Secured Information Transfer Power by Modified and Optimized RSA Cryptosystem

Prashnatita Pal

July 30, 2023

Secured Information Transfer Power by Modified and Optimized RSA Cryptosystem

Prashnatita Pal[1]*,
[1],[2]Electronics & Communication Engineering, National Institute of Technology, Patna, India
*Corresponding Author & Email: prashnatitap@gmail.com

Abstract: Information are transferred through machines, the internet, and communications networks in the digital realm. In these kinds of circumstances, data security and confidentiality are paramount. Through the application of cryptography, it can shield data from unauthorized access. To enhance security, try to improve the RSA cryptosystem used in this research by introducing more prime numbers and adjusting the public key. After modification, compared the original and modified RSA algorithms and demonstrated, via qualitative and quantitative analysis, whether our modified algorithm is superior with respect to the original

Keywords – Information Security, Optimization, Modified RSA. Computational complexity, Keystream expansion

1. INTRODUCTION

Currently, we can observe that a significant quantity of data is being produced and sent through the internet due to the fast progress and development of numerous multimedia technologies. This makes editing, changing, and duplicating digital material relatively simple. Digital papers may also be duplicated and distributed quickly, which exposes them to a variety of dangers and puts our essential data at risk. There must be a solution to this significant privacy and security problem. As a result, privacy and security have become crucial.

We may use cryptography to safeguard our sensitive data and information while it is transferred over the internet. Information security is becoming more and more crucial as a result of the internet and telecommunications industries' fast and considerable expansion. The greatest method for safeguarding our confidential data and information is cryptography. Through the art of cryptography, we can safeguard and secure our data and information from unauthorized access. Data decryption & encryption techniques are the core concepts of cryptography. By converting plain text into an unintelligible format that an outsider cannot read, encryption reduces the likelihood that data may be attacked. Encryption is simply reversed during decryption. From the encrypted data, it transforms back to our original data. Two categories of cryptography algorithms exist. The first kind of decryption and encryption techniques are known as symmetric key type of cryptography. Second, there is asymmetric key cryptography, which employs two main keys: a public key use for encryption and a private key applicable for decryption.

The RSA cryptosystem has several benefits, which help explain why it is so widely used and well-liked in the cryptography community. The following are some benefits of using the RSA cryptosystem:

i) Security: The mathematical hardness of factoring big prime numbers is the main part for RSA. Because it is computationally impossible to broken RSA encryption without knowledge the private key, it offers a high degree of protection. The difficulty of factoring enormous numbers into their factors is the foundation of RSA's security.

ii) Asymmetric Key Encryption: RSA uses a pair of keys—a public key used for encryption and a private key used for decryption—to achieve asymmetric key encryption. This makes it possible for parties to communicate securely and confidentially without having to beforehand provide a secret key.

iii) Key Distribution: Because each participant may independently create their own key pair, RSA does not need a secure key distribution method. While the use of private keys is put away a secret, the public keys may be freely disseminated. Because of this, RSA is especially helpful in situations when secure exchange of key is unfeasible or difficult.

iv) Versatility: RSA has applications in key exchange, encryption, decryption, and digital signatures. Due to its adaptability, it is a commonly used algorithm for a variety of cryptographic applications, including secure email, secure file transfers, HTTPS-encrypted online surfing and secure communication protocols like SSH.

v) Standardization and Support: Due to significant research, testing, and standardization, RSA has gained broad acceptance in cryptographic protocols and libraries. It is simple to include RSA encryption into programs since RSA technology is incorporated into many software libraries and programming languages.

vi) Performance: Despite the computationally demanding nature of RSA encryption and decryption processes, the algorithm's performance may be enhanced by adopting effective implementations and hardware acceleration methods. Furthermore, symmetric session keys—which are used to encrypt the actual data effectively symmetrically—are often encrypted and decrypted using RSA. Asymmetric and symmetric encryption working together to provide a balance between security and speed.

It is important to know that RSA has flaws and restrictions, just like every other cryptographic method. Larger key sizes could be needed to maintain the same degree of security as computational power increases. If not properly designed and deployed, RSA is also vulnerable to attacks including side-channel attacks, padding oracle attacks, and timing attacks. Nevertheless, when built and utilised securely, RSA continues to be extensively used and trusted.

The RSA algorithm has been chosen as our preferred algorithm for implementation even though there are many cryptography algorithms available. Rivest, Shamir, and Adlemen are the acronym for RSA, which falls under the heading of asymmetric key cryptography. The most popular algorithm for creating security measures is this one. To boost the security level of the method, Finally, using qualitative and quantitative analysis, we compared the RSA and modified RSA algorithms.

The compares of standard RSA(SRSA) and Modified RSA(MRSA) algorithms fairly. Therefore, our goal is that any data or information is transferring from source to destination must be safe and secure from unauthorized users. So, we can do this using cryptography.

2. RELATED WORK

To provide readers with a broader understanding of the cryptography area, this part addressed the numerous outcomes attained using other earlier techniques. Numerous scholars conducted in-depth investigations in this area: Designing and implementing an RSA cryptosystem using Fernet cipher encryption was Babu and Vijayalakshmi's [1] work. A sophisticated and intricate method of encryption is built using multi-layer encryption. The research speeds up encryption and decryption by allowing message transmission across an unsafe network. A public key is used by the authors of [2] to propose an improved symmetric key cryptosystem that makes it harder to decrypt the original communication. The key size in the symmetric key scheme was 512 bits. The present ones, which are only effective for effectively transferring tiny quantities of information, are inefficient for encrypting vast amounts of data. Another research from [3] adds jumbled textual randomization and the RSA method to the electronic health record to assure improved encryption. By using the idea of scrambled alphanumeric randomization, the approach suggested a safe way of data decryption and encryption to give users a clear grasp of how the RSA works during encryption/ decryption process. Every letter is given a distinct numerical or letter sequence using innovative alphanumerical procedures. This method's drawback is that it takes longer to encrypt and decode data because of slower processing speed. The RSA Cryptosystem was used in the text file Security by the authors of [4]. The goal was to create and then implement an alternative RSA Cryptosystem ensuring the privacy of text files. The study is limited because it does not strengthen or offer RSA protection against popular RSA attacks. An effective cryptographic technique for text message security against brute force assaults was suggested in research by [5]. The goal of the study is to simplify and condense the ciphertext of the key. The method also makes encrypting communications simple. Some assaults may not be successfully resisted by the approaches. [6] created unique RSA algorithms for communicating via wireless devices that use unexpected bio signals and one-time encryption keys. The premier number generator (TPRNG) used in this design reduces the encryption key to a smaller, consumable RSA encryption key that is impossible to know or predict using biological signals. Verilog is used to implement algorithms. Only a real-time environment is appropriate for and suitable to the approach. Similar research in [7] suggested a strategy where the general key is obtained using optimization methods. The highest point of the Signals as Noise ratio is used to assess the optimization performance. The approach fends against statistical and differential assaults separately. The authors of [8] used new encryption technology to provide effective security and produce the outer system as a waveform so that the original data would not be modified or attacked. Timing attacks are resisted by the algorithm. The RSA cryptographic algorithm was created by authors in [9] employing three keys, however, the research still needs some protection to address the issue of information transfer from the server. The authors of [10] suggested a framework that offers dual layer of security for the RSA; however, this security mechanism is not universally applicable since the methods for key production have made the proposed system more complicated. [11] also created an improved RSA cryptosystem design. Four huge prime integers are used in the procedure. The computational and spatial complexity of the innovative system is greater than that of traditional RSA due to these many primes. Multiplying two significant figures yields the general component of n. Four significant prime numbers are multiplied to get the amount of encryption and decryption. Brute Forte Attack is resistant to the newly developed algorithm. Additionally, the new method is much more effective than Classical RSA. Authors [12] suggested a parallel method using a novel parallel type of data structure called a simultaneous search list of character in blocks. The RSA Cryptosystem is intended to be executed at a faster pace, and its security is not addressed by the system. Using the use of several keys for connection, the author [13] suggested an Optimized type of CRT-RSA Method for safe and reliable transmission; the level of security was compared to traditional RSA. The results of the experiments showed that the suggested algorithm increases security and reduces the participation of outsiders in communication, but it has the disadvantage that it uses more resources than traditional RSA. To increase security of data in the ambiance of cloud context, the results of [14] propose Quasai modified levy flying distribution for the RSA. The encryption system developed by RSA handles safe key creation and data protection, protecting data from unauthorized access. The suggested strategy [15] uses the Cuckoo Search Algorithms (CSA) to protect and solve data integrity issues while introducing an effective RSA cryptosystem. To prevent brute force attacks and improve key encryption, CSA is used. The suggested technique increases the length of the private key while still operating more quickly than traditional RSA. In their work, [16] performed double encryption utilizing both RSA and AES, encrypting the file twice. When compared to traditional RSA, the approaches boost security since the appropriate keys are produced during algorithm execution. [17] proposes a secured message transmission in the cloud utilizing the RSA method and an improved play fair cipher; the study's goals are to protect the key and offer security for the data transferred. The suggested system encrypts the content employing the play fair cipher in the first step, then conducts an XOR calculation on the text in the second stage, then uses the RSA algorithm to complete the process in the third stage. Though more computationally intensive, the suggested approach raises RSA's security level over traditional RSA. By combining the RSA and AES techniques with confirmation from a third party, the author [18] adds an estimate and guarantees the confidentiality of encrypted data. By preventing unauthorized access to the data, the system carefully managed security and privacy concerns and ensured authentication.

