Received: February 28, 2024. Revised: April 26, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Dynamic Approach for Time Reduction in RSA Algorithm through Adaptive Data Encryption and Decryption

Pradeep Krishnadoss[1]* **Palani Thanaraj Krishnan[1]** **Nishanth Paramasivam[1]**
Deepesh Sai Kesavan[1] **Anish Thishyaa Raagav[1]**

[1]*Vellore Institute of Technology, Chennai, India*
* Corresponding author's Email: pradeep.k@vit.ac.in

Abstract: With increasing data transmission and the escalating risks of data theft, the demand for fast and robust encryption algorithms has intensified. The Rivest Shamir Adleman (RSA) algorithm is a widely used asymmetric cryptographic solution for secure data communication. However, the computational resources required for encryption and decryption processes can become significant, particularly as the volume of data increases. This research paper proposes a dynamic approach to enhance the RSA algorithm tailored explicitly for large-scale data encryption and decryption. The proposed method involves storing the encrypted value of repeated elements, aiming to optimize performance. To evaluate the efficacy of this approach, extensive experiments were conducted on various datasets comprising paragraphs of different sizes. The experiments involved encrypting and decrypting paragraphs containing 10, 50, 100, 200, 500, 1000, and 10,000 words. Comparative analyses were performed against existing RSA, El Gamal, and AES algorithms. Results indicate that the proposed dynamic RSA approach consistently outperforms traditional RSA in terms of encryption time, decryption time, and total execution time across all tested paragraph sizes. The regression analysis revealed stark differences between the two approaches. The regression line for traditional RSA exhibited a significantly steeper slope 1.75 and a substantially higher intercept 1001.42 compared to the proposed dynamic RSA approach, which demonstrated a lower slope 0.0113 and a much smaller intercept 28.58. These findings underscore the superior scalability and efficiency of the proposed dynamic RSA approach over traditional RSA, particularly in handling larger volumes of data while maintaining lower computational overheads.

Keywords: Cryptography, Key generation, Encryption, Decryption, Public key, Private key, RSA, Dynamic approach, Asymmetric algorithms.

1. Introduction

Researchers and scientists have faced critical challenges to ensure robust data security in the current digital landscape [1]. The demand for security measures has increased due to the increase in penetration attacks in which unauthorized users coordinate during data communication. The exponential growths of internet usage and alarming surge in cyberattacks have further complicated the security landscape, posing numerous issues in the vast web domain. Hence, Encryption and Decryption play crucial roles in our everyday lives, ensuring secure communication and transactions [2]. A readable message is transformed into an unreadable one, called the ciphertext message, through encryption, and decryption is the process of converting a ciphertext message to its original form. When information is being encrypted and decrypted, mathematical functions or procedures known as cryptographic algorithms scramble the plain text and render it unreadable to prevent unauthorized access [3]. To control the encryption and decryption processes, cryptographic algorithms use a piece of information called the cryptographic key [4, 5]. It determines how the plaintext (original message) is transformed into ciphertext (encrypted message) and vice versa. Digital signatures, authentication, and data encryption all require cryptographic techniques. Based on the kinds of keys and encryption techniques, cryptography is divided into two categories: i)

Symmetric Key Cryptography (Secret key), ii) Asymmetric Key Cryptography (Public key)

The symmetric algorithm, also known as the secret-key algorithm, uses a shared key for both data encryption and Decryption. These are widely used algorithms for data encryption and Decryption, ensuring the confidentiality and integrity of sensitive information [6]. The strength of a symmetric algorithm is based on different factors such as key size, block size, number of rounds and resistance to attack. Even though symmetric algorithms are efficient, there are a few limitations, like key distribution challenges and a need for built-in features like authentication and forward security, which may require the asymmetric algorithm to meet specific security requirements.

Asymmetric key encryption, also known as public key encryption, uses the concept of key encryption. Unlike symmetric encryption, it uses a pair of keys [7]. Encryption uses the public key, and Decryption uses the private key where both are mathematically related. The asymmetric encryption process is briefly categorized into three processes:

(i) Key generation: receiver generates the pair of keys and shares the public key with others while keeping the private key secret.
(ii) Encryption: The person sending the message to the receiver uses the public key and encrypts the data to an unreadable form
(iii) Decryption: receiver enters the private key and decrypts the data converting it to its original readable form

The main advantage of the usage of asymmetric encryption is its use of two keys and separate keys for encryption and Decryption. Even if someone gets hold of the public key, they cannot decrypt the data since a private key is needed.

RSA(Rivest-Shamir-Adleman) algorithm is a widely used asymmetric encryption technique in modern data security systems. Based on the principle of key encryption, RSA uses a pair of keys, namely public and private keys, for the encryption and decryption process. This algorithm has gained significant prominence due to its robustness in ensuring secure data transmission [8]. The RSA algorithm works by utilizing the mathematical property of large prime numbers and modular arithmetic. During the encryption process, the sender utilizes the public key to convert the message into ciper text format that can only be decrypted using the corresponding private key held by the recipient. Although RSA has many benefits, it is important to consider its limitation, especially in terms of computational speed when handling large amounts of data. Increasing data size in RSA algorithm leads to significant computational complexity, causing resource-intensive and time-consuming encryption and decryption processes. This poses challenges for real-time efficient data processing.

