Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Al-Qadisiyah
College of Computer Science and Information Technology
Department of Computer Science

An Efficient Development Method for Speed up and Secure RSA Encryption Algorithm

A Thesis

Submitted to the Council of the College of Computer Science and Information Technology at the University of Al-Qadisiyah in Partial Fulfilment of the Requirements for the Degree of Master in Computer Science

By

Ali Najam Mahawash Al-Jubouri

Supervised by:

Assistant. Prof. Dr. Rana Jumaa Surayh Al-Janabi

2021 A.D **1443 A.H**

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

﴿قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا إِنَّكَ

أَنْتَ الْعَلِيمُ الْحَكِيمُ﴾

صدق الله العلي العظيم

سورة البقرة: الآية (32)

Supervisor Certificate

*I certify that thesis entitled "**An efficient development method for speed up and secure RSA encryption algorithm**" is prepared and written under my supervision at the department of Computer Science / College of Computer Science and Information Technology / University of Al-Qadisiyah as a partial fulfilment of the requirements of the degree of Master in Computer Science.*

Signature:
Supervisor Name: ***Dr. Rana Jumaa surayh AL-Janabi***
Date: / /2021

Head of Department Certificate

In view of the available recommendations, I forward the thesis entitled

*"**An efficient development method for speed up and secure RSA encryption algorithm**" for debate by the examination committee.*

Signature:
Head Name: ***Dr. Qusay Omran Mosa***
Head of the Department of Computer Science
Date: / 12 /2021

Certificate of the Examination Committee

*We, the undersigned, certify that (**Ali Najam Mahawash Al-Jubouri**) candidate for the degree of Master in Computer Science, has presented this thesis entitled (**An efficient development method for speed up and secure RSA encryption algorithm**) for debate examination. The examination committee confirms that this thesis is accepted in form and content and displays a satisfactory knowledge in the field of study based on the candidate demonstration during the debate examination held on: **24- November - 2021***

Signature:
Name: Dr. Sahar Adill Kadum
Title: Assistant Professor
Date: / / 2021
(Chairman)

Signature:
Name: Dr. Mustafa Jawad Radif
Title: Assistant Professor
Date: / /2021
(Member)

Signature:
Name: Dr. Luma Salal Hasan
Title: Dr.
Date: / / 2021
(Member)

Signature:
Name: Dr. Rana Jumaa surayh
Title: Assistant Professor
Date: / / 2021
(Supervisor and member)

Signature:
Name: Dr. Dhiah Eadan Jabor AL-shammary
Title: Assistant Professor
Date: / / 2021
(Dean of College of Computer Science and Information Technology)

Dedication

I would like to dedicate this work...

To Imam the twelfth, redeemer Al-Imam al-Mahdi (peace be upon him)

To my dear mother and my father's soul and I shouldn't forget my teachers

who played the most important role in helping me and providing useful

knowledge, particularly my supervisor,

To my dear brothers.

For all friends

To everyone who encourages and helps me to finish this search.

Ali Najam Mahawash

Acknowledgments

In the beginning and before everything, I thank God and praise him helping me to finish my thesis.

Then all thanks and appreciations to my supervisor

*" **Dr. Rana Al-janabi** " for granting me the time, effort, guidance, and encouragement to complete my thesis, I am very*

grateful to her and I feel proud because she is the Supervisor of my thesis.

***I would** like to extend my thanks to the professors of the Department of Computer Science and Information Technology at the University of Al-Qadisiyah, where I completed my research.*

***I would** like to present special thanks to my mother and wife for their efforts and support to me.*

As well as I don't forget to present my thanks to all my friends and everyone who contributes and encourages me to complete my thesis

Ali Najam Mahawash Al-Jubouri

2021

Abstract

The use of digital information has spread dramatically across the world. It is used in banks, financial markets, digital currencies, and more. This information is vulnerable to threats by hackers since information is over the network, and it may be insecure, causing violations across certain networks. RSA algorithm is used to provide confidentiality and authentication for this information.

RSA algorithm **(Rivest–Shamir–Adleman)** is a public-key cryptosystem widely used, Pretty Good Privacy (PGP), for secure data transmission. The acronym RSA comes from the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1978. It works by encrypting a message using a public key that's tied to a specific user; or they use it simply encrypts the session key with the receiver's public key and then appends the RSA-encrypted session key to the beginning encrypted document. Both the document and session key are sent together to the receiver.

The most important problems of the original RSA algorithm are slow speed and deterministic. This thesis presents a proposed algorithm for RSA. The proposed one is considered probabilistic through the use of a pseudo-random number generator (PRNG) with RSA, as well as, use Optimal Asymmetric Encryption Padding (OAEP) to reach a high level of security by preventing chosen-ciphertext attack which is one of the most popular attacks against RSA algorithm. To solve the slow speed of the RSA algorithm, RSA Key is divided into several keys to encrypt and decrypt blocks using multiple public and private keys according to the order of the random generator. The sequence of public and private keys depends on the seed number for the pseudo-random generator.

Dividing the key into several parts leads to an increase in the speed of encryption and decryption process. Using PRNG, OAEP and multiple keys can increase security. The proposed algorithm is resistant to several types of attack, brute force, chosen ciphertext and integer factorization problem attack, brute force attacks are impractical with the key length, chosen ciphertext attack cannot when OAEP is used. The integer factorization problem algorithm fails to factorize (N) with the size that is used.

Finally, when comparing the speed of the proposed RSA algorithm with the RSA algorithm (for key size 2048). The speed for the proposed algorithm has increased approximately as follows: segment 2 is twice, segment 4 is three times, segment 8 is four times and segment 16 is five times as fast as the RSA algorithm.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Full-Form
RSA	**Rivest-Shamir-Adleman**
PGP	**Pretty Good Privacy**
PKA	**public-keys algorithms**
MRSA	**Modified RSA**
AES	**Advanced Encryption Standard**
DES	**Data Encryption Standard**
RC2	**Rivest Cipher 2**
RC4	**Rivest Cipher 4**
DSA	**Digital Signature Algorithm**
PT	**Plain Text**
CT	**Cipher Text**
GCD	**The greatest common divisor**
MIME	**Multipurpose Internet Mail Extensions**
ASCII	**American Standard Code for Information Interchange**
SMTP	**simple mail transfer protocol**
OAEP	**Optimal Asymmetric Encryption Padding**
TLS	**Transport Layer Security**
SSL	**Secure Sockets Layer**
MR	**Miller Rabin**
RNG	**Random Number Generator**
HRNGS	**hardware-based number generators**
PRNG	**pseudorandom number generators**
MITM	**Man in the Middle**
Seg	**Segment**

TDES	**Triple Data Encryption Standard**
PR	**Private Key**
RC5	**Rivest Cipher 5**
MB	**megabyte**
MS	**Milliseconds**
FFT	**fast Fourier transform**
PK	**Public Key**
CCA	**Chosen cipher text attack**

List of the algorithms

Chapter one

General Introduction

1.1 Introduction

The security of data to maintain its confidentiality, proper access control, integrity, and availability is a significant issue in data communication. In today's information age, the need to protect communications from prying eyes is greater than ever before. As a result, the science of encryption plays a central role in mobile phone communication, electronic commerce, streaming services, private e-mail, and transmissions[1].

cryptography algorithms are used to achieve security. There are two types of cryptography algorithms symmetric and asymmetric, with symmetric-key algorithms, a single key is created via symmetric- or secret-key techniques, and it is utilized for both encryption and decryption. If numerous people intend to share a message encrypted in this manner, the secret key must be securely communicated[2].

On the other hand, asymmetric- or public-keys algorithms (PKA) utilize a pair of keys, one to generate the ciphertext and the other to decrypt it. The former is known as the public key and can be shared with anyone, while the latter is the private key, and is kept secret. A message encrypted by one of the keys can only be decrypted by the other. Increased data security is the primary benefit of asymmetric cryptography. It is the most secure encryption process because users are never required to reveal or share their private keys, thus decreasing the chances of a cybercriminal discovering a user's private key during transmission[3].

Asymmetric cryptography is more secure because it employs two separate keys: a public key that is only used to encrypt messages and can be shared with anybody, and a private key that is only used to decrypt messages and never needs to be shared[1]. As in the figure (1.1)

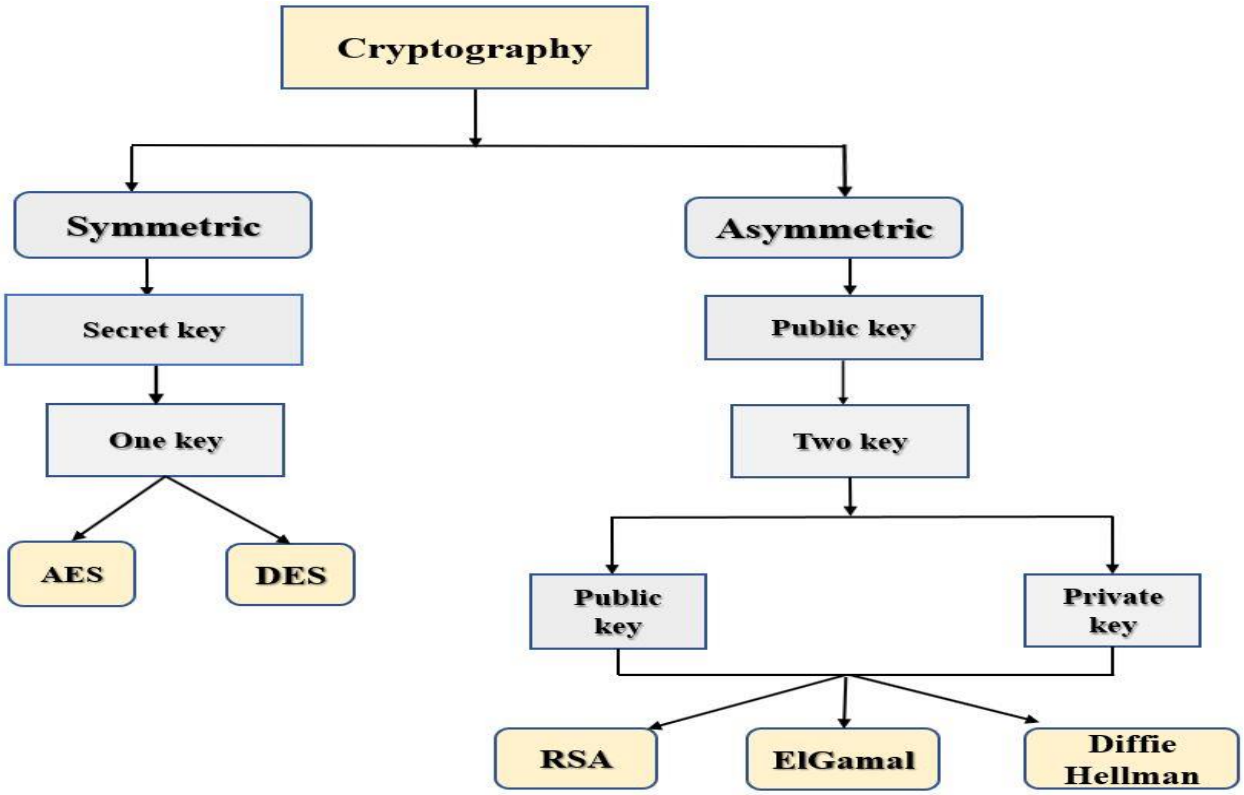


Figure (1.1). Types of Cryptography

1.2 Problem Statement

There are many problems with the RSA algorithm, the security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, Factoring the modulus of N is currently the most effective method for breaking the RSA algorithm, will reveal the secret key, which can then be used to decrypt any ciphertext All of the problems are summarized as follows:

1. Breaking RSA encryption (extracting the private key from the public key) is known as an RSA problem.
2. RSA security is based on the practical difficulty to factorize (N) into two large random prime numbers, the "factorization problem".
3. finding the decryption exponent d is equivalent to decomposing N into two random prime numbers big.

4. There is other problem of RSA algorithm. It is slow and deterministic algorithm.

On the other hand, finding the decryption exponent d is equivalent to factoring (N). So, the encryption and decryption process remain slow, and in general, the original RSA algorithm is a deterministic algorithm so it always produces the same ciphertext from a given plaintext and same key, so it is vulnerable to attack.

1.3 Previous Related Work

Many scientists have put forward some ideas to improve the RSA algorithm. The algorithms used in each case are discussed, along with a few recent and major modifications suggested.

Amare A. Ayele and Vuda Sreenivasarao proposed using two public key inputs, similar to the standard RSA algorithm. It uses some mathematical relationships and a pair of public keys that are sent separately. instead of directly sending a value. These additions make the algorithm harder to crack since an attacker needs to factorize two numbers instead of one, but the time needed for transformation is doubled[4].

R Patidar and R Bhartiya introduced a new algorithm principle that uses three prime numbers. They also proposed storing key parameters offline. A database that is identical across all networks is used to store the RSA key pairs. All parameters used in the RSA algorithm are stored before the algorithm is performed [5].

This increases the speed of the RSA algorithm during data exchange over the network[6]. Using three primes increases the complexity and time required to do the transformations.

This implementation can only be used in a closed network system and cannot be used in public systems on the internet without the predefinition of such offline datasets, which is not possible, limiting the use-case of this implementation[6].

R Minni, K Sultania, worked on eliminating the distribution of n, which is the large number whose factor, if found, compromises the RSA algorithm[7]. However, this implementation still suffers from the drawback of a time increase proportional to the key size.

M. A. Islam and B. Shabnam, has proposed a method instead of two prime numbers used "n" distinct prime numbers, which increases the attacking time to find the big prime number, Modified RSA (MRSA) needs a longer key generation period since it is based on a large factor value "N" The longer it takes to generate a key, the longer it takes to break the system[8].
The disadvantage of this implementation is that takes time longer.

I. Jahan, M. Asif, and L. J. Rozario has been published introducing a new algorithm This study focuses on number theory and public-key cryptosystems, with the goal of making the RSA cryptosystem more secure. To encrypt the message, the RSA cryptosystem generates a single public key. While it is difficult to find the factors of n to obtain p and q, (two large prime numbers), our proposed algorithm encryption keys are sent separately rather than all at once[9].
The proposed RSA algorithm is used in a context that demands high security while still being slow.
Narander Kumar and Chaudhary proposed a modified RSA algorithm in 2016, to build an RSA algorithm that is based on n-prime numbers. this strategy employs prime numbers Because big primes numbers are difficult to factorize[10].

Aarushi, Shitanshu, introduce entitled "Modified RSA Cryptographic System with Two Public keys and Chinese Remainder Theorem".

The aim of the research is to increase network security and improve the speed of the decryption process in the RSA algorithm, where the two researchers suggested using four random numbers to generate public and private keys.

The algorithm's private key values were calculated with a more complex degree than the process of generating private keys in the basic algorithm, and the decoding speed was improved using the Chinese remainder theory, after conducting the experiment process for the proposed design, it was found that the decoding time The encryption of the RSA algorithm according to the developed model for 512-bit data was 12 milliseconds, compared to the basic algorithm, where the encryption time was 14 milliseconds[11].

Farheen Sultana, et al, provide a modify: "Study of data encryption using AES and RSA" The study aimed to secure the confidentiality and protection of data transmitted over a computer network, using a system A hybrid of AES and RSA algorithms, which was based on MATLAB without mentioning the type and the specifications of the device to perform the following, first: encrypting the message text with the secret key of the AES algorithm, and secondly: encrypting the secret key using the public key of the RSA algorithm.

In the research, a comparison was made between the AES and RSA algorithms in terms of encryption time and security level.

The result of the comparison shows that the AES algorithm Faster than the RSA algorithm, while the latter is more secure than AES because it depends on two keys in encryption and decryption[12].

All of them have a problem about security, efficiency and performance. So that we will be tried to solve this problem through a proposed approach.

1.4 Aim of the thesis

The main aim of this thesis is to develop and build for keys an RSA cipher algorithm that efficiently provides secure communication to enable users to safely communicate and send encrypted data (such as images and text) via an internet network not secure. The following steps summarize the development of the algorithm:

1. Development of the RSA algorithm cipher method by using multiple random 2048-bit public and private keys.
2. The newly developed RSA algorithm requires less time and is faster at encrypting random data.
3. The encryption process of the modified RSA algorithm relies on a secret key and a seed key that must be confidentially agreed upon by the sender and the recipient. This random sequence is a necessary component of the encryption process to avoid an exhaustive search attack.
4. Reduce the amount of information accessible by unauthorized parties.
5. Create an optimal encryption algorithm based on random dynamic keys.

1.5 Contributions

1- The proposed RSA algorithm Encrypt and decrypt data using multiple keys to increase security and less time.

2- we added in the suggested structure, message encrypt and decrypt in a non-sequential (random way) using several public and private keys whose order is tied to a random generator (PRNG) by (seed key).

3- In proposed structure was used of Optimal Asymmetric Encryption Padding (OAEP) in RSA make the algorithm more randomness and convert it from deterministic into a probabilistic scheme.

4- Because the proposed algorithm is non-deterministic, so it is secure from the Chosen ciphertext attack.

1.6 Thesis Layout

The thesis of the study is structured as follows:

Chapter 1: Provides an overview of the many types of cryptography. In this chapter, there is a brief introduction to the issue statement and the work done by many researchers in the field of a cryptographic method for data protection.

Chapter 2: This chapter some background material about the types of cryptographic systems information about the RSA cryptosystem, including definitions, structure, advantages/Disadvantages, Strengths, and Weaknesses (Factorization Problem).

Chapter 3: Includes a detailed description and various security issues related to RSA. Introduces the proposed design, explains the practical stages of encryption and decryption a new algorithm is proposed which is stated in this chapter.

chapter 4: Presented the results of the performance and security of the proposed design and includes a comparison between the proposed approach and the original RSA algorithm.

Chapter 5: Introduces the conclusion and proposals for future work of the work done in this thesis.

Chapter Two

Theoretical Background

2.1 Introduction

Computer security has become an increasing concern for enterprises and individuals alike. Protecting information and data is relatively simple when stored on a single device, as a computer may be physically secured and requires login information to access. But as networks and the internet are becoming much more common for messages, e-mails, and other data transmissions, it is essential to protect transferred data from access by unauthorized individuals, which is a more challenging task [17].

Passwords are no longer considered to be as reliable for this activity since their small range makes them easy to guess. Additionally, if the password range is limited, a brute force search can be used to crack it [18].

In addition, even well-established cryptography methods become less effective as technology progresses and computers become more powerful, necessitating the constant improvement of these algorithms to ensure they remain secure capable of protecting transmitted information.

This can be ensured by developing the proposed algorithms and improving existing ones. To keep pace with the development of modern technical systems with the huge increase in the volume of data required to be processed in order to increase the speed of data encryption, however, conducting the development process to improve the performance of the algorithm's work requires studying and analyzing its working mechanism and how it is used in the encryption and decryption processes based on the concepts of number theory.

The Figure (2.1) below illustrates cryptography operation.

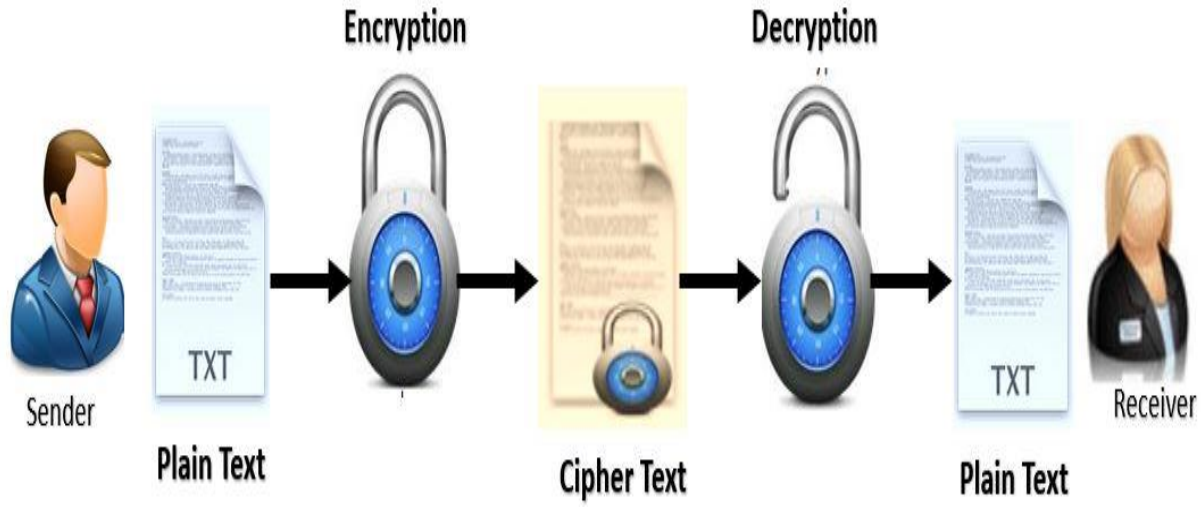


Figure (2.1). Encryption and Decryption

2.2 Type of Cryptography

Cryptography is divided into two types: 1) symmetric cryptography and 2) asymmetric cryptography. Following is a flowchart that illustrates the types of cryptography (2.2).