The RSA cryptosystem has undergone several revisions and variations throughout the years with the purpose of addressing different issues or adding new capabilities. Here are some examples of several RSA-modified cryptosystems:
Chinese Remainder Theorem (CRT) RSA [19]: This update uses the CRT to accelerate the decryption of an RSA key using the Chinese Remainder Theorem (CRT) RSA. To minimise the amount of modular exponentiation necessary, it requires pre-calculating certain values during the key creation phase and employing them throughout the decryption process.
Multi-Prime RSA [20]: This technique generates keys by employing numerous prime numbers, as opposed to simply two very big prime numbers. The efficiency of decryption, key generation, and encryption, procedures may increase because of this improvement.
Blinded RSA [21]: Blinded RSA employs randomization throughout the encryption and decryption procedures to thwart side-channel attacks. Blinding, or randomization, conceals important information to prevent leakage via auxiliary channels like time or power usage.

Multi-Exponent RSA [22]: In traditional RSA, encryption and decryption are accomplished using the same exponent. Different exponents are utilised for encryption and decryption in multi-exponent RSA, enabling optimised performance in certain circumstances. By carefully choosing the exponents, it may increase the effectiveness of either encryption or decryption procedures.

Montgomery RSA [23]: The modular exponentiation processes in RSA may be accelerated using the montgomery reduction approach. It speeds up computing by switching out the conventional modular exponentiation for a sequence of multiplications and modular reductions.

Homomorphic RSA [24] is a modified version of RSA that enables calculations to be done directly on encrypted data without having to first decode it. Using this characteristic, computations may be safely outsourced to unreliable servers while maintaining data secrecy.

Balanced RSA [25] is a variant that seeks to equally split the computing workload between operations using the public and private keys. By using many exponents with distinct mathematical features, it achieves this equilibrium. It's important to remember that the security and applicability of these modified RSA cryptosystems might change based on the implementation details, key sizes, parameter selections, and the application's cryptographic objectives. When contemplating the implementation and usage of modified RSA schemes, careful study, assessment, and adherence to best practises are required.
Due to various security flaws in RSA, including ciphertext attacks, brute-force attacks, typical modulus type attacks, measuring attacks, and others, various researchers worked hard to introduce a variety of modified RSA algorithms. Others RSA with different encryption techniques to address the flaws. However, several of the suggested algorithms are still susceptible to certain assaults in one way or another, and some of the most effective methods may have longer calculation times, use more memory, and use more computing resources.

2.1. Study motivation

The uncovered certain RSA flaws, which render RSA vulnerable to certain common attacks including a thorough search, timed attacks, popular mod attacks, and a few gaps in the previous research, based on a comprehensive assessment of the available publications. Therefore, the study is now focused on applying the suggested MRSA to enhance RSA security without employing over three primes or lengthening the key while still providing adequate security against a variety of attacks. To resist attacks on information security, this research provides an enhanced stochastic bit-insertion method with a modified RSA.

3. PROPOSED MODIFIED RSA (MRSA)

There have been several modifications and variants of the RSA cryptosystem proposed over the years. While these modifications aim to address certain limitations or provide additional features, they can also introduce new vulnerabilities or drawbacks. Here are some drawbacks associated with different types of modified RSA cryptosystems:

1. Low Public Key Efficiency: Some modified RSA schemes, such as RSA with Chinese Remainder Theorem (CRT), can improve the efficiency of decryption by using the CRT to speed up the modular exponentiation process. However, these schemes often require additional parameters and computations, which increase the size of the public key. This can lead to larger storage requirements and increased transmission overhead.

2. Reduced Security Margin: Certain modifications to RSA, such as RSA with small public exponents (e.g., small Fermat primes), aim to improve efficiency by using smaller exponents. While this can accelerate decryption and encryption, it can also reduce the security margin of the algorithm. Small exponents make the encryption more vulnerable to attacks like the Wiener's attack, where an attacker can factorize the modulus when the exponent is too small.

3. Vulnerability Attacks in Side-Channel: Some implementations or modifications of RSA may be susceptible to side-channel attacks. Side-channel attacks exploit information leaked during the cryptographic operation, such as power consumption, timing information, or electromagnetic radiation. If proper countermeasures are not taken, sensitive data may be extracted via side-channel attacks., including the private key.

4. Incompatibility and Lack of Standardization: Modified RSA schemes often introduce variations in the algorithm, parameter choices, or key formats. This can lead to compatibility issues between different implementations or systems. Lack of standardization may result in interoperability problems and difficulties in securely exchanging keys or encrypted data between different RSA variants.

5. Limited Adoption and Support: New modifications of RSA may lack the extensive testing, analysis, and peer review that the original RSA algorithm has undergone. As a result, they may not have the same level of trust, support, and available cryptographic libraries or tools. This limited adoption can make it more challenging to integrate and deploy modified RSA schemes in real-world applications.

6. Increased Key Size and Computation Complexity: Some modifications of RSA aim to enhance security by using larger key sizes or more complex mathematical operations. While this can provide additional

security, it also leads to increased computational requirements. Larger key sizes can impact encryption and decryption performance, requiring more computational resources and potentially slowing down the cryptographic operations.

It is the important to note that the specific drawbacks and vulnerabilities associated with modified RSA schemes can vary depending on the variant and its implementation. When considering the use of modified RSA schemes, it is crucial to thoroughly analyze their security, evaluate their performance implications, and ensure they meet the required standards and best practices of secure cryptography.

This section explains the two-phased process utilized to secure and protect information. The fundamental principle of our suggested approach (MRSA) is a multi-prime and subsequent stage of the MRSA encryption operations, which is a bit insertion algorithm. The key generation technique (n, e, d) and encryption mechanism make up the first step of encryption. rather than using the two prime numbers required by traditional RSA, the key preparation process uses four different prime integers (p, q, r, and s) to generate both the public as well as private keys that are used to encrypt and decrypt type of data, respectively. The result of our suggested approach (MRSA), which has been transformed into text (Ciphertext), was acquired, and it was then converted to a binary type of format to produce the ciphertext.