Finding new and efficient approaches is a never-ending task in the world of academic research. One such approach that has gained significant attraction in recent years is the dynamic approach. Repeating complex problems are challenges that persistently arise in various domains [9]. Conventional problem-solving methods need help to provide effective solutions due to the dynamic nature of these problems. The dynamic approach follows two ways: (i) Overlapping subproblem - We store the solution to the overlapping subproblem (ii) Optimal Substructure - If the solution for the subproblem is optimal, then the solution of the original problem is also optimal. By adopting the dynamic approach, researchers make effective strategies to address complex problems.

In this paper, we have proposed a dynamic approach on the RSA algorithm that involves storing each character's encrypted and decrypted values for potential reuse to enhance the efficiency of the encryption process. This approach is highly advantageous when the data has a repetitive nature, where the same character combination frequently appears in different contexts. By utilizing the repository of encrypted and decrypted values, the encryption process can be done quickly by retrieving and using the pre-computed values, reducing the computation overhead and increasing the speed of encryption and decryption compared to the original RSA algorithm that does the same computation again and again for the repetition of the same character. The reduction in the overall execution time can improve the usage of RSA for large data. Experiments have been conducted using local laptop (i7-11th gen) on paragraphs of varying length to prove the reduction in overall execution time and the same has been presented Section 5. Additionally, this dynamic approach does not compromise data security, as the stored values are securely managed and can only be accessed by authorized parties.

The paper has been organized as follows: Section 2 contains an overview of the recent literatures related to RSA and hybrid RSA. The proposed efficient dynamic RSA is described in Section 3. Section 4 lists out the various parameters used for comparing the algorithms. The obtained results and the respective discussions ore done in Section 5. Conclusion is present in Section 6.

2. Related work

Over the years, researchers have made significant efforts to enhance the performance of the RSA algorithm, primarily focusing on reducing the computational time required for encryption and decryption operations. Several techniques have been proposed and explored to achieve this objective, aiming to strike a balance between security and efficiency.

To increase security, an enhanced key expansion technique for AES is presented. It tackles the issue of varying operation times in AES procedures by making AES operations simpler and assuring consistent computational resource utilisation [10]. Using four prime numbers, the Chinese remainder theorem and Montgomery modular multiplication increase the effectiveness of RSA operations. These improvements enable a hybrid encryption system that combines RSA and AES and offers easy key management and speedy encryption and decoding. Experimental findings demonstrate that the optimised algorithm is quick and practicable for real-world use. Our proposed methodology ensure the security of RSA along with the reduction in computational time but the compromise security for computation time.

New hybrid cryptosystems like Twofish, AES, RSA, and ElGamal that combine symmetric and asymmetric algorithms have been proposed. The primary goals are to raise performance, increase speed, evaluate efficiency, and evaluate against other algorithms [11]. Two hybrid algorithms-AES+RSA and Twofish+RSA are presented. Their security and efficacy are assessed, and the findings are used to emphasise the advantages of each hybrid algorithm. While Twofish+RSA has advantages in terms of calculation speed, cypher text size, and memory consumption, AES+RSA is determined to be much safer. Hybrid cryptosystems, in particular, are new methods to cryptography. As in the hybrid techniques either time or security is being compromised but in our proposed methodology the security of traditional RSA is maintained along with improved in computation time.

Cloud computing is used to store private data and access shared resources. It highlights the significance of strong encryption and access control to maintain privacy and security [12]. The computational and storage costs of encryption techniques, including RSA, KP-ABE, CP-ABE, and AES, are compared. Multi-threading is used on contemporary CPUs to increase RSA encryption’s speed to boost effectiveness. Information about cloud data security and advancements that will speed up and improve the effectiveness of encryption processes are proposed. Instead of multi-threading we can use dynamic method to increase the encryption speed which may reduce the computation cost.

The method for enhancing the prime number generation process which is essential for cryptographic systems is presented in the study [13]. The authors speed up primality testing for 1024-bit RSA key pairs by around 30% by skipping the initial step of looking for minor prime factors. With shared RSA keys being used in decentralised systems, this is especially advantageous because it conserves computational resources and lessens communication complexity. The study also introduces a more basic mathematical function that can be applied to the design of new public-key encryption schemes and random number generators. As the strength of RSA relies on prime number, on choosing minor prime number the security is comprised as compared to our proposed methodology.

The issue of optimizing the performance of RSA encryption while ensuring information security is a critical concern addressed in this research paper [14]. The authors propose a modification to the RSA algorithm by integrating the Euclidean technique. By leveraging the Euclidean algorithm, the proposed method aims to enhance the speed, throughput, power consumption, and avalanche effect of RSA encryption. The experimental results and mathematical justification provided support the efficacy of the proposed algorithm, demonstrating its potential to improve data security in certain applications. The mentioned paper primarily focuses on optimizing key generation within the RSA encryption process. However, our methodology goes further by optimizing both encryption and decryption times.