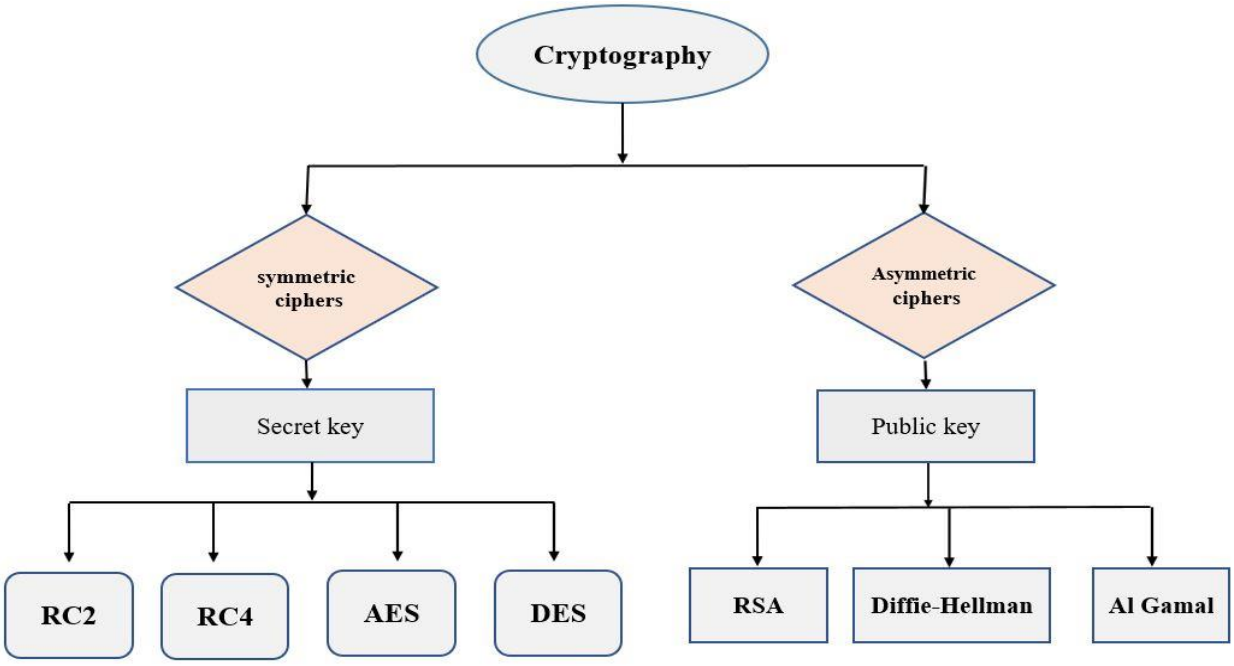


Figure (2.2). Classification of Cryptography

2.2.1 Symmetric Cryptography

It is the oldest form of cryptography and has long been used to secure data. The systems that use this algorithm must provide a safe method of transferring the private key to the recipient, because if the transfer is intercepted, the algorithm and the system are quickly compromised. As a result, when designing symmetric encryption algorithms, determining a secure method of transferring the key is vital[2].

Symmetric-key algorithms are cryptographic algorithms that encrypt plaintext and decrypt ciphertext using the same cryptographic key. Either the keys are identical, or one of the keys is the result of performing a simple transformation on the other [9]. In practice, the keys represent a shared secret used to maintain a secure information link between two or more authorized parties [10].

Requiring that all parties have access to the secret key is one key drawback of symmetric-key encryption compared to asymmetric (or public-key) encryption [11].

2.2.1.1 Advantages of Symmetric Encryption

Symmetric encryption more efficient and requires less time than asymmetric encryption, due to its lesser bandwidth and computational requirements. As a result, it is usually used to encrypt large amounts of data. A good use case involves an offline system such as a database. In this case, only administrators of the database might have access to the key. The speed of symmetric key encryption is substantially faster than the speed of asymmetric key encryption. Reduces the number of computing resources used. Single-key encryption requires fewer computing resources than public-key encryption[13].

Figure (2.3) depicts symmetric coding in further detail.

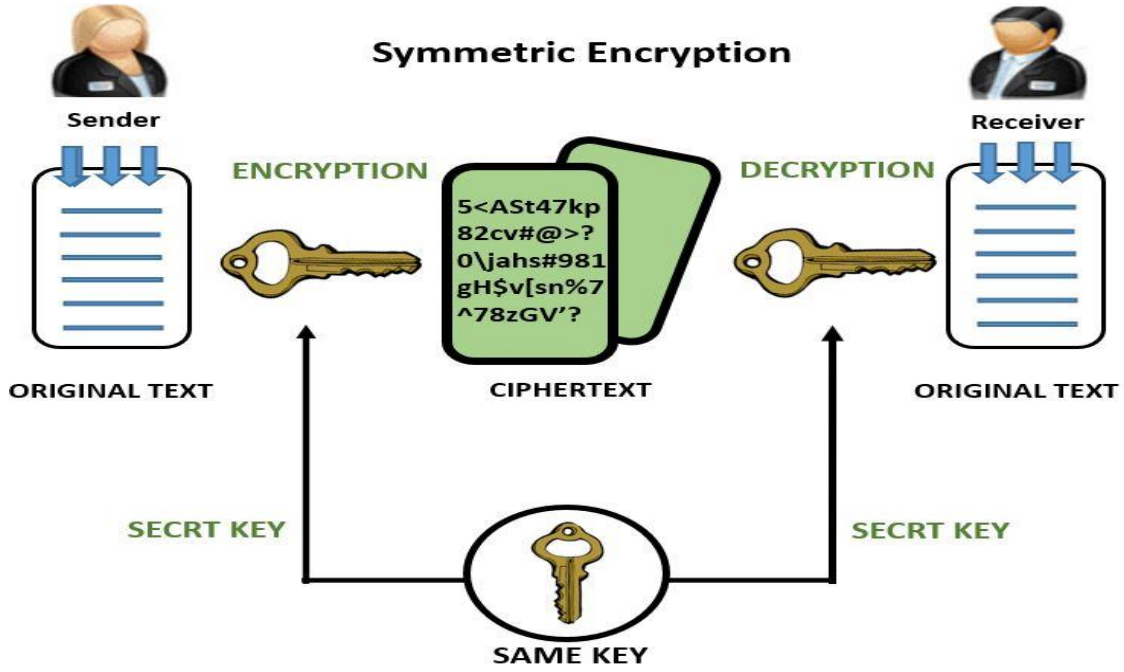


Figure (2.3). Symmetric key Cryptography

2.2.1.2 Disadvantages of Symmetric Encryption

1. The need for a secure route for secret key exchange: In symmetric key encryption, sharing the secret key at the start is a concern. It must be exchanged in such a way that it remains confidential.
2. Origin and authenticity of the message cannot be guaranteed: Since both sender and receiver use the same key, messages cannot be verified to have come from a particular user, this might be a problem.
3. The only secure way of exchanging keys would be exchanging personally[14].

And Some examples applications of symmetric cryptography

1. Electronic payments where protecting any personally identifiable information is essential to the prevention of identity theft or fraudulent charges.
2. Digital signatures algorithms (DSA) to confirm the identity of a message's sender.

3. Random number generation.
4. Hashing algorithms[14].

2.2.1.3 Symmetric encryption algorithms Examples

1. AES (Advanced Encryption Standard) and DES (Data Encryption Standard), both block ciphers.
2. RC4 (Rivest Cipher 4), a stream cipher.

2.2.2 Asymmetric Cryptography

Asymmetric cryptography, also known as public-key cryptography, is the second type of cryptography that encrypts and decrypts data using separate private and public keys. The keys are only two big numbers related mathematically; they are asymmetric, which means that they are not identical.

The public key is published to everyone. The private key, on the other hand, must not be revealed. the public key can be used to encrypt a message. The other key is used only for decryption[15]. Figure (2.4) depicts asymmetric coding in further detail.

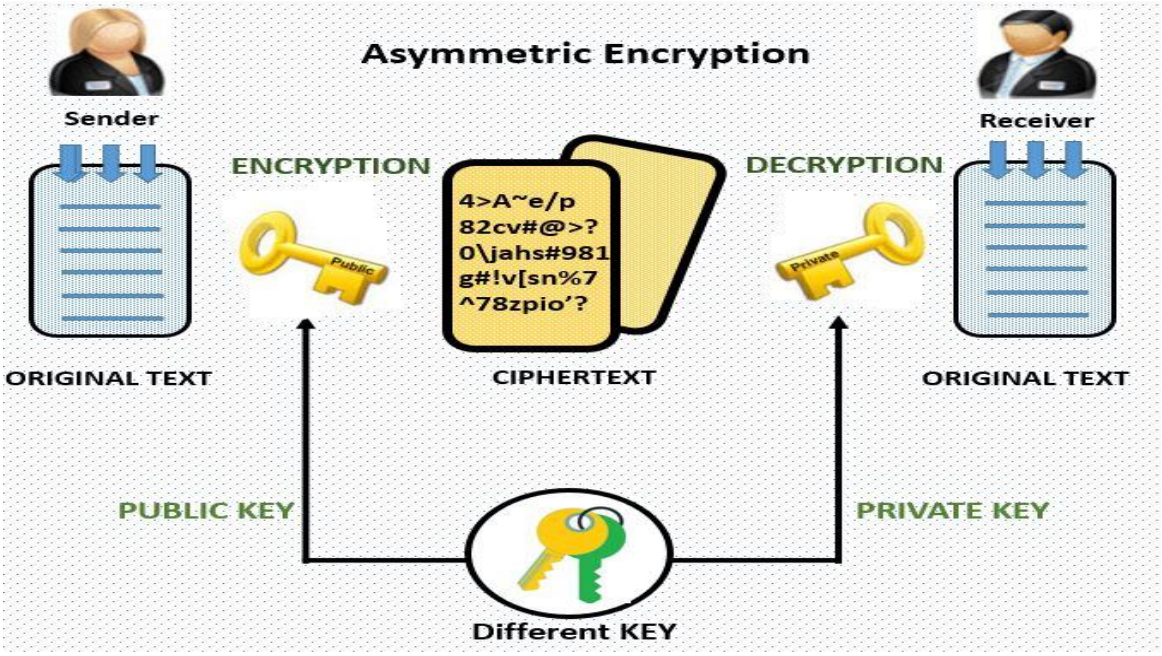


Figure (2.4) Asymmetric key Cryptography

2.2.2.1 Services of Asymmetric Cryptography

1. Confidentiality

Confidentiality is the most common use of Asymmetric Encryption. It is accomplished by having the sender encrypt the sensitive data with the receiver's public key and then having the receiver decrypt it with their private key[16].

2. Authenticity using Digital Signatures

A sender encrypts a message using their private key and exchanges it with the receiver. By decrypting the message with the sender's public key, the receiver can verify that the message was sent from the expected party, hence ascertaining the sender's authenticity[16].

3. Information integrity

A hash of the data to be exchanged is created and encrypted using the sender's private key. The encrypted hash and data are then exchanged with the receiver.

Using the sender's public key, the receiver decrypts the hash and generates a hash from the sent data. Any difference between the two hashes indicates that the content was altered after signing and that integrity has been lost. This kind of integrity check is used in digital currency transactions[15].

2.2.2.2 Advantages and Disadvantages of asymmetric cryptography

1. Increased confidentiality and security are the main benefits of public-key cryptography. Private keys never need to be transferred or released to others. This is in contrast to symmetric cryptography where the secret key must be transmitted, presenting a potential risk.
2. Another essential benefit of asymmetric algorithms is that they can provide a method of verifying digital signatures. On the other hand, authentication via secret-key systems requires certain secrets to be shared and often.
3. The encryption/decryption process is in reverse relation to your key length, the drawback is that it is not suited for the encryption of huge messages.
4. The downsides of using public-key cryptography are the required time and computational resources[17].

Although secret-key encryption techniques are often faster than their public-key counterparts, asymmetric encryption offers advantages that make it more appealing for many scenarios[18].

And Examples of Asymmetric Encryption Algorithms:

1. RSA (Rivest–Shamir–Adleman) Asymmetric Encryption Algorithm.
2. The (El Gamal) encryption system,
3. Diffie–Hellman key exchange algorithm.

Comparison Between Symmetric and Asymmetric Key Encryption

Symmetric Key Encryption	Asymmetric Key Encryption
It only requires a single key for both encryption and decryption.	It requires two key one to encrypt and the other one to decrypt.
The size of cipher text is same or smaller than the original plain text.	The size of cipher text is same or larger than the original plain text.
The encryption process is very fast.	The encryption process is slow.
It is used when a large amount of data is required to transfer.	It is used to transfer small amount of data.
It only provides confidentiality.	It provides confidentiality, authenticity and non-repudiation.
Examples: 3DES, AES, DES and RC4	Examples: Diffie-Hellman, ECC, El Gamal, DSA and RSA.
In symmetric key encryption, resource utilization is low as compared to asymmetric key encryption.	In asymmetric key encryption, resource utilization is high.

2.3 Mathematical Background of Algorithms

Before discussing the details of algorithms, it's important to understand how they're created. Algorithms are calculated values that are based on mathematical theory, functions, logic, and other factors. These algorithms are primarily based on mathematical theories. In the design of cryptographic algorithms, several mathematical concepts from numbers theory are important.

This section offers a description of the principles, along with evidence of the theorems used in these algorithms. The different theorems that are further used in the RSA algorithm in our work plan have been elucidated[18].

2.3.1 Number Theory

The theory of numbers in cryptography is commonly used. It is particularly important for design public key algorithms. Only the aspects of number theory that are relevant to the RSA algorithm in cryptography will be covered.

2.3.2 Modular Arithmetic

In some situations, we think about the remainder of an integer when it is divided by some specified positive integer. Let ‘a’ be an integer and ‘m’ be a positive integer. We denote by a mod m = r, the remainder when ‘a’ is divided by ‘m’.

Example: 17 mod 3 = 2, 5 mod 2 =1, –8 mod 7 = 6 and so on.

Basically, r ≡ a (mod m) if r = a - km for some integer k. If ‘r’ is non-negative and ‘a’ is between 0 and m, you can think of ‘a’ as the remainder of ‘r’ when divided by ‘m’. Sometimes, ‘a’ is called the residue of ‘r’, modulo m. Sometimes ‘r’ is called congruent to a, modulo m. The triple equals sign ‘≡’ denotes congruence.

Modular arithmetic is just like other arithmetic in which commutative, associative, and distributive laws obey. Also, reducing each intermediate result modulo m yields the same result as doing the whole calculation and then reducing the end result modulo m[19].

$(a + b) \bmod m = [(a \bmod m) + (b \bmod m)] \bmod m$

$(a - b) \bmod m = [(a \bmod m) - (b \bmod m)] \bmod m$

$(a * b) \bmod m = [(a \bmod m) * (b \bmod m)] \bmod m$

$(a * (b + c)) \bmod m = [((a * b) \bmod m) + ((a * c) \bmod m)] \bmod m$

2.3.3 Prime Numbers

A central concern of number theory is the study of prime numbers. Prime numbers play a very big role in cryptography, An integer $p > 1$ is a prime number if and only if its only divisors are ± 1 and $\pm p$ (It is an integer greater than one, and is divisible only by itself and by one) Prime numbers play a critical role in number theory and in the cryptographic techniques. If P is the set of all prime numbers[18].

Example of prime numbers in the following form: **2, 3, 7, 23, 29, 163 and many more.**

The rest of the numbers that are greater than one and are not prime are called Composite numbers, an example of a Composite numbers: **4 (since it is divisible by 2),100 (is divisible by 2 and 5).**

This means if we want to know the number x is prime or not, we will search from the beginning 2(because one not prime) until we reach the root of x. We test each of these numbers, is x divisible? On it, if this is achieved, we will have known that the number is not prime, and if it is not achieved, then the number is prime. Regard to

small numbers, the process of determining the prime number is simple, as in the following example:

For example, we have the number **101**, we start the test from **2**, to the root of **101**, which is **10**.

Is 101 divisible by 2. No

Is 101 divisible by 3, no?

Is 101 divisible by 4, 5, 6, 7, 8 and 9? Until we reach 10, and also does not accept, if The result is that **101** is a prime number[20].

For many cryptographic algorithms, it is necessary to select one or more very large prime numbers at random. Thus, we are faced with the task of determining whether a given large number is prime. There is no simple yet efficient means of accomplishing this task[18][21].

2.3.4 Greatest common divisor (GCD)

The greatest common divisor (GCD), also called the greatest common factor, of two numbers is the largest number that divides them both. Let a and b be integers, not both zero. Then the greatest common divisor (GCD) of a and b is the largest positive integer which is a factor of both a and b. We use gcd (a, b) to denote this largest positive factor. For instance, the greatest common factor of 20 and 15 is 5, since 5 divides both 20 and 15 and no larger number has this property. The concept is easily extended to sets of more than two numbers: the GCD of a set of numbers is the largest number dividing each of them[22].

The GCD is used for a variety of applications in number theory, particularly in modular arithmetic and thus encryption algorithms such as RSA. It is also used for simpler applications, such as simplifying fractions. This makes the GCD a rather

fundamental concept to number theory, and as such a number of algorithms have been discovered to efficiently compute it[23].

What is the greatest common divisor of 54 and 24?

The number 54 can be expressed as a product of two integers in several different ways:

54*1 = 27*2 = 18*3 = 9*6

Thus, the divisors of 54 are: 1,2,3,6,9,18,27,54
Similarly, the divisors of 24 are: 1,2,3,4,6,8,12,24

The list of numbers that the two numbers (54 and 24) have in common are the common denominators of 54 and 24:1,2,3,6

The largest of these is 6. That is, the greatest common divisor of 54 and 24 is 6, One writes: gcd(54,24)=6[24].

Two numbers are said to be "prime among themselves" Prime Relatively if their greatest common divisor is 1. The following pairs of numbers are first added to each other Prime Relatively 8 and 9, because their greatest common denominator is 1. GCD (44,23) =1, The greatest common divisor is calculated using Euclid's algorithm, which is one of the most important and oldest algorithms. For the two variables(a,b), we get the greatest common divisor by following the algorithm (2.1) below [25].

Algorithm (2.1) greatest common divisor (GCD)

Input**: number **a** and number **b

Output**: greatest common divisor of **a** and **b

***Begin**:*
***Step1**: IF (b==0) then*

***Return** (a)*

***Step2**: Else*

***Step3**: GCD (b, a MOD b)*

***Step 4**: Repeat step3*

***Return** GCD*
End

An example of calculating the greatest common divisor of the two numbers, GCD (2881, 2345).

GCD (2345, 2881 mod 2345) = GCD (2345, 536)

GCD (536, 2345 mod 536) = GCD (536, 201)

GCD (201, 536 mod 201) = GCD (201, 134)

GCD (134, 201 mod 134) = GCD (134, 67)

GCD (67, 134 mod 67) = GCD (67, 0)

For the two of them, it is the biggest common denominator. GCD (2345,2881) =67

Example: gcd (9,22) = 1. Here 9 and 22 are not real prime numbers but only 1 is the common factor between them so, 9 and 22 are relatively prime. A prime number is relatively prime to all other numbers except its multiples[22].

2.3.5 Euclid's algorithm

Euclidean algorithm is a simple procedure for determining the greatest common divisor of two positive integers. Nonzero b is defined to be a divisor of a if a = mb for some m, where a, b, and m are integers[26]. We will use the notation gcd (a, b) to mean the greatest common divisor of a and b. The positive integer c is said to be the greatest common divisor of a and b if c is a divisor of a and of b and any divisor of a and b is a divisor of c.

The Euclidean algorithm is based on the following theorem: For any nonnegative integer a and any positive integer b, gcd (a,b) = gcd(b, a mod b)[18].

An example could be if a = 20 and b = 15, gcd (20, 15).

This sequence helps supply an algorithm to compute the Greatest Common Divisor for two numbers. and the way it works is that a function is declared as gcd and takes two parameters which are a and b; returns the Greatest Common Divisor[26][27].

Example: gcd (48,14) by Euclid's theorem is

gcd (48,14) = gcd (14,6)

gcd (14,6) = gcd (6,2)

gcd (6,2) = gcd (2, 0)

The gcd (a,b) is the last non-zero remainder. Therefore, in above example gcd (48,14) = 2 [19].

2.3.6 Euler's Theorem

Before presenting Euler's theorem, we need to introduce an important quantity in number theory, referred to as Euler's totient function and written φ (n), defined as the number of positive integers less than n and relatively prime to **n**. By convention, φ (1) = 1[28].

1). These numbers are also used in public-key algorithms.

Suppose, if n is prime, then φ(n) = n- 1.

If n = p * q, where p and q are prime,

then φ(n) = (p-1) (q-1)

Example: φ (37) =36 and φ (35) = 24 for a prime number p, φ (p) = p-1

Now suppose that we have two prime numbers p and q, with p not equal q. Then it can be show that for n = pq, φ (n) = φ (pq) = φ (p) x φ (q) = (p-1) * (q-1)

Euler's theorem states that for every a and n that are relatively prime [18].

2.4 Multiplicative inverse (MI)

The multiplicative inverse of a number, say N, is represented by 1/N or N^{-1}. It is also called reciprocal**,** derived from the Latin word "reciprocus."' The meaning of inverse is "opposite." The reciprocal of a number obtained is such that when it is multiplied with the original number, the value equals the identity 1.

In other words, it is a method of dividing a number by its own to generate identity 1, such as N/N = 1.