3.1. RSA algorithm modified.

Step 1: Choose four different significant prime integer numbers (p, q, r, and s) at random.
. # 1st Modification
Step 2: Calculate encryption and decryption, n. The n, is calculated as the multiplication of the four prime numbers: $n = p * q * r * s$
Step 3: Calculate the totient function of n. The following formula is used to compute the totient function, φ (n):
: $\varphi(n) = (p - 1) * (r - 1) * (s - 1) * (q - 1)$
Step 4: Choose an integer e based on certain criteria. Choose an integer e.
such that: $1 < e < \varphi(n)$ GCD $(e, \varphi(n)) = 1$ (e and φ(n) are co-prime) e has a tiny bit-length
Step 5: Choose a random variable f. Let f = (e * a) + b # 2nd Modification (a, b are integers and value of a, b is determined using PSO optimising technique.)
Step 6: Calculate the value of d. Calculate d such that: $(d * e) \text{ Mod } (\varphi(n)) = 1$
Step 7: The public key is (f, n). The public key is composed of the modified value f and the modulus n.
Step 8: The private key is (d, n). The private key is composed of the calculated value d and
the modulus n.
For message encryption:
The sender encrypted message M using the following technique:
Cipher text message created after encryption: C = M^f Mod (n), where C is the ciphertext.
For message decryption:
The receiver decodes the encrypted text using the following technique: The original information, M = C^d Mod (n).

3.2. Analysis of security and efficiency assessment of Modified RSA (MRSA)

a. **Compare the amount of time required by various algorithms** shown in table 1, we have contrasted the outcomes of the various methods like Key Generation Time, Decryption Time, and Encryption Time, described here. We applied 20-bit modules and 20-bit messages using python libraries. Figure 1, 2 and 3 shows the comparison between Key generation time, Decryption time Encryption Time, with respect to 20-bit messages of different algorithms.

b. **Execution time:** The effectiveness of any specific encryption and decryption approach is dependent upon the speed at which a cryptographic algorithm is implemented, and the period of execution identifies the algorithm's speed or slowness. Tables 2 show the encryption and decryption result compares of the proposed algorithms MRSA and SRSA. It is evident from the findings in Tables 2 that the computational complexity of both encryption and decryption on MRSA has a more advanced cryptosystem than SRSA, indicating that it will be more complicated, and the attackers need considerably more time to breach easily than that of the SRSA. Encryption and decryption graphical representations Figure displays the time of the proposed MRSA and SRSA algorithms when plotted against the message sizes in bytes at Figure 4 and Figure 5 correspondingly. When plotted against the message, the graphical comparison of the encryption times for the proposed algorithms MRSA and conventional RSA sizes are shown same bytes. At Figure 7 and 8, the graphical comparison Comparing suggested Algorithm MRSA and SRSA's decryption times as compared to the message sizes (measured in bytes).

c. **Computational Complexity**: Let's examine each stage of the modified key creation, message encryption, and text decryption procedure in terms of its temporal complexity.

A. Key Generation,

a. Producing the four unique randomized prime numbers that are p, q, r, and s Prime number generation is required in this stage, and probabilistic techniques like the Miller-Rabin test used to do this. Assuming that n is the key's bit length, the total amount of time complexity of creating each prime may be expressed as O (n^2 * (log n) ^3). The overall complexity of time for this phase would be around O (4 * n^2 * (log n) ^3) = O (n^2 * (log n) ^3) because we are producing four primes.
b. Since the total amount of prime variables is constant, multiplication of the four prime integers to get n = p * q * r * s may be performed in O (1) time.
c. Calculating (n) entails multiplying the variances of the prime components while deducting 1 from each. This yields the Euler's totient functional (n). We have four prime factors, thus O (4) = O (1) may be used to denote the temporal complexity of this step.
d. Calculating e: Obtaining an appropriate encryption exponent e requires meeting several requirements and looking for a co-prime with n. The GCD procedure and iterating over numbers are often used in this process. The temporal complexity is often expressed as O(n), which is a small number.
e. Finding f = (e * a) + b: This stage requires basic arithmetic operations and takes O (n_e * n_a) or O (n_e + n_b) depending on the dominant factor. Note that if 'e', 'a', and 'b' are not arrays but scalar values, the time complexity remains O (1), O (1) time to complete.

Finding an integer that meets the criteria for (e * d) Mod (n) = 1 is required to calculate the decryption exponent d. Algorithms like the enhanced Euclidean algorithm or its modular inverse may be used. Depending on the method employed, the computational complexity for this phase is generally O (log n) or O(n3).

The production of the four prime numbers (O (n^2 * (log n) ^3)) and the determination of d (O (log n) or O(n^3)) would take most of the key generation process's time.

B. Message Encryption and Message Decryption:

Both encrypting and decrypting messages need modular exponentiation, which is possible with the help of practical algorithms like the square-and-multiply technique. Modular exponentiation generally has a temporal complexity of O(k^3), where k is the modulus n's bit length. As a result, O(k^3) is a good approximation for the temporal complexity of the encryption and decryption procedures.

Algorithm for Big-O- Notation of MRSA

- Step1: randomly choose the four important prime p, q, r, and s. The technique used to test for primality often determines how time-consuming it is to generate prime numbers. The average time complexity of creating a prime number of bit length n is O(n3), supposing a probabilistic primality test like the Miller-Rabin test is used. The temporal complexity for this phase is O(n3) increased by four since we are separately producing four prime numbers, which reduces to O(n3).
- Step2: to determine the modulus, n, for encryption and decryption. This phase consists of a straightforward multiplication operation with an O (1) time complexity.
- Step 3: Calculate n's totient function in step three. Because it just requires basic arithmetic operations, computing the totient function using the above formula has an O (1) time complexity.
- Step 4: Based on certain criteria, choose an integer e. Depending on the technique used, selecting an integer e has a temporal complexity. The worst-case time complexity, if we use a random search strategy to locate an adequate e, would be O((n)). The search space for e is, however, often constrained, making the time complexity O (1).
- Step 5: Pick a random variable f in step five. This phase consists of a straightforward arithmetic operation with an O (1) time complexity.

- Step 6: Calculate the value of d in step six. A modular equation must be solved to calculate d. Using techniques like the extension of Euclidean algorithm, determining the modular inverse of e modulo n has an O(log(n)) or O(log(n)) time complexity.
- Step 7 and 8: Making public and private keys, both actions are O (1) since they both entail giving variables values.

In conclusion, the time complexity of Step 1 (creating prime numbers) and Step 6 (calculating the modular inverse) are the two steps that take up much of the key generation process' total duration. As a result, O(n^3) for prime generation and O(log(n)) for computing the modular inverse have the highest time complexity. It's crucial to remember that although encryption and decryption may be carried out several times for various communications, key creation is normally carried out only once. As a result, the key generation process would dominate the total time complexity of RSA.

The modified RSA technique has an O (n^2*(log n) ^3) time complexity for key creation, where n is the bit length of the key, and an O(k^3) time complexity for encryption and decryption, where k is the bit length of the modulus.

Table 1. Comparison table of algorithms using 20-bits modules size and 20-bits message size

S.No	**Algorithm**	**Time of key Generation (µsec)**	**Time of Encryption (µsec)**	**Time of Decryption (µsec)**
1	RSA [30]	989	989	1998
2	Prime numbers RSA [31]	988	1988	13992
4	Diffie-key-RSA [32]	999	1000	999
5	IRSA [33]	905	985	2010
6	MREA [34]	1598	3174	3896
7.	**MRSA**	**1875**	**4057**	**14976**

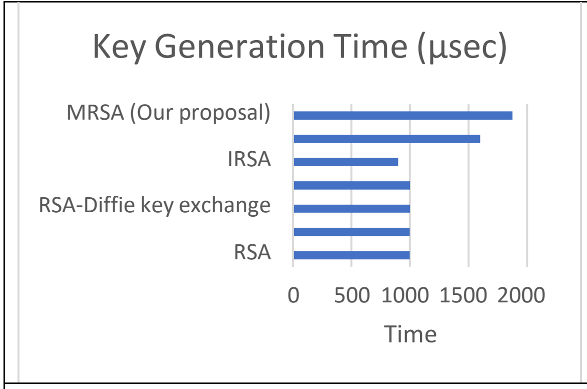


Figure 1 shows that the outcomes Key Generation Time with consedering 20-bit modules and 20-bit messages using python libraries.

