hybrid data encryption technique using RSA and Blowfish for cloud computing on FPGAs. *In 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences.* 1-5. IEEE.

Wigderson, A. (2019). Mathematics and Computation: *A Theory Revolutionizing Technology and Science*. Princeton University Press.

Mahmood, K., Chaudhry, S. A., Naqvi, H., Kumari, S., Li, X., and Sangaiah, A. K. (2018). Elliptic curve cryptography based lightweight authentication scheme for smart grid communication. *Future Generation Computer Systems*. 81(1): 557-565.

Johnson, D., Menezes, A., and Vanstone, S. (2001). The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1): 36-63.

Zarni S, Thi T.S., Kaythi W. M., and Zin M.W. (2019). Performance comparison of asymmetric cryptography (case study-mail message). *APTIKOM Journal on Computer Science and Information Technologies*. 4(3):105-111.

Suguna, S., Dhanakoti, V., and Manjupriya, R. (2016). A Study on Symmetric and Asymmetric Key Encryption Algorithms. *International Research Journal of Engineering and Technology* (IRJET). 3(4): 27-31.

Aljawarneh, S., and Yassein, M. B. (2017). A resource-efficient encryption algorithm for multimedia big data. *Multimedia Tools and Applications*, 76(21): 22703-22724.

Abdullah, A. (2017). Advanced encryption standard (AES) algorithm to encrypt and decrypt data. *Cryptography and Network Security*, 16(1):1-10.

Mushtaq, M. F., Jamel, S., Disina, A. H., Pindar, Z. A., Shakir, N. S. A., and Deris, M. M. (2017). A survey on the cryptographic encryption algorithms. *International Journal of Advanced Computer Science and Applications*, *8*(11): 333-344.

Nureni A. A. and Sayyidina'Aliyy B.A. (2018). Comparative Analysis of Encryption Algorithms. *Covenant Journal of Informatics & Communication Technology*. 6(1):13-22.

Putri P.S, Erna B. N and Muhammad Z. (2020, June). Comparative study of LUC, Elgamal and RSA Algorithms in Encoding Texts. *In 2020 3rd International Conference on*

Mechanical, Electronics, Computer, and Industrial Technology (MECnIT). 148-151. IEEE.

Kumar, P., Rawat, S., Choudhury, T., and Pradhan, S. (2016, November). A performance based comparison of various symmetric cryptographic algorithms in run-time scenario. In 2016 International Conference System Modeling & Advancement in Research Trends (SMART). 37-41. IEEE.

Arora, Rachna, Anshu Parashar, and Cloud Computing Is Transforming. "Secure user data in cloud computing using encryption algorithms." *International journal of engineering research and applications*. 3(4): 1922-1926.

Boni, Sharad, Jaimik Bhatt, and Santosh Bhat (2015). "Improving the Diffie-Hellman Key Exchange Algorithm by Proposing the Multiplicative Key Exchange Algorithm." *International Journal of Computer Applications* 130(15): 8-19.

Yassein, M. B., Aljawarneh, S., Qawasmeh, E., Mardini, W., and Khamayseh, Y. (2017, August). Comprehensive study of symmetric key and asymmetric key encryption algorithms. *In 2017 International conference on engineering and technology* (ICET). 1-7. IEEE.

Kansal, S., and Mittal, M. (2014, December). Performance evaluation of various symmetric encryption algorithms. *In 2014 International Conference on Parallel, Distributed and Grid Computing*. 105-109. IEEE.

Costello, C., Longa, P., and Naehrig, M. (2016, August). Efficient algorithms for supersingular isogeny Diffie-Hellman. *In Annual International Cryptology Conference*. 572-601. Springer, Berlin, Heidelberg.

Suganya, K., and Ramya, K. (2014). Performance study on Diffie Hellman Key Exchange Algorithm. *International Journal for Research in Applied Science and Engineering Technology*. 12(7): 68-75.

Kaur, M., and Kaur, J. (2017). Data Encryption Using Different Techniques: *A Review. International Journal of Advanced Research in Computer Science*, 8(4). 12-19

Vasundhara, S. (2017). Elliptic curve Cryptography and Diffie-Hellman Key exchange. *IOSR Journal of Mathematics.* 13(1): 56-61.

Biswas, B. (2012). On a key exchange technique, avoiding man-in-the-middle-attack. *Journal of Global Research in Computer Science*, 3(9): 28-30.

Rabah, K. (2005). Theory and implementation of data encryption standard: *A review. Information Technology Journal*, 4(4): 307-325.

Suo, H., Wan, J., Zou, C., and Liu, J. (2012, March). Security on the internet of things: a review. *In 2012 International Conference on Computer Science and Electronics Engineering.* 648-651. IEEE.

Oduyiga, A. O. (2018). Security in Cloud Storage: A Suitable Security Algorithm for Data Protection.

Thambiraja, E., Ramesh, G., and Umarani, D. R. (2012). A survey on various most common encryption techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(7):7-15.