- **Fact:** When a number is multiplied by its multiplicative inverse, the resultant value equals one

The multiplicative inverse is an essential operation in cryptographic systems; public-key cryptography has given rise to such a need, in which we need to generate a related public/private pair of numbers, each of which is the inverse of the other. One of the best methods for calculating the multiplicative inverse is the Extended-Euclidean method.

- Choose two different prime numbers, p, and q.
- Compute n = p q.
- Compute $\varphi(n)=(p-1)\,(q-1)$.
- Choose an integer e such that $1< e < \varphi(n)1$ and **e** with **φ(n)** are coprime. Then, e is released as the public key exponent.
- Determine d as $d\equiv e^{-1} \bmod \varphi(n)$; i.e., d is the multiplicative inverse of e (modulo φ(n)). We keep d as the private key exponent.
  Thus, the public key is (e, n), and the private key is (d, n).

Based on this description, e and n directly determine d. In other words, the public key is used to derive the private key. The private key cannot be directly determined from the public key due to the computational effort required to the find the multiplicative inverse. To understand this further, we must examine what the key generator knows which an attacker does not, and how it is used to determine the private key.

In general, $a^{-1} \equiv x \pmod n$ has a unique solution if 'a' and 'n' are relatively prime. If 'a' and 'n' are not relatively prime then $a^{-1} \equiv x \pmod n$ has no solution. If **n** is a prime number, then every number from 1 to n, 1 is relatively prime to n and has exactly one inverse modulo **n** in that range[19].

Example:

To find the inverse of 3 (mod 26)

Since gcd (26,3) =1, an inverse exists to we can use the Euclidean algorithm to solve for it. 23

26 =3. 8 +2

3 =1 · 2+1

2=2. 1+0

GCD (3,26) =1

you now must "backtrack" your steps like this:

1 = 3-2

1=3-(26-8. 3) =3-26+8.3

1= 1.3-26+8.3 =9.3-26

$3^{-1} = 9$

When taken modulo 26, this last equation gives $1 \equiv 3 \cdot 9 \bmod 26$, demonstrating that 9 is the inverse of 3 modulo 26.

2.5 System encode Base64

Base64 is a group of encoding schemes used to similar binary-to-text encoding using a fixed set of 6-bit text characters in order to transfer it across a channel which does not reliably support content besides text.

The name Base64 is derives from the fact that the encoding uses 64 characters, which typically include lowercase and uppercase letters (62 characters) and 2 extra symbols which vary between implementations, although “+” and “/” are typically the 63rd

and 64th characters[29]. The term Base64 originates from a specific MIME content transfer encoding Each base64 digit represents exactly 6 bits of data. The encoding works by every transform the 8-bits into 6-bits character[30]. The "=" symbol is used to pad the encoded output if the number of bytes of the unencoded input is not a multiple of 3. Base64 is most commonly used to encode data to be transferred over the web[31].

Up until the late 1960s, different computer systems used various schemes to encode their data. When ASCII was introduced in 1968, it used a 7-bit encoding and was adopted as the standard to data exchange. Initially, some protocols only transferred data in groups of 7 or even 6 bits. Furthermore, encodings for line endings and ASCII characters 10 and 13 (line feed and carriage return) differed across systems. This meant that Base64 was an appealing format for transferring data between these systems. Base64 is also used in programming to represent binary data as text[32].

The Base64 algorithm is good at an information security system for encryption and decryption. Base64 is easy to apply; it simply transforms 8-bit characters into 6-bit characters. However, Base64 is an encoding technology that simply delivers data in a different format.

It communicates the same data in a different syntax. The Base64 algorithm encodes the characters into a limited character set to make transmissions easier. Base64 algorithm is one of the ideal encryption processes for data transmission [30]. Base64 is not an encryption method, but it is a standard encoding for data There is no such thing as a secret, protection, or encryption [33]. The history of Base64 begins with e-mail. At that time, e-mail was sent via SMTP (simple mail transfer protocol) to a mail server, then sent to a mailbox at the mail server destination. A protocol is a set of rules by which computers communicate via a network[30][34]. Sometimes,

Base64 is used to encrypt the plaintext, but it does not have the key. Everyone can decrypt the message by knowing the table pattern.

Although it cannot be a standalone algorithm, it might be combined with another method to increase the level of security. Base64 is an encoding that uses a concept of modern encryption algorithms, but the Base64 mode is easier in its implementation than others. Base64 is a general term for some similar encoding schemes that encode binary data and translate it.

It is typically used when there is a need to encode binary data that needs in algorithms cryptography, Base64 was used in the proposed algorithm within RSA NEW encoding and decoding to encoding text and color images before encrypted and decrypt [30]. The below table (2.1) provides us with a Base64 encoding and decoding table.

Table (2.1) Encoding and Decoding Base64 string

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
Padding		=									

2.6 Padding

Padding in cryptography is the act of adding extra data to the beginning, end, or middle of plaintext before encrypting it. In classical cryptography, this involved adding zeros to obscure common message patterns.

2.6.1 Padding in RSA

Padding is used in both block ciphers and RSA. With block ciphers, padding is used if the size of the plaintext does not match the block size. Padding is essential to the function of RSA, however. By adding zeros padding, we can ensure that the ciphertexts will look the same when a given piece of data is encrypted twice. This can also eliminate leaking and other weaknesses (leaking occurs when a message is encrypted using different RSA keys).

Unlike block ciphers which use no more than one byte, padding used in RSA typically has a minimum requirement of several dozen bytes. Optimal Asymmetric Encryption Padding (OAEP) is the padding used for the RSA encryption[35].

2.6.2 Optimal Asymmetric Encryption Padding (OAEP) in RSA

Optimal Asymmetric Encryption Padding (OAEP) is a method of converting the RSA trapdoor permutation into a chosen-ciphertext secure system in the random oracle model. Then it is a form of Feistel network which uses two random oracles, G and H, to process the plaintext before performing asymmetric encryption.

When combined with any secure trapdoor one-way permutation, the random oracle model proves that this processing results in a combined scheme that is semantically secure under a chosen-plaintext attack. When implemented with certain trapdoor permutations (such as RSA), OAEP also proves to be secure against chosen-ciphertext attacks. An all-or-nothing transform can be built using OAEP[36].

2.6.3 OAEP goals

It will be satisfying the following goals:

1- Adding an element of randomness that can be used to convert a deterministic encryption scheme (e.g., traditional RSA) into a probabilistic scheme.

2- Preventing partial decryption of ciphertexts (or other information leakages) by ensuring that an unauthorized party cannot decrypt any portion of the ciphertext without inverting the one-way trapdoor permutation beforehand.

2.7 RSA Algorithm

RSA (Rivest–Shamir–Adleman) is public-key cryptosystems widely used for secure data transmission. The acronym RSA comes from the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1978[37].

Similar to other asymmetric algorithm designs, the key used to encrypt the data is shared and publicly available, while the secret decryption key is kept private. A user of the RSA algorithm generates two keys based on two very large prime numbers. They publish the public key which can be used by anyone to encrypt a message that can only be decrypted with the private key, allowing anyone to send them a securely encrypted message [16].

The security of the RSA algorithm lies in the difficulty of factoring the product of two large prime numbers. This is known as the "factorization problem" and breaking RSA encryption is known as the "RSA problem". Whether the RSA problem is actually as difficult as the factoring problem is currently an open question [17]. and although there are currently no known methods of breaking RSA encryption if a long enough key is used, due to it being a relatively slow algorithm it is not commonly used to encrypt data directly. Instead, RSA is often used to encrypt and share a symmetric key which is used to encrypt and decrypt the bulk of the data to be transmitted.

2.7.1 Usage of RSA

The RSA algorithm was quickly adopted in the early days of the internet as a standard encryption tool, being utilized by the TLS (Transport Layer Security) and SSL (Secure Sockets Layer) protocols, for internet communications such as e-mail and browsing, Netscape Navigator and Microsoft Internet Explorer are examples of early internet-based software to incorporate the RSA algorithm.

2.7.2 Structure of RSA

RSA uses two keys, one for encryption known as the public key and the other for decryption known as the private key. Therefore, we can summarize the steps to use the algorithm as follow:

1. Two prime numbers are chosen and kept secret p, q.
2. Compute n=pq. n is used as the modulus for both the public and private keys. Compute the key length in bits as.
3. Compute $\lambda(n) = \text{lcm}(p-1, q-1)$. This is kept secret.
4. Choose an integer e such that $1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1$; (e and $\lambda(n)$ are coprime).

5. Calculate d as the multiplicative inverse of e modulo λ(n). d ≡ e−1 (mod λ(n)).
6. (e, n) pair is released as the public key, and the rest are kept secret
7. (d, n, λ(n)) are used as a private key.

2.7.3 The Encryption process of RSA

A sender uses the intended receiver's public key (e, n) to encrypt some plaintext, such as a message or an image. The sender translates the letters into their numerical equivalents (if needed) and then forms plaintext blocks, m, such that the nonnegative integer m is less than n. The sender then uses the following encryption algorithm to encrypt m (which is handled with

$m^e \equiv c \ (mod\ n)$).

The ciphertext is represented by c and is sent to the receiver.

2.7.4 The decryption process of RSA

Decryption is the converse methodology of encryption, which shifts over the encrypted content into the unique plain content. In the decryption process,

is $c^d \equiv (m^e)^d \equiv m \ (mod\ n)$

The private key (d) is employed to decode the ciphertext and image.

RSA algorithm is an asymmetric cryptography algorithm which means that there should be two keys involve while communicating, i.e., a public key and a private key.

RSA Algorithm Example

- Choose p = 3 and q =11
- Compute n = p *q = 3* 11 = 33
- Compute φ(n) = (p-1) * (q-1) = 2*10 =20

- Choose e such that 1< e < φ(n) and e and φ(n) are coprime. let e =7
- Compute a value for d such that (d*e) % φ(n) =1. One solution is d =3[(3*7) % 20 =1]
- Public key is (e, n) ➔(7,33)
- Private key is (d, n)➔(3,33)
- The encryption of m =2 is c = 2∧7 % 33 = 29
- The decryption of c = 29 is m = 29∧3 % 33 = 2

2.8 Key Sizes

A cryptographic key is what defines the maximum theoretical security of an algorithm. More specifically, longer keys result in more secure encryption. This is due to the nature of brute-force attacks. The security of symmetric algorithms is dependent on the key length. Public-key systems, however, also depend on the ease of integer factorization and other mathematical problem, which is easier than brute force. Therefore, asymmetric keys are often longer to achieve the same level of security as their symmetric equivalents.

The NIST has recommended a minimum key length of 2048 bits since 2015[38], an update to the 1024-bit recommendation made in 2002[18]. However, the advent of quantum computers may render current algorithms week.

2.9 The Security of RSA

RSA security is based on the practical difficulty to factorize (N) into two large random prime numbers, the "factorization problem". which many cryptologists have tried to accomplish[39]. If anyone can get the factors p or q of n, it would be easy to find φ (n) and d since e is known. Many studies have shown that if R is a large

composite number, then it is hard to obtain the prime factors of n. Thus, hacking or cracking the RSA cryptosystem by factoring n would not be easy.

Nevertheless, there might be other ways to obtain d. It can be obtained by finding φ(n) from n, such that find φ(n)= φ(pq)= φ(p) φ(q)=(p-1)(q-1). Then p and q, the factors of n, can be found easily. Note that finding φ(n) is not easier than factoring n. Moreover, when p and q both have approximately 300 decimal digits, n=pq has approximately 600 decimal digits. Several million

years of computation are required to factor an integer of this size, even with the fastest factorization algorithms.[39]. RSA's safety depends on discover various numbers such as p, q, and d. If these parameters are not adequately taken into account, a number of Various problems will arise[40].

2.9.1 Weakness (Factorization Problem)

If a hacker only has the public key (n,e) and wishes to decode a ciphertext message c into plain text m, this adversary must try to find the factor in order to determine d [41]. Therefore, number factorization is a serious threat against RSA. Today, factorization is still an unsolved problem. However, there do exist inefficient algorithms which can correctly factor big numbers. The standard RSA algorithm is “deterministic”, meaning that when any given piece of data is encrypted, it will always produce the same ciphertext.

To secure RSA, a proper padding system must be used in order to pseudo-randomize the generated ciphertext. Since textbook RSA does not include any random factors, i.e., it is deterministic. This is because the public key is known to everyone, including potential attackers, and anyone can use it to encrypt messages. Padding the plaintext with using a system such as OAEP before being encrypted will result in a different ciphertext every time, thus defeating such an attack.

2.9.2 Strength of RSA

It is considered infeasible to attempt to crack RSA encryption because no known algorithm can do so effectively, requiring a huge amount of time and computational power. The strength of RSA is a result of two mathematical issues, which are listed below[42].

2.9.2.1 RSA Problem

The aim is to find a value for m using the formula $c = m^e \bmod n$, where (e, n) is the public RSA key and c is the ciphertext. An efficient method of computing this has yet to be found, but if one were to be developed, the security of RSA would be compromised[42].

2.9.2.2 Factoring large numbers

The easiest way to crack an RSA encryption is to factor the numbers in the public key to find the two prime numbers that were used to produce it in the first place. The private key's numbers can then be calculated quickly and efficiently. If the keys are long enough, this process is thought to be impossible. In 2005, the largest number factorized successfully was a 663-bit number.

Modern RSA keys are typically 1024 to 2048 bits long, which is long enough to make them unbreakable for the time being. It is also believed that 1024-bit numbers will be breakable before long. As a result, using at least 2048-bit RSA keys is currently recommended. The use of a quantum computer, could significantly improve the efficiency of factoring large numbers.

If such a machine were to be developed, the RSA algorithm would be rendered useless because it would be too easy to crack. It will be several years before a quantum computer of this magnitude is created[42].

2.10 The advantages and disadvantages of RSA Algorithm

RSA offers a high degree of security and safety, which result from the complex mathematics involved in the algorithm. As the factorization of large prime numbers is a difficult task, breaking RSA is highly challenging. And since the public key is only used to encrypt the data, it can be shared with anyone.

On the other hand, RSA requires that a third party verify the authenticity of public keys, which can compromise the algorithm if the public key system is tampered with. It is also slow when encrypting large amounts of data, making it unideal for mass data encryption and transmission.

This means that both symmetric and asymmetric encryption techniques are vital to the encryption and transfer of sensitive data[43].

2.11 Miller–Rabin (MR)

When we want to test large random numbers, are they prime or not, we use the Miller-Rabin function, which is a probabilistic algorithm.

Miller Rabin is a relatively simple extension of Fermat's little theorem that allows us to test for primality with a much higher probability than Fermat's little theorem. It was named after Michael Rabin discovered a randomized polynomial-time algorithm to test if a number was prime in 1980.

The most widely used primality test algorithm is still this one. If anyone wants to use the RSA public-key cryptosystem, they must first produce a private key, which comprises two large prime numbers, and a public key, which is made up of the product of those two prime numbers[44]. To do this, we must be able to find large prime numbers, which are essential for securing our data. Miller Rabin is a quick way to see if large numbers are prime.

This algorithm, also known as the Rabin-miller primality test, decides if a number is prime. It is similar to other tests, including the Fermat primality test and the Solovay-

Strassen primality test. This test is based on an equality, or a set of equalities, that hold for prime values and then checks to see if they hold for the number being tested for primality.

It is the most widely used primality checking algorithm and is found in various software libraries that use RSA encryption. Miller Rabin shows that a number is composite, as opposed to other primality tests that show that a number is prime. Therefore, this test may be referred to as a compositeness test rather than a primality test[45].

Algorithm (2.2) ***Miller Rabin Algorithm Testing***

Input: *" n > 3, an odd integer to be tested for primality"*
Input: "*k, the number of rounds k >0, m is odd"*

Output: *"composite if* ***n*** *composite, otherwise probably prime"*

Begin*:*
Step1*:* $(n-1) = 2^k.m$
Step2: *choose* ***a*** *is random number integer such that* $1 < a < n-1$
If $(a^m \bmod n) = 1$ *or -1 then*
Return *"probably prime"*

Step3*:* *for j =0 to k-1 do*
If $(a^{2jq} \bmod n) = -1$
Return *"probably prime"*
If $(a^{2jq} \bmod n) = 1$
Return **"**composite"
End

Miller-Rabin Algorithm yields a number that is not necessarily a prime. However, the algorithm can yield a number that is almost certainly a prime. It is based on the

conclusion that if n is prime, then either the first element in the list of residues, or remainders, (a q, a2q,..., a 2k-1q, a 2kq) modulo n equals 1, or some element in the list equals (n-1); otherwise n is composite (i.e., not a prime). On the other hand, if the condition is met, that does not necessarily mean that n is prime. For example, if n = 2047 = 23 x 89, then n-1 = 2 x 1023. Computing, 21023 mod 2047 = 1, so that 2047 meets the condition but is not prime[18].

This algorithm is used for the primality testing of large numbers. The Miller-Rabin test is often chosen for cryptographic application as it runs faster than other primality tests and offers a minimal chance of failure[46].

2.12 Random number generation (RNG)

Random number generation is a technique in which special algorithms are used to produce sequences of numbers that generally cannot be predicted. Randomness has many applications in modern computation which led to the development of random number generators (RNG).

Truly random hardware-based number generators, or HRNG, are one type of random number generators. The other type is pseudorandom number generators, or PRNG, which generate sequences of numbers that appear to be random but are in fact deterministic and depend on an initial value known as a “seed”. Statistical tests that attempt to analyze the unpredictability of pseudorandom generators are used with varying degrees of success and that are designed primarily for use in cryptography[47].

Cryptography requires a high degree of randomness and must be randomized by design, although there are other applications, such as simulations and statistics, where randomness is desirable or necessary. Its use in cryptography stems from the fact that the sender and receiver will both generate the same number set as the shared key.

2.13 Efficient Operation Using the Public Key

The operation speed of RSA can be increased by choosing an appropriate value for e, typically, a value which has only two 1 bits (such as 3 and 17) is used in order to reduce the number of multiplications required for exponentiation. However, a small value such as 3 renders RSA vulnerable to simple attacks, therefore $(2^{16}+1)=65537$ is the most commonly chosen value[48].

The user is required to choose a value of e that is relatively prime to φ (n), meaning that values of p or q not congruent to 1 must be rejected, this is because that if, for example, the value 65537 is chosen for e, gcd(φ(n,e) might not be equal to 1[49].

2.14 Cryptanalytic Attack

Cryptanalysis is important during the construction of a block cipher to determine how secure the cipher is, how good the proposed algorithm is, and whether the proposed structure has any faults. Cryptanalysis results can thus be used to verify a cipher's strength.

To determine the weak points of a cryptographic system, it is important to attack the system. These attacks are called Cryptanalytic attacks. The attacks rely on the nature of the algorithm and also knowledge of the general characteristics of the plaintext, Therefore, the nature of the plaintext should be known before trying to use the attacks. There are many various cryptanalytic attacks viz. brute-force attack and chosen ciphertext attack[50].

2.14.1 Brute-Force Attack

A brute-force search is a cryptanalytic attack that attempts to decrypt any encrypted data for any encryption scheme that uses keys, such as the RSA algorithm. brute-force attack just relies on his ability to try all possible keys until the correct

key is found A brute-force attack cannot be avoided, but it can be made infeasible[51].

2.14.2 Chosen cipher text attack (CCA)

Chosen ciphertext attack refers to a situation in which the attacker has the capacity to select ciphertexts (C) and examine their decryptions – plaintexts (M). It's essentially the same scenario as a selected plaintext attack, except it's applied to a decryption function rather than an encryption function. Using this information, the adversary can attempt to obtain the concealed secret key used for decryption[52].

Chapter Three

Proposed enhancement for RSA Algorithm

3.1 Introduction

In recent years, the amount of information sent via the Internet has skyrocketed, and practically everyone in the globe has a computer connected to the Internet. Depending on the nature of the information sent, it could take the shape of text, photos, emails, etc. Some of this information could be classified as sensitive or military-related. This mandates the use of encryption techniques to encrypt data before transmitting and protect the data from external assaults like Man in the Middle (MITM) as well as the ability to send electronic documents from over a computer network not secure. In the research area, there are many encryption and decryption algorithms available; one such algorithm is known as the RSA (Rivest-Shamir-Adleman) algorithm.

This chapter presents (discusses) the proposed enhancement method of key generation in RSA algorithm to again strengthen security and multiple (public and private) keys that resulted in speed up key generation method in RSA algorithm:

1- The first part of this chapter explains the generation of multiple public and private keys.
2- The second part explains the preprocessing stage, which includes base64 encoding and turning it to bytes for processing using ASCII.
3- The third part of this chapter covers the encryption process using multiple public keys.
4- The fourth part explains how to use PRNG in the encryption and decryption process.
5- Finally, the decryption process using multiple private keys. The proposed algorithm is discussed and implemented in c# .net (visual studio).