Communication security is crucial due to the widespread use of hidden contact and communication channels. The security of the data kept on their servers is a problem for cloud service providers who gather and retain uploaded data [15]. To address the concern that cloud providers might gain access to customer data, A robust encryption method that improves the key-generating process was proposed to increase security. In terms of response time, the new technique outperforms the well-known RSA algorithm. The encryption and decryption procedures in the research use ASCII and EBCDIC codes. Our proposed method employs dynamic programming techniques, resulting in reduced computation time with less overhead compared to the method mentioned above [16]. The RSA algorithm, which has extended key sizes, a high memory requirement, and a poor execution speed, was the subject of the research. The execution times of the encryption and decryption operations were measured in three trials. The researchers' programming approaches accelerated the encryption and decryption processes by 14% and 22%, respectively. These results demonstrate the possibility of RSA algorithm optimisation to deliver faster and more effective encryption and decryption, concluding that the key length is proportional to the time spent in encryption and decryption. The above paper suggest that strength of RSA depends on key size and computation time is issue in RSA which can be resolved using our proposed methodology.

The conventional RSA encryption algorithm is sluggish because it depends on complicated processes and long key lengths. In order to increase the efficiency and security of RSA encryption, a new algorithm dubbed PMKRSA was created in this work. The PMKRSA technique divides the plaintext into pieces and simultaneously encrypts each with a different key pair [17]. This parallelisation makes faster implementation possible, and the technique was also made GPU-friendly. According to the results, the CPU+GPU version's average encryption and decryption times were substantially faster than the CPU version for files ranging from 1 MB to 100 MB. Comparatively, the above mentioned technique might introduce added complexity in handling multiple key pairs, potentially resulting in increased overhead for key management and storage, in contrast to our proposed methodology.

A hybrid encryption method was developed that combines the RSA and DH algorithms to improve data security while it is being transmitted [18]. To speed up encryption execution, decrease the time required to create images and keys, and boost overall security against hacking; prime numbers are created from random images and used in XOR operations for encryption. The study highlights the possibilities for secure data communication by showing how effective this strategy is compared to other approaches. The drawback of the aforementioned methodology is that it is more sophisticated than our suggested methodology because it makes use of image processing techniques, which may require additional computing resources.

While hybrid algorithm can offer certain advantages in terms of reducing execution time, they also come with their fair share of disadvantage i.e. Increased complexity, higher resources requirement, Compatibility issues. So it is better to use the existing asymmetric algorithm to execute with high computational speed. By improving the asymmetric algorithm the security and computation speed is improved at the same time.

3. Proposed methodology

In the realm of cybersecurity, fast encryption algorithms hold immense significance. With the increase in data breaches and cyber threats, the timeliness of encryption becomes crucial. Efficient encryption techniques ensure that sensitive information is securely protected, minimizing exposure to potential attacks. Hence, fast encryption is a vital cornerstone of modern cybersecurity, safeguarding sensitive data with efficiency and speed.

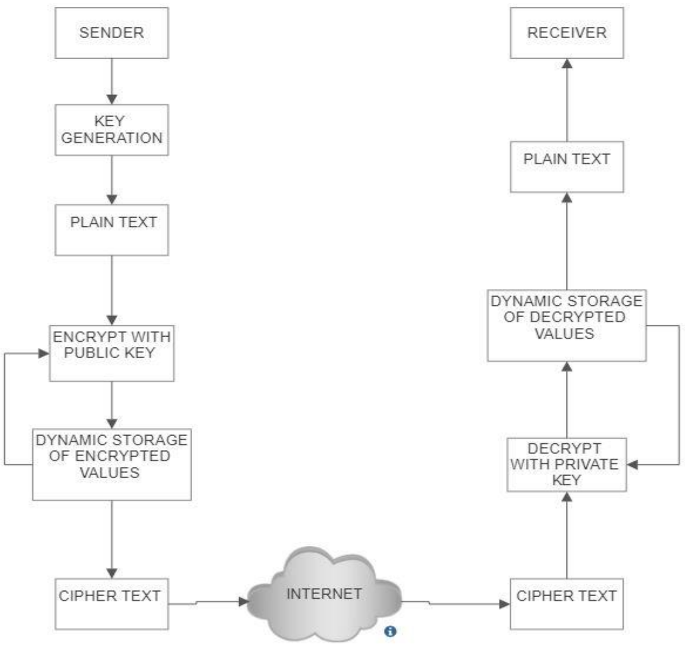

Figure. 1 Block diagram of our proposed RSA algorithm

Our research paper focuses on encrypting and decrypting data using a dynamic approach strategy that uses the Rivest-Shamir-Adleman (RSA) algorithm for the encryption and decryption process.Our suggested method, known as “Dynamic Approach for Time Reduction in RSA Algorithm through Adaptive Data Encryption and Decryption”, uses a dynamic approach in which computed encryption and decryption values of each different character is stored and used when there is a reputation of character in the given data which enhance the efficiency of the RSA encryption algorithm. The process of key generation by generating two random large prime numbers and encryption of the plain text using the primary key, and decryption of cipher text using the private key all are done using the RSA algorithm, which reduces the computation overhead the dynamic approach of storing the different character is used. Fig. 1 shows the different stages of proposed RSA algorithm.