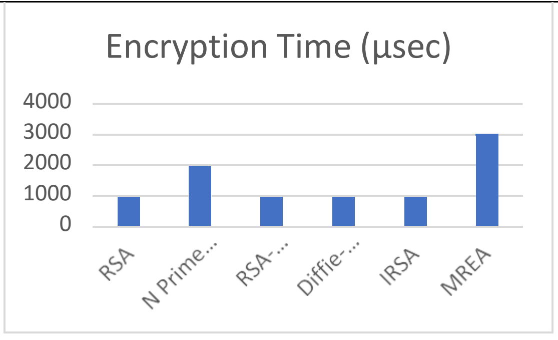


Figure 2 shows that the outcomes Encryption Time, described here with considering 20-bit modules and 20-bit messages using python libraries.

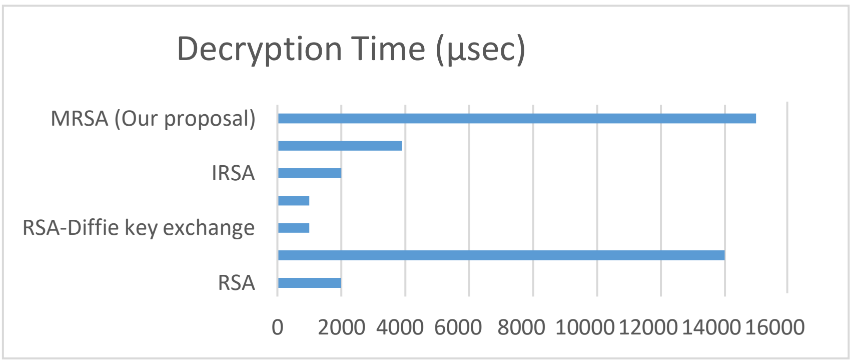


Figure 3 shows that the outcomes Decryption Time, described here with considering 20-bit modules and 20-bit messages using python libraries.

Table 2. Encryption and Decryption tine (m-Sec) of MRSA and RSA with distinct sizes(byte)

Size of Message	**Encryption time MRSA**	Encryption time SRSA	**Decryption time MRSA**	Decryption time SRSA
20	**4.35**	3.12	**41.67**	4.51
30	**3.43**	3.15	**44.00**	9.96
40	**2.95**	3.39	**27.67**	11.65
50	**3.77**	3.4	**20.16**	12.99
60	**3.65**	3.51	**18.5**	14.57
70	**3.5**	3.67	**17.67**	15.98

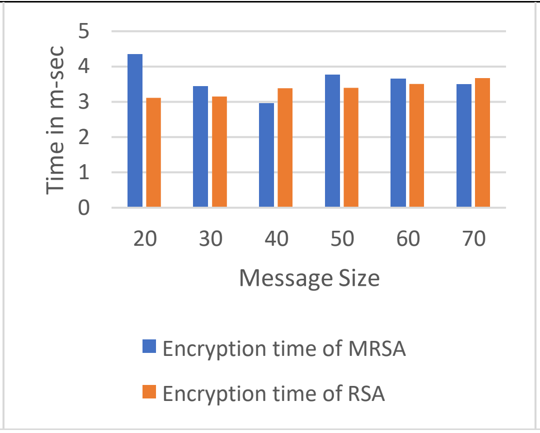


Figure 4. Plotting the proposed MRSA and SRSA algorithms' Encryption times versus message sizes in bytes.

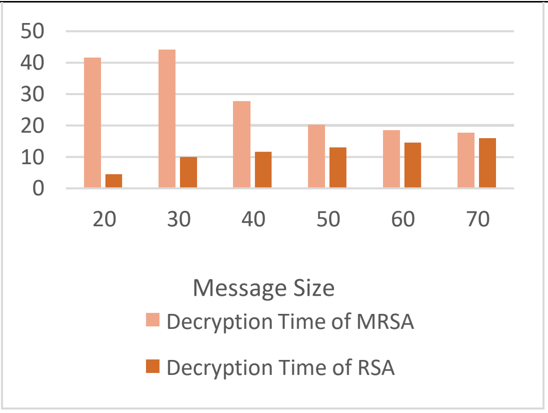


Figure 5. Plotting the proposed MRSA and SRSA algorithms' Decryption times versus message sizes in bytes

Table 3. Analysis of MRSA and Classical RSA algorithm with different parameter

Encryption Technique	**Complexity of Encryption**	**Complexity of Decryption**
MRSA	O(n^3)	O(n^3)
RSA	O(n^2)	O(n^3)
MREA [34]	O (n)	O(n^3)

MRSA, a novel security method for messages, was created in this study. The outcomes demonstrated that the computationally complex nature associated with the MRSA technique is greater compared to the algorithm used by RSA because of the time required for execution suffered, which supports the notion that always exists a compromise between protection and execution time. The difficulty indicator for the algorithm aids in assessing the efficacy of MRSA for complexity. The algorithm's sensitivity to a little modification to Plaintext. As a result, the greater the security level, the bigger the avalanche, which implies the MRSA algorithm is susceptible to even a little change in terms of Plaintext by causing obvious modifications in the form of ciphertext.

4. Optimization of MRSA Decryption and Encryption process using Parallel of computing resources technique

To address the MRSA algorithm's computational issue, it is suggested an innovative approach. As is well known, calculating MRSA on a single computer takes longer to finish. Therefore, we suggested that they calculate the MRSA algorithm in a portion of their sequence using a distributed computing approach. Our primary research focuses on the MRSA algorithm's decryption and encryption processes. This method's fundamental concept is to divide out the computing work involved in the decryption and encryption processes across several computers. The job will then be conducted by those computer employees in parallel [26]. Therefore, the speed of encryption

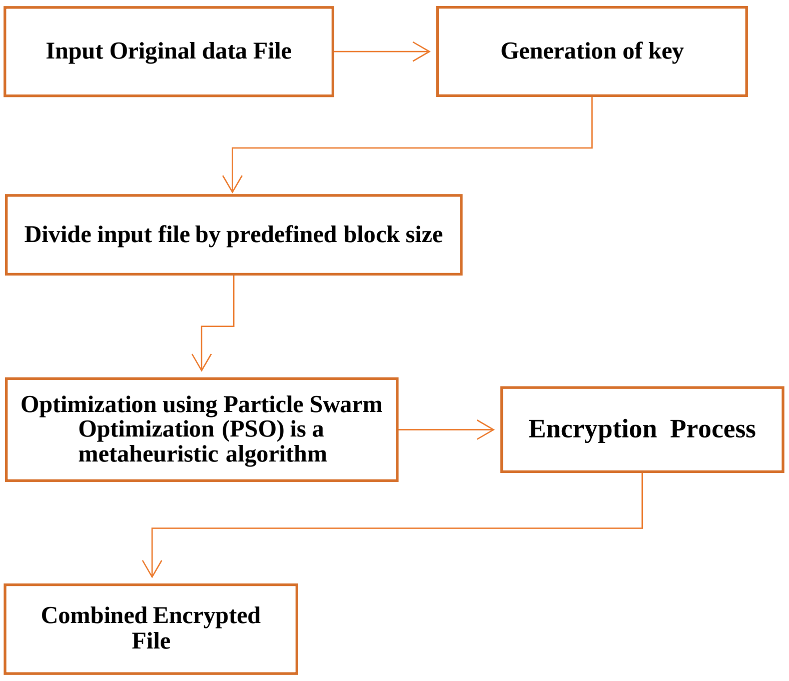

Input Original data File
Generation of key
Divide input file by predefined block size
Optimization using Particle Swarm Optimization (PSO) is a metaheuristic algorithm
Encryption Process
Combined Encrypted File