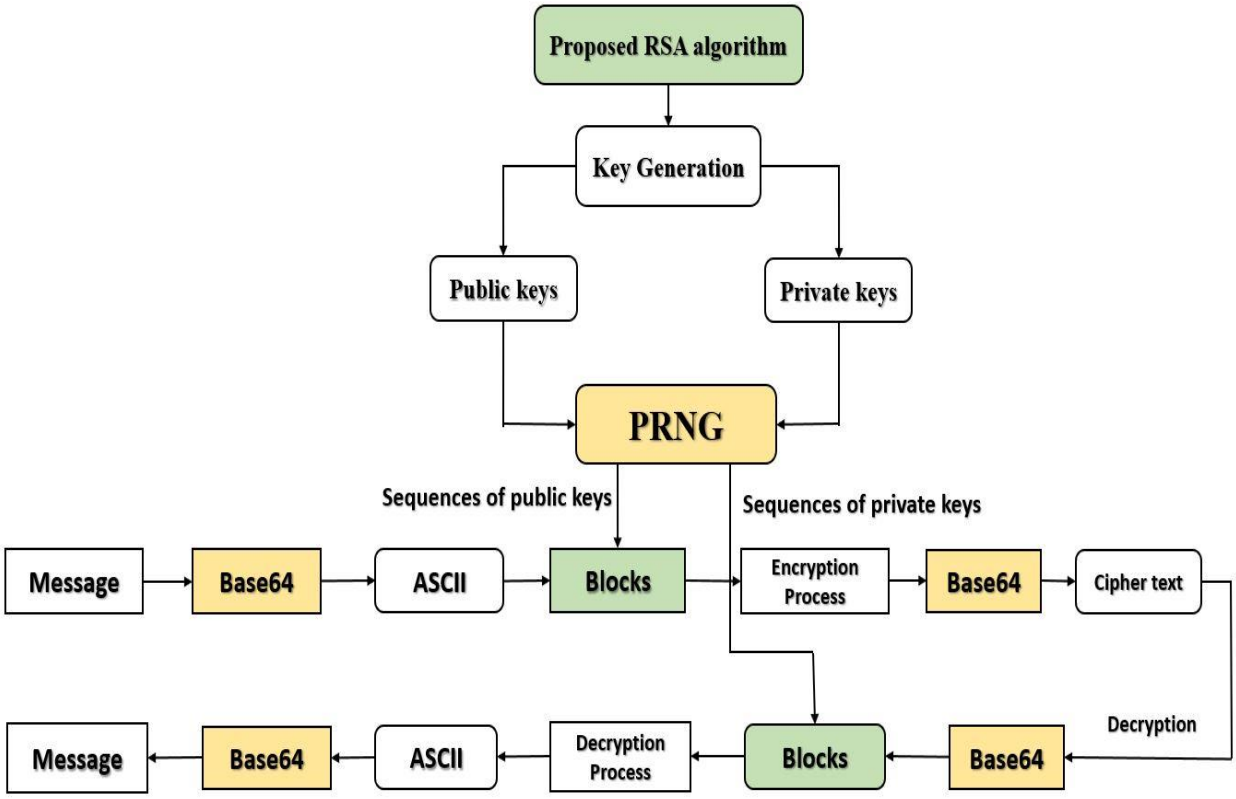


Figure (3.1) structure proposed RSA algorithm, Source: Prepared by the researcher.

3.2 Proposed Solution

The related work explains when complexity grows, time increases. The proposed algorithm does not alter the basic structure of the RSA algorithm, but it mixes the structure of RSA with block cipher characteristics. Where encryption and decryption blocks are set to length with 2048 bits as an example, then the 2048-bit block is segmented, the segment can be 64 bits, 128 bits, 256 bits, 512 bits, and so on. To generate the same number of RSA key pairs according to the segment numbers.

Each segment is encrypted with a random key. The key is randomly selected using PSNR. The random generator with a seed is used to preserve the order of the keys as well as add a random component by removing one key from the chain to produce

different output from the same input. The same seed is used later during decryption to reproduce the same random sequence that is used in encryption.

The proposed algorithm is more speed because of dividing key size and more secure because of using multiple keys, the general structure of the proposed algorithm illustrated in Figure (3.2).

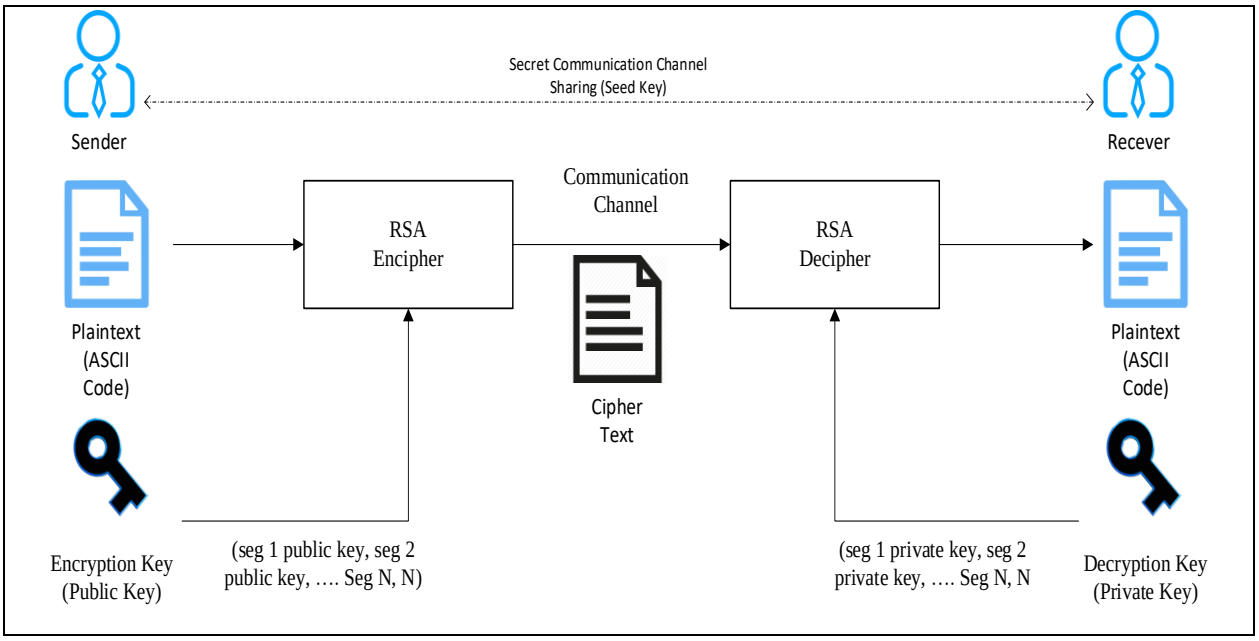


Figure (3.2) The General Structure of proposed RSA algorithm, Source: Prepared by the researcher.

The proposed RSA algorithm can be used to encrypt text messages (plain text). This is achieved by selecting key size and dividing the size of this key into specific sections to encrypt and decrypt blocks using a mixture of several public keys and private keys according to the order of keys that are produced by the random generator, so each block is encrypting with multiple keys according to key order.

Small RSA keys are vulnerable to factorization attacks, whereas large keys are very slow and inefficient. To overcome the problem of increasing security at the expense of time when using larger keys in the RSA algorithm, we increase the complexity

and use larger block sizes without sacrificing speed. The modified RSA algorithm used to encode standard text is then compared with the original RSA algorithm. The time problem was overcome by dividing the key into several parts, which makes the proposed RSA algorithm faster than the original RSA algorithm and makes the time less.

The proposed algorithm is identical to the original RSA algorithm but the key size is divided according to user-defined segments that are either 2, 4, 8, and 16. So there are multiple keys for the proposed algorithm. After that, the original message is split into blocks. each block is encrypted with multiple public keys according to the order that is generated random generator that contains an initial value for the sequence of these segments which is called (seed key). The seed key must be shared between sender and recipient to know the key sequence. so that it becomes It is very difficult to solve, increasing the security of the system. Usually, large keys in the RSA algorithm are used which are very slow since small RSA keys are vulnerable to factorization attacks.

we decrease the time for encryption and decryption process in the proposed RSA algorithm. As a result, this method is more efficient, secure. The Proposed Algorithm is depicted in the block diagram as in Figure (3.3).

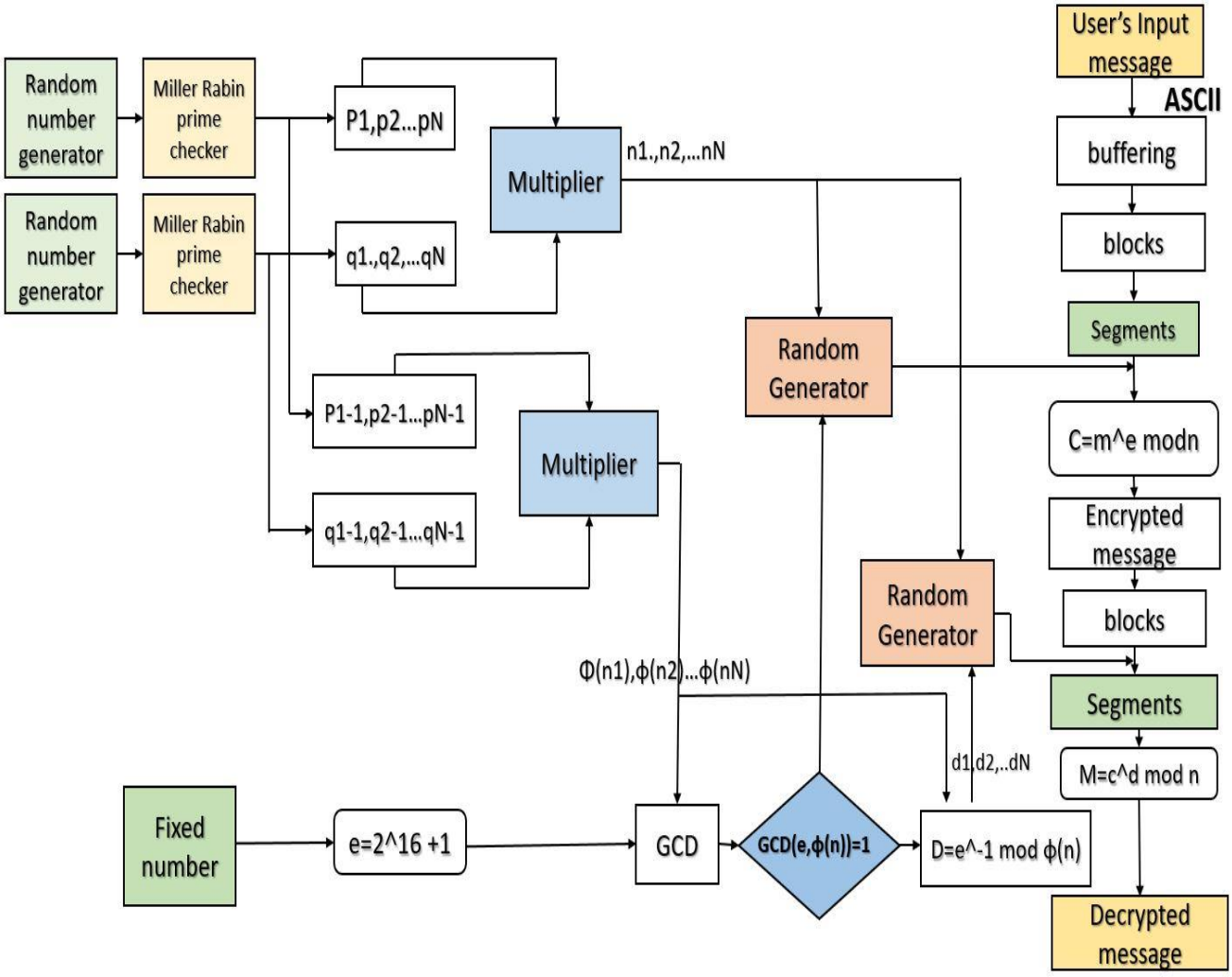


Figure (3.3). Block Diagram of proposed RSA Algorithm, Source: Prepared by the researcher.

3.3 Characteristics of the Proposed RSA Algorithm

A proposed model has the following features:

1- The proposed RSA algorithm increases the complexity of the key generation process, by increasing the number of primes that are used.
2- The fundamental benefit is that the suggested architecture can be utilized to exchange data and public keys across an unsafe cryptographic channel.
3- Encrypting data using multiple public keys can increase security.

4- The proposed algorithm reduces encryption and decryption time.

5- The use of large initial random numbers within the proposed approach increases the security and reliability of the new method so that it is more difficult for hackers.

6- In the suggested structure, message segments are encrypted and decrypted in a non-sequential (random way) using several public and private keys whose order is tied to a random generator by (primary key).

3.4 Proposed Algorithms

To illustrate more details about flowchart functions several algorithms have been constructed.

Each algorithm example a specific such as:

1- key generation as explained in the algorithm (3.1).

2- Encryption process as explained in the algorithm (3.2)

3- Decryption process as explained in the algorithm (3.3)

Algorithm (3.1) Initialization and key generation, Source: Prepared by the researcher.

Input: *Segment Size, (seed key)*

Output: *Public Key, Private Key for each segment*

Begin:

Step1: *Select the block size*

Step2: *Calculate number of segments N = block size / Segment Size.*

Step3: *Generate pair of keys for each segment number (N).*

Step4: *The public key (seg 1 public key, seg 2 public key, …. Seg N, N).*

Step5*: The private key (seg 1 private key, seg 2 private key, …. Seg N, N).*

Step6: *Seed Key is pre-shared between sender and receiver using a secure method.*

End

Algorithm (3.2) The proposed RSA Encryption Algorithm, Source: Prepared by the researcher.

Input: *Public keys, Bits To Encrypt, Seed Key*

Output: *Encrypted Bits*

Begin:

Step1: *Encrypted Bits = empty*

Step2: *From* ***Public key*** *extract* ***List of Public RSA Keys*** *and number of segments (**N**)*

Step3: ***Generator*** *= Initialize sequence generator with (**Seed Key**)*

Step4: *While* ***Bits To Encrypt*** *not empty:*

a. ***Block*** *= cut a block of 2048 bits from* ***Bits To Encrypt***
b. *Add padding if needed*
c. ***Encrypted Block*** *= Initialize empty block 2048bits*
d. ***List of Segments*** *= Divide* ***Block*** *to the desired number of segments (**N**)*
e. *For each segment in the* ***List of Segments***
    i. ***Sequence*** *=* ***Generator****. Get Number (0,* ***N****-1)*
    ii. *Encrypt segment with RSA using* ***List of Public RSA Keys*** *[**Sequence**]*
    iii. *Append to result to* ***Encrypted Block***

Step5: *Next segment*

f. *Append Encrypted Block* ***To*** *Encrypted Bits*

Step6: *Go to step 2.*

Step7: *Output* ***Encrypted Bits***

End

Algorithm (3.3) The proposed RSA Decryption algorithm, Source: Prepared by the researcher.

***Input: Private key**, **Bits To Decrypt**, Seed Key*

Output: Decrypted Bits

Begin:

***Step1: Decrypted Bits** = empty*

Step2: From **Private keys** extract **List of Private RSA Keys** and number of segments (**N**)

***Step3: Generator** = Initialize sequence generator with (**Seed Key**)*

***Step4:** While **Bits To Decrypt** not empty:*

a. ***Block**= cut a block of 2048 bits from **Bits To Decrypt***
b. *Decrypted **Block** = Initialize empty block 2048bits*
c. ***Encrypted Block** = Initialize empty block 2048bits*
d. ***List of Segments** = Divide **Block** to the desired number of segments (**N**)*
e. *For each segment in the **List of Segments***
   i. ***Sequence** = **Generator**. Get Number (0, **N**-1)*
   ii. *Encrypt segment with RSA using **List of Private RSA Keys** [**Sequence**]*
   iii. *Append to result to **Decrypted Block***

***Step5:** Next segment*

f. *Removed padding if found*
g. *Append **Decrypted Block** to **Decrypted Bits***

***Step6:** Go to step 2.*

Step7:** Output **Decrypted Block

End

3.5 Practical Implementation of proposed RSA Algorithm

To implement the RSA or generate a modified RSA algorithm programmatically, one has to focus on the following:

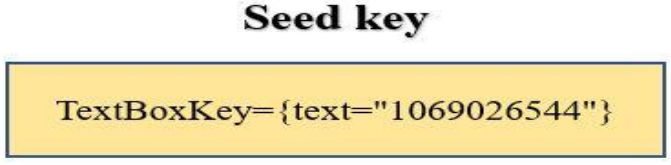
- The key generation
- The process of encryption
- The process of decryption

3.5.1 key generation

1- An example of key size is (2048 bits) because it is larger than 1024 to prevent analyzing it in the future. a very large number is used, so it is 2048 bits (256 bytes) for the proposed algorithm.

2- In the next step, the key must be divided into specific segments according to the user's choice (2, 4, 8, 16). These partitions are equivalent to the basic key partitions of the original RSA algorithm so that the keys are compared in terms of standard size between the proposed algorithm and the original RSA algorithm.

3- Random key is then generated and is considered important for arranging the sequence of the public and private keys for the segments and it is called (the seed key), which in turn must be sent secretly to the recipient so that he knows the sequence of keys that decrypt the message in the order in which they have encrypted so that it can be decoding and restoring the original text, for example, seed key= ("1805961104").



4- To calculate the size of the segment, their size in bits must be determined according to the following equation:

[SegmentSizeInBits = (KeySizeInBits / (int)segmentMode) – 8] ---------- (Equation 1)

The size of the segment in bit is equal to the size of the key (2048 bits) divided by the number of segments specified by the user minus one byte to be less than the original key, as well as its size in bytes only, we divide it by eight according to the following equation:

[SegmentSizeInBytes = SegmentSizeInBits / 8] ---------- (**Equation 2)**

5- To calculate the size of the message block that we want to encrypt in bits, as calculated by the following equation:

[BlockSizeInBits = SegmentSizeInBits * (int) segmentMode] --------- (Equation 3)

The message block is determined in bytes according to the following equation:

[BlockSizeInBytes = BlockSizeInBits / 8] ---------- (Equation 4)

6- After dividing the key into segments, each segment is considered as an independent key for the RSA algorithm, where public and the private key is generated for each these segments, we determine the size of the key in bits as well as in bytes and we are working to create a large random prime number representing (p) through the random number generation function (Get random prime) is a function that generates empty bytes according to the size of the key we entered into this function.

will dictate these empty bytes with integers and we will have a random integer candidate number representing the number (p), in the initial form.

7- This candidate integer random number can be a prime number or not be a prime number, depending on what is entered in a series of tests to make sure that it is a prime number.

3.5.1.1 Prime number test series

When testing the candidate random integer, it must not be an even number because even numbers are not prime numbers at all, except for the number (2), which is the only even prime number, so the test is for odd numbers only, this represents the first test. After the candidate random integer is an odd and not even number, the absolute function (Abs) is entered in order to make sure that the candidate number is a positive and not a negative number, this second convert the number to positive.

After this, the candidate number must be entered in the Miller Rabin function, which is a probability function to test the number is prime or Not prime, because it is a probabilistic algorithm and this algorithm was previously explained, we give the integer candidate for the Miller Rabin algorithm as well as the number of instances in which we test the number to ensure that it is likely to be a prime number, the following is algorithm for miller Rabin test.

Finally, we understand how to test for an equal value set that applies to prime values, then examine if they hold or not a number we wish to test for primality. After we assess the likelihood of the random prime integer that the first candidate represents (p) by the using method (Miller Rabin) we carry out the identical preceding methods to generate and determine the other prime number representing (q). At this point, we have two random prime numbers primes are p and q.

3.5.1.2 Key building stages

To construct the public and private keys we need two random prime numbers that were generated earlier, we calculate (n) according to the following equation: **n= p*q**

we add a basic test that discovers the number of bytes in the key (n) if it is equal to the number of bytes in the original key or not.

If the size of (n) in bytes is less than the size of the bytes of the key, both p and q are deleted and rebuild each of the primes and multiply them until the size (n) is equal to the size of the primary key.

Following Equation represents Euler function and assign it a symbol (phi), based on the equation below.

[**Phi = (P - 1) * (Q - 1)]**

The Euler function is a fixed number that denotes the number of positive integers between 1 and (n).

3.5.1.3 The public-key for RSA

In the RSA cryptosystem, the number 65537 is widely used as a public exponent. In the proposed design for the modified algorithm, the public key value (e) was used fixed because it is exposed to everyone and is equal to (e= (65537) =2^16+1) which is a prime number and this value speeds the encryption process more efficiently because there are only two specified bits in this number, See the chapter two section (2.13). This value is considered prime because it is large enough to avoid the assaults that tiny exponents leave RSA vulnerable to. e=2^16+1 is a good compromise between performance and security.

It's small enough to provide good performance and has a low hamming weight, but not too little to be dangerous. Explanation: For the calculation of the modular exponentiation, a smaller exponent with a lower hamming weight (fewer '1' bits) provides superior speed. Because e=2^16+1 only contains two '1' bits, it has a low hamming weight and a large enough value to ensure appropriate security. The value of (e) should ideally meet two requirements:

1. (e) must be a prime value.

2. Check the following equation **[1< e < phi]**

Now we insert the function of the greatest common divisor between the two values (e, Phi) The greatest common divisor of two numbers is the largest integer that divides both numbers. If the two numbers share no factors, like 14 and 25, then the gcd (14,25) = 1, must be **gcd (e, phi) =1.**

Following the discovery of the public key, d is calculated immediately from e and $\phi(n)$. (i.e., the private key is derived from the public key). that computing the multiplicative inverse in this manner is difficult If this weren't the case, any attacker could figure out the private key from public key. why an attacker can't get private key? The problem here is that the private exponent is the modular inverse of the public one, but modulo $\phi(n)$, not n: as in the equation

$$d \equiv e^{-1} (\mathrm{mod} \phi(n))$$

Finding the multiplicative inverse is in fact computationally feasible. The prime numbers p and q are not public (although n = pq is). An attacker cannot therefore know $\varphi(n)$, which is required to derive **d** from e. The strength of the algorithm rests on the difficulty of factoring n (i.e. of finding p and q, and thence $\varphi(n)$ and thence **d**). In other words, (d, p, q, $\phi(n)$) should be secret, as opposed to information on the public key and the ciphertext being publicly exposed.

