


Improving the Robustness of RSA Encryption Through Input-Based Key Generation

Dua M. Ghadi 

Wasit Education Directorate, Ministry of Education, Baghdad 10011, Iraq

Corresponding Author Email: mdua1093@gmail.com



Copyright: ©2024 The author. This article is published by IIETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/mmep.110124>

ABSTRACT

Received: 19 May 2023

Revised: 25 July 2023

Accepted: 11 August 2023

Available online: 30 January 2024

Keywords:

decryption, encryption, hexadecimal values, input-based cryptography, variable-length input, key generation method, NIST randomness tests, Rivest-Shamir-Adleman (RSA)

In cryptography, we use different methods to hide information and make sure it's safe when shared. This keeps hackers from getting at it. The RSA cryptosystem is a way to protect messages that uses two different keys. In this paper, a new method is suggested. It includes changing messages into hexadecimal values and then turning them into decimals. Public and private keys are generated based on the input of message's length, adding an increase of complexity to enhance the security of the cryptosystem. The proposed algorithm uses two different keys to encrypt and decrypt each character, this makes the cryptosystem increasing the difficulty for attackers trying to hack it. A comparison is made between the proposed algorithm and the original RSA, using NIST tests and measuring the running time of key generation, encoding, and decoding operations. The results show that the new algorithm provides a secure transmitting of data. The proposed algorithm enhances security over the standard RSA algorithm by using hexadecimal conversion, multiple keys, dual key encryption per one-character, increased randomness, and a more advanced cryptography method, offering improved resistance against attacks and protecting data.

1. INTRODUCTION

The continuous progress in telecommunication and networking technologies has sparked a rising trend in the utilization of message exchange, involving the storage and transfer of diverse content and its accompanying information [1]. This growing reliance on message exchange underscores the critical significance of data security in today. For instance, in e-commerce, secure message exchange plays a vital role in protecting personal and financial details during online transactions. Similarly, in healthcare, it ensures the confidentiality of patient medical records, safeguarding sensitive data. In the financial sector, secure message exchange is essential for protecting sensitive financial information during electronic transactions. Moreover, secure message exchange is crucial in remote work and virtual collaboration, safeguarding intellectual property and proprietary business information. These are just a few examples of the numerous applications that highlight the importance of secure message exchange and data security. Overall, robust data security measures are essential to maintain integrity, privacy, and trust in message exchange.

Cryptography is used to protect data and communications from hackers by transforming information into formats that are unreadable unless the user has a particular key or method of decrypting it. Cryptography has grown over time to become a crucial component of modern communication safety, with applications ranging from banking services to secure texting to securing sensitive data and others.

Numerous encryption techniques are widely available and

employed to protect information. Encryption involves the use of a key to transform data into an unreadable format, rendering it incomprehensible to unauthorized parties. To revert the data to its original form, decryption using the appropriate key is necessary. Key generators create new keys for encryption systems. There are different sorts of keys in these algorithms [2].

Symmetric cryptography, also referred to as secret-key or traditional cryptography, utilizes a single key for both encoding and decoding information. In simpler terms, both the sender and the recipient use the same key to encrypt and decrypt messages. This type of cryptography is relatively simple and efficient, but it requires that both parties share the key, which can be difficult to do securely.

Another type of cryptography is referred to as "asymmetric cryptography" or "public-key cryptography," which is a more advanced form of encryption. This method uses a pair of keys used for encryption, referred to as the public and private keys, to ensure the security of information. The public key is available to everyone who wants to send a message, while the private key is securely held by the owner of the public key. This allows for secure communication between two parties, as the sender can encrypt a message with the recipient's public key and only the recipient can decrypt it with their private key.

The major applications of public key cryptography are authentication, non-repudiation, and key exchange [3, 4]. The widely adopted public-key cryptography algorithm in use today is the RSA (Rivest-Shamir-Adleman) algorithm, which is recognized as the most well-known and extensively used public-key cryptography system. The acronym RSA

represents the surnames of its creators: Ronald Rivest, Adi Shamir, and Leonard Adleman [5].