Figure 6. Block diagram of Optimization technique of MRSA encryption process using Parallel Computing

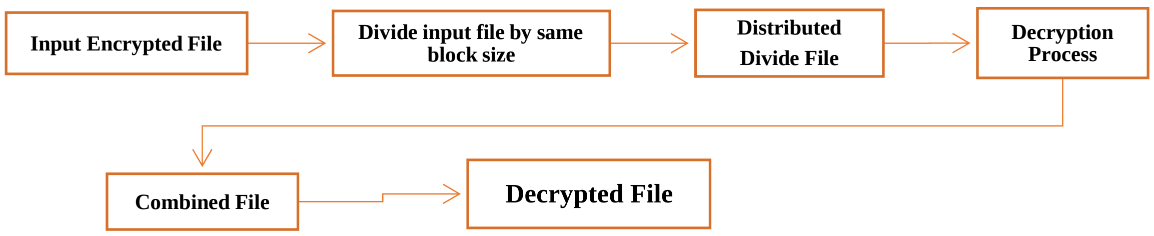


Figure 7. Block diagram of Optimization technique of MRSA Decryption process using Parallel Computing

and decryption using this approach will be quicker. The RSA technique had a computational issue that made it take longer to finish [27], hence in this study, it is developed a new way (Figure 6 and 7). It is suggested that the MRSA algorithm be computed using a distributed computing technique in parts of their series [28]. We solely work on the MRSA algorithm's decryption and encryption processes. The key concept is to divide out the computing work involved in the decryption and encryption processes across a few machines [29].

Depending on the size of the key used, the file that will be encrypted or decrypted will be divided into several parts. A single computer that we dubbed broker is responsible for managing this procedure. Following that, a group of computers (later referred to as workers) get the divided file (later referred to as pieces) and the produced key. Then, using the same randomly generated key, each worker encrypts or decrypts those parts in simultaneously. Broker should then combine encrypted or decrypted components into a finished file. On Figure 2 above, an illustration of our suggested methodology is shown.

To implement this technique, write a simple Python programmed. using Python XMLRPC. Additionally, it can change the Python Encryption Library (Crypto) to support the MRSA algorithm. Every computer utilized had this Python program running as a daemon. It is evaluated with a variety of file sizes, including 2MB, 10MB, 20MB, and 100MB. The encryption and decryption processes must be assessed on every testing file. It can get the completion time for each tested file. In the next step, compare that completion time against the completion time of a single computer technique. However, to ensured that the specifications of this single computer were equivalent to those of all other dispersed computers. The testing process log provides details on the completion times for each testing file. Each procedure's results are shown in Table 4.

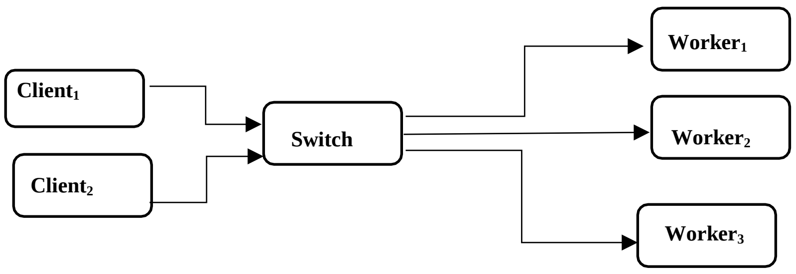


Figure 8: Testbed

When implementing parallel computing methods for MRSA, it's important to consider the trade-offs between computational speedup, communication overhead, and memory requirements. Additionally, ensure the security of private keys, protect against side-channel attacks, and adhere to cryptographic best practices to maintain the confidentiality and integrity of the MRSA system.

1. Modular Exponentiation:
   - Break down the modular exponentiation operation, which is the most computationally intensive step in MRSA, into smaller exponentiation tasks.
   - Divide the exponent into chunks and distribute the computation across multiple processing units.
   - Each processing unit independently performs the modular exponentiation for its assigned chunk.
   - Use algorithms such as the square-and-multiply algorithm or the Montgomery ladder algorithm, which are amenable to parallelization.
2. Parallel Prime Number Generation:
   - MRSA relies on prime numbers for key generation.
   - Implement parallel algorithms for prime number generation, as searching for large prime numbers can be time-consuming.
   - Techniques like sieving or primality testing can be parallelized to speed up the process.
   - Divide the range of prime numbers to be generated among multiple processing units, with each unit searching for primes independently.
   - Utilize techniques like segmented sieves or parallelized primality tests to distribute the workload effectively.
3. Task Partitioning:
   - Divide the encryption and decryption operations into smaller tasks that can be processed independently.
   - For encryption, break the message into blocks or chunks and assign each block to a separate processing unit.
   - Each processing unit independently performs modular exponentiation for its assigned block.

- For decryption, divide the ciphertext into blocks and distribute them among processing units to perform the modular exponentiation using the private key.

4. Inter-process Communication:
   - Develop efficient mechanisms for communication and synchronization between parallel processes.
   - As MRSA operations involve modular reduction, ensure that intermediate results are shared and combined correctly.
   - Utilize techniques like shared memory or message passing to exchange data between parallel processes.
   - For example, during modular exponentiation, each processing unit can compute its part of the exponentiation and then share the intermediate result with other units for further computation.
5. Load Balancing:
   - Distribute the workload evenly across the available processing units to ensure efficient utilization of resources.
   - Implement load balancing algorithms that consider the computational capabilities of each processing unit.
   - Monitor the progress and performance of each unit and dynamically adjust the workload distribution to balance the computation.
6. Parallel Key Generation:
   - MRSA key generation involves generating prime numbers and performing complex computations.
   - Utilize parallel computing to speed up the key generation process.
   - Divide the prime number generation and key computation tasks across multiple processing units to generate keys in parallel.
   - Apply parallel algorithms for prime number generation and utilize multiple units to perform the necessary computations simultaneously.
7. GPU Acceleration:
   - Graphics Processing Units (GPUs) are highly suitable for performing parallel computations.
   - Utilize GPUs to accelerate modular exponentiation and other computationally intensive operations in MRSA.
   - Implement parallel algorithms using GPU libraries such as CUDA or OpenCL to leverage the parallel processing power of GPUs.
   - Utilize techniques like data parallelism, where multiple threads or blocks of threads perform computations simultaneously.
8. Thread-level Parallelism:
   - Exploit thread-level parallelism within each processing unit.
   - Use multi-threading to execute multiple threads within a single processing unit.
   - Each thread can handle modular exponentiation for different blocks or chunks concurrently.
   - Utilize multi-core processors or multi-threading capabilities to maximize parallelism and speed up the computation.

Table 4. Result of Encryption and Decryption

File Size	Time required for Encryption process		Time required for Encryption process	
	With optimization	With out optimization	With optimization	With out optimization
2	4	9	10	44
10	14	62	77	395
20	27	135	141	801
100	138	671	873	4552

The experiment results (as per table 4) demonstrated that using the distributed computing strategy resulted in a three times quicker completion of the encryption procedure for the smallest test files. In 2 MB files, the suggested technique completes the task in 4 seconds compared to 9 seconds for the single way. Additionally, the encryption procedure is completed more quickly on bigger files. The suggested approach takes 138 seconds to process 100MB files, compared to 671 seconds for the single method. Suggested solution completed the decryption procedure 4,4 times quicker than the minimum tested files. In 2 MB files, suggested way completes the task in 10 seconds as opposed to 44 seconds for the single method. A greater file size also results in a higher completion rate for the decryption procedure. Our suggested approach takes 873 seconds to process 100MB files, compared to 4552 seconds for the single method (it is around five times quicker). In conclusion, the completion time of the encrypted data service has risen thanks to our suggested way.