D= e ∧-1 mod phi

Upon completion of generating the public key and the private key in relation to the first segment of the original static key that was divided by the user into (2, 4, 8, 16) it generates the public and private key, in relation to the next segment, and so on to the rest of the other segments. For example, when the User chooses split (4) The

algorithm will generate four public keys and four private keys, for each segment. The following is an example of public and private key.

Public key (N, e):

N=5369301298078350748289985228140971308725926678075090836014783176909153933969

e= 65537

private key (N, d):

N=5369301298078350748289985228140971308725926678075090836014783176909153933969

d=213995376513735022270373093304609870117830739302635981759490153111255332273

In the proposed design algorithm, the public key value (e) was used fixed because it is exposed to all and equal to (65537) which is a prime number and this value speeds up the encoding process more efficiently, but modulo N is a variable every time the output is a variable due to a change in the value of (N), as we note in some mathematical examples.

$2^{32} \bmod 67 = 33$

$33^{32} \bmod 37 = 12$

$12^{32} \bmod 41 = 16$

$2^{32} \bmod 37 = 7$

$7^{32} \bmod 67 = 19$

$19^{32} \bmod 41 = 10$

3.5.2 Encrypt the plain message with the proposed RSA

In this section, preprocessing the plaintext is done. by converting message to bytes, then encrypt the buffers using the proposed RSA algorithms.

3.5.2.1 Preprocessing of Text

A type of encoding is therefore necessary, including Encoding the plain text into an ASCII compatible environment, such as base64 string format, the output may now be translated into ASCII code, and finally, the ASCII code should be converted into a byte array for the plain text to be a encode. See Algorithm (3.4) below, for more information.

Algorithm (3.4) Encode message (Normalization)
***Input**: Plain Text (Messages, Symbols)*
***Output**: array of bytes*
Begin**:* ***Step1**: convert message using Base64and convert it to ASCII.* ***Step2**: Get the ASCII characters of the input message as array of byte* ***End

3.5.2.2 Encrypt Text

When encrypting a particular message, the Base64 system is used to and convert to a message with specific characters, Base64 system. It is a 64-character, printable-

character-based representation of binary data. Base64 encoding requires that every three 8-bit bytes be converted into four 6-bit bytes.

Base64 is a popular encoding method today because it is fast and simple to translate. We convert plain text (all world languages and all existing symbols) into a message with specific characters, and the message must be converted into a series of bytes by ASCII to be encoded in the proposed algorithm.

The printable characters include the letters A-Z, a-z, numbers from 0 to 9, and "+ /", so the table size is 64. Convert the output to an ASCII string into an array of bytes. Convert that byte array into a large integer.

For that, you need a library with support for arbitrary large integers (e.g., BigInteger) For encryption, in order to deal with integers by raising the exponent and other mathematical operations. In other words, we transform the message into a sequence of bytes by encoding Unicode, it is characterized by converting every character, symbol, or any language in the world into bytes, and each character is stored in 32 bits, i.e., 4 bytes, and convert it by system Base64 to a message with specific characters so that it may be specified within a certain range of 0 to 64, and encode the message using ASCII so that it can be converted into specific bytes. Each character is stored in one byte so that the message is bytes (buffering). look the Figure (3.5) below.

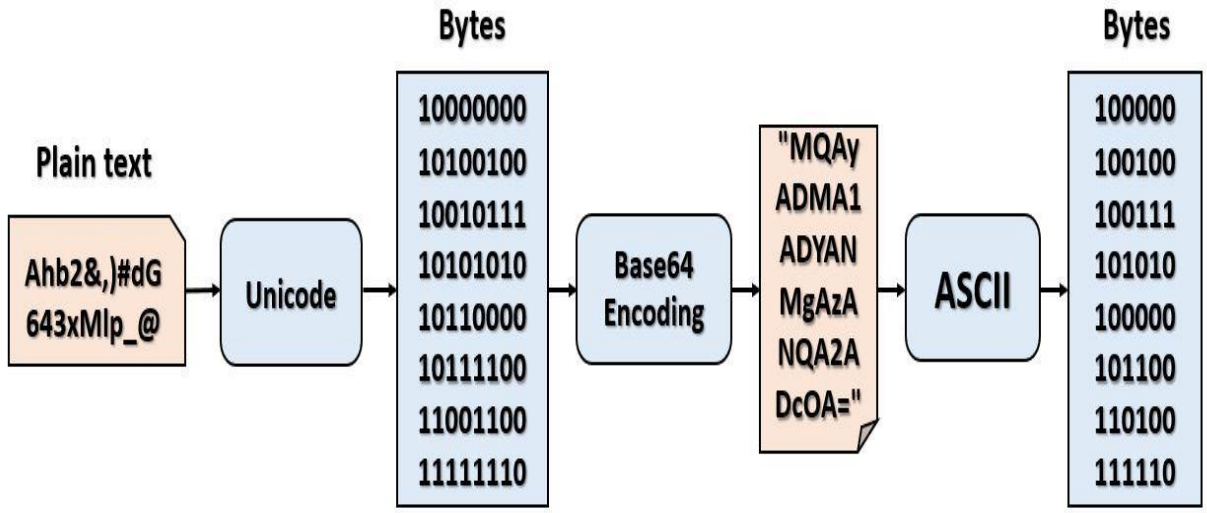


Figure (3.5) Encoding text in Base64 system, Source: Prepared by the researcher.

3.5.2.3 Encryption Buffering

After converting text into bytes using ASCII, the encryption process should be done using various public keys. This procedure requires the following inputs:

1. Generate the seed key and size of the block must equal to the size the key which is used.
2. The buffer must be padded using (AOEP) to complete block size

```
remainder = buffer.Length % BytesInBlock
```

   The reminder represents the size of last block. So, no. of zeros can be added.
3. Following is the relationship that determines the length of the padding that will be added.

```
Plain Buffer List . Add(0)
```

4. The number of blocks is counted in buffer.

3.5.2.4 Padding OAEP

If the plaintext size does not match the block size, Optimal Asymmetric Encryption (OAEP) padding is employed. Padding, which is required for the text function, is a type of Feistel network that processes the explicit text using two random units, G and H, before executing asymmetric encryption. It is the technique of adding data to the beginning, end, or middle of plaintext before it is encoded in order to obscure message patterns. As a result, we included OAEP in the suggested algorithm; additionally, OAEP is utilized to protect against certain ciphertext attacks.

3.5.2.5 Block Encryption

At the beginning, the number of blocks must be calculated for the message to be encrypted. So, all blocks must encrypt one by one. The user selects the numbers of segment, such as (2, 4, 8, 16). Steps of the block encryption as follows:

1. Array of integer is created, each element in that array contain a number which represent the order of the public and private keys.

```
Selector[ i ] = Rng.Next(0, RsaAlgos.Length - 1)
```

The following figures show the random generator and public and private key sequences and their random order.

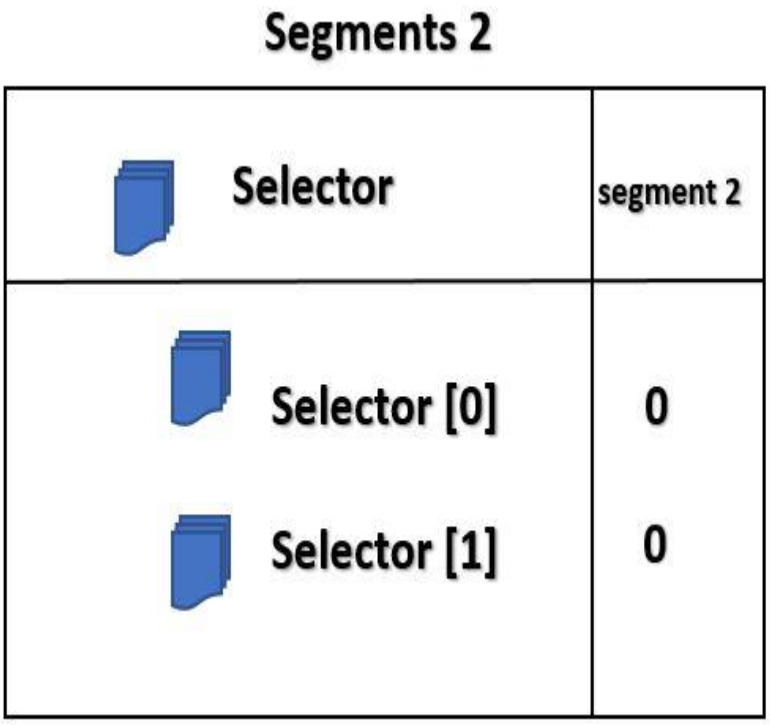


Figure (3.6) shows the random generator two segments

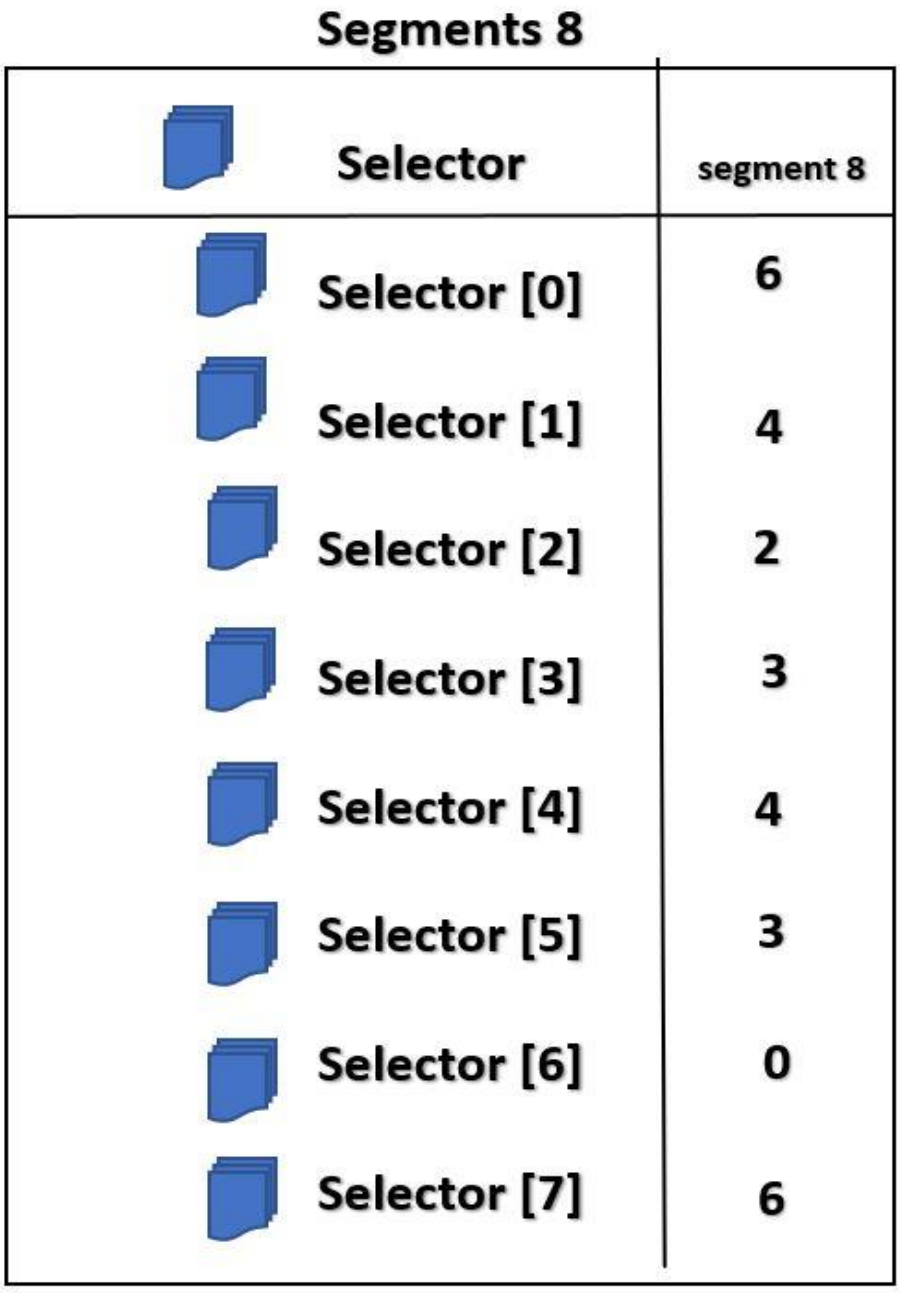


Figure (3.7) shows the random generator eight segments

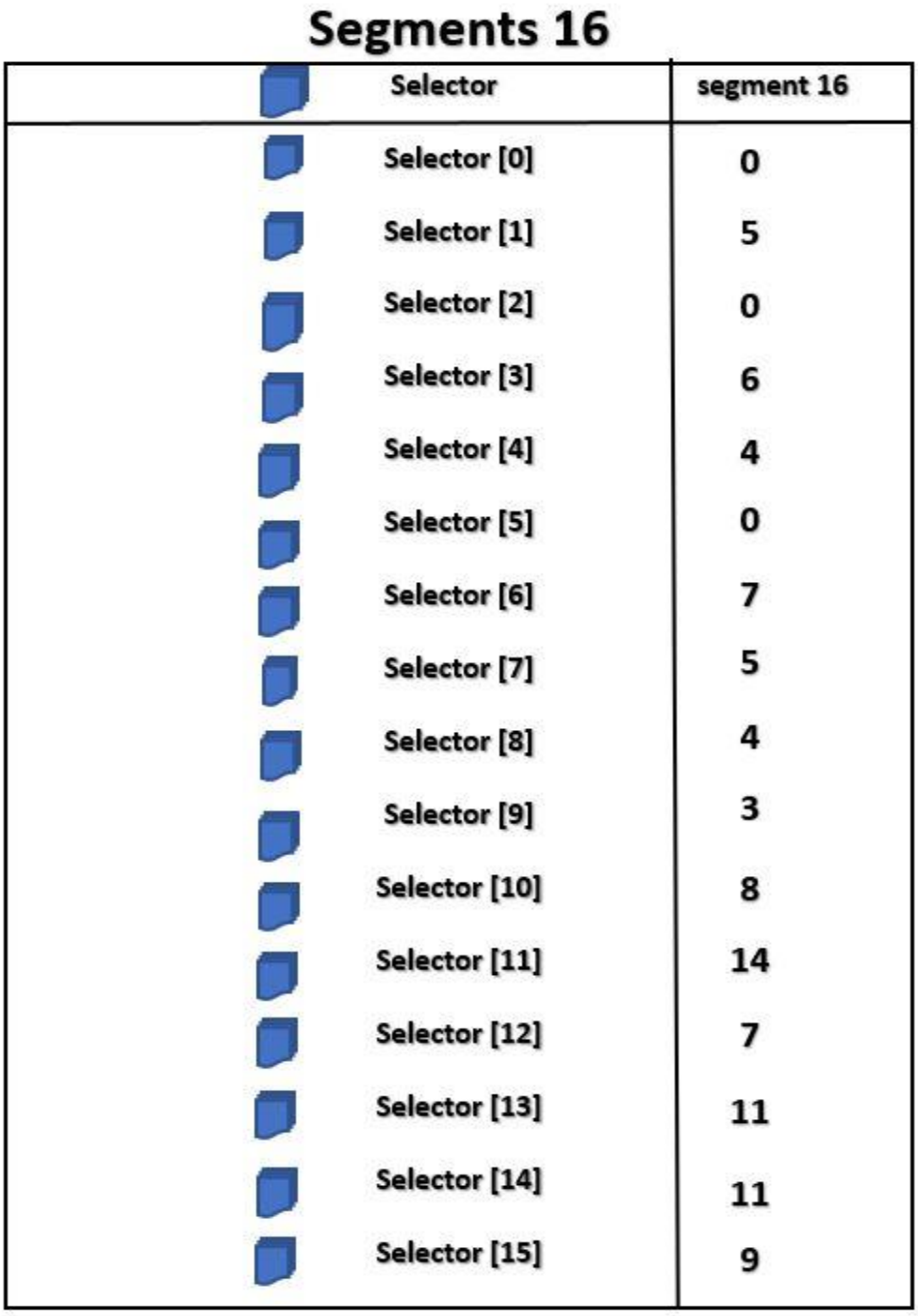


Figure (3.8) shows the random generator sixteen segments

2. Each block is divided into multiple segments according to user selection.
3. Each segment is converted from a byte to a single BigInteger (m). The encryption process is done according to the following equation.

$$Cipher = M \wedge e \bmod N$$

4. Each segment is converted from BigInteger to bytes.
5. All segments are encrypted in block and collect bytes for all parts of the block until the first block has been encrypted. Then it can be moved to the second block to complete the message.
6. Once all the blocks of the message have been encrypted, the encrypted buffer will be returned and converted to special letters and numbers using (Base64) to obtain the ciphertext.

3.5.3 Decrypt the cipher Text with the proposed RSA

As described in this section is Decrypt a message that normally includes symbols, characters, and numbers that are transformed into plaintext (the original message) is the inverse of the encryption process, with the processes reversed.

3.5.3.1 Decrypt cipher Text

Convert the encrypted message into bytes using the Base64 system These bytes are forwarded to the spooler's decrypt function so that the encrypted message is decoded.

3.5.3.2 Decrypt buffering

1. Decrypting the buffer needs to read the seed key of the random generator, the block size in bytes, and the number of blocks in the encrypted message as in the following equation:

No_Of_Blocks = plainBuffer List.Count / BytesInBlock

2. All bytes are divided to blocks.
3. Select the number of blocks.
4. The first block is taken and submitted to the block decryption function.

3.5.3.3 Block Decryption

1. pseudo-random number generators (PRNG) are used that order the sequence of private keys for segments that are taken randomly according to the seed key and store that arrangement in an array to use in the decryption process as in figures (3.6), (3.7), (3.8).
2. Each block is divided into multiple segments according to user selection (2, 4, 8 or 16).
3. Each segment is converted from a byte to a single BigInteger (c). So that we can perform mathematical operations on it by raising the exponent and modular and using the private key to decrypt the text, is done according to the following equation:

$$Plain = C \wedge d \bmod N$$

4. Each segment is converted from BigInteger to bytes.
5. All segments are decrypted in block and collect bytes for all parts of the block until the first block has been decrypted. Then we move to the second block to complete the message.

6. After this, the padding appended at the end of the buffer is processed to return the encrypted message's original text After decryption.
7. Finally, a message consisting of a set of bytes is passed to a preprocessing.

3.5.3.4 Preprocessing of cipher Text

The result is processed, once all the blocks of the message have been decrypted, the decrypted buffer will be returned and converted to specific characters by ASCII and using the Decode Base64 system to obtain the plaintext, According to the following figure (3.9).

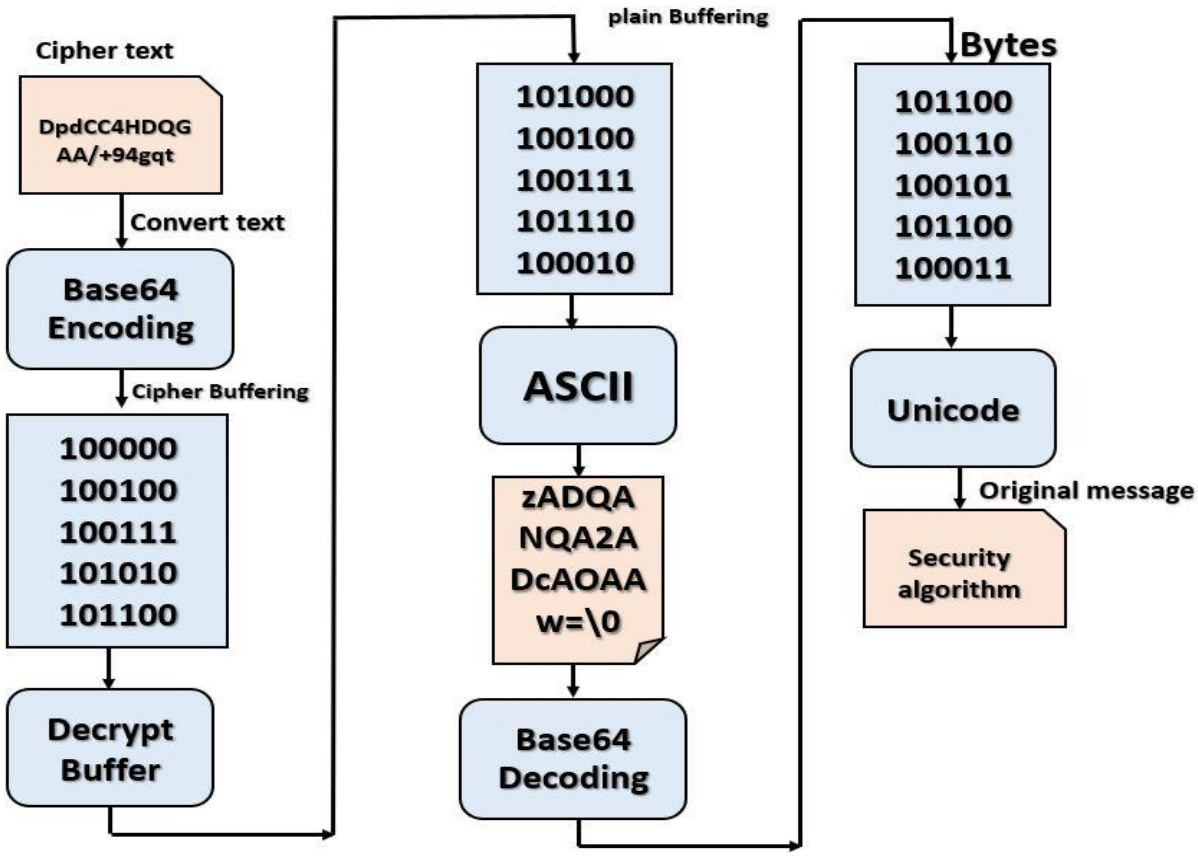


Figure (3.9) Decoding text in Base64 system, Source: Prepared by the researcher.