At a high level, the RSA algorithm works as follows: a user generates a pair of keys - a public key and a private key. The public key is distributed to anyone who wants to send a message, while the private key is kept secure by the owner of the public key. When someone wants to send an encrypted message to the owner of the public key, they use the recipient's public key to encrypt the message. The encrypted message can only be decrypted using the corresponding private key held by the recipient. This ensures that only the intended recipient can access the original content of the message [5]. The RSA algorithm is considered a safe method of encryption because it relies on the complexity of factorizing primes of large numbers, which is a widely recognized mathematical problem that currently has no efficient algorithm to solve [6, 7]. Therefore, it is widely preferred over other cryptography algorithms, such as Advanced Encryption Standard (AES), a symmetric key algorithm [8], and the Goldwasser-Micali (GM) algorithm, an asymmetric key algorithm [3]. The current RSA implementations are more efficient owing to their flexible key size range, which spans from 2 to 2048 bits. The algorithm's security is contingent on the key size selected by the user or programmer. Larger key sizes are generally considered more secure because they increase the computational effort required to break the encryption. The security of RSA is based on the difficulty of factoring large composite numbers into their prime factors. As the key size increases, the number of possible factorizations grows exponentially, making it increasingly challenging to find the prime factors and decrypt the message without the corresponding private key. On the other hand, the mention of a 512-bit key size in various applications is concerning. A 512-bit key size is now considered insecure due to advancements in computing power and factoring algorithms. The computational resources required for factoring 512-bit keys have become increasingly accessible, rendering them inadequate for ensuring secure communication or data protection. Furthermore, in addition to this method, a 1024-bit key length is employed. As a result, the 512-bit key size is still prevalent in various applications [9, 10]. The RSA cryptosystem has been the target of various attacks over the years, and it is essential to secure it against these attacks. A common attack method is known as a brute force attack, in which all possible combinations of keys are tested until the correct one is identified. An attack on the RSA algorithm is an attempt to decrypt encrypted data without knowing the private key. The RSA algorithm is secure because it is very difficult to factor large numbers. However, if an attacker has enough time and resources, they can try all possible keys until they find the correct one. This is known as a brute force attack [11]. The proposed method addresses this issue by generating multiple keys based on the length of inputs. This makes it much more difficult for an attacker to try all possible keys. The aim of this paper is to propose a novel technique for enhancing the security of the RSA cryptosystem against brute force attacks. The proposed technique generates multiple keys based on the length of the input message. This makes it much more difficult for an attacker to try all possible keys, i.e., ensuring that the encryption scheme remains secure even if one key is compromised.

2. RSA CRYPTOSYSTEM

The RSA Cryptosystem is a commonly utilized public key

encryption method that was initially introduced by Rivest et al. in 1978 [5], as previously noted. This cryptographic technique is founded on the challenge of factoring large numbers and has emerged as a favored approach for ensuring secure network communication. The difficulty of factoring large numbers is the basis of RSA security. It is believed to be computationally infeasible to factor a large number that is the product of two large prime numbers. This is because the number of possible factors of a large number grows exponentially with the size of the number. In this section, we will provide a detailed examination of the RSA public key cryptosystem algorithm, drawing from the studies conducted in references [12, 13].

RSA Key Generation Steps

- 1-Randomly take two large prime numbers (p and q) of equal length.
- 2-Calculate the product of p and q to obtain n , which is the public modulus.
- 3-Compute the totient function of Euler:

$$\varphi(n) = (p - 1) (q - 1)$$

where, the totient function of Euler, $\varphi(n)$, is the number of positive integers less than or equal to n that are relatively prime to n . In other words, $\varphi(n)$ is the number of integers that cannot be divided by any of the prime factors of n .

- 4-Select a public key e , such that e is an integer and $gcd(e, \varphi(n)) = 1$.
- 5-Calculate the private decryption exponent d using the extended Euclidean algorithm: $d = e^{-1} \bmod \varphi(n)$.

RSA Encryption Steps

- 1-Select a plaintext message M such that M is less than n .
- 2-Compute the ciphertext C by raising M to the power of the public encryption exponent e and taking the result modulo n : $C = M^e \bmod n$.

RSA Decryption Steps

- 1-Compute the plaintext M by raising the ciphertext C to the power of the private decryption exponent d and taking the result modulo n : $M = C^d \bmod n$.
- 2-Retrieve the message M .