Singh, G. (2013). A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *International Journal of Computer Applications*, 67(19): 19-25.

Singh, P., and Singh, K. (2013). Image encryption and decryption using the blowfish algorithm in MATLAB. *International Journal of Scientific & Engineering Research*. 4(7): 150-154.

Rahim, R. (2017). Combination of the Blowfish and Lempel-Ziv-Welch algorithms for text compression.

Lee, B. H., Dewi, E. K., and Wajdi, M. F. (2018, April). Data security in cloud computing using AES under HEROKU cloud. *In 2018 27th Wireless and Optical Communication Conference*. 1-5. IEEE.

Talirongan, H., Sison, A. M., and Medina, R. P. (2018, November). Modified Advanced Encryption Standard using Butterfly Effect. *In 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management*. 1-6. IEEE.

APPENDIX

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Security;
using System.Security.Cryptography;
using System.IO;
using System.Collections;
using System.Diagnostics;

namespace File_Encrypter_Decrypyer
{
   public partial class frmEncDec: Form
   {
      private static int _keyLength = 0;
      private String _key = null;
      private String _cryptedText = "";
      private static Boolean _flag = false;
      private String _randomKey = "";
      string[] FileName, FilePath;



      private static String APPLICATION_NAME = "Time and Space Complexities of
RSA and ElGamal on Mixed Data";

      //image
      static int RSA_P = 0;
      static int RSA_Q = 0;
      static int RSA_E = 0;
      static int d = 0;
      static int n = 0;

      static string loadImage = "";
      static string loadcipher = "";
      static string loadvideo = "";

   //image stop
```

```
        public frmEncDec()
        {
            InitializeComponent();
        }

        private void frmEncDec_Load(object sender, EventArgs e)
        {
            btnEncrypt.Enabled = false;
            btnDecrypt.Enabled = false;
            disable_all();
            _flag = true; // by default ELGAMAL encryption is chosen
        }

        //Image Encryption and Decrypion Process
        public string encrypt(string imageToEncrypt)
        {
            //Stopwatch stopwatch = new Stopwatch();
            //stopwatch.Start();
            string hex = imageToEncrypt;
            char[] ar = hex.ToCharArray();
            String c = "";
            //MessageBox.Show("ar");
            progressBar1.Maximum = ar.Length;
            for (int i = 0; i < ar.Length; i++)
            {
                Application.DoEvents();
                progressBar1.Value = i;
                if (c == "")
                    c = c + File_Encrypter_Decrypyer.RSAalgorithm.BigMod(ar[i], RSA_E, n);
                else
                    c = c + "-" + File_Encrypter_Decrypyer.RSAalgorithm.BigMod(ar[i],
RSA_E, n);
            }
            return c;
            //stopwatch.Stop();
            //TimeSpan duration = stopwatch.Elapsed;
            //label6.Text = "Time Used: " + duration.TotalSeconds;
            //label5.Text = "Space Used: " + (pictureBox1.Width/pictureBox1.Height)
/1024.0 + "KB"; //rTxtBox.TextLength / 1024.0 + "KB";
        }

        public string decrypt(String imageToDecrypt)
        {
            char[] ar = imageToDecrypt.ToCharArray();
            int i = 0, j = 0;
```

```
            string c = "", dc = "";
            progressBar2.Maximum = ar.Length;
            try
            {
               for (; i < ar.Length; i++)
               {
                  Application.DoEvents();
                  c = "";
                  progressBar2.Value = i;
                  for (j = i; ar[j] != '-'; j++)
                     c = c + ar[j];
                  i = j;

                  int xx = Convert.ToInt16(c);
                  dc = dc + ((char)File_Encrypter_Decrypyer.RSAalgorithm.BigMod(xx, d,
n)).ToString();
               }
            }
            catch (Exception ex) { }

            return dc;
         }
      //End of the Process
         public String ReadFullFileData(string fileName)
         {
            TextReader tr = null;
            try
            {
               tr = File.OpenText(fileName);

               if (tr != null)
                  return tr.ReadToEnd();
            }
            catch (Exception ex)
            {
               MessageBox.Show(ex.Message, APPLICATION_NAME,
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            finally
            {
               tr.Close();
            }

            return null;
         }
```

```
        public void OpenFile(String title)
        {
            OpenFileDialog fileDialog = new OpenFileDialog();
            fileDialog.Title = title;
            fileDialog.Filter = "files (*.txt)|*.txt|Word Document (*.docx)|*.docx|PDF
(*.pdf)|*.pdf"; //"TEXT Files(*.txt)|*.txt";

            if (fileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                txtPath.Text = fileDialog.FileName;
                string dataToEncrypt = ReadFullFileData(fileDialog.FileName);
                rTxtBox.Text = dataToEncrypt;
                chkData.Checked = true;
            }
            else
            {
                return;
            }
        }