3.6 Summary

The proposed RSA algorithm can be used to encrypt text messages (plain text) This is achieved by selecting key size and dividing the size of this key into specific sections. The proposed algorithm is identical to the original RSA algorithm but the key size is divided according to user-defined segments that are either 2, 4, 8, and 16. As a result, this method is more efficient, secure. so that it is very difficult to solve.

The proposed RSA algorithm increases the complexity of the key generation process, by increasing the number of primes that are used. In the suggested structure, message segments are encrypted and decrypted in a non-sequential (random way) using several public and private keys whose order is tied to a random generator.

Padding in the RSA algorithm is the act of adding extra data to the beginning, end, or middle of plaintext before encrypting it. this involved adding zeros to obscure common message patterns. Optimal Asymmetric Encryption Padding OAEP is the padding used for the RSA encryption.

Chapter four

Implementation and Results Discussion

CHAPTER FOUR

Implementation and Results and Discussion

4.1 Introduction

This chapter explains the result obtained by implementing the proposed RSA algorithm described in chapter 3 and discusses these results. by comparing the proposed approach with the original RSA algorithm in terms of speed, and security with various key sizes.

4.2 Performance Measurements

The performance measures, for the proposed RSA, are the time required for encryption and decryption file, second, the speed parameter for calculation the encryption and decryption file, and last the security is evaluated.

4.3 Testing

All tests are performed on a computer with specifications shown in table (4.1).

Table (4.1) Test System Specifications

Test System Specifications		
1	Operating System (OS)	Windows 10 19043 (21H1)
2	Processor (CPU)	Intel Core i7 8750H with 2.2Ghz Base Clock
3	Random Access Memory (RAM)	16GB DDR4 clocked at 2667MHz

In general, the model's functioning mechanism in the programming environment is designed to perform encryption and decryption for each of the basic and developed encryption algorithms, as shown in the figure (4.1)

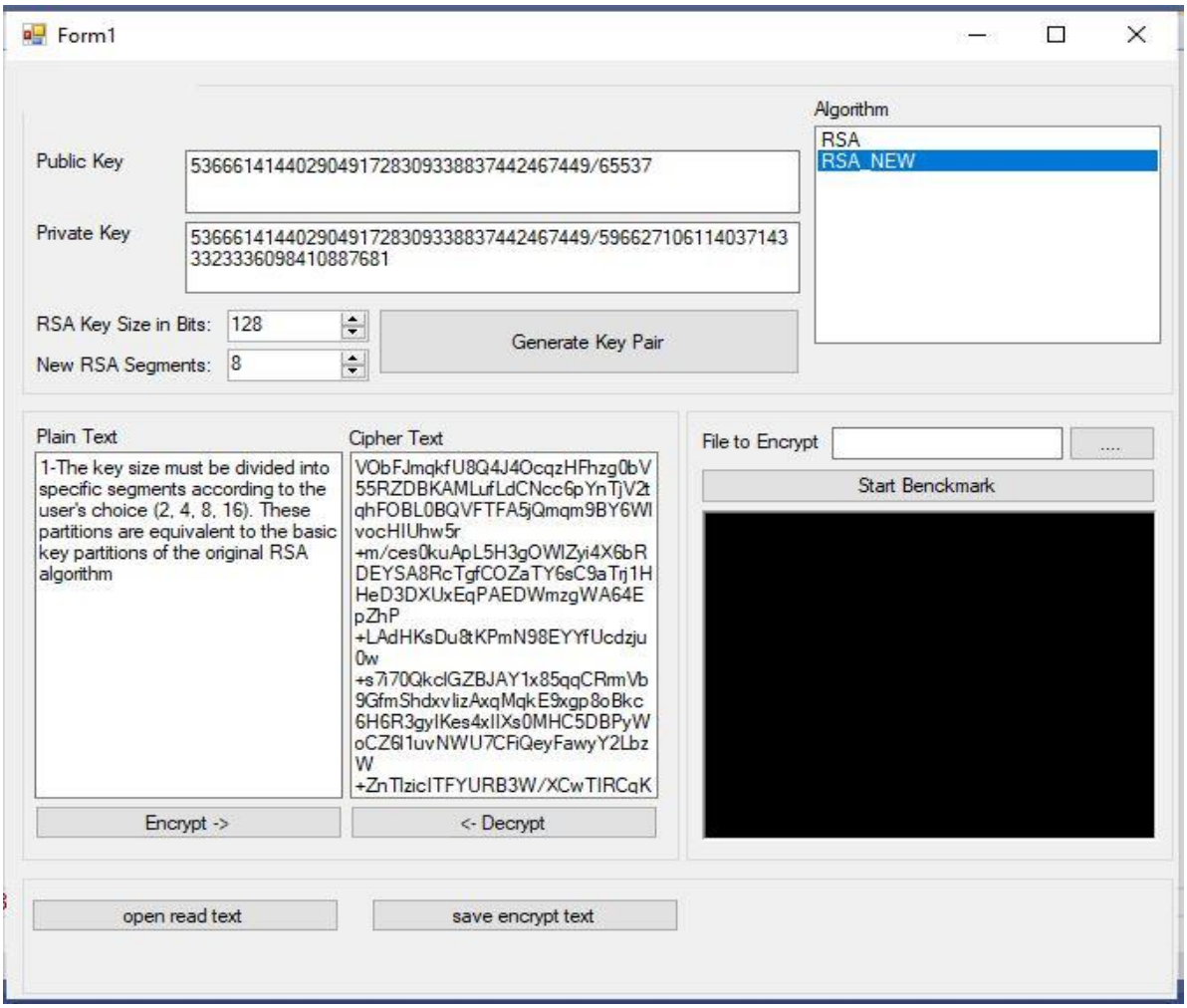


Figure (4.1) The main interface of the proposed algorithm, Source: Prepared by the researcher.

4.4 Implementation

There are three stages to execute the proposed algorithm:

4.4.1 Key generation

1- The key size must be divided into specific segments according to the user's choice (2, 4, 8, 16). These partitions are equivalent to the basic key partitions of the original RSA algorithm.

2- Each segment is considered as an independent key for the RSA algorithm, where the public and private key is generated for each of these segments, the size of the key in bits are determined.
3- Generate the odd numbers (p and q) randomly, these numbers may be prime or not, and tested by the Miller Rabin algorithm.
4- N is calculated according to the following equation: N= p*q.
5- Calculated the Phi according to the equation below: **Phi = (p - 1) * (q - 1).**
6- Finally, compute (d) according to the following equation: **d≡e^-1(modϕ(n)).**
7- Public and private key sequences are arranged randomly depending on PRNG.

The following is an example, the key size is 2048 is divided into 4 segments, so, the key size becomes 512 bits for each segment. Four public and private keys are used for encryption and decryption. For each segment, the following are public and private keys.

Public keys (N,e):

N1=*22212722372975676089111559326928884659995907637818964056436716467912945213395552702151470349122284011902261282734886662350528341443980661673988730648059 7*

e1=*65537*

N2=*37685626991474665953336594846939979193613248517587852297503565202377388030153762354388482940149639031260102308289072786006222009750916764197452645222993*

e2=*65537*

N3=*1343574442618576290796390002395344767313987368463764939198101050392258059630056023483463385533272729067554386170296779718763089052217283874952597854345357*

e3=*65537*

N4=*1560651547752990907678173551593038746010695464398745360576970223652259557450375489007948976762695716519293798576959506279479807744173214730517314901591939*

Private keys (N, d):

N1=*2221272237297567608911155932692888465999590763781896405643671646791294521339555270215147034912228401190226128273488666235052834144398066167398873064805 97*

d1=*1654472786234393781549188484344273579494087668694775335231856627042285285634896295336259085911697442091014543222208743976350977156601516345911662026301 45*

N2=*3768562699147466595333659484693997919361324851758785229750356520237738803015376235438848294014963903126010230828907278600622200975091676419745264522299 3*

d2=*9302807169508478352580670940595312318145095663785896126905605655493253941306108146128951982737047900370142801929856470501622373749829418564755651544033*

N3=*1343574442618576290796390002395344767313987368463764939198101050392258059630056023483463385533272729067554386170296779718763089052217283874952597854345357*

d3=*28687057960482991343994820183282816316011757094333678371300224297936840301512632346972356341195665847350839784147207403421069271583240244423891714817497 7*

N4=*15606515477529909076781735515930387460106954643987453605769702236522595574503754890079489767626957165192937985769595062794798077441732147305173149015919 39*

d4=*90659574094792433047310290801682374993321615309099645425524258639646571572837043664497185338104227347905864158229444373410738062210788098000806415899306 5*

Another example, if key size (2048) is divided into 8 segments, eight public and private keys will be generated, the following explains these keys:

Public keys (N, e):

N1=*80340647890773644216140460405606177547515191266553162509037820564793637010 39*

e1=*65537*

N2=*15989320267600455539594622753322403278937956280548393377570563624702030248843*

e2=*65537*

N3=*17346895307474806142894966361407657089639181618180328156921672850524978544697*

e3=*65537*

N4=666704496719632233146450949600249464255365927581755990493788910029932742521

e4=65537

N5=194467582748104394582538868541848496988703580906725531858981088602425384813 53

e5=65537

N6=*1037595192244456755867532805456267620004951557106502190695517829374945274218 3*

e6=*65537*

N7=*1037595192244456755867532805456267620004951557106502190695517829374945274218 3*

e7=*65537*

N8=1037595192244456755867532805456267620004951557106502190695517829374945274218 3

e8=65537

Private keys (N, d):

N1=*8034064789077364421614046040560617754751519126655316250903782056479363701039*

d1=*5043157168589555441060473321369962827173906504376218603939165340207065511705*

N2=*1598932026760045553959462275332240327893795628054839337757056362470203024884 3*

d2=*1207207464731583900017824325887889034640141271540660321489592891821924248835 3*

N3=*1734689530747480614289496636140765708963918161818032815692167285052497854469 7*

d3=*3126501478125217970915289724292342425494242012679010260956843796500925376529*

N4=*6667044967196322331464509496002494642553659275817559904937889100299327425 21*

d4=*1571313556667445789886641342680539729243266515867321664194622831846968248 25*

N5=*1944675827481043945825388685418484969887035809067255318589810886024253848135 3*

d5=*1237539863907783676466631986726022462928339341923643134321652294155207030339 3*

N6=*1037595192244456755867532805456267620004951557106502190695517829374945274218 3*

d6=*4222290642227660286896139949755588622817391093400879638638464371316304976113*

N7=*1037595192244456755867532805456267620004951557106502190695517829374945274218 3*

d7=*42229064222766028689613994975558862281739109340087963863846437 1316304976113*

N8=*10375951922444567558675328054562676200049515571065021906955178 293749452742183*

d8=*42229064222766028689613994975558862281739109340087963863846437 1316304976113*

4.4.2 Encryption

When encrypting a particular message, the Base64 system is used to and convert to a message with specific characters and the message must be converted into a series of bytes by ASCII to be encoded in the proposed algorithm.

1- Add zeros to complete the block size in the last block.

2- The proposed RSA algorithm uses randomized padding (we will use the OAEP padding).

3- PRNG are used to select Public (N) and private keys and store that arrangement in array to use it later for encryption and decryption process.

4- Each block is divided into multiple segments according to user selection.

5- Each segment is converted from a byte to a single BigInteger (m). The encryption process is done according to the following equation:

c=m ^e mod n

6- Each segment is converted from BigInteger to bytes.

7- All segments are encrypted in block and collect bytes for all parts of the block until the first block has been encrypted. Then it can be moved to the second block to complete the message.

8- Once all the blocks of the message have been encrypted, the encrypted buffer will be returned and converted to special letters and numbers using (Base64) to obtain the ciphertext.

4.4.3 Decryption

The following represent the stages that are needed to decrypt the message in the proposed algorithm:

1. Decoding the encrypted message, which consists of symbols, letters, and numbers that have been transformed from (DecodeBase64) system into bytes. These bytes are used in the decrypt buffering function.
2. All bytes are divided to blocks.
3. PRNG are used to select private keys and store that arrangement in an array to use in the decryption process.
4. Each block is divided into multiple segments according to user selection.
5. Each segment is converted from a byte to a single BigInteger (c). The decryption process is done according to the following equation:

   $m=c^{\wedge}d \bmod n$
6. Each segment is converted from BigInteger to bytes.
7. All segments are decrypted in block and collect bytes for all parts of the block until the first block has been decrypted. Then it can be moved to the second block to complete the message.
8. Once all the blocks of the message have been decrypted, the decrypted buffer will be returned and converted to specific characters by ASCII and using the Decode Base64 system to obtain the plaintext.

4.5 Example of proposed RSA algorithm (4 segments)

PLAIN TEXT MESSAGE:

364896564397011459724575230981320914472256689895283825736949671285
945813416963547069428149180284612060787604195762648578260270922849

141447807801785790099965423415165889627382848490753196984483917921
633499173120483800789218174753482435328041525968282689242448449908
7383676687756808368

Public keys:

N1=*359557945626895706029209358606159458076028232774171904479940880*
039826030573670348152090427458587042740131583412224644326522075737
778207424132031862387667 3

e1=*65537*

N2=*630041772140322885552680297537207868038540924372949818335594721*
004059772311351998907806239287401246481669015328572401425592321913
704667306507466064105513

e2=*65537*

N3=*469376045812522401467826736690437436344553790278511876225191867*
673522183590723002493509013856174665632433337419755080386500306517
806536804440195799711174 7

e3=*65537*

N4=*531749391813446873936487206861315289052117768299718357293654737*
931545301847331447865514677683611903734660080409463908966700510691
718622708857777442540383

e4=*65537*

private keys:

N1=359557945626895706029209358606159458076028232774171904479940880039826030573670348152090427458587042740131583412224644326522075737782074241320318623876673

d1=*33285046619830226572907064066522862142018335984522079547268769170294956206220690672333928277629974223642505900533237285443336879069837554719683508675244 9*

N2=*63004177214032288555268029753720786803854092437294981833559472100405977231135199890780623928740124648166901532857240142559232191370466730650746606410551 3*

d2=*40405657389044007018598887507040063923676511056952684536437502668417279140399989128844534353060415106356561421594297176766000904537403513880511544427633*

N3=*46937604581252240146782673669043743634455379027851187622519186673522183590723002493509013856174665632433337419755080386500306517806536804440195799711174 7*

d3=*22485101482039368286742481954310211505008264409103072087660250936344796890773342064655900614473716859073031064803837768794234780612564527513981298411067 3*

N4=*53174939181344687393648720686131528905211776829971835729365473931545301847331447865514677683611903734660080409463908966700510691718622270885777744254038 3*

d4=*4049074820186803301482591142592288048713991349806848788071249234905224138103559483626569269742709948069165737139537204329573335461268780403172532649080 33*

CIPHER_TEXT:

KRw1vlhbinhipa3aLaQ5X2oO0v4DR6gyeonmQsExC0ovFwtt5CMQSVJ31bJST7JadgSkpDD8s+j+E3hTjmkeAUFVKmelbgK23KOEHLjRUa6tXBHmbV0Pq28uCvfiz3NqMSUKIpuQw2E4NF6JhqGQnoTf0ZcN87WxlHYLbHkedwZRvCcMtoKtkQXQ

XW41SihPPQ94x44TKp7JhiL215qwicpPZfA8QGiZphIl+d2RJEslNviEv7ALgHZS1

f+4ygsCCqsQy/hnlpgfgYgsr1LX1JrNdt72U/YUKsqbDvDNXYLgKmi3u4V9zxoMrv

elRWqqrNT+VrkrWmeABcEM0w+LA5juKZpw1p74cnDYQJpGs3YHfPnwsFoC+s

dA43Ax7SGyKC2E602H9INMnWgVN4eN6Ko10A8KN3cvN30d4thhQgXGYd+qR

mXwCy3JM7d/zdc59PLcTSYS4uMa7k2cBfKJfcN5VLO8hEaY+jcZrEZZUCymX+

QhgbiHqAn1N+76eeIKiWAhE30eZyxtRqIxlMJPkl3M5iRSVEmJ3fjRBa0wS/Ex0o

HkwWVkjHY4e0vI67oIZ0BPsVBuWAUQ22fQ2wWnAphYEamQHx2rCT4MlipP42

XOzQQuX9qoi3MzXKMTACR2CS4JA1C8YCn5Y2COh48lYB5xj74W5W8aTMaM

0OkrUQSgWRZj5tlfDJDl5LkuzdEjXCeiIVqoJxSWX+qzGEbxJxvhqq8z4wCmOS5s

aous8k/yAcDPcrNg52T7aXwGhSsGEHqOfP3/AYrcUuS83UaRPpB+jKImlFK1iYp

Tpm4YaW7dy+Nb3Ed7OuAYO659RT2ku5XYI3ZWWyBjod6RUUBgBeYec819iBqb

VyEJOdPL2WxpbHeXhxud3Pkoc0BnB86E7cpn5Y7rQjnQoOdSTrUttz/a3I/kl/MlHi

R6VcPdJAczR5nD6r5+MWr/v4xPCIIhVmhMWkojFaTUIURafOxazzlp9HK7mFM

qK6DueXTtXDrYHT0MjBkvoNeXBPCwWhsG

4.6 Testing methodology

By Comparing the regular RSA algorithm that uses key length (2048 bit) against the proposed RSA algorithm that uses segments 2, 4, 8, or 16.

The comparison is done to show:

1. Performance: Time that are needed to encrypt and decrypt data.
2. Strength: According to integer factorization problem test and brute-force attack.

All tests are preformed using the same keys on all algorithms to ensure a fair comparison .The files are used to test the algorithms mentioned above are obtained from [https://sample-videos.com], multiple file formats each with multiple sizes are used to perform testing process.

The files are

1. Randomly generated text files with 2 sizes 10,000 and 100,000 characters.
2. JPG image files with 3 sizes 100kb, 500kb and 1mb.
3. MP4 video files with 2 sizes 2mb and 5mb.

And the following Testing steps:

1- Chose encryption keys to use on all algorithm variations, the important factor to consider in this step is to reuse the same key in all algorithms to provide consistent results and fair comparison.

2- Select a file to test.

3- Apply the RSA variations, RSA 2048, while recording the time used to encrypt the file and then decrypt the encrypted file, after knowing the original file size, we can calculate the encryption and decryption Speeds.

4- Apply New RSA 2, New RSA 4, New RSA 8 and New RSA 16, while also recording encryption and decryption times and deriving the speeds in the same manner as above.

5- Using the speed numbers collected above we can compare the algorithms performance.

6- Apply the statistical testing on original file and all outputted encrypted files from step 3 and 4 and record the results.

4.7 Results

We tested the traditional RSA algorithm as a reference with key sizes (2048) and tested the proposed structure with 2 segmentation each is 1024bit, 4 segmentations each is 512bit, 8 segmentations each is 256bit, and finally 16 segmentations each is 128bit. Speed tests are performed on a file of size 10,174,700 bytes (Approximately 10MB). Speed results can be seen in **Table** (4.2)**.**

Table (4.2) Encryption and Decryption performance speed results

sequence	Algorithm	Encryption speed in KB/s	Decryption speed KB/s
1-	**RSA 2048**	**515.6**	**2.8**
2-	**Proposed (16 Segments)**	**3434.5**	**316.6**
3-	**Proposed (8 Segments)**	**2575.5**	**121.1**
4-	**Proposed (4 Segments)**	**1526.9**	**35.0**
5-	**Proposed (2 Segments**	**894.7**	**10.3**

time tests are performed on another file of size 9507843 Bytes (Approximately 9 MB). time results can be seen in Table (4.3).

Table (4.3) Encryption and Decryption performance time results

sequence	Algorithm	Encryption Time in ms	Decryption Time in ms
1-	**RSA 2048**	**26635**	**4749015**
2-	**Proposed (16 Segments)**	**4258**	**38145**
3-	**Proposed (8 Segments)**	**5368**	**109678**
4-	**Proposed (4 Segments)**	**8367**	**362710**
5-	**Proposed (2 Segments**	**14340**	**1280451**

According to the tables (4, 2), (4, 3), decryption process is much slower than encryption process, this because that the public key (65537) is fixed and small

number so the calculation speed is high while the private key is large number which leads in turn that calculation process is slow as result when use smaller the key size, the method become faster, the proposed algorithm consumes less time for all segments sizes.