3.1 Proposed RSA algorithm

1. Select two large prime numbers p and q.
2. Compute modules
$$n = p * q$$
3. Compute
$$\varphi(n) = (p-1) * (q-1)$$
4. Compute public key by selecting e which is range of (1, φ(n)) and relative prime of φ(n)
5. Compute private key d that satisfy
$$ed = 1\, mod\, (p - 1)\, (q - 1)$$
6. Let the messages be
$$1 = ASCII('a'),$$
$$M2 = ASCII('b')$$
$$\dots\dots$$
$$\dots\dots.$$
$$M26 = ASCII('z')$$
7. Encrypt the message
$$C1 = M1\, e\, mod\, n$$
$$C2 = M2\, e\, mod\, n$$
$$\dots$$
$$\dots$$
$$C26 = M26\, e\, mod\, n$$
8. Each time a new letter is encountered, save its encrypted value in a dictionary. If a letter is encountered again, use the stored encrypted value instead of recalculating it.
9. Decrypt the encrypted message
$$M1 = C1\, d\, mod\, n,$$
$$M2 = C2\, d\, mod\, n$$
$$\dots$$
$$\dots$$
$$M26 = C26\, d\, mod\, n$$
10. Each time a new decrypted value is encountered, save its decrypted value along with message in an dictionary. If a same decrypted value is encounter again, use the stored decrypted value instead of recalculating it.

Sieve of Eratosthenes algorithm is used to efficiently find all the prime numbers up to a given limit. The function encrypt() is used to encrypt a given message using RSA algorithm. The function takes a message as input that needs to be encrypted. It uses the recipient’s public key, specifically the modulus n and the public exponent e, for encryption. The function converts each character of the message into its corresponding ASCII value. It applies modular exponentiation to each ASCII value using the public key components n and e. This process involves raising the ASCII value to the power of e and then taking the modulus n of the result. The resulting encrypted values represent the ciphertext, which is the encrypted form of the original message. The function returns the ciphertext as the output.

```
function encrypt(message):
   e = public_key
   encrypted_text = 1
   while e > 0:
      encrypted_text *=
message
      encrypted_text %= n
      e -= 1
   return encrypted_text
```

Code block 2: Encrypt

The function encoder() is used to convert the plaintext message into an representation suitable for encryption, and it also applies the dynamic approach to check if the character is previously encrypted or not. The function takes a message as input, typically in the form of a string. It iterates over each character of the message. For each character, the function performs a lookup to check if it has been previously encoded and stored in a dictionary (‘enq’). If so, it retrieves the previously assigned encoded value for that character. If the character has not been previously encoded, the function encrypts the ASCII value of the character using the ‘encrypt()’ function. The function records the encoded value for the character in the dictionary for future reference. The encoded values for all the characters are collected and returned as the output.

```
function encoder(message):
    encoded = []
    for letter in message:
        if letter in enq.keys():
            encoded.append(enq[letter])
        else:
            eq = encrypt(ord(letter))
            encoded.append(eq)
            enq[letter] = eq
    return encoded
```

Code block 3: Encoder

The function decrypt() is used to decrypt an encrypted message using the RSA algorithm. The function takes the encrypted ciphertext as input. It uses the recipient's private key, specifically the modulus n and the private exponent d, for decryption. The function applies modular exponentiation to the ciphertext using the private key components n and d. This process involves raising the ciphertext to the power of d and then taking the modulus n of the result. The resulting decrypted values represent the original ASCII values of the characters in the plaintext message. The function converts each decrypted ASCII value back into its corresponding character. The decrypted characters are concatenated to reconstruct the original plaintext message. The function returns the plaintext message as the output.

```
function
decrypt(encrypted_text):
     d = private_key
    decrypted = 1
    while d > 0:
        decrypted *= encrypted_text
        decrypted %= n
        d -= 1
    return decrypted
```

Code block 4: Decrypt

The function decoder() is used to convert the encrypted value into an representation suitable for decryption and it also apply the dynamic approach to check if the character is previously decrypted or not. The function takes the decrypted numerical values as input. It iterates over each numerical value. For each value, the function performs a lookup to check if it has been previously decoded and stored in a dictionary. If so, it retrieves the previously assigned decoded character for that value. If the value has not been previously decoded, the function converts it into its corresponding ASCII character. The function records the decoded character for the value in the dictionary for future reference. The decoded characters are concatenated to reconstruct the original plaintext message. The function returns the plaintext message as the output.

```
function decoder(encoded):
    s = ‘ ‘
    for num in encoded:
        if num in deq.keys():
            s += deq[num]
        else:
            q = chr(decrypt(num))
            s += q
            deq[num] = q
    return s
```

Code block 5: decoder

4. Analysis and results

We have implemented our approach of dynamically improving the RSA algorithm with the help of Python programming language. We have compared our algorithm with the original RSA and El Gamal encryption algorithm. Fig. 2 shows Flow chart of our proposed RSA algorithm. We have considered the following parameters for comparison of the algorithms:

4.1 Execution time

Execution time is the algorithm's overall time to complete the execution and the processes. Execution time has two times- encryption and decryption time. Encryption time is the overall time the code takes to convert the original message into the cipher text. Decryption time is the time the algorithm takes to convert the cypher text back into the message, .i.e, in a readable format. We have compared our improved rsa algorithm with the El Gamal encryption algorithm based on execution time. Our improved rsa algorithm is also compared based on encryption and decryption times with the original rsa algorithm. We have also compared a symmetric algorithm AES based on execution time.