        public void SaveFile(String title)
        {
            TextWriter tw = null;
            try
            {
                SaveFileDialog saveFileDialog = new SaveFileDialog();
                saveFileDialog.Title = title;
                saveFileDialog.Filter = "TEXT Files(*.txt)|*.txt";
                if (saveFileDialog.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
                {
                    tw = File.CreateText(saveFileDialog.FileName);
                    tw.WriteLine(rTxtBox.Text);
                    /*if ((myStream = browsedfile.OpenFile()) != null)
                    {
                        txtBrowse.Text = browsedfile.SafeFileName;
                        string strfilename = browsedfile.FileName;
                        string filetext = File.ReadAllText(strfilename);
                        txtEncryptPlainText.Text = filetext;
                    }*/
                }
                else
                {
                    return;
                }
            }
```

```
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, APPLICATION_NAME,
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            finally
            {
                tw.Close();
            }
        }

        public static String RandomKeyString() // random key generator
        {
            StringBuilder builder = new StringBuilder();
            Random r = new Random();
            char ch;
            int size = r.Next(35, 70);

            for (int i = 0; i < size; i++)
            {
                ch = Convert.ToChar(r.Next(65, 122));
                builder.Append(ch);
            }

            return builder.ToString();
        }

        private void btnBrowse_Click(object sender, EventArgs e)
        {
            OpenFile("Open A File To Encrypt");
            btnEncrypt.Enabled = true;
            btnDecrypt.Enabled = true;
        }

        public void ELGAMALEncryption()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
            ELGAMALEncryptionDecryption ElGamalEnc = new
ELGAMALEncryptionDecryption();
            ElGamalEnc.EncryptionKey = _randomKey;
            lblKey.Text = "Elgamal Encryption Key";
            rTxtKey.Text = _randomKey;
            ElGamalEnc.InClearText = rTxtBox.Text;
            ElGamalEnc.ELGAMALEncryption();
            this._cryptedText = ElGamalEnc.CryptedText;
```

```
            rTxtBox.Clear();
            rTxtBox.Text = this._cryptedText;
            stopwatch.Stop();
            TimeSpan duration = stopwatch.Elapsed;
            label6.Text = "Time Used: " + duration.TotalSeconds * 60;
            label5.Text = "Space Used: " + rTxtBox.TextLength / 1024.0 + "KB";
        }

        public void ELGAMALDecryption()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
            ELGAMALEncryptionDecryption ElGamalEnc = new
ELGAMALEncryptionDecryption();
            ElGamalEnc.EncryptionKey = _randomKey;

            lblKey.Text = "ElGamal Decryption Key";
            rTxtKey.Text = _randomKey;

            ElGamalEnc.CryptedText = rTxtBox.Text;
            ElGamalEnc.ELGAMALDecryption();
            rTxtBox.Clear();
            rTxtBox.Text = ElGamalEnc.InClearText;
            stopwatch.Stop();
            TimeSpan decrytime = stopwatch.Elapsed;
            label6.Text = "Time Used: " + decrytime.TotalSeconds * 60;
            label5.Text = "Space Used: " + rTxtBox.TextLength / 1024.0 + "KB";
        }

        public void RSAEncryption()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
            _keyLength = Convert.ToInt32(numericUpDown.Value.ToString());
            string encryptedString =
RSAEncryptionDecryption.RSAEncryption(rTxtBox.Text, _keyLength, _key);
            lblKey.Text = "RSA Encryption Key";
            rTxtKey.Text = _key;
            rTxtBox.Clear();
            rTxtBox.Text = encryptedString;
            chkData.Checked = true;
            //label6.Text = "Time Used: " + (rTxtBox.TextLength / _keyLength)/60.0;
            //Stopwatch stopwatch = new Stopwatch();
            stopwatch.Stop();
            TimeSpan duration = stopwatch.Elapsed;
            label6.Text = "Time Used: " + duration.TotalSeconds;
```

```
            label5.Text = "Space Used: " + rTxtBox.TextLength/1024.0 + "KB";
        }

        public void RSADecryption()
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
            string decryptedString =
RSAEncryptionDecryption.RSADecryption(rTxtBox.Text, _keyLength, _key);
            rTxtBox.Clear();
            rTxtBox.Text = decryptedString;
            //calculate time of decryption
            stopwatch.Stop();
            TimeSpan decrytime = stopwatch.Elapsed;
            label6.Text = "Time Used: " + decrytime.TotalSeconds;
            label5.Text = "Space Used: " + rTxtBox.TextLength/1024.0 + "KB";
        }

        private void btnEncrypt_Click(object sender, EventArgs e)
        {
            try
            {
                if (String.IsNullOrEmpty(rTxtKey.Text))
                {
                    MessageBox.Show("You must generate key first to encrypt or decrypt text.",
APPLICATION_NAME, MessageBoxButtons.OK, MessageBoxIcon.Information);
                    return;
                }
                else if (!String.IsNullOrEmpty(rTxtKey.Text))
                {
                    if (rTxtBox.Text != null && !chkData.Checked) // data is from file
                    {
                        if (_flag) // user select ELGAMAL encrytion from Encryption Menu
                        {
                            ELGAMALEncryption();
                        }
                        else // user select RSA encrytion from Encryption Menu
                        {
                            RSAEncryption();
                        }