Applying testing steps on Randomly generated text files with 2 sizes 10,000 and 100,000 characters, the speed results are in tables (4.4), (4.5) and figures (4.2), (4.3).

Table (4.4) Time and speed of encryption and decryption comparison between the proposed method and RSA algorithm of Random Text File with 10,000 Characters

RandomText-10000char.txt					
No	Algorithm	Encryption Time In ms	Encryption Speed in KB/s	Decryption Time In ms	Decryption Speed in KB/s
1	RSA 2048	195.00	515.94	35,165.00	2.85
2	New RSA 2	105.00	960.61	9,406.00	10.64
3	New RSA 4	62.00	1,639.23	2,679.00	37.34
4	New RSA 8	38.00	2,721.68	850.00	117.87
5	New RSA 16	29.00	3,681.10	312.00	320.77

Table (4.5) Time and speed of encryption and decryption comparison between the proposed method and RSA algorithm of Random Text File with 100,000 Characters

RandomText-100000char.txt					
No	Algorithm	Encryption Time In ms	Encryption Speed in KB/s	Decryption Time In ms	Decryption Speed in KB/s
1	RSA 2048	1,966.00	510.70	351,190.00	2.85
2	New RSA 2	1,046.00	963.79	93,803.00	10.66
3	New RSA 4	618.00	1,644.12	26,402.00	37.88
4	New RSA 8	385.00	2,681.68	8,147.00	122.77
5	New RSA 16	309.00	3,452.27	2,944.00	339.70

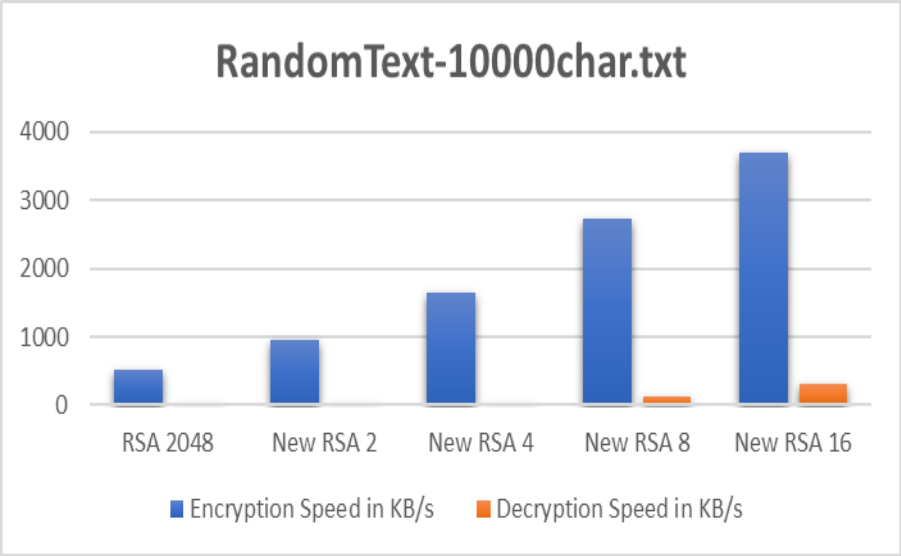


Figure (4.2) Test speed results for encrypting and decrypting comparison between the proposed method and RSA algorithm of Random Text File

with 10,000 Characters

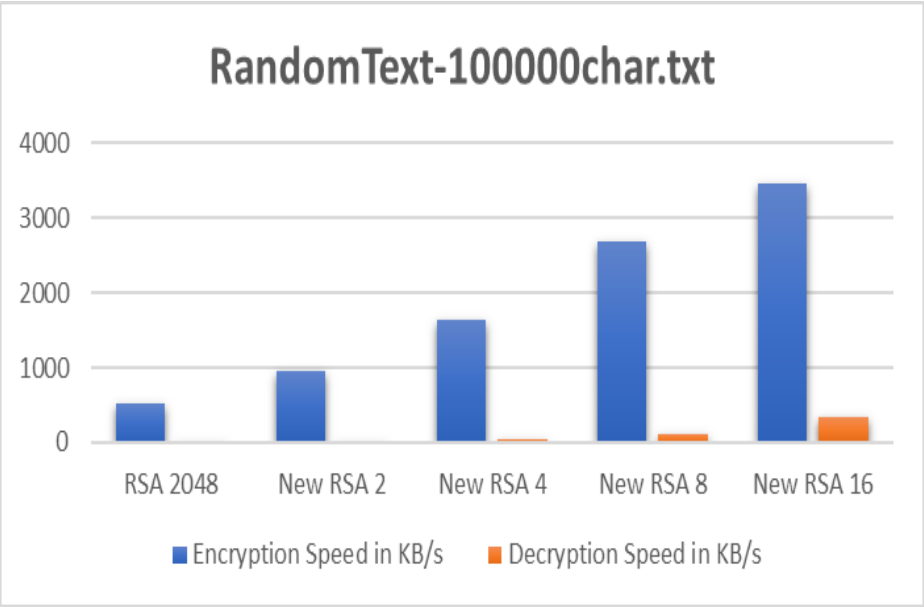


Figure (4.3) Test speed results for encrypting and decrypting comparison between the proposed method and RSA algorithm of Random Text File

with 100,000 Characters

Applying testing steps on JPG image files with 3 sizes 100kb, 500kb and 1mb, the speed results are in tables (4.6), (4.7), (4.8) and figures (4.4), (4.5), (4.6).

Table (4.6) Time and speed of encryption and decryption comparison between the proposed method and RSA algorithm of JPG image files 100kb

SampleJPGImage_100kb.jpg					
No	Algorithm	Encryption Time In ms	Encryption Speed in KB/s	Decryption Time In ms	Decryption Speed in KB/s
1	RSA 2048	202.00	512.00	36,029.00	2.86
2	New RSA 2	110.00	942.55	9,850.00	10.44
3	New RSA 4	62.00	1,684.65	2,699.00	38.09
4	New RSA 8	37.00	2,871.35	832.00	123.70
5	New RSA 16	57.00	1,926.74	290.00	355.03

Table (4.7) Time and speed of encryption and decryption comparison between the proposed method and RSA algorithm of JPG image files 500kb

SampleJPGImage_500kb.jpg					
No	Algorithm	Encryption Time In ms	Encryption Speed in KB/s	Decryption Time In ms	Decryption Speed in KB/s
1	RSA 2048	1,003.00	512.51	179,337.00	2.86
2	New RSA 2	545.00	946.97	48,166.00	10.63
3	New RSA 4	315.00	1,651.40	13,709.00	37.35
4	New RSA 8	196.00	2,697.14	4,047.00	126.54
5	New RSA 16	155.00	3,524.54	1,537.00	333.22

Table (4.8) Time and speed of encryption and decryption comparison between the proposed method and RSA algorithm of JPG image files 1mb

SampleJPGImage_1mb.jpg					
No	Algorithm	Encryption Time In ms	Encryption Speed in KB/s	Decryption Time In ms	Decryption Speed in KB/s
1	RSA 2048	2,031.00	510.87	361,687.00	2.86
2	New RSA 2	1,116.00	933.39	99,196.00	10.42
3	New RSA 4	633.00	1,658.54	27,399.00	37.72
4	New RSA 8	400.00	2,666.88	8,289.00	124.67
5	New RSA 16	316.00	3,488.41	3,042.00	339.72

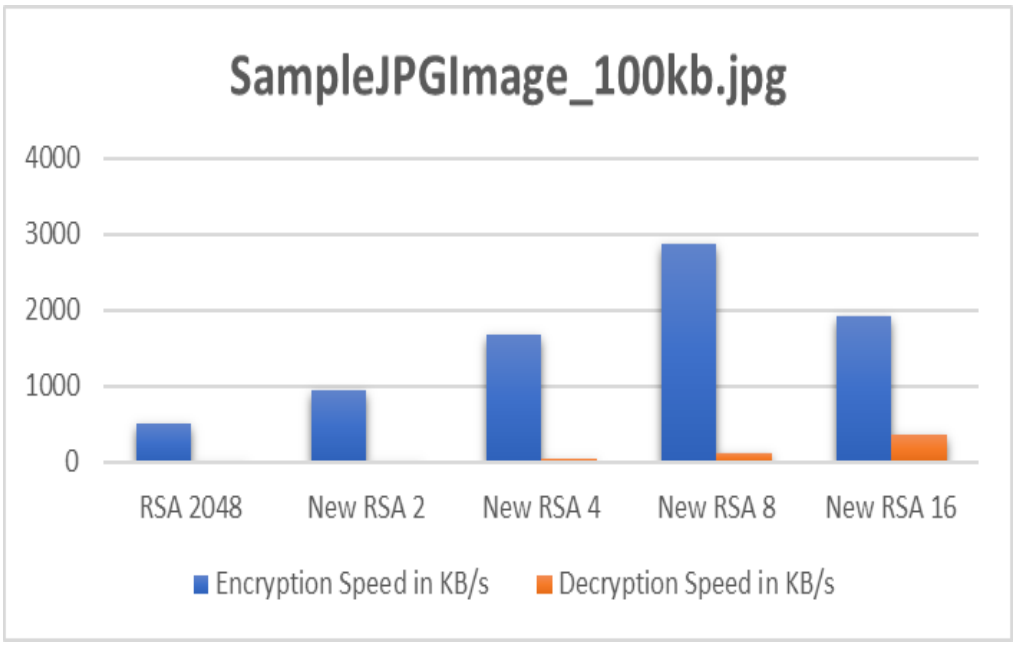


Figure (4.4) Test speed results for encrypting and decrypting comparison between the proposed method and RSA algorithm of JPG Image File 100kb.jpg

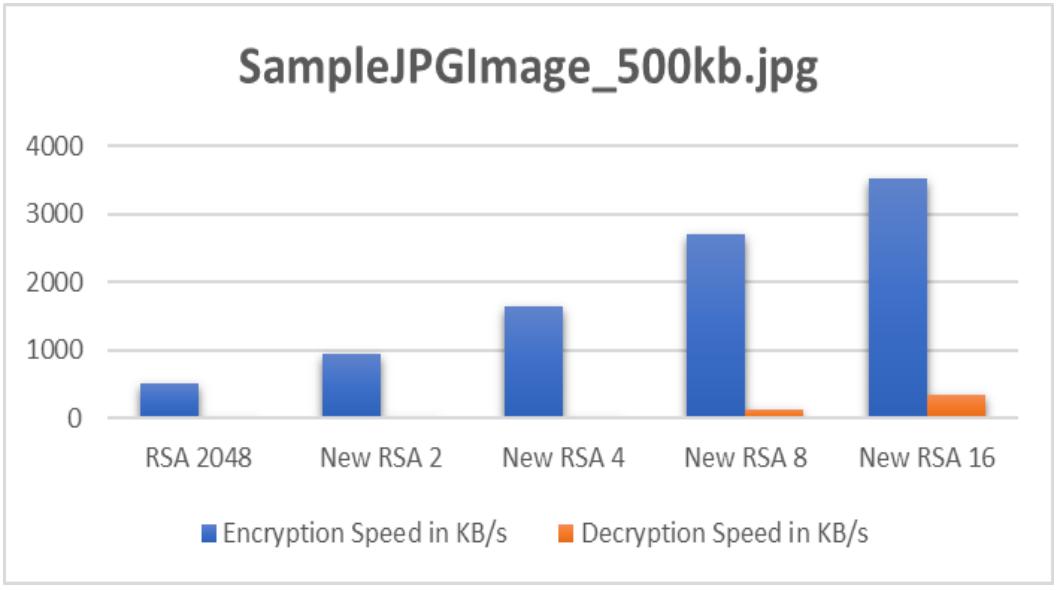


Figure (4.5) Test speed results for encrypting and decrypting comparison between the proposed method and RSA algorithm of JPG Image File 500kb.jpg

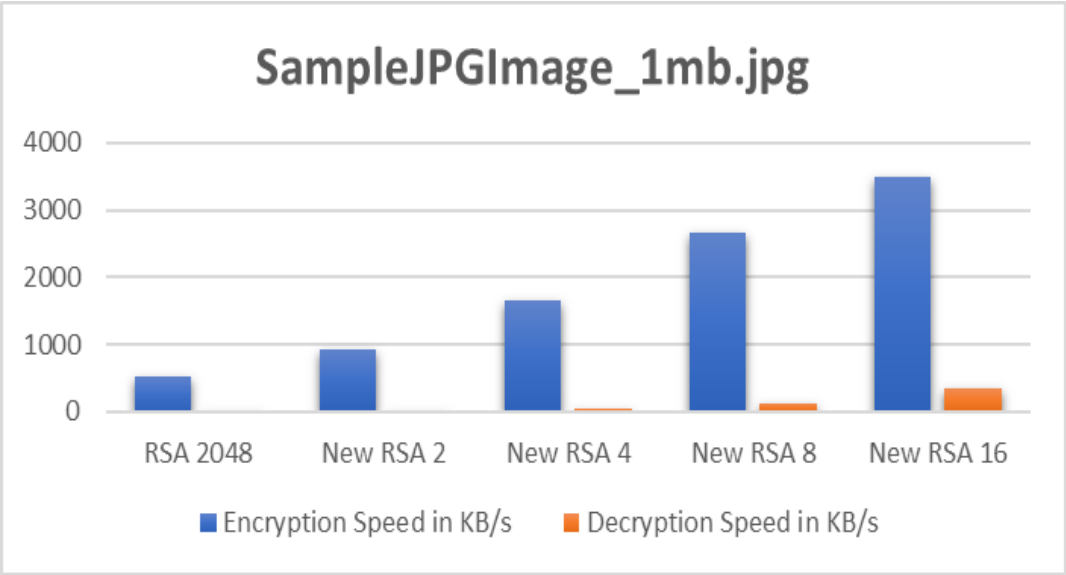


Figure (4.6) Test speed results for encrypting and decrypting comparison between the proposed method and RSA algorithm of JPG Image File 1mb.jpg

Applying testing steps on MP4 video files with 2 sizes 2mb and 5mb, the speed results are in tables (4.9), (4.10) and figures (4.7), (4.8).

Table (4.9) Time and speed of encryption and decryption comparison between the proposed method and RSA algorithm of 2mb MP4 Video File

big_buck_bunny_720p_2mb.mp4					
No	Algorithm	Encryption Time In ms	Encryption Speed in KB/s	Decryption Time In ms	Decryption Speed in KB/s
1	RSA 2048	4,172.00	507.28	739,194.00	2.85
2	New RSA 2	2,222.00	956.14	198,372.00	10.63
3	New RSA 4	1,333.00	1,606.48	57,137.00	36.89
4	New RSA 8	817.00	2,663.40	17,408.00	121.09
5	New RSA 16	625.00	3,597.52	6,143.00	343.14

Table (4.10) Time and speed of encryption and decryption comparison between the proposed method and RSA algorithm of 5mb MP4 Video File

big_buck_bunny_720p_5mb.mp4					
No	Algorithm	Encryption Time In ms	Encryption Speed in KB/s	Decryption Time In ms	Decryption Speed in KB/s
1	RSA 2048	10,365.00	508.89	1,841,846.00	2.85
2	New RSA 2	5,605.00	944.76	495,486.00	10.60
3	New RSA 4	3,242.00	1,646.31	140,790.00	37.32
4	New RSA 8	1,981.00	2,737.69	41,594.00	126.31
5	New RSA 16	1573	3562.843	15811	332.3054

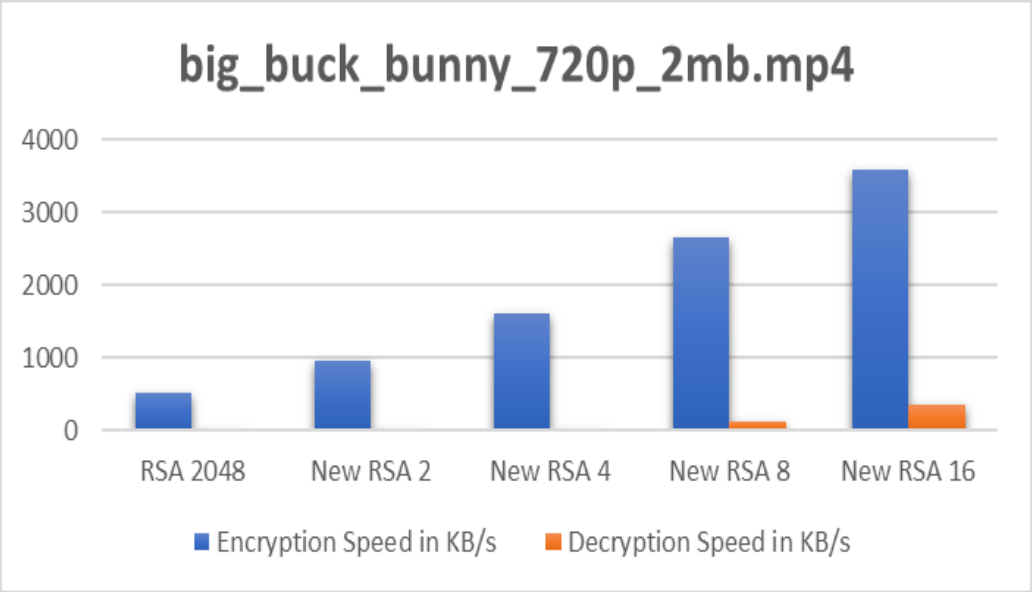


Figure (4.7) Test speed results for encrypting and decrypting comparison between the proposed method and RSA algorithm of 2mb.MP4 Video File

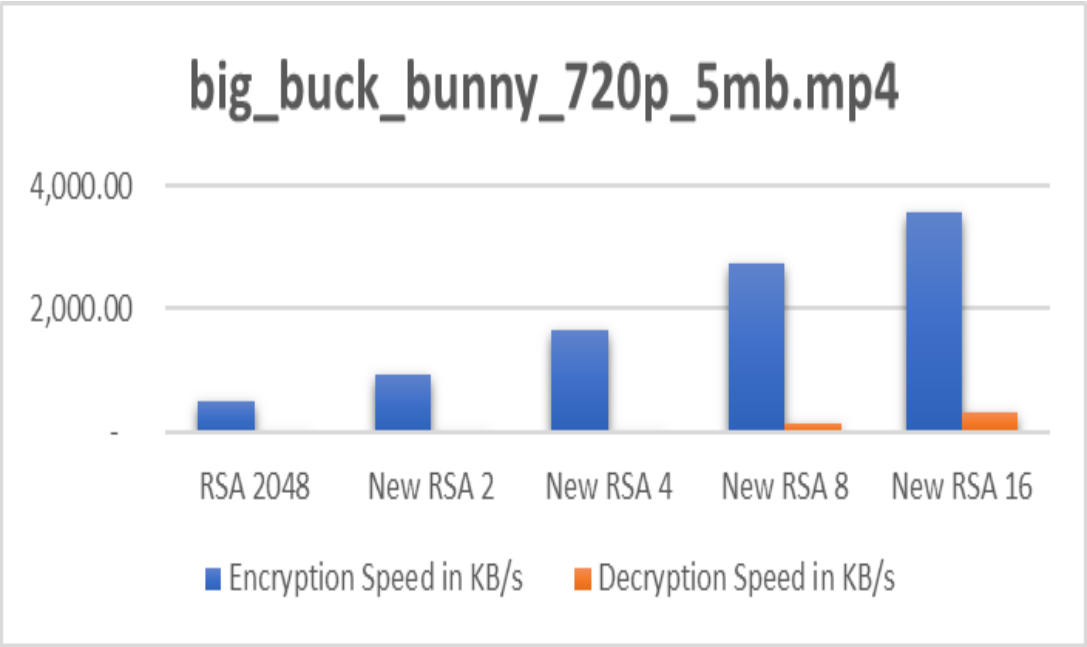


Figure (4.8) Test speed results for encrypting and decryption comparison between the proposed method and RSA algorithm of 5mb.MP4 Video File

The average speed of each algorithm can be calculated by applying the mean function for tables (4.4), (4.5), (4.6), (4.7), (4.8), (4.9), and (4.10), the average encryption speeds and average decryption speeds are shown in table (4.11) and also plotted in figure (4.9).