There are a number of known attacks against RSA, but they all require the attacker to know the public modulus n . The most common attack is the brute-force attack, which simply tries all possible values of d until it finds one that decrypts the ciphertext correctly. This attack is impractical for large values of n , but it is possible for small values of n .

The modification proposed for RSA, which is explaining it in the next section, is considered the solution for enhancing the security of the cryptosystem because, as mentioned in the earlier section, it makes it much more difficult for an attacker to try all possible keys, including that the encryption scheme remains secure even if one key is compromised.

3. MODIFICATION PROPOSED FOR THE RSA CRYPTOSYSTEM

In this section, we explain the proposed solution, which will

add security and complexity to the RSA cryptosystem, which depends on generating keys and input lengths and the way to encode and decrypt messages by using the generated keys.

The modification proposed focuses on increasing the complexity of procedures for creating keys with the encryption and decryption processes of the RSA cryptosystem. The main difficulty in many situations is that it is readily broken since keys depending on n are simply computed.

Since n in RSA is the product of two large prime numbers, its value can be easily verified. However, if an attacker manages to discover the value of n , they may be able to find the keys and compromise the security of the cryptosystem.

To improve the complexity and security of the RSA cryptosystem, several significant modifications have been proposed and explained in this section, as follows:

In this modification, we encrypt a message after converting each character into a hexadecimal ASCII value of two digits $M_j = (m_i m_{i+1})_{16}$, where $j = 1, 2, 3, \dots$ and $i = 2j - 1$. Then, separate them into two values $M_j = (m_i, m_{i+1})_{16}$, and convert each value of m_i and m_{i+1} to decimal values $M_j = (m_i, m_{i+1})_{10}$ [14] as follow:

The message $M_j = M_1 M_2 M_3 \dots M_j$. The Converting as below:

$$M_1 = (m_1 m_2)_{16} = (m_1, m_2)_{16} = (m_1, m_2)_{10} \text{ for } j = 1.$$

$$M_2 = (m_3 m_4)_{16} = (m_3, m_4)_{16} = (m_3, m_4)_{10} \text{ for } j = 2.$$

$$M_3 = (m_5 m_6)_{16} = (m_5, m_6)_{16} = (m_5, m_6)_{10} \text{ for } j = 3.$$

...

$M_j = (m_i m_{i+1})_{16} = (m_i, m_{i+1})_{16} = (m_i, m_{i+1})_{10}$, where j represents the length of message.

In this research paper, we have made modifications to the RSA cryptosystem algorithm by adding public and private keys that depend on the length of the message to be encrypted. Like the conventional RSA algorithm, the computation of the public modulus n in our proposed algorithm involves utilizing two large prime numbers p and q , which are multiplied to compute n , i.e., $n = p * q$. Then compute $\varphi(n) = (p - 1)(q - 1)$. In adding new public keys e_i , where $i = 1, 2, 3, \dots, j$ and $j = \text{length of message}$ are selected as follows: $1 < e_i < \varphi(n)$ such that $\gcd(e_i, \varphi(n)) = 1$. After that, calculate the private keys d_i by using: $d_i = e_i^{-1} \text{ mod } \varphi(n)$, $i = 1, 2, 3, \dots, j$.

For encryption procedures, this proposed modification uses these exponents to encrypt messages using public keys e_i as follows:

$$\begin{aligned} C_j &= (c_i, c_{i+1}) = (m_i^{e_i}, m_{i+1}^{e_{i+1}}) \text{ mod } n \\ &= (m_i^{e_{i+1}}, m_{i+1}^{e_{i+2}}) \text{ mod } n \\ &\quad \vdots \\ &= (m_i^{e_{i+j}}, m_{i+1}^{e_i}) \text{ mod } n \end{aligned}$$

where, C_j is the ciphertext.

Similarly, for decryption procedures, use exponents private keys d_i to decrypt and find the message M_j as follows:

$$\begin{aligned} M_j &= (m_i, m_{i+1}) = (c_i^{d_i}, c_{i+1}^{d_{i+1}}) \text{ mod } n \\ &= (c_i^{d_{i+1}}, c_{i+1}^{d_{i+2}}) \text{ mod } n \\ &\quad \vdots \\ &= (c_i^{d_{i+j}}, c_{i+1}^{d_i}) \text{ mod } n \end{aligned}$$

In the case of brute force attack against RSA, the process of obtaining the private key (d) through a brute force attack involves systematically testing all possible values until the correct one is found. In this scenario, the attacker iterates

through a range of numbers, starting from 1, in order to identify a value (d) that satisfies the equation: $C^d \text{ mod } n = M$, where M represents the original message.