4.2 Energy

By using the dynamic approach for improving our algorithm, we have made more efficient use of algorithms and data structures by minimising unnecessary computations and memory operations. We also reduce the redundant calculations and optimise the memory access, reducing the energy

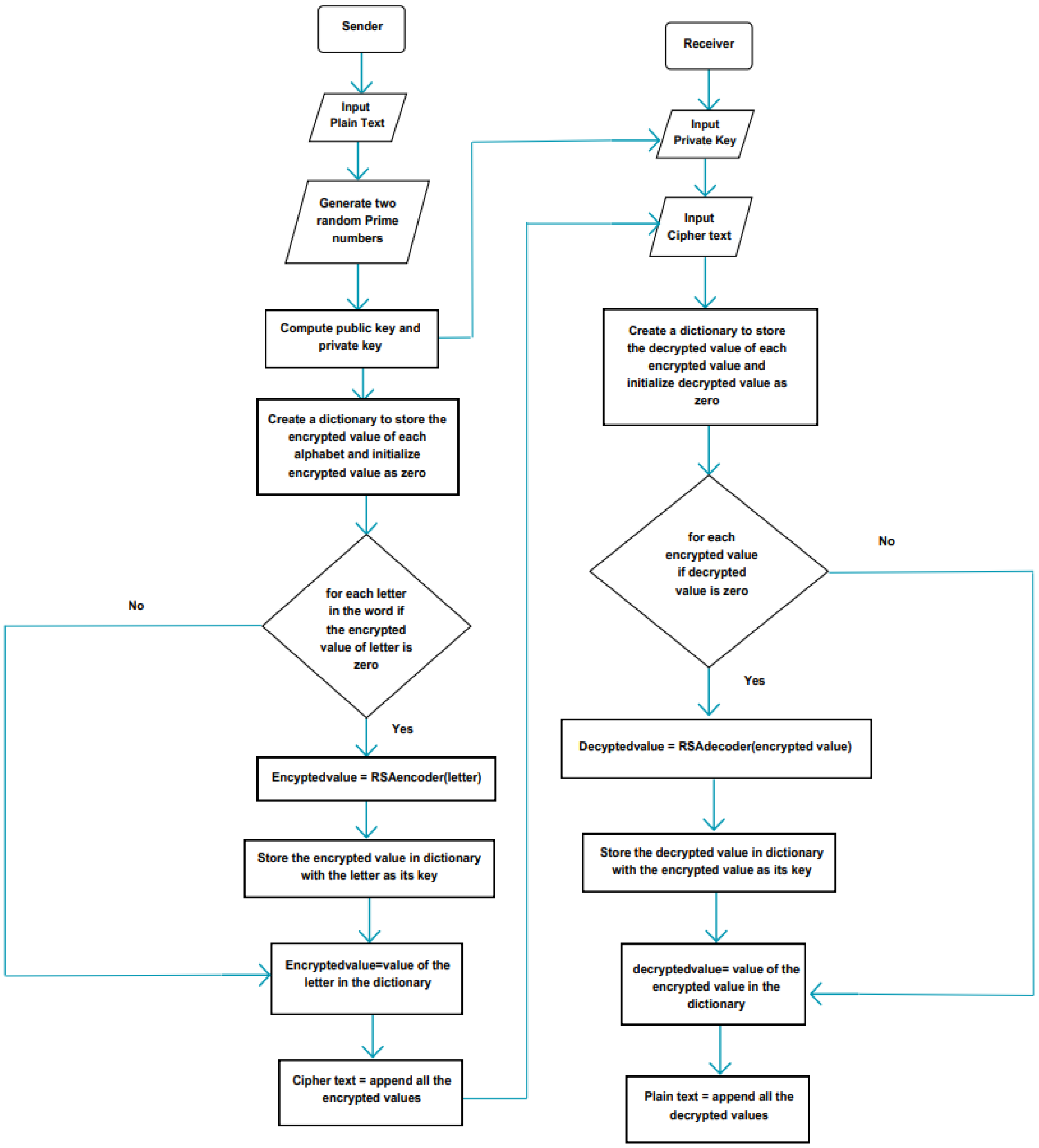
Sender
Input Plain Text
Generate two random Prime numbers
Compute public key and private key
Create a dictionary to store the encrypted value of each alphabet and initialize encrypted value as zero
for each letter in the word if the encrypted value of letter is zero
No
Yes
Encyptedvalue = RSAencoder(letter)
Store the encrypted value in dictionary with the letter as its key
Encryptedvalue=value of the letter in the dictionary
Cipher text = append all the encrypted values
Receiver
Input Private Key
Input Cipher text
Create a dictionary to store the decrypted value of each encrypted value and initialize decrypted value as zero
for each encrypted value if decrypted value is zero
No
Yes
Decyptedvalue = RSAdecoder(encrypted value)
Store the decrypted value in dictionary with the encrypted value as its key
decryptedvalue= value of the encrypted value in the dictionary
Plain text = append all the decrypted values

Figure. 2 Flow chart of our proposed RSA algorithm

required to perform the operations. This helps in reducing the energy consumption of the algorithm without compromising the functionality of the algorithm.