Table (4.11) Average Speed for every tested algorithm

Average Speed			
No	**Algorithm**	**Encryption Speed in KB/s**	**Decryption Speed in KB/s**
1	**RSA 2048**	**511.17**	**2.85**
2	**New RSA 2**	**949.74**	**10.58**
3	**New RSA 4**	**1,647.25**	**37.51**
4	**New RSA 8**	**2,719.98**	**123.28**
5	**New RSA 16**	**3,319.06**	**337.70**

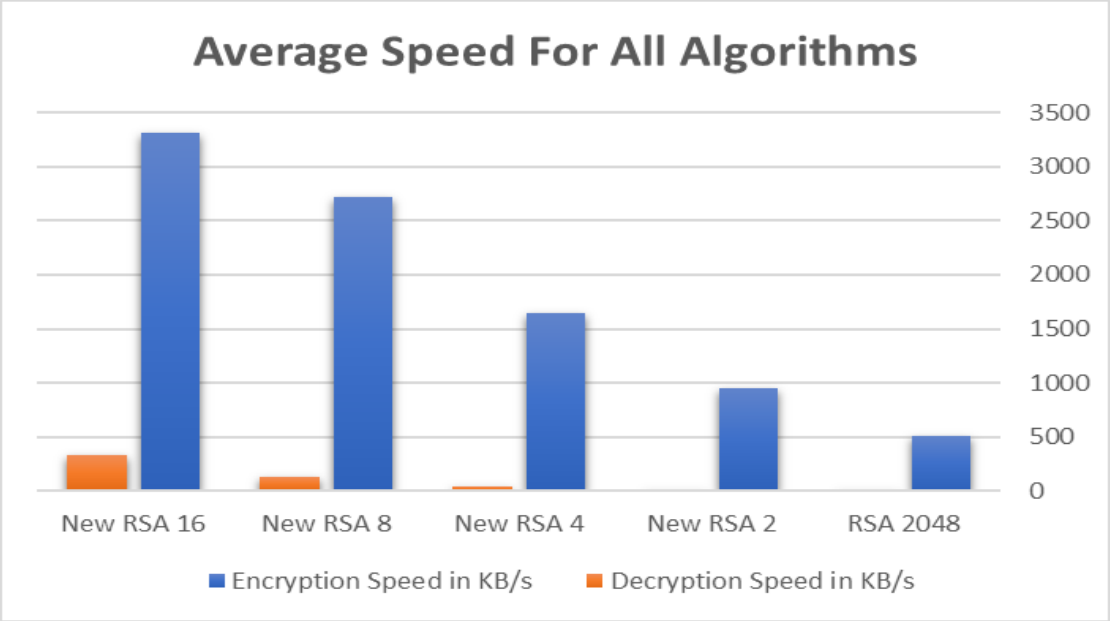


Figure (4.9) Average Speed for every tested algorithm

4.8 Discussion

4.8.1 Speed comparison

After looking at the performance results in Average Speed table (4.11), and figure (4.9), with each key size increase we get almost half the encryption speed and one third the decryption speed. When looking at rows (1-5), the speed of encryption is increased when no. of segment is increased in the proposed RSA algorithm, also the speed of the New RSA is higher than the speed of the RSA 2048 in row five while using the same block size.

4.8.2 The problem of the factorization in cryptographic systems RSA

Today, there is no algorithm for factoring in a large random number factoring a number, we know is the product of two primes should be simpler than factoring a

number we don't know is the product of two primes. The modulus n (which we know is the product of two big random primes) and the encryption exponent e make up RSA's public key, the decryption exponent **d** is the private key.

The suggested algorithm's random huge integer online representing **N** was tested using the prime factorization problem and Decomposition into prime factors, and the outcome was found to be infinity According to the following site (https://www.calculatorsoup.com/calculators/math/prime-factors.php) Figure (4.10) is a screenshot from the webpage where the test was performed.

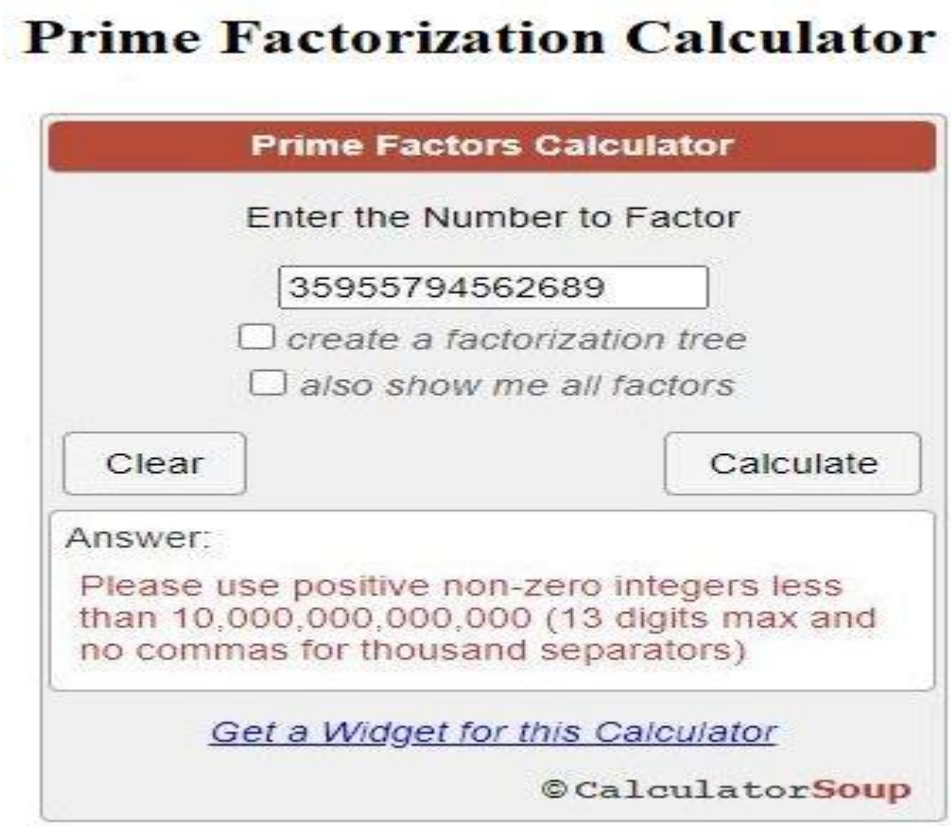


Figure (4.10) tested integer factorization problem

4.8.3 Brute force attack

Brute force assaults are used to try all possible keys until the correct one is found. Hackers continue to use this strategy. Although a brute-force attack cannot be avoided, it can be rendered ineffective. So, to eliminate this weakness, because it uses numerous public and private keys for encryption and decryption, the proposed algorithm is more secure against Brute force attacks than the RSA cryptosystem.

If the encrypted text and original message are obtained, the private key should be extracted for all possible combinations. However, because many private keys are used in the proposed approach, it will be difficult for an intruder to decrypt the ciphertext if only one private key is discovered, because the second key is still secure, and if brute force attacks are used on all the keys, the attacker's time will increase exponentially.

As a result, the proposed approach is more secure against brute force attacks based on Modulus factoring and tracing private keys. As a result, the security of the proposed approach has enhanced tremendously against Brute-force assaults.

4.8.4 Chosen cipher text attack

RSA algorithm can be attacked using chosen ciphertext attack because it is deterministic algorithm. Because the proposed algorithm is non-deterministic, so it is secure from the Chosen cipher text attack.

4.9 Summary

We presented a proposed algorithm for RSA that resists several types of attack, brute force, chosen ciphertext, and correct attack of the factoring problem, Splitting the key into several parts speeds up the encryption and decryption process by using

multiple keys whose order is related to the randomizer through the use of a pseudo-random number generator (PRNG) with RSA to increase security.

As well as using Optimal Asymmetric Encryption Padding (OAEP) to achieve a high level of randomness, and converting the proposed RSA algorithm to a probabilistic algorithm.

Finally, after looking at all the results of the proposed algorithm in all the tables above, we notice that the proposed RSA speed is higher than the RSA speed while using the same block size.

Chapter five

Conclusion and Future work

CHAPTER FIVE

CONCLUSIONS AND FUTURE WORKS

5.1 Conclusion

Depending on the results, conclusions can be drawn as follows:

1- Using several public and private keys for each segment make the proposed algorithm be more sophisticated and faster.
2- The original RSA algorithm is a deterministic technique in general, in any way, the random generator with the proposed RSA algorithm can be used to increase randomness and the proposed algorithm would be non-deterministic algorithm.
3- The proposed RSA algorithm is non- deterministic when using the optimal asymmetric encryption padding system (OAEP), if a different padding system is used, the proposed algorithm may be deterministic.
4- The proposed algorithm in all of its possible segmentations is faster than the original algorithm.
5- when using the random generator with the proposed technique. and does not require the padding (OAEP) mechanism, the last key of the public and private keys can be removed and to make the method non deterministic.

5.2 Future work

There are many suggestions can be used to improve the efficiency of the proposed RSA algorithm described in this thesis as follows.

1- The keys of the proposed system are stored and recalled offline before the operation begins. As a result, the speed required for encryption and decryption increases compared to standard RSA, during storage, the private

key must be encrypted using a symmetric encryption method (AES or DES) to ensure protection.

2- Three prime numbers (P, q, S) or more can be used in the suggested method, instead of two numbers, it generates a huge prime number that symbolizes n.

3- We propose that the algorithm be improved further by merging the proposed RSA method with the ElGamal algorithm.

References

References

[1] F. Mallouli, A. Hellal, N. S. Saeed, and F. A. Alzahrani, "A Survey on Cryptography: Comparative Study between RSA vs ECC Algorithms, and RSA vs El-Gamal Algorithms," in *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2019, pp. 173–176.

[2] G. Ramakrishnan, "Design and Verification of an RSA Encryption Core," 2019.

[3] B. Habib, B. Cambou, D. Booher, and C. Philabaum, "Public key exchange scheme that is addressable (PKA)," in *2017 IEEE Conference on Communications and Network Security (CNS)*, 2017, pp. 392–393.

[4] A. A. Ayele and V. Sreenivasarao, "A modified RSA encryption technique based on multiple public keys," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 1, no. 4, pp. 859–864, 2013.

[5] R. Patidar and R. Bhartiya, "Modified RSA cryptosystem based on offline storage and prime number," in *2013 IEEE International Conference on Computational Intelligence and Computing Research*, 2013, pp. 1–6.

[6] R. Patidar and R. Bhartiya, "Modified RSA cryptosystem based on offline storage and prime number," *2013 IEEE Int. Conf. Comput. Intell. Comput. Res. IEEE ICCIC 2013*, pp. 1–6, 2013, doi: 10.1109/ICCIC.2013.6724176.

[7] R. Minni, K. Sultania, S. Mishra, and D. R. Vincent, "An algorithm to enhance security in RSA," in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 2013, pp. 1–4.

[8] M. A. Islam, M. A. Islam, N. Islam, and B. Shabnam, “A modified and secured RSA public key cryptosystem based on ‘n’ prime numbers,” *J. Comput. Commun.*, vol. 6, no. 03, p. 78, 2018.

[9] I. Jahan, M. Asif, and L. J. Rozario, “Improved RSA cryptosystem based on the study of number theory and public key cryptosystems,” *Am. J. Eng. Res.*, vol. 4, no. 1, pp. 143–149, 2015.

[10] N. Kumar and P. Chaudhary, “Implementation of modified RSA cryptosystem for data encryption and decryption based on n prime number and bit stuffing,” in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, 2016, pp. 1–6.

[11] A. Rai and S. Jain, “Modified RSA Cryptographic System with Two Public keys and Chinese Remainder Theorem,” *Int. J. Comput. Sci. Eng.*, vol. 4, no. 7, 2017.

[12] F. Sultana, B. Choudhury, M. S. Shobha, and J. Mungara, “A Study on Data Encryption Using AES and RSA,” *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 5, no. 2, pp. 1302–1309, 2017.

[13] V. Lozupone, “Analyze encryption and public key infrastructure (PKI),” *Int. J. Inf. Manage.*, vol. 38, no. 1, pp. 42–44, 2018.

[14] M. U. Bokhari and Q. M. Shallal, “A review on symmetric key encryption techniques in cryptography,” *Int. J. Comput. Appl.*, vol. 147, no. 10, 2016.

[15] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, “Comprehensive study of symmetric key and asymmetric key encryption algorithms,” in *2017 international conference on engineering and technology (ICET)*, 2017, pp. 1–7.

[16] O. G. Abood and S. K. Guirguis, “A survey on cryptography algorithms,” *Int. J. Sci. Res. Publ.*, vol. 8, no. 7, pp. 495–516, 2018.

[17] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, “A comparative survey of symmetric and asymmetric key cryptography,” in *2014 international conference on electronics, communication and computational engineering (ICECCE)*, 2014, pp. 83–93.

[18] S. Patel and P. P. Nayak, “A Novel Method of Encryption using Modified RSA Algorithm and Chinese Remainder Theorem.” 2009.

[19] F. Shaheen, “M. Wasim Munir.”

[20] M. R. Joshi and R. A. Karkade, “Network security with cryptography,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 4, no. 1, pp. 201–204, 2015.

[21] C. Cooper, G. Woltman, S. Kurowski, and A. Blosser, “GIMPS Project Discovers Largest Known Prime Number.” .

[22] M. Siddhartha, J. Rodriques, and B. R. Chandavarkar, “Greatest common divisor and its applications in security: Case study,” in *2020 International Conference on Interdisciplinary Cyber Physical Systems (ICPS)*, 2020, pp. 51–57.

[23] I. Marouf, M. M. Asad, and Q. A. Al-Haija, “Reviewing and analyzing efficient GCD/LCM algorithms for cryptographic design,” *Int. J. New Comput. Archit. their Appl. (IJNCAA), By Soc. Digit. Inf. Wirel. Commun.*, vol. 7, no. 1, pp. 1–7, 2017.

[24] S. Beslin and S. Ligh, “Greatest common divisor matrices,” *Linear Algebra Appl.*, vol. 118, pp. 69–76, 1989.

[25] W. Stein, *Elementary number theory: primes, congruences, and secrets: a computational approach*. Springer Science & Business Media, 2008.

[26] M. Orozco and D. Gardiner, "'Euclidean' Number Theory," 2020.

[27] K. Nguyen, "Cryptography and Number Theory," *RN*, vol. 55, p. 7.

[28] "Theorems of Fermat, Euler, and Wilson." Jun. 13, 2020, [Online]. Available: https://math.libretexts.org/@go/page/8835.

[29] H. Nurdiyanto, R. Rahim, A. S. Ahmar, M. Syahril, M. Dahria, and H. Ahmad, "Secure a Transaction Activity with Base64 Algorithm and Word Auto Key Encryption Algorithm," in *Journal of Physics: Conference Series*, 2018, vol. 1028, no. 1, p. 12053.

[30] A. P. U. Siahaan, "Base64 character encoding and decoding modeling," 2017.

[31] S. Wen and W. Dang, "Research on Base64 Encoding Algorithm and PHP Implementation," in *2018 26th International Conference on Geoinformatics*, 2018, pp. 1–5.

[32] W. Muła and D. Lemire, "Faster Base64 encoding and decoding using AVX2 instructions," *ACM Trans. Web*, vol. 12, no. 3, pp. 1–26, 2018.

[33] L. Cantara, "METS: The metadata encoding and transmission standard," *Cat. Classif. Q.*, vol. 40, no. 3–4, pp. 237–253, 2005.

[34] D. Shah, "Digital security using cryptographic message digest algorithm," *Int. J. Adv. Res. Comput. Sci. Manag. Stud.*, vol. 3, no. 10, pp. 215–219, 2015.

[35] N. Cao, A. O'Neill, and M. Zaheri, "Toward RSA-OAEP without random oracles," in *IACR International Conference on Public-Key Cryptography*, 2020, pp. 279–308.

[36] D. Boneh, "Simplified OAEP for the RSA and Rabin functions," in *Annual*

International Cryptology Conference, 2001, pp. 275–291.

[37] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[38] E. Barker and Q. Dang, "Nist special publication 800-57 part 1, revision 4," *NIST, Tech. Rep*, vol. 16, 2016.

[39] K. D. M. AlSabti and H. R. Hashim, "A new approach for image encryption in the modified RSA cryptosystem using MATLAB," *Glob. J. Pure Appl. Math.*, vol. 12, no. 4, pp. 3631–3640, 2016.

[40] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," *IEEE J. Sel. areas Commun.*, vol. 24, no. 2, pp. 247–260, 2006.

[41] H. Mittal and A. Kakkar, "Optimized Encryption Algorithm using Dynamic Keys." 2014.

[42] B. S. Jones and S. D. A. Salagean, "No Title Final Year Project RSA Public-Key Cryptography," vol. 1,900 KB, p. 56, 2007, [Online]. Available: https://www.steve-j.co.uk/files/rsaproject.pdf.

[43] T. S. Obaid, "Study A Public Key in RSA Algorithm," *Eur. J. Eng. Technol. Res.*, vol. 5, no. 4, pp. 395–398, 2020.

[44] G. L. Miller, "Riemann's hypothesis and tests for primality," *J. Comput. Syst. Sci.*, vol. 13, no. 3, pp. 300–317, 1976.

[45] J. Rajput and A. Bajpai, "Study on deterministic and probabilistic computation of primality Test," 2019.

[46] R. Pavuluru, "Miller-Rabin." Desember, 2015.

[47] P. L'Ecuyer, "Random number generation," in *Handbook of computational statistics*, Springer, 2012, pp. 35–71.

[48] A. Okhrimenko and V. Kovtun, "Experimental research of developed arithmetic transformations according to RSA," in *XX International conference of higher education students and young scientists «POLIT. Challenges of science today: Modern information and communication technologies in aviation», Kyiv*, 2020, pp. 30–31.

[49] D. Boneh and H. Shacham, "Fast variants of RSA," *CryptoBytes*, vol. 5, no. 1, pp. 1–9, 2002.

[50] S. A. Jothi, "Evaluation of Symmetric Key Cryptosystem Based On Randomized Key Block Cipher Algorithm to Cryptanalytic Attacks," *Evaluation*, vol. 9, no. 2, 2019.

[51] C. J. Mok and C. W. Chuah, "An Intelligence Brute Force Attack on RSA Cryptosystem," *Commun. Comput. Appl. Math.*, vol. 1, no. 1, 2019.

[52] R. Novak, "SPA-based adaptive chosen-ciphertext attack on RSA implementation," in *International Workshop on Public Key Cryptography*, 2002, pp. 252–262.

الملخص

انتشر استخدام المعلومات الرقمية بشكل كبير في جميع أنحاء العالم. يتم استخدامه في البنوك والأسواق المالية والعملات الرقمية والمزيد. هذه المعلومات عرضة للتهديدات من قبل المتسللين لأن المعلومات عبر الشبكة، وقد تكون غير آمنة، مما يتسبب في حدوث انتهاكات عبر شبكات معينة. تُستخدم خوارزمية RSA لتوفير السرية والمصادقة لهذه المعلومات.

تعتبر خوارزمية RSA أشهر أنظمة تشفير المفتاح العام وتستخدم على نطاق واسع، Pretty Good Privacy (PGP)، وتستخدم نظام تشفير RSA وغالبًا ما تستخدم لإرسال رسائل مشفرة بين الأشخاص. إنه يعمل عن طريق تشفير رسالة باستخدام مفتاح عام مرتبط بمستخدم معين؛ عندما يتلقى هذا المستخدم الرسالة، فإنه يستخدمها ببساطة لتشفير مفتاح الجلسة بالمفتاح العام للمستقبل ثم يقوم بإلحاق مفتاح الجلسة المشفر RSA ببداية مستند المرسل المشفر بمفتاح الجلسة. يتم إرسال كل من المستند ومفتاح الجلسة معًا إلى جهاز الاستقبال.

أهم مشاكل خوارزمية RSA هي السرعة البطيئة والحتمية. تقدم هذه الأطروحة خوارزمية مقترحة لـ .RSA يعتبر الخيار المقترح احتماليًا من خلال استخدام مولد الأرقام العشوائية الزائفة (PRNG) معRSA ، وكذلك استخدام حشوة التشفير غير المتماثل الأمثل (OAEP) للوصول إلى مستوى عالٍ من الأمان عن طريق منع هجوم النص المشفر المختار وهو واحد من أكثر الهجمات شيوعًا ضد خوارزمية .RSA لحل السرعة البطيئة لخوارزميةRSA ، ينقسم مفتاح RSA إلى عدة مفاتيح لتشفير وفك تشفير الكتل باستخدام مفاتيح عامة وخاصة متعددة وفقًا لترتيب المولد العشوائي. يعتمد تسلسل المفاتيح الخاصة على الرقم الأولي للمولد العشوائي الزائف.

يؤدي تقسيم المفتاح إلى عدة أجزاء إلى زيادة سرعة عملية التشفير وفك التشفير. يمكن أن يؤدي استخدام PRNGو OAEPومفاتيح متعددة إلى زيادة الأمان. الخوارزمية المقترحة تقاوم عدة أنواع من الهجوم والقوة الغاشمة والنص المشفر المختار والهجوم الصحيح لمشكلة العوامل، وهجمات القوة الغاشمة غير عملية مع طول المفتاح، ولا يمكن استخدام هجوم النص المشفر المختار عند استخدام .OAEP فشلت خوارزمية مشكلة عامل العدد الصحيح في تحليل (N) بالحجم المستخدم.

أخيرًا، عند مقارنة سرعة خوارزمية RSA المقترحة مع خوارزمية RSA (لحجم المفتاح 2048). زادت نتيجة الخوارزمية المقترحة تقريبًا على النحو التالي: الجزء 2 مرتين، والجزء 4 ثلاث مرات، والجزء 8 أربع مرات، والجزء 16 أسرع بخمس مرات من خوارزمية.RSA

جمهورية العراق
وزارة التعليم العالي والبحث العلمي العراقية
جامعة القادسية
كلية علوم الحاسوب وتكنولوجيا المعلومات
قسم علوم الحاسوب

طريقة تطوير فعالة لتسريع وتأمين خوارزمية تشفير RSA

رسالة ماجستير

مقدمة الى مجلس كلية علوم الحاسوب وتكنولوجيا المعلومات في جامعة القادسية

كجزء من متطلبات نيل شهادة الماجستير في تخصص علوم الحاسوب

مقدمة بواسطة

علي نجم مهاوش الجبوري

تحت اشراف:

الأستاذ المساعد الدكتورة رنا جمعة سريح الجنابي

2021A.D. **1443 A.H.**