                        DialogResult result = MessageBox.Show("Want to save encrypted text to
file?", APPLICATION_NAME, MessageBoxButtons.YesNo,
MessageBoxIcon.Information);
                        if (result == System.Windows.Forms.DialogResult.Yes)
                        {
```

```
                    // then save the encrypted text to file
                    SaveFile("Save a encrypted text to file");
                }
                else
                {
                    return;
                }
            }
            else if (chkData.Checked && rTxtBox.Text != null)
            {
                // user write directly to a rich text box, not open a file from the browse
button
                // now first convert text to encrypted text, then prompt a user to save that
text or not

                if (_flag)
                {
                    ELGAMALEncryption();
                }
                else
                {
                    RSAEncryption();
                }

                DialogResult result = MessageBox.Show("Want to save encrypted text to
file?", APPLICATION_NAME, MessageBoxButtons.YesNo,
MessageBoxIcon.Information);
                if (result == System.Windows.Forms.DialogResult.Yes)
                {
                    // then save the encrypted text to file
                    SaveFile("Save a encrypted text to file");
                }
                else
                {
                    return;
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, APPLICATION_NAME,
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
```

```
        private void btnDecrypt_Click(object sender, EventArgs e)
        {
            try
            {
                if (rTxtBox.Text != null && !chkData.Checked)
                {
                    // means data that is to be decrypted is from a file, so first open the file by
clicking the button browse

                    OpenFile("Open A Encrypted File To Decrypt");

                    if (_flag)
                    {
                        ELGAMALDecryption();
                    }
                    else
                    {
                        RSADecryption();
                    }

                    DialogResult result = MessageBox.Show("Want to save decrypted text to
file?", APPLICATION_NAME, MessageBoxButtons.YesNo,
MessageBoxIcon.Information);
                    if (result == System.Windows.Forms.DialogResult.Yes)
                    {
                        // then save the encrypted text to file
                        SaveFile("Save a decrypted text to file");
                    }
                    else
                    {
                        return;
                    }
                }
                else if (chkData.Checked && rTxtBox.Text != null)
                {
                    // means data that is to be decrypted is written by the user directly to a rich
textbox,
                    // so simple decrypt it and prompt the user for its saving

                    if (_flag)
                    {
                        ELGAMALDecryption();
                    }
                    else
                    {
                        RSADecryption();
```

```
                }

                DialogResult result = MessageBox.Show("Want to save decrypted text to
file?", APPLICATION_NAME, MessageBoxButtons.YesNo,
MessageBoxIcon.Information);
                if (result == System.Windows.Forms.DialogResult.Yes)
                {
                    // then save the encrypted text to file
                    SaveFile("Save a decrypted text to file");
                }
                else
                {
                    return;
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, APPLICATION_NAME,
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }

    private void chkData_CheckedChanged(object sender, EventArgs e)
    {
        if (chkData.Checked)
        {
            btnEncrypt.Enabled = true;
            btnDecrypt.Enabled = true;
        }
        else
        {
            btnEncrypt.Enabled = false;
            btnDecrypt.Enabled = false;
        }
    }

    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void ElGamalToolStripMenuItem_Click(object sender, EventArgs e)
    {
        _flag = true;
    }
```

```
        private void rSAToolStripMenuItem_Click(object sender, EventArgs e)
        {
            _flag = false;
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            rTxtBox.Clear();
            rTxtKey.Clear();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void btnGenerateKey_Click(object sender, EventArgs e)
        {
            if (_flag)
            {
                lblKey.Text = "ElGamal Encryption Key";
                _randomKey = RandomKeyString(); // generating random key
                rTxtKey.Text = _randomKey;
            }
            else
            {
                _key = RSAEncryptionDecryption.RSAGenerateKey(_keyLength);
                string bitStrengthString = _key.Substring(0, _key.IndexOf("</BitStrength>") +
14);
                _key = _key.Replace(bitStrengthString, "");
                lblKey.Text = "RSA Encryption Key";
                rTxtKey.Text = _key;
            }
        }

        private void btnImage_Click(object sender, EventArgs e)
        {
            OpenFileDialog open1 = new OpenFileDialog();
            open1.Filter = "Image File(*.JPG)|*.JPG";

            if (open1.ShowDialog() == DialogResult.OK)
            {
                textBox1.Text = open1.FileName;
                pictureBox1.Image = Image.FromFile(textBox1.Text);
                button2.Enabled = true;
```