Metaheuristic optimization algorithms are powerful optimization techniques that are used to solve complex problems where traditional optimization methods may not be effective. They are inspired by natural or biological processes and offer a flexible and efficient approach to finding high-quality solutions.

In summary, the choice of metaheuristic optimization algorithm depends on the specific problem characteristics and requirements. Genetic Algorithms are suitable for large search spaces and multi-modal problems. Particle Swarm Optimization excels in continuous optimization with smooth landscapes. Simulated Annealing is effective for escaping local optima and handling problems with many local optima. Each algorithm has its strengths and weaknesses, and it is recommended to experiment and compare their performance on the specific problem at hand.

Out of above algorithms suitable for this work is Particle Swarm Optimization (PSO). PSO is a metaheuristic optimization algorithm inspired by the collective behaviours of bird flocks or fish schools. It mimics the social interactions and movement patterns of particles in a search space to find optimal solutions. Here are the key components and steps involved in PSO:

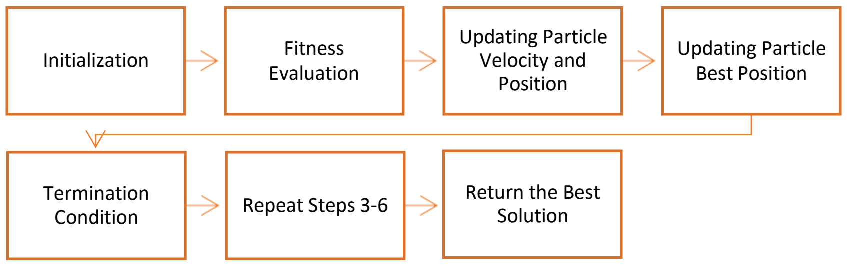
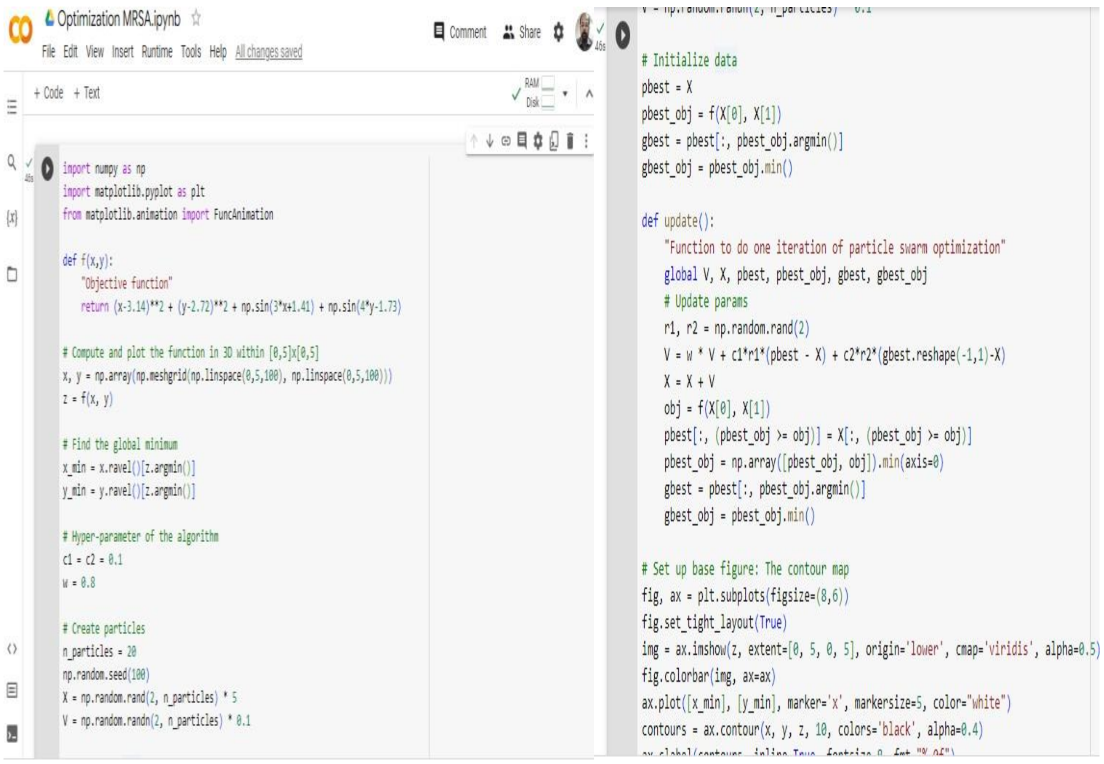


Figure 9: Block diagram for PSO optimization Technique

Process for PSO optimization:

1. Initialization:
    - Define the search space, which consists of variables and their corresponding ranges.
    - Initialize a population of particles within the search space.
    - Assign random positions and velocities to each particle.
    - Initialize the best-known position (pbest) for each particle as its initial position.
2. Fitness Evaluation:
    - Evaluate the fitness of each particle based on its position in the search space.
    - The fitness function represents the objective or cost function that needs to be minimized or maximized.
3. Updating Particle Velocity and Position:
    - Update the velocity of each particle based on its previous velocity, cognitive component, and social component.
    - The cognitive component represents the particle's memory of its best-known position (pbest).
    - The social component represents the influence of the best-known position among all particles (gbest).
    - The velocity update equation typically includes inertia, cognitive acceleration coefficient (c1), and social acceleration coefficient (c2).
    - Update the position of each particle based on its new velocity.
4. Updating Particle Best Position:
    - Update the best-known position (pbest) for each particle if its current fitness is better than its previous best-known fitness.
5. Updating Global Best Position:
    - Identify the particle with the best fitness value among all particles.
    - Update the global best-known position (gbest) with the position of the best particle found so far.

6. Termination Condition:
    - Check if the termination condition is met. This can be a maximum number of iterations, reaching a desired fitness value, or other predefined criteria.
7. Repeat Steps 3-6:
    - Repeat the velocity and position updates, as well as the updates of pbest and gbest, until the termination condition is satisfied.
8. Return the Best Solution:
    - Once the algorithm terminates, return the particle with the best fitness value (gbest) as the optimal solution.



```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

def f(x,y):
    "Objective function"
    return (x-3.14)**2 + (y-2.72)**2 + np.sin(3*x+1.41) + np.sin(4*y-1.73)

# Compute and plot the function in 3D within [0,5]x[0,5]
x, y = np.array(np.meshgrid(np.linspace(0,5,100), np.linspace(0,5,100)))
z = f(x, y)

# Find the global minimum
x_min = x.ravel()[z.argmin()]
y_min = y.ravel()[z.argmin()]

# Hyper-parameter of the algorithm
c1 = c2 = 0.1
w = 0.8

# Create particles
n_particles = 20
np.random.seed(100)
X = np.random.rand(2, n_particles) * 5
V = np.random.randn(2, n_particles) * 0.1
```

```python
# Initialize data
pbest = X
pbest_obj = f(X[0], X[1])
gbest = pbest[:, pbest_obj.argmin()]
gbest_obj = pbest_obj.min()

def update():
    "Function to do one iteration of particle swarm optimization"
    global V, X, pbest, pbest_obj, gbest, gbest_obj
    # Update params
    r1, r2 = np.random.rand(2)
    V = w * V + c1*r1*(pbest - X) + c2*r2*(gbest.reshape(-1,1)-X)
    X = X + V
    obj = f(X[0], X[1])
    pbest[:, (pbest_obj >= obj)] = X[:, (pbest_obj >= obj)]
    pbest_obj = np.array([pbest_obj, obj]).min(axis=0)
    gbest = pbest[:, pbest_obj.argmin()]
    gbest_obj = pbest_obj.min()

# Set up base figure: The contour map
fig, ax = plt.subplots(figsize=(8,6))
fig.set_tight_layout(True)
img = ax.imshow(z, extent=[0, 5, 0, 5], origin='lower', cmap='viridis', alpha=0.5)
fig.colorbar(img, ax=ax)
ax.plot([x_min], [y_min], marker='x', markersize=5, color="white")
contours = ax.contour(x, y, z, 10, colors='black', alpha=0.4)
```

Figure 10: Code for PSO Optimization technique of MRSA

PSO is characterized by its simplicity and ease of implementation. It does not require gradient information and can handle both continuous and discrete variables. Some variations of PSO, such as constriction coefficient PSO or adaptive PSO, introduce additional mechanisms to enhance convergence speed and balance exploration and exploitation.
However, PSO tends to converge the value a, b (as per indication at MRSA algorithm) to local optima and can suffer from premature convergence to a minimum timing. It is having proposed hybrid approaches, such as combining PSO with local MRSA methods. Careful parameter a, b are tuning for achieving good performance in this problem domains.
PSO has been successfully applied to MRSA function optimization. It is particularly effective in continuous optimization problems with smooth fitness landscapes and problems that require balancing exploration and exploitation.

