Research Article AI and Intelligent Systems: Engineering, Medicine & Society

A Numerical and Security Analysis of RSA: From