```
            FileInfo fi = new FileInfo(textBox1.Text);

            label9.Text = "File Name: " + fi.Name;
            label10.Text = "Image Resolution: " +
pictureBox1.Image.PhysicalDimension.Height + " X " +
pictureBox1.Image.PhysicalDimension.Width;

            double imageMB = ((fi.Length / 1024f) / 1024f);

            label11.Text = "Image Size: " + imageMB.ToString(".##") + "MB";
        }
        else
        {
            textBox1.Text = "";
            label9.Text = "File Name: ";
            label10.Text = "Image Resolution: ";
            label11.Text = "Image Size: ";

            //pictureBox1.Image = Properties.Resources.blank;
            button2.Enabled = false;

        }
    }

    private void disable_all()
    {
        button2.Enabled = false;
        groupBox4.Enabled = false;
        button4.Enabled = false;
        button5.Enabled = false;

        button9.Enabled = false;
        groupBox5.Enabled = false;
        button7.Enabled = false;
        button6.Enabled = false;
    }

    private void enable_all()
    {
        btnImage.Enabled = true;
        button2.Enabled = true;
        groupBox4.Enabled = true;
        button4.Enabled = true;
        button5.Enabled = true;
    }
```

```
        private void button5_Click(object sender, EventArgs e)
        {

        }

        private void button4_Click(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            loadImage =
BitConverter.ToString(File_Encrypter_Decrypyer.library.ConvertImageToByte(pictureB
ox1.Image));
            MessageBox.Show("Image Load Successfully");
            groupBox4.Enabled = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            loadImage =
BitConverter.ToString(File_Encrypter_Decrypyer.library.ConvertImageToByte(pictureB
ox1.Image));
            rTxtBox.Text = loadImage;
            MessageBox.Show("Image Load Successfully");
            groupBox4.Enabled = true;
        }

        private void button3_Click_1(object sender, EventArgs e)
        {
            if (button3.Text == "Set Details")
            {
                if (textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "")
                {
                    MessageBox.Show("Enter Valid Detail For RSA", "ERROR");
                }

                else
                {
```

```
                if
(File_Encrypter_Decrypyer.library.IsPrime(Convert.ToInt16(textBox2.Text)))
                {
                    RSA_P = Convert.ToInt16(textBox2.Text);
                }
                else
                {
                    textBox2.Text = "";
                    MessageBox.Show("Enter Prime Number");
                    return;
                }
                if
(File_Encrypter_Decrypyer.library.IsPrime(Convert.ToInt16(textBox3.Text)))
                {
                    RSA_Q = Convert.ToInt16(textBox3.Text);
                }
                else
                {
                    textBox3.Text = "";
                    MessageBox.Show("Enter Prime Number");
                    return;
                }

                RSA_E = Convert.ToInt16(textBox4.Text);

                textBox2.Enabled = false;
                textBox3.Enabled = false;
                textBox4.Enabled = false;
                button4.Enabled = true;
                button3.Text = "ReSet Details";
            }
        }
        else
        {
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            textBox2.Enabled = true;
            textBox3.Enabled = true;
            textBox4.Enabled = true;
            button4.Enabled = false;
            button3.Text = "Set Details";
        }
    }

    private void button5_Click_1(object sender, EventArgs e)
```

```
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
            n = File_Encrypter_Decrypyer.RSAalgorithm.n_value(RSA_P, RSA_Q);
            btnImage.Enabled = false;
            disable_all();
            String en = encrypt(loadImage);
            File.WriteAllText(textBox5.Text, en);
            //MessageBox.Show("Image Encrypted Successfully");
            btnImage.Enabled = true;
            stopwatch.Stop();
            TimeSpan duration = stopwatch.Elapsed;
            label6.Text = "Time Used: " + duration.TotalSeconds;
            label5.Text = "Space Used: " +
(1024)/(pictureBox1.Image.PhysicalDimension.Width +
pictureBox1.Image.PhysicalDimension.Height) + "KB"; //rTxtBox.TextLength / 1024.0
+ "KB";
        }

        private void button4_Click_1(object sender, EventArgs e)
        {
            SaveFileDialog save1 = new SaveFileDialog();
            save1.Filter = "TEXT|*.txt";
            if (save1.ShowDialog() == DialogResult.OK)
            {
                textBox5.Text = save1.FileName;
                button5.Enabled = true;
            }
            else
            {
                textBox5.Text = "";
                button5.Enabled = false;
            }
        }

        private void button10_Click(object sender, EventArgs e)
        {
            OpenFileDialog open1 = new OpenFileDialog();
            open1.Filter = "TEXT|*.txt";
            if (open1.ShowDialog() == DialogResult.OK)
            {
                textBox7.Text = open1.FileName;
                button9.Enabled = true;
            }
            else
            {
```