4. PROPOSED METHOD TO SECURE INFORMATION TRANSFER

The represents the block diagram of MRSA i.e., the Modified RSA algorithm where we send an image that converts to a cipher type image and then vice versa. shown in figure 11. The proposed block diagram of the MRSA Encryption Algorithm consists of three stages: i. Generation of Key, ii. Encryption process, iii. Decryption process.

Each stage has its work. At first, we import data from the plain image, and from there we randomly generate four prime numbers. Then we generate two keys i.e., the public key as well as the private key. Then we apply the

MRSA encryption algorithm and there we use the public key to get an encrypted image. We can see that we plain image is changed into an encrypted or chipper image. From the encrypted image, we import data, and then we apply the MRSA decryption algorithm and there we use the private key to get back our original plain image. We can see that an encrypted image is converted back to a plain image.

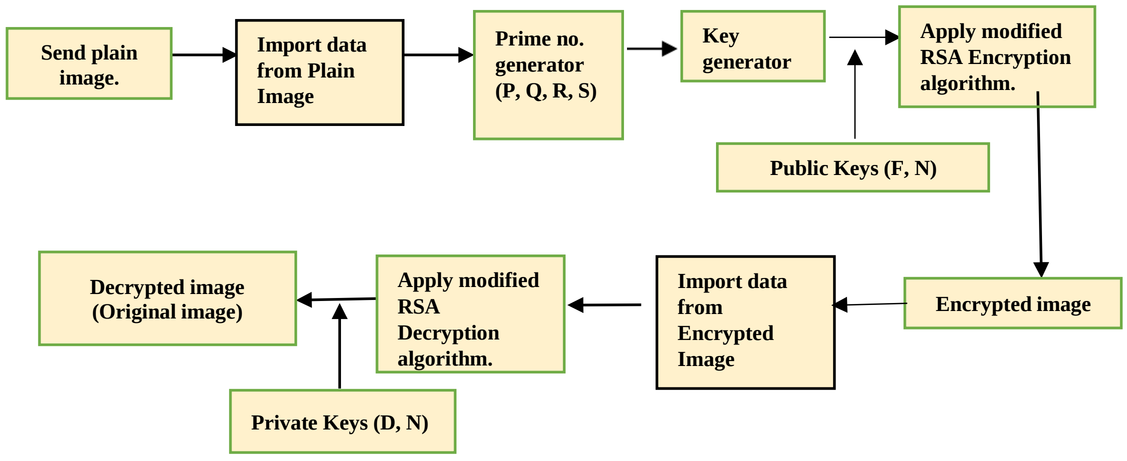

Send plain image.
Import data from Plain Image
Prime no. generator (P, Q, R, S)
Key generator
Apply modified RSA Encryption algorithm.
Public Keys (F, N)
Encrypted image
Import data from Encrypted Image
Apply modified RSA Decryption algorithm.
Decrypted image (Original image)
Private Keys (D, N)

Figure 11. Block representation of MRSA

6. CONCLUSION

Nowadays cyber-attack and stealing confidential digital data is a common crime that we face in various sectors of life. So, it is important to secure our data from the cyber attacker and unwanted users. After a deep study of the cryptosystem, we chose RSA, which is a part of the asymmetric cryptosystem, to complete our article as it is more secure and advanced. It uses public and private keys to encrypt and decrypt the data, so it is more secure to send data from one user to another via this encryption system. After choosing the encryption system, we worked on the conventional algorithm and tried our modifications to make it more secure and reliable. After completing the modification on the conventional cryptosystem, we compared both algorithms by performing a security analysis by taking various image security analysis parameters. After the result analysis, it was observed that the modified algorithm showed more satisfactory results and by comparing the various statistical data obtained from the calculations, the modified algorithm proved to be more secure and reliable as compared to the conventional algorithm.

While doing the result analysis on the running time of both algorithms, it is showing that the modified algorithm took more time to encrypt. This increase in time is because the modified algorithm is more complex than the conventional one and the computational time has been increased in the modified algorithm.

REFERENCES

[1] D.S. Babu, Y. Vijayalakshmi, Enhancement of E-commerce security through asymmetric key Algorithm, Computer Communication Elsevier, vol.153, pp. 125-134, 2020.
[2] Pandey K, Rangari KV, Shina K. An Enhanced Symmetric Key Cryptosystem Algorithm to Improve Data Security. Int J Computer Application 2013;74 (20):29–33.
[3] Osamor VC, Edosomwan I. Employing Scrambled Alpha-numeric Randomization and RSA Algorithm to Ensure Enhanced Encryption in Electronic Medical Records. Informat Med Unlock 2021; 25:1–10.
[4] Gong, Lihua, Kaide Qiu, Chengzhi Deng, and Nanrun Zhou. "An optical image compression and encryption scheme based on compressive sensing and RSA algorithm." Optics and Lasers in Engineering 121 (2019): 169-180.
[5] Joshi A, Wazid M, Goudar RH. An Efficient Cryptographic Scheme for Text Message Protection against Brute force and Cryptanalytic Attacks. Procedia Computer Sci 2015; 48:360–6.
[6] Yu H, Kim Y. New RSA Encryption Mechanism Using One Time Encryption Keys and Unpredictable Bio Signal for Wireless Communication Devices. Electron MDPI 2020;9(246):1–10.
[7] Shankar K. An Optimal RSA Encryption Algorithm for Secret Images. Int J Pure Appl Math 2018;118(2):2491–500.