4.3 Cost

Efficient usage of the resources such as memory and network bandwidth by optimising the usage of the resources, we can reduce the overall cost. Caching and memorisation technique has been used, i.e. storing already computed values to avoid re-computations. we have used a dictionary so that the original word and its computed value can be stored in the same place without using different data structures. Without reducing the correctness of the algorithm, we have implemented resource utilisation. Hence the overall cost can be reduced.

5. Result and discussion

We have varied the message size (in words) and observed its impact on both encryption and decryption time (in ms) in the case of the proposed

Table 1. Observed Encryption time of Proposed RSA and Classic RSA

ENCRYPTION TIME (s)		
No of words	Proposed RSA	RSA
10	0.0027	0.00429
50	0.0048	0.00309
100	0.00275	0.0033
200	0.0051	0.0056
500	0.00471	0.00728571
1000	0.0048	0.0282
10000	0.031	0.0737

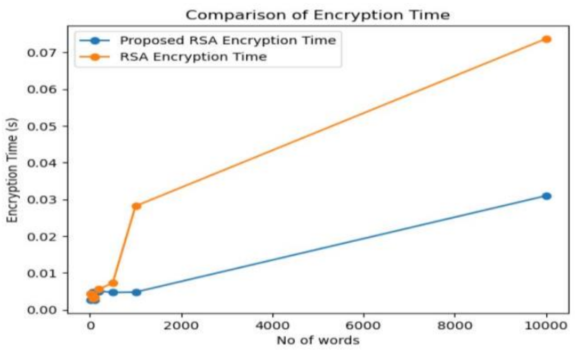

Figure. 3 Comparison between the Encryption times of Proposed RSA and Classic RSA

Table 2. Observed Decryption time of Proposed RSA and Classic RSA

DECRYPTION TIME (s)		
No of words	Proposed RSA	RSA
10	0.0168	0.07355
50	0.01429	0.4678
100	0.02951	0.609
200	0.0279	1.72541
500	0.034	2.503571
1000	0.0338	4.0054
10000	0.109	18.28

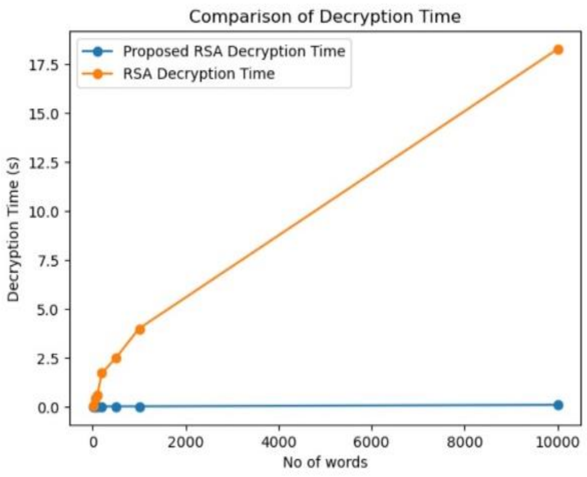

Figure. 4 Comparison between the Decryption times of Proposed RSA and Classic RSA

Table 3. Observed Execution time of Proposed RSA, Classic RSA, El Gamal and AES Algorithms

TOTAL EXECUTION TIME(ms)				
No of words	PROPOSED RSA	RSA	EL Gamal	AES
10	20.8	70.6	14.9	1.19
50	20.9	473	48.71	16.5
100	35.96	614.51	86.2	36
200	36	1734	192.9	42.95
500	39	2513	601.8	56
1000	40.2	4017.4	899.2	72
10000	141	18360	8901	671

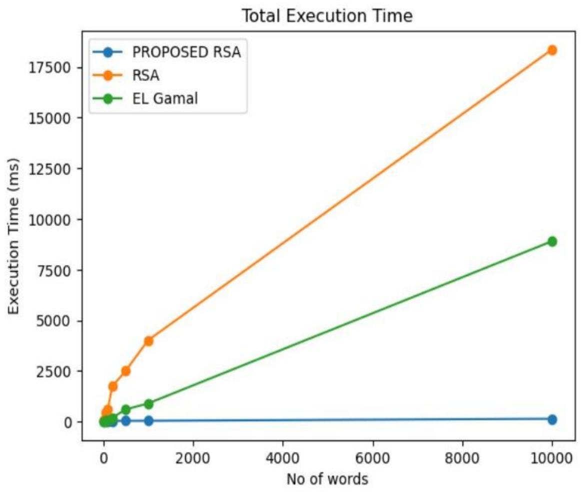

Figure. 5 Comparison between the Execution times of Asymmetric algorithms- Proposed RSA, Classic RSA and El Gamal