```
        textBox7.Text = "";
        button9.Enabled = false;
    }
}

private void button9_Click(object sender, EventArgs e)
{
    loadcipher = File.ReadAllText(textBox7.Text);
    MessageBox.Show("Load Cipher Image Successfully");
    groupBox5.Enabled = true;
}

private void button8_Click(object sender, EventArgs e)
{
    if (button8.Text == "Set Details")
    {
        if (textBox9.Text == "" || textBox8.Text == "")
        {
            MessageBox.Show("Enter Valid Detail For RSA", "ERROR");
        }
        else
        {
            if (Convert.ToInt16(textBox9.Text) > 0)
            {
                d = Convert.ToInt16(textBox9.Text);
            }
            else
            {
                textBox9.Text = "";
                MessageBox.Show("Enter Valid Number");
                return;
            }
            if (Convert.ToInt16(textBox8.Text) > 0)
            {
                n = Convert.ToInt16(textBox8.Text);
            }
            else
            {
                textBox8.Text = "";
                MessageBox.Show("Enter Valid Number");
                return;
            }

            textBox9.Enabled = false;
            textBox8.Enabled = false;
            button8.Text = "ReSet Details";
```

```
                button7.Enabled = true;

            }
        }
        else
        {
            textBox9.Text = "";
            textBox8.Text = "";
            textBox9.Enabled = true;
            textBox8.Enabled = true;
            button8.Text = "Set Details";
            button7.Enabled = false;
        }
    }

    private void button7_Click(object sender, EventArgs e)
    {
        SaveFileDialog save1 = new SaveFileDialog();
        save1.Filter = "JPG|*.JPG";
        if (save1.ShowDialog() == DialogResult.OK)
        {
            textBox6.Text = save1.FileName;
            button6.Enabled = true;
        }
        else
        {
            textBox6.Text = "";
            button6.Enabled = false;
        }
    }

    private void button6_Click(object sender, EventArgs e)
    {
        Stopwatch stopwatch = new Stopwatch();
        stopwatch.Start();
        button10.Enabled = false;
        disable_all();
        String de = decrypt(loadcipher);
        pictureBox1.Image =
File_Encrypter_Decrypyer.library.ConvertByteToImage(File_Encrypter_Decrypyer.libra
ry.DecodeHex(de));
        //MessageBox.Show("Decryption Done");
        pictureBox1.Image.Save(textBox6.Text,
System.Drawing.Imaging.ImageFormat.Jpeg);
        //MessageBox.Show("Picture Saved");
        button10.Enabled = true;
```

```
            TimeSpan duration = stopwatch.Elapsed;
            label6.Text = "Time Used: " + duration.TotalSeconds;
            label5.Text = "Space Used: " + (1024) /
(pictureBox1.Image.PhysicalDimension.Width +
pictureBox1.Image.PhysicalDimension.Height) + "KB"; //rTxtBox.TextLength / 1024.0
+ "KB";
        }

        private void btnElga_Click(object sender, EventArgs e)
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
            ELGAMALEncryptionDecryption ElGamalEnc = new
ELGAMALEncryptionDecryption();
            string jay = encrypt(loadImage);
            ElGamalEnc.EncryptionKey = _randomKey;
            lblKey.Text = "Elgamal Encryption Key";
            rTxtKey.Text = _randomKey;
            ElGamalEnc.InClearText = jay;
            ElGamalEnc.ELGAMALEncryption();
            this._cryptedText = ElGamalEnc.CryptedText;
            rTxtBox.Clear();
            rTxtBox.Text = this._cryptedText;
            stopwatch.Stop();
            TimeSpan duration = stopwatch.Elapsed;
            label6.Text = "Time Used: " + duration.TotalSeconds;
            label5.Text = "Space Used: " + rTxtBox.TextLength / 1024.0 + "KB";
        }

        private void btnVideo_Click(object sender, EventArgs e)
        {
            OpenFileDialog open1 = new OpenFileDialog();
            open1.Filter = "Video
File(mp3,wav,mp4,mov,wmv,mpg,avi,3gp,flv)|*.mp3;*.wav;*.mp4;*.3gp;*.avi;*.mov;*.
flv;*.wmv;*.mpg|all files|*.*";
            if (open1.ShowDialog() == DialogResult.OK)
            {
                textBox10.Text = open1.FileName;
                FileName = open1.SafeFileNames;
                FilePath = open1.FileNames;
                for (int i = 0; i <= FileName.Length - 1; i++)
                {
                    textBox10.Text=textBox10.Text + FileName[i];
                }
                axWindowsMediaPlayer1.URL = open1.FileName;
            }
```