Regarding the proposed modification, the private keys are represented by the exponents ($d_i, d_{i+1}, d_{i+2}, \dots$). This modification renders the brute force attack computationally infeasible. Even if the attacker manages to discover one of the private key exponents, such as d_1 , the message remains unreadable or unknown unless they also find d_2 . This is because the proposed modification employs two keys for encrypting and decrypting each character of the message. Consequently, the attacker would need to search through an exponentially large number of potential combinations, making the task exceedingly complex and impractical to break the cryptosystem within a reasonable timeframe.

3.1 Proposed algorithm

The message $M_j = \dots$

Keys Creation

1. Randomly take two large prime numbers, p and q .
2. Calculate $n = (p)(q)$.
3. Compute $\varphi(n) = (p - 1)(q - 1)$.
4. Randomly generate public keys e_i such that $1 < e_i < \varphi(n)$ and $\gcd(e_i, \varphi(n)) = 1$.
5. Calculate $d_i = e_i^{-1} \text{ mod } \varphi(n)$, $i = 1, 2, 3, \dots, j$.

Encryption

1. Given the message M_j , where $M_j = M_1 M_2 M_3 \dots M_j$ and $j = \text{length of message } M_j$.
2. Convert M_j into a hexadecimal ASCII value $(m_i m_{i+1})_{16}$.
3. Rewrite M_j as $(m_i, m_{i+1})_{16}$ and then convert into decimal values $(m_i, m_{i+1})_{10}$.
4. Compute C_j by:

$$\begin{aligned} C_1 &= (m_i^{e_i}, m_{i+1}^{e_{i+1}})_{10} \text{ mod } n = (c_i, c_{i+1}) \\ C_2 &= (m_i^{e_{i+1}}, m_{i+1}^{e_{i+2}})_{10} \text{ mod } n = (c_i, c_{i+1}) \\ &\quad \vdots \\ C_j &= (m_i^{e_{i+j}}, m_{i+1}^{e_i})_{10} \text{ mod } n = (c_i, c_{i+1}) \end{aligned}$$
5. Send the ciphertext C_j to the other side.

Decryption

1. Received the ciphertext C_j .
2. Compute M_j :

$$\begin{aligned} M_1 &= (c_i^{d_i}, c_{i+1}^{d_{i+1}}) \text{ mod } n = (m_i, m_{i+1})_{10} \\ M_2 &= (c_i^{d_{i+1}}, m_{i+1}^{d_{i+2}}) \text{ mod } n = (m_i, m_{i+1})_{10} \\ &\quad \vdots \\ M_j &= (c_i^{d_{i+j}}, c_{i+1}^{d_i}) \text{ mod } n = (m_i, m_{i+1})_{10} \end{aligned}$$
3. Convert $(m_i, m_{i+1})_{10}$ to hexadecimal ASCII value $(m_i, m_{i+1})_{16}$.
4. Rewrite $(m_i, m_{i+1})_{16}$ as $(m_i m_{i+1})_{16}$.
5. Recovered the original message M_j .

3.2 Implementation

The following example shows the implementation of the proposed algorithm above.

Message = 'Hello'

Keys Creation

1. Take two prime numbers, $p = 5$ and $q = 17$.
2. Compute $n = 5 * 17 = 85$.
3. Compute $\varphi(n) = (5 - 1)(17 - 1) = 64$.
4. Generate random public keys e_i such that $1 < e_i < \varphi(n)$ and $\gcd(e_i, \varphi(n)) = 1, i = 1,2,3,4,5$. We choose $e_1 = 3, e_2 = 5, e_3 = 7, e_4 = 9, e_5 = 11$.
5. Compute $d_i = e_i^{-1} \bmod \varphi(n)$: we get $d_1 = 43, d_2 = 13, d_3 = 55, d_4 = 57$ and $d_5 = 35$.