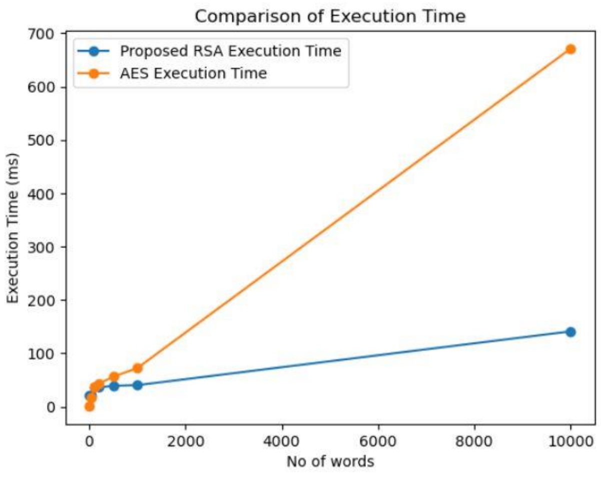

Figure. 6 Comparison between the Execution times of Asymmetric algorithm- Proposed RSA and Symmetric algorithm- AES algorithm

dynamic approach-based RSA algorithm and Classical RSA, as shown in Tables 1 and 2. From observing the outcomes in Figs. 3 and 4, it is evident that the execution of both encryption and decryption of the proposed RSA is much faster than classical RSA, that justifies that the encryption of message can be done much faster in the proposed algorithm than the classical RSA algorithm.

We have also compared the total execution time between the proposed RSA, classical RSA [21], El Gamal [23], and AES algorithms [22] by varying the message size (in words), as shown in Table 3. From observing the outcomes in Figs. 5 and 6, it is evident that the total execution time of the proposed RSA algorithm is much faster than another asymmetric algorithm, i.e. classical RSA, EL Gamal and comparatively faster execution time as compared to the symmetric algorithm, i.e. AES algorithm, as the message size increases.

A faster algorithm requires fewer computational resources, such as memory. By reducing the execution time of an algorithm, the cost required in terms of resources can be reduced [19]. In the modern era, cloud computing platforms such as amazon web service (AWS) employ virtualization techniques to consolidate physical resources into virtual instances. The cost of using cloud resources is calculated in terms of time. Hence by reducing the execution time, usage of cloud resources is minimized, leading to cost saving in terms of hardware requirement.

The algorithm is often used in real-time applications where speed is crucial. Reducing the time required for an algorithm can improve operational efficiency and reduce associated costs [20]. As the size of the input data increases, algorithms with higher time complexity can become prohibitively expensive to execute. By reducing the time complexity, an algorithm can scale more effectively.

The dynamic approach focuses on optimizing algorithms and techniques to reduce the time required for computational tasks. By executing computations more efficiently, the processor can complete the tasks faster, leading to a shorter duration of high activity. As a result, the processor can enter low-power states sooner, reducing overall energy consumption. In traditional approaches, where computations may be less optimized, the CPU may have to remain active longer to complete tasks. This leads to increased idle time, where the CPU remains powered on but not actively processing. By reducing computational time through the dynamic approach, the CPU can enter idle states earlier, minimizing energy usage during these idle periods. Energy usage depends not solely on the processor but also involves other system components such as memory, storage, and network interfaces. By reducing computational time, the overall system can complete tasks faster, enabling these components to enter low-power states sooner and reduce energy consumption. Coordinated optimization across the system can lead to significant energy savings.

6. Conclusion

In this paper, we have used a dynamic approach to enhance the RSA algorithm, focusing on computational speed and optimizing the encryption and decryption processes. Our suggested dynamic approach stores the encryption and decryption value of every character, and once the character gets repeated, the already stored encrypted and decrypted value is being used, which reduces computational complexity. To evaluate the efficacy of this approach, extensive experiments were conducted on various datasets comprising paragraphs of different sizes. The experiments involved encrypting and decrypting paragraphs containing 10, 50, 100, 200, 500, 1000, and 10,000 words. Comparative analyses were performed against existing RSA, El Gamal, and AES algorithms. Results indicate that the proposed dynamic RSA approach consistently outperforms traditional RSA in terms of encryption time, decryption time, and total execution time across all tested paragraph sizes. The regression analysis revealed stark differences between the two approaches. Furthermore, the Proposed RSA minimized the energy consumption and cost reduction for the encryption and decryption process in data transmission. In future, the security of the algorithm can be maintained along with the faster computation by erasing the stored values on both sender and receiver side.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

“Conceptualization, Pradeep Krishnadoss and Palani Thanaraj Krishnan; methodology, Nishanth Paramasivam; software, Deepesh Sai Kesavan; validation, Anish Thishyaa Raagav; formal analysis, Pradeep Krishnadoss; investigation, Pradeep Krishnadoss; resources, Deepesh Sai Kesavan; data curation, Deepesh Sai Kesavan; writing—original draft preparation, Anish Thishyaa Raagav; writing—review and editing, Nishanth Paramasivam; formal analysis, Pradeep Krishnadoss; investigation, Pradeep Krishnadoss; visualization, Pradeep Krishnadoss; supervision, Pradeep Krishnadoss”.

References

[1] Y. Li and Q. Liu, “A comprehensive review study of cyber-attacks and cyber security;

Emerging trends and recent developments", *Energy Reports,* Vol. 7, pp. 8176-8186, 2021.