```
        }

        private void btnLoadVideo_Click(object sender, EventArgs e)
        {
            //byte[] bytes =System.IO.File.ReadAllBytes(Selected-File);
            //txtyes.text=BitConverter.ToString(byte);
            byte[] bytes = System.IO.File.ReadAllBytes(axWindowsMediaPlayer1.URL);
            loadvideo = BitConverter.ToString(bytes);
            rTxtBox.Text = loadvideo;
            File.WriteAllText(textBox10.Text, loadvideo);
            MessageBox.Show("Video Load Successfully");
            //loadvideo =
BitConverter.ToString(File_Encrypter_Decrypyer.library.ConvertImageToByte(pictureB
ox1.Image));
        }

        private void btnKeyGen_Click(object sender, EventArgs e)
        {
            if (_flag)
            {
                lblKey.Text = "ElGamal Encryption Key";
                _randomKey = RandomKeyString(); // generating random key
                rTxtKey.Text = _randomKey;
            }
            else
            {
                _key = RSAEncryptionDecryption.RSAGenerateKey(_keyLength);
                string bitStrengthString = _key.Substring(0, _key.IndexOf("</BitStrength>") +
14);
                _key = _key.Replace(bitStrengthString, "");
                lblKey.Text = "RSA Encryption Key";
                rTxtKey.Text = _key;
            }
        }

        private void btnEncryptVideo_Click(object sender, EventArgs e)
        {
            Stopwatch stopwatch = new Stopwatch();
            stopwatch.Start();
            String en = encrypt(loadvideo);
            //File.WriteAllText(textBox5.Text, en);
            string encryptedString =
RSAEncryptionDecryption.RSAEncryption(rTxtBox.Text, _keyLength, _key);
            rTxtBox.Text = encryptedString;
            stopwatch.Stop();
            TimeSpan duration = stopwatch.Elapsed;
```

```
            label6.Text = "Time Used: " + duration.TotalSeconds;
            label5.Text = "Space Used: " + (1024) / (axWindowsMediaPlayer1.URL.Length);
//(pictureBox1.Image.PhysicalDimension.Width +
pictureBox1.Image.PhysicalDimension.Height) + "KB"; //rTxtBox.TextLength / 1024.0
+ "KB";
        }

        private void btnUploadAudio_Click(object sender, EventArgs e)
        {
            OpenFileDialog open1 = new OpenFileDialog();
            open1.Filter = "Audio File(*.mp3)|*.wav;*.ogg"; //open1.Filter = "Image
File(*.JPG)|*.JPG";
            if (open1.ShowDialog() == DialogResult.OK)
            {
                textBox11.Text = open1.FileName;
                FileName = open1.SafeFileNames;
                FilePath = open1.FileNames;
                for (int i = 0; i <= FileName.Length - 1; i++)
                {
                    textBox11.Text = textBox10.Text + FileName[i];
                }
                axWindowsMediaPlayer2.URL = open1.FileName;
            }
        }

        private void btnAudio_Click(object sender, EventArgs e)
        {
            //var bytes = File.ReadAllBytes(axWindowsMediaPlayer2.URL);
            //string result = System.Text.Encoding.ASCII.GetString(bytes);
              //  rTxtBox.Text=result;
                byte[] bytes = System.IO.File.ReadAllBytes(axWindowsMediaPlayer2.URL);
                loadvideo = BitConverter.ToString(bytes);
                rTxtBox.Text = loadvideo;
                MessageBox.Show("Audio File Loaded Successfully");
        }
    }
}



namespace File_Encrypter_Decrypyer
{
    partial class FrmSplash
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
```

```
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(FrmSplash));
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.label3 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.timer1 = new System.Windows.Forms.Timer(this.components);
            this.progressBar1 = new System.Windows.Forms.ProgressBar();
            this.label9 = new System.Windows.Forms.Label();
            this.label10 = new System.Windows.Forms.Label();
            this.groupBox1.SuspendLayout();
            this.SuspendLayout();
            //
            // groupBox1
            //
```

```
            this.groupBox1.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("groupBox1.BackgroundImage")));
            this.groupBox1.Controls.Add(this.label3);
            this.groupBox1.Controls.Add(this.label2);
            this.groupBox1.Controls.Add(this.label1);
            this.groupBox1.Font = new System.Drawing.Font("Tahoma", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.groupBox1.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(64)))),
((int)(((byte)(0)))));
            this.groupBox1.Location = new System.Drawing.Point(12, 12);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(509, 153);
            this.groupBox1.TabIndex = 4;
            this.groupBox1.TabStop = false;
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.BackColor = System.Drawing.Color.Transparent;
            this.label3.Font = new System.Drawing.Font("Georgia", 21.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label3.ForeColor = System.Drawing.Color.Blue;
            this.label3.Location = new System.Drawing.Point(23, 104);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(451, 34);
            this.label3.TabIndex = 5;
            this.label3.Text = "Cryptographic on Mixed Data";
            this.label3.Click += new System.EventHandler(this.label3_Click);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.BackColor = System.Drawing.Color.Transparent;
            this.label2.Font = new System.Drawing.Font("Monotype Corsiva", 26.25F,
((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Italic))), System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label2.ForeColor = System.Drawing.Color.Blue;
            this.label2.Location = new System.Drawing.Point(99, 61);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(303, 43);
            this.label2.TabIndex = 4;
            this.label2.Text = "of ElGamal and RSA";
            //
            // label1
            //
```