[8] Bangera KN, Reddy NVS, Paddambail Y, Shivaprasad G. In: Multilayer Security using RSA Cryptography and dual audio Stenography. Information and Communication Technology (RTEICT); 2017. p. 492–5.
[9] Stergio C, Kim KE, Gupta BG. Secure an Integration of IoT and Cloud Computing. Future Gener Comput Syst 2018;78(6):964–75.
[10] Abdulshaheed HR, Binti SA, Sadiq II. Proposed Smart Solution Based on Cloud Computing and Wireless Sensing. Int J Pure Apl Math 2018, 2018,119 (18):427–49
[11] [18] Thangavel M, Varalakshmi P, Murrali M, Nithy K. An Enhanced and Secure RSA Key Generation Scheme. J Inform Security Application 2015, 2015,20(4):3–10.
[12] Rawat A, Sehgal K, Tiwari A, Sharma A, Joshi A. A Novel Accelerated Implementation of RSA Using Parallel Processing. J Discr Math Sci Cryptogr 2019;22(2):309–22.
[13] R. Abid, C. Iwendi, A. Javed, M. Rizwan and Z. Jaid ''An Optimized Homomorphic CRT-RSA Algorithm for Secure and Efficient Communication" Personal and Ubiquitous Computing Springer, 2021. http:/doi.org/10.1007/ 500779-021607-3,
[14] Kaliyamoorthy P, Ramalingam AC. QMLFD Based RSA cryptosystem for Enhancing Data security in the public cloud system. Wireless Personal Communication, Springer 2022;122:752–82.
[15] R. Shree, C. Chelvan and M. Rajesh ''An Efficient RSA Cryptosystem by Applying Cuckoo Search Optimization Techniques," Concurrency and Computation: Practice and experience. Wiley Online Library, vol. 31, no. 12. http:/doi.org/ 10.1002/cpe.4845, 2019.
[16] K. Jaspin, S. Selva, S. Sahana and G. Thamnas ''Efficient and Secured File Transfer in Cloud through Double Encryption using AES and RSA Algorithm." International of Emerging Smart Computing and Informatics. IEEE, pp. 791- 796, 2021. doi:/10.1109/ESCI50559.2021.9397005
[17] P. Hemanth, N. Raj and N. Yadva ''A Secured Message Transfer Using RSA Algorithm an Improved Playfair Cipher in Cloud Computing. ''International Conference of Convergence in Technology I2CT" IEEE, pp.931-936, 2017. Doi:/ 10.1109/I2CT.2017.8226265.
[18] S. Almanaun, M. Mahmood and M. Amin ''Ensuring the Security of Encrypted Information with Hybrid of AES and RSA Algorithm with the Third-Party Confirmation," 5th International Conference of Intelligent Computing and Control System, IEEE, pp.337-343, 2021. Doi:/10.1109/ ICICCS51141.2021.9432174.
[19] . Chung-Hsien Wu, Jin-Hua Hong and Cheng-Wen Wu, "RSA cryptosystem design based on the Chinese remainder theorem," Proceedings of the ASP-DAC 2001. Asia and South Pacific Design Automation Conference 2001 (Cat. No.01EX455), Yokohama, Japan, 2001, pp. 391-395, Doi: 10.1109/ASPDAC.2001.913338.
[20] To cite this article: M Ghazali Kamardan et al Modified Multi Prime RSA Cryptosystem 2018 J. Phys.: Conf. Ser. 995 01203
[21] Ueno, Rei & Homma, Naofumi. (2023). How Secure is Exponent-blinded RSA–CRT with Sliding Window Exponentiation? IACR Transactions on Cryptographic Hardware and Embedded Systems. 2023. 241-269. 10.46586/tches. v2023.i2.241-269.
[22] Takayasu, A., Kunihiro, N. (2014). Cryptanalysis of RSA with Multiple Small Secret Exponents. In: Susilo, W., Mu, Y. (eds) Information Security and Privacy. ACISP 2014. Lecture Notes in Computer Science, vol 8544. Springer, Cham. https://doi.org/10.1007/978-3-319-08344-5_12
[23] Venkatalakshmi, K & Gayathri, Pamuru & Likhitha, Tadakaluru & Shinde, Sushmitha & Kumar, MOV. (2022). Design of Montgomery Multiplier – RSA Algorithm. Journal of Physics: Conference Series. 2325. 012022. 10.1088/1742-6596/2325/1/012022.
[24] Kiratsata, Harsh J. and Panchal, Mahesh, A Novel Homomorphic Encryption based RSA Algorithm for Machine Learning (May 25, 2021). Proceedings of the International Conference on Smart Data Intelligence (ICSMDI 2021), Available at SSRN: https://ssrn.com/abstract=3852596 or http://dx.doi.org/10.2139/ssrn.3852596
[25] Cherkaoui-Semmouni, M., Nitaj, A., Susilo, W., Tonien, J.: Cryptanalysis of RSA variants with primes sharing most significant bits. In: Liu, J.K., Katsikas, S., Meng, W., Susilo, W., Intan, R. (eds.) ISC 2021. LNCS, vol. 13118, pp. 42–53. Springer, Cham (2021).
[26] Diaz J, Muñoz-Caro C and Niño A 2012 A urvey of parallel programming models and tools in the multi and any-core era IEEE Transactions on Parallel and Distributed Systems
[27] Amalarethinam I G and Leena H M 2017 Enhanced RSA Algorithm with Varying Key Sizes for Data Security in Cloud Proceedings - 2nd World Congress on Computing and Communication Technologies, WCCCT 2017
[28] Khanezaei N and Hanapi Z M 2014 A framework based on RSA and AES encryption algorithms for cloud computing services Proceedings - 2014 IEEE Conference on System, Process and Control, ICSPC 2014
[29] Jadeja Y and Modi K 2012 Cloud computing - Concepts, architecture and challenges 2012 International Conference on Computing, Electronics and Electrical Technologies, ICCEET 2012
[30] B. R. Ambedkar, S.S. Bedi, 'A New Factorization Method to Factorize RSA Public Key Encryption', International Journal of Computer Science Issues, Vol. 8, Issue 6, No 1, Nov 2011.
[31] B.P. Urbana Ivy, P. Mandiwa, M. Kumar, 'A Modified RSA Cryptosystem based on 'n' Prime Number', International Journal of Engineering and Computer Science ISSN: 2319-7242 Vol. 1 Issue 2 Nov 2012 pp. 63-66.
[32] S. Gupta and J. Sharma, 'A Hybrid Encryption Algorithm based on RSA and Diffie- Hellman', 2012 IEEE International Conference on Computational Intelligence and Computing Research.
[33] A. Bhattacharjee, C. Khaskel, D. Basu, D. R. Vincent P.M., 'Hybrid Security Approach by Combining Diffie Hellman and RSA Algorithm', International Journal of Pharmacy and Technology Dec. 2016 Vol. 8 Issue No. 4 pp. 26560-26567.
[34] R. S. Dhakar A. K. Gupta and P. Sharma, 'Modified RSA encryption algorithm (MREA)', 2012 2nd International Conference on Advanced Computing and Communication Technologies IEEE. pp. 426-429

[35] M Ganavi, S Prabhudeva, “A Secure Data Transmission using Modified RSA and Random Pixel Replacement Steganography”, 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), IEEE Xplore, Coimbatore, India, November 29, 2018.

[36] Faten H. Mohammed Sediq Al-Kadei, Huda Abdalkaream Mardan, Nevart A. Minas, “Speed Up Image Encryption by Using RSA Algorithm”, 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE Xplore, Coimbatore, India, April 23, 2020.

[37] Sudipto Majumder, Md. Mahfuzur Rahman, “Implementation of security-enhanced image steganography with the incorporation of modified RSA algorithm", 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), IEEE Xplore, April 4, 2019.

[38] Oluwakemi Christiana Abikoye, Haruna Ahmed Dokoro, Abdullahi Abubakar, Akande Noah Oluwatobi, "Modified Advanced Encryption Standard Algorithm for Information Security" ResearchGate, December 2019.

[39] Sarjiyus, O., “Enhancing RSA Security Capability Using Public Key Modification”, International Journal of Research and Scientific Innovation (IJRSI) | Volume VII, Issue IX, ISSN 2321–2705, September 2020

[40] M Ganavi, S Prabhudeva, “A Secure Data Transmission using Modified RSA and Random Pixel Replacement Steganography”, 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), IEEE Xplore, Coimbatore, India, November 29, 2018.

[41] Faten H. Mohammed Sediq Al-Kadei, Huda Abdalkaream Mardan, Nevart A. Minas, “Speed Up Image Encryption by Using RSA Algorithm”, 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE Xplore, Coimbatore, India, April 23, 2020.

[42] Sudipto Majumder, Md. Mahfuzur Rahman, "Implementation of security-enhanced image steganography with the incorporation of modified RSA algorithm", 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), IEEE Xplore, April 4, 2019.

[43] R.L. Rivest, A. Shamir, and L. Adleman, ‘A Method for Obtaining Digital signatures and Public-Key Cryptosystems,’ Communications of the ACM, Feb. 1978 vol. 21(2) pp. 120- 126’.