[2] M. Panda, "Performance analysis of encryption algorithms for security", In: *Proc. of 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, pp. 278-284, 2016.

[3] O. G. Abood and S. K. Guirguis, "A survey on cryptography algorithms", *International Journal of Scientific and Research Publications*, Vol. 8, No. 7, pp. 495-516, 2018.

[4] A. S. Khader and D. Lai, "Preventing man-in-the-middle attack in Diffie-Hellman key exchange protocol", In: *Proc. of 2015 22nd international conference on telecommunications (ICT)*, pp. 204-208. IEEE, 2015.

[5] C. Gupta and N. V. S. Reddy, "Enhancement of Security of Diffie-Hellman Key Exchange Protocol using RSA Cryptography", In: *Proc. of Journal of Physics: Conference Series*, Vol. 2161, No. 1, p. 012014. IOP Publishing, 2022.

[6] V. Shashidhara and S. Kengond, "Performance analysis of symmetric key cryptographic algorithms", In: *Proc. of 2018 international conference on communication and signal processing (ICCSP)*, pp. 0411-0415, 2018.

[7] A. G. Khan, S. Basharat, and M. U. Riaz, "Analysis of asymmetric cryptography in information security based on computational study to ensure confidentiality during information exchange", *International Journal of Scientific & Engineering Research*, Vol. 9, No. 11, pp. 992-999, 2018

[8] S. Nisha and M. Farik, "Rsa public key cryptography algorithm–a review", *International Journal of Scientific & Technology Research*, Vol. 6, No. 7, pp. 187-191, 2017.

[9] B. Bhowmik "Dynamic programming. Its principles, applications, strengths, and limitations", C*criterion*, Vol. 4, No. 7, 2010.

[10] J. Liu, C. Fan, X. Tian, and Q. Ding, "Optimization of AES and RSA algorithm and its mixed encryption system", In: *Proc. of Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceedings of the Thirteenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, August, 12-15, 2017, Matsue, Shimane, Japan, Part II 13*, pp. 393-403, 2018.

[11] E Jintcharadze and M Iavich, "Hybrid implementation of Twofish, AES, ElGamal and RSA cryptosystems", In: *Proc. of 2020 IEEE East-West Design & Test Symposium (EWDTS),* pp. 1-5, 2020.

[12] P. Gupta, DK Verma, and AK Singh, "Improving RSA algorithm using multi-threading model for outsourced data security in cloud storage", In: *Proc. of 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence),* pp. 14-15, 2018.

[13] V. Dimitrov, L. Vigneri, and V. Attias, "Fast generation of RSA keys using smooth integers", *IEEE Transactions on Computers,* Vol. 71, No. 7, pp. 1575-1585, 2021.

[14] R. F. S. Lizy, "Improvement of RSA Algorithm Using Euclidean Technique", *Turkish Journal of Computer and Mathematics Education (TURCOMAT),* Vol. 12, No. 3, pp. 4694-4700, 2021.

[15] Taneja, R. K. Shukla, and R. S. Shukla, "Improvisation of RSA Algorithm in Respect to Time and Security with the Proposed (AEA) Algorithm", In: *Proc. of Journal of Physics: Conference Series*, Vol. 1998, No. 1, p. 012036, 2021.

[16] F. H. M. S. Al-Kadei, and H. A. Mardan, "Speed up image encryption by using RSA algorithm", In: *Proc. of 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 1302-1307, 2020.

[17] J. J. Liu, K. T. Tsang, and Y. H. Deng, "A variant RSA acceleration with parallelisation", *International Journal of Parallel, Emergent and Distributed Systems*, Vol. 37, No. 3, pp. 318-332, 2022.

[18] S. B. B. Priyadarshini, S. Rath, Sm Patel, A Udgata, A Mohanta S. R. ali, S. Panigrahi, and P. Sahu, "A hybrid random image generation strategy (HR-IGS) for securing plain text data in networks", *Journal of Theoretical and Applied information technology,* Vol. 101, No. 6, 2023.

[19] S. K. Gorva and L. C. Anandachar, "Effective Load Balancing and Security in Cloud using Modified Particle Swarm Optimization Technique and Enhanced Elliptic Curve Cryptography Algorithm", *International Journal of Intelligent Engineering & Systems*, Vol. 15, No. 2, 2022, doi: 10.22266/ijies2022.0430.18.

[20] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities", *IEEE Access*, Vol. 9, pp. 28177-28193, 2021.

[21] Parkar, P. Ashwini, N. Madhuri Gedam, N. Ansari, and S. Therese, “Performance level evaluation of cryptographic algorithms”, In: *Proc. of Intelligent Computing and Networking*, pp. 157-167, 2021.

[22] N. Alenezi, Mohammed, H. Alabdulrazzaq, and Q. Nada Mohammad, “Symmetric encryption algorithms: Review and evaluation study”, *International Journal of Communication Networks and Information Security* Vol.12, No. 2 pp.256-272, 2020.

[23] J. Mohammed, Saja, and B. Dujan Taha, “Performance evaluation of RSA, ElGamal, and Paillier partial homomorphic encryption algorithms.” In: *Proc. of 2022 International Conference on Computer Science and Software Engineering (CSASE)*, pp. 89-94, 2022.