Encryption

1. Convert 'Hello' into a hexadecimal ASCII value: $H: (48)_{16}, e: (65)_{16}, l: (6C)_{16}, l:(6C)_{16}, o: (6F)_{16}$
2. Rewrite and convert to decimal as: $(4, 8)_{16} = (4, 8)_{10}, (6, 5)_{16} = (6, 5)_{10}, (6, C)_{16} = (6, 12)_{10}, (6, C)_{16} = (6, 12)_{10}, (6, F)_{16} = (6, 15)_{10}$.
3. Compute $C_j, j = 1,2,3,4,5$:

$$C_1 = (4^{e_1}, 8^{e_2}) \bmod 85$$

$$= (4^3 \bmod 85, 8^5 \bmod 85)$$

$$= (64, 43).$$

$$C_2 = (6^{e_2}, 5^{e_3}) \bmod 85$$

$$= (6^5 \bmod 85, 5^7 \bmod 85)$$

$$= (41, 10).$$
 :

$$C_5 = (6^{e_5}, 15^{e_1}) \bmod 85$$

$$= (6^{11} \bmod 85, 15^3 \bmod 85)$$

$$= (56, 60).$$
4. Send the ciphertext C_1, C_2, \dots, C_5 to the other side

Decryption

1. Received the ciphertext C_1, C_2, \dots, C_5 .
2. Compute $M_j, j = 1,2,3,4,5$:

$$M_1 = (64^{d_1}, 43^{d_2}) \bmod 85$$

$$= (64^{43} \bmod 85, 43^{13} \bmod 85)$$

$$= (4, 8).$$

$$M_2 = (41^{d_2}, 10^{d_3}) \bmod 85$$

$$= (41^{13} \bmod 85, 10^{55} \bmod 85)$$

$$= (6, 5).$$
 :

$$M_5 = (56^{d_5}, 60^{d_1}) \bmod 85$$

$$= (56^{35} \bmod 85, 60^{43} \bmod 85)$$

$$= (6, 15).$$

3. Convert $(4,8)_{10}, (6,5)_{10}, \dots, (6,15)_{10}$ into hexadecimal $(4,8)_{16}, (6,5)_{16}, \dots, (6,F)_{16}$.
4. Rewrite $(48)_{16}, (65)_{16}, \dots, (6F)_{16}$.
5. Find the character that corresponds to $(48)_{16} = H, (65)_{16} = e, \dots, (6F)_{16} = o$.
6. Recovered the original message 'Hello'.

4. RESULTS AND DISCUSSION

The proposed algorithm has been implemented in the MATLAB R2018b (9.5.0.944444) 64-bit software on Core i3 computer with CPU 2.00GHz and RAM 4GB. As we know, when an attacker wants to break a cryptosystem, they attack the private key. The security of the RSA is predicated on the complexity of factoring large numbers and the potency of the private key, thereby affirming its resilience. The proposed algorithm uses characters with their hexadecimal values and converts them into pairs of values (m_i, m_{i+1}) [14]. This provides a higher level of security for exchanges between two parties. To encrypt and decrypt messages, the proposed algorithm utilizes randomly generated public and private keys (e_i and d_i) that match the message length.

Our proposed algorithm makes the cryptosystem more sophisticated and secure by generating multiple public and private keys and a method of embedding messages using hexadecimal ASCII values, in addition to encryption and decryption procedures. where each character of the message is encrypted with two keys; on the other hand, it is decrypted with two private keys, as shown in steps 4 and 2 of the proposed algorithms of RSA in the stages of encryption and decryption. The use of multiple keys ensures that even if one key is compromised, the entire system remains secure, while embedding messages using hexadecimal ASCII values adds an extra layer of protection against attacks.

The Figures 1 and 2 and Tables 1 and 2 presented display the duration of generating public and private keys (e_i, d_i, e , and d), as well as the encryption and decryption times in seconds of 1000-character messages, utilizing distinct prime numbers for both the proposed algorithm and the original RSA algorithm.

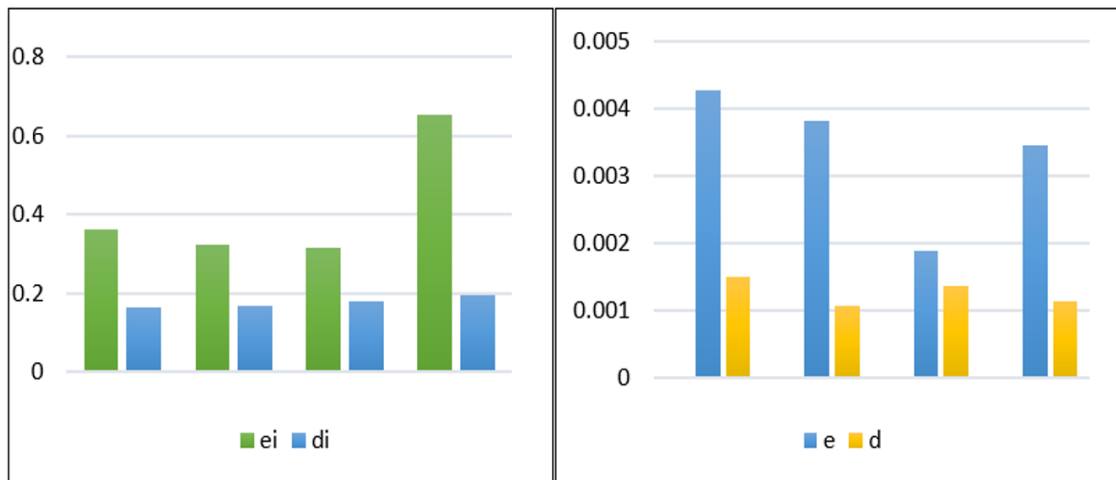


Figure 1. The running time of generating keys (e_i, d_i, e and d) for 1000-characters

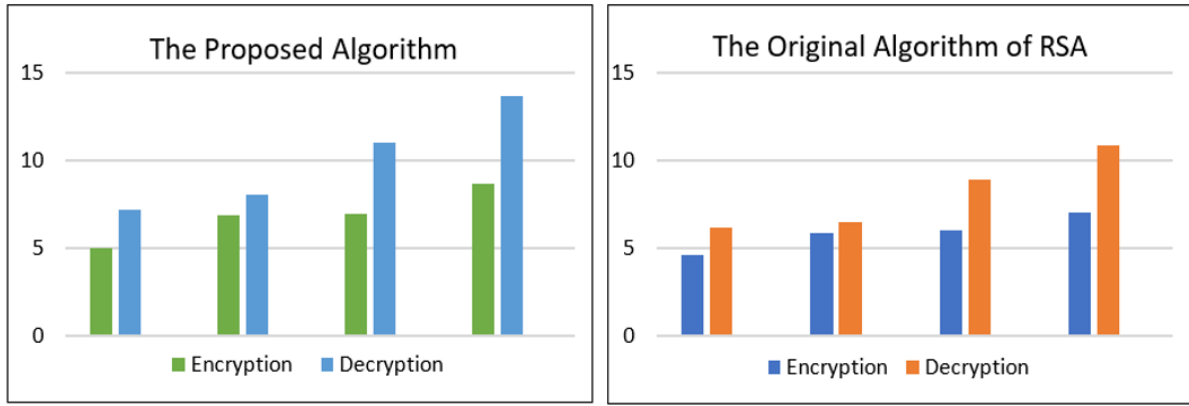


Figure 2. The running time encryption and decryption of messages (1000-characters)

Table 1. The running times of the proposed algorithm in seconds

p	q	e_i	d_i	Encryption	Decryption
58363	53269	0.362074	0.165056	5.021223	7.178098
130253	130241	0.323466	0.168802	6.843971	8.034885
2305337	2303669	0.314585	0.178270	6.931602	11.050265
23569129	23568971	0.653467	0.195785	8.654589	13.656344

Table 2. The running times of the original RSA algorithm in seconds

p	q	e	d	Encryption	Decryption
58363	53269	0.004274	0.001500	4.640346	6.170683
130253	130241	0.003816	0.001065	5.876735	6.466873
2305337	2303669	0.001881	0.001352	6.017149	8.931112
23569129	23568971	0.003457	0.001136	7.028774	10.839939

The results indicate that the proposed algorithm takes longer to generate keys than the original RSA technique. Due to the running time for generating e_i (public keys) being longer than d_i (private keys) due to several factors such as random generation, the proposed algorithm specifies that e_i is randomly generated within the range of 1 to $\phi(n)$. Generating a random number based on the input length and due gcd computation steps to verify this condition may require additional time. In the case of the calculation of d_i , it involves computing the modular inverse of e_i modulo $\phi(n)$. Depending on the specific implementation and the algorithm used for modular inversion, this operation will be computationally faster than the random generation and gcd computation steps involved in generating e_i .

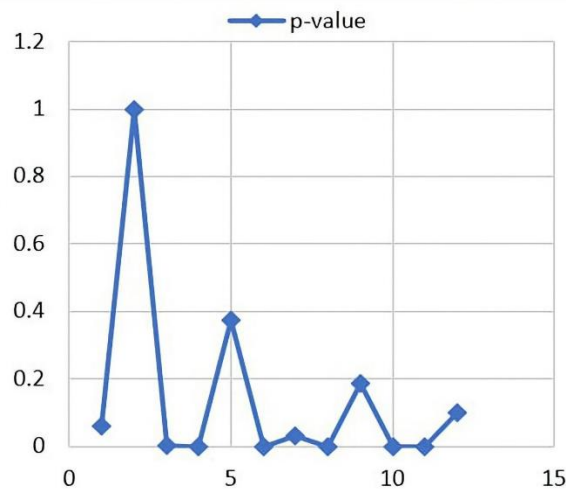
Nevertheless, this elongated key generation process can be regarded as a benefit as it bolsters the system's security by generating numerous keys based on the input length, thereby adding an extra layer of complexity. Furthermore, the

encryption and decryption processes of the proposed RSA algorithm require more time compared to those of the original RSA algorithm, as they necessitate using two keys for each character. Despite these downsides, the proposed RSA algorithm enhances security through its key generation, encryption, and decryption methods.

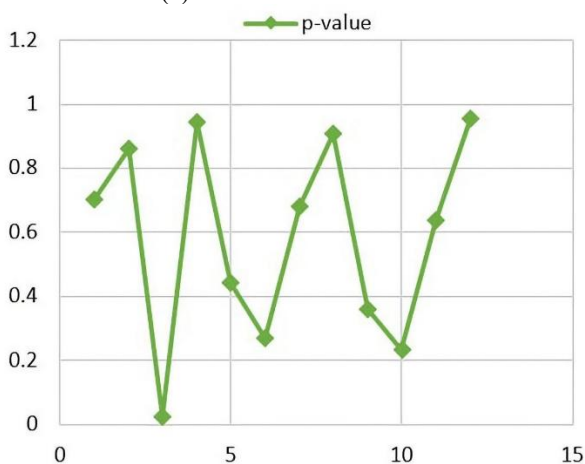
To evaluate the proposed algorithm's performance in terms of randomness, we subjected it to the NIST tests [15], using 12 tests chosen based on the length of the message encrypted (10000 bits). As known from the NIST tests, a p-value greater than or equal to 0.01 indicates randomness (otherwise, it failed). The results of the NIST tests showed that the proposed algorithm generated more randomness than the original RSA algorithm. Specifically, the output of the proposed algorithm passed all the NIST randomness tests, indicating that it generated a highly random sequence of numbers. In contrast, the output of the original RSA algorithm failed some of the tests, as shown in Table 3 and Figure 3.

Table 3. The NIST tests of the proposed algorithm and the original RSA for 10000 bits

Test	P-Value (RSA)	P-Value (Proposed Algorithm)
Frequency test	0.0601080779223996	0.703945415151674
Block frequency	0.999999667966886	0.862371323589811
Runs	0.0016	0.022520239477322
Longest run of ones in block	4.0334e-45	0.944987682549921
Binary matrix rank	0.374305808177149	0.441305948144981
Discrete fourier transform	7.2189e-07	0.270811569981268
Overlapping template matching	0.0313561580237405	0.681484410476162
Universal	0	0.90839709352016
Linear complexity	0.1865343653042280	0.359316182992329
Serial	0	0.233767801899238
Approximate entropy	4.40301060682201e-152	0.637758780173094
Cumulative sums	0.0999915723875517	0.953761550531937



(a) The NIST tests of RSA



(b) The NIST tests of the proposed algorithm

Figure 3. The NIST tests each method

5. CONCLUSIONS

The proposed RSA algorithm in this research utilizes hexadecimal values and multiple keys to enhance security for exchanging messages between two parties. Based on the length of the message, the algorithm generates several pairs of public and private keys and encrypts each character using two keys, ensuring that the encryption scheme remains secure even if one key is compromised. The NIST tests showed that the proposed algorithm generated more randomness than the original RSA algorithm. Although the key generation, encryption, and decryption processes take longer than those of the original RSA, the added complexity provides improved security against attacks. Overall, the proposed RSA algorithm offers a more sophisticated and secure method of encryption for exchanging sensitive information. It leverages hexadecimal values and multiple keys, enhancing security while introducing a slight increase in computational complexity. The significance of this work lies in providing a robust and secure method for exchanging sensitive information. By enhancing the encryption process, the proposed RSA algorithm offers a viable solution for secure communication, contributing to the field of encryption methods by introducing a novel approach that fortifies data protection.

REFERENCES

- [1] Salman, L.A., Hashim, A.T., Hasan, A.M. (2022). Selective medical image encryption using polynomial-based secret image sharing and chaotic map. *International Journal of Safety and Security Engineering*, 12(3): 357-369. <http://doi.org/10.18280/ijss.120310>
- [2] Khan, Z. (2016). Efficient design and implementation of elliptic curve cryptography on FPGA. Doctoral dissertation, University of Sheffield.
- [3] Priya, N., Kannan, M. (2017). Comparative study of RSA and probabilistic encryption. *International Journal of Engineering and Computer Science*, 6(1): 19867-19871. <http://doi.org/10.18535/ijecs/v6i1.04>
- [4] Nisha, S., Farik, M. (2017). RSA public key cryptography algorithm - A review. *International Journal of Scientific and Technological Research*, 6(7): 187-191.
- [5] Rivest, R.L., Shamir, A., Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2): 120-126. <http://doi.org/10.1145/359340.359342>
- [6] Singamaneni K.K., Naidu P.S., Kumar P.V.S. (2018). Efficient quantum cryptography technique for key distribution. *Journal Européen des Systèmes Automatisés*, 51(4-6): 283-293. <http://doi.org/10.3166/jesa.51.283-293>

- [7] Huang, X., Wijesekera S., Sharma D. (2009). Quantum cryptography for wireless network communications. In 2009 4th International Symposium on Wireless Pervasive Computing, Melbourne, VIC, Australia, pp. 1-5. <http://doi.org/10.1109/ISWPC.2009.4800604>
- [8] Pethe, H.B., Pande, S.R. (2017). Comparative study and analysis of cryptographic algorithms AES and RSA. *International Journal of Advance Research in Computer Science and Management Studies*, 3(1): 48-56.
- [9] Yu, Y., Xue, L., Au, M.H., et al. (2016). Cloud data integrity checking with an identity-based auditing mechanism from RSA. *Future Generation Computer Systems*, 62: 85-91. <https://doi.org/10.1016/j.future.2016.02.003>
- [10] Al-Barazanchi, I., Shawkat, S.A., Hameed, M.H., Al-Badri, K.S.L. (2019). Modified RSA-based algorithm: A double secure approach. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 17(6): 2818-2825. <http://doi.org/10.12928/telkomnika.v17i6.13201>
- [11] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A. (1997). *Handbook of Applied Cryptography*. CRC Press. <http://doi.org/10.1201/9780429466335>
- [12] Bruce, S. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*.-2nd.
- [13] Nivetha, A., Preethy Mary, S., Santosh Kumar, J. (2015). Modified RSA encryption algorithm using four keys. *International Journal of Engineering Research & Technology (IJERT)*, 3(7): 1-5.
- [14] Kurt, M., Yerlikaya, T. (2013). A new modified cryptosystem based on Menezes Vanstone elliptic curve cryptography algorithm that uses characters' hexadecimal values. In 2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAECE), Konya, Turkey, pp. 449-453. <http://doi.org/10.1109/TAECE.2013.6557316>
- [15] Rukhin, A.L., Soto, J., Nechvatal, J.R., et al. (2010). Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards & Technology.