```
            this.label1.AutoSize = true;
            this.label1.BackColor = System.Drawing.Color.Transparent;
            this.label1.Font = new System.Drawing.Font("Georgia", 21.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.ForeColor = System.Drawing.Color.Blue;
            this.label1.Location = new System.Drawing.Point(23, 11);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(457, 34);
            this.label1.TabIndex = 3;
            this.label1.Text = "Time and Space Complexities ";
            this.label1.Click += new System.EventHandler(this.label1_Click);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.BackColor = System.Drawing.Color.Transparent;
            this.label8.Font = new System.Drawing.Font("Georgia", 15.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label8.ForeColor = System.Drawing.Color.White;
            this.label8.Location = new System.Drawing.Point(39, 253);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(228, 25);
            this.label8.TabIndex = 13;
            this.label8.Text = "Prof. A. E. Okeyinka";
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.BackColor = System.Drawing.Color.Transparent;
            this.label7.Font = new System.Drawing.Font("Georgia", 15.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label7.ForeColor = System.Drawing.Color.White;
            this.label7.Location = new System.Drawing.Point(182, 209);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(172, 25);
            this.label7.TabIndex = 12;
            this.label7.Text = "19PGCD00079";
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.BackColor = System.Drawing.Color.Transparent;
            this.label6.Font = new System.Drawing.Font("Georgia", 15.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label6.ForeColor = System.Drawing.Color.White;
            this.label6.Location = new System.Drawing.Point(82, 184);
```

```
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(367, 25);
            this.label6.TabIndex = 11;
            this.label6.Text = "ADENIYI ABIDEMI EMMANUEL";
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.BackColor = System.Drawing.Color.Transparent;
            this.label5.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label5.ForeColor = System.Drawing.Color.Red;
            this.label5.Location = new System.Drawing.Point(83, 234);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(108, 19);
            this.label5.TabIndex = 10;
            this.label5.Text = "Supervised by:";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.BackColor = System.Drawing.Color.Transparent;
            this.label4.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.ForeColor = System.Drawing.Color.Red;
            this.label4.Location = new System.Drawing.Point(219, 168);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(106, 19);
            this.label4.TabIndex = 9;
            this.label4.Text = "Developed by:";
            //
            // timer1
            //
            this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
            //
            // progressBar1
            //
            this.progressBar1.BackColor = System.Drawing.Color.Red;
            this.progressBar1.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(64)))), ((int)(((byte)(64)))),
((int)(((byte)(64)))));
            this.progressBar1.Location = new System.Drawing.Point(35, 281);
            this.progressBar1.Name = "progressBar1";
            this.progressBar1.Size = new System.Drawing.Size(457, 23);
            this.progressBar1.TabIndex = 14;
            //
```

```
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.BackColor = System.Drawing.Color.Transparent;
            this.label9.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label9.ForeColor = System.Drawing.Color.Red;
            this.label9.Location = new System.Drawing.Point(364, 234);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(132, 19);
            this.label9.TabIndex = 15;
            this.label9.Text = "Co-Supervised by:";
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.BackColor = System.Drawing.Color.Transparent;
            this.label10.Font = new System.Drawing.Font("Georgia", 15.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label10.ForeColor = System.Drawing.Color.White;
            this.label10.Location = new System.Drawing.Point(303, 253);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(193, 25);
            this.label10.TabIndex = 16;
            this.label10.Text = "Dr. M.O. Adebiyi";
            //
            // FrmSplash
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.BackgroundImage")));
            this.ClientSize = new System.Drawing.Size(531, 304);
            this.Controls.Add(this.label10);
            this.Controls.Add(this.label9);
            this.Controls.Add(this.progressBar1);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.groupBox1);
            this.Name = "FrmSplash";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Splash Screen";
            this.Load += new System.EventHandler(this.FrmSplash_Load);
```

```
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.ProgressBar progressBar1;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Label label10;
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.Drawing;

namespace File_Encrypter_Decrypyer
{
    class RSA algorithm
    {

        public static long square(long a)
        {
            return (a * a);
        }

        public static long BigMod(int b, int p, int m) //b^p%m=?
        {
            if (p == 0)
                return 1;
            else if (p % 2 == 0)
```

```
            return square(BigMod(b, p / 2, m)) % m;
        else
            return ((b % m) * BigMod(b, p - 1, m)) % m;
    }

    public static int n_value(int prime1, int prime2)
    {
        return (prime1 * prime2);
    }

    public static int cal_phi(int prime1, int prime2)
    {
        return ((prime1 - 1) * (prime2 - 1));
    }

    public static Int32 cal_privateKey(int phi, int e, int n)
    {
        int d = 0;
        int RES = 0;

        for (d = 1; ; d++)
        {
            RES = (d * e) % phi;
            if (RES == 1) break;
        }
        return d;
    }


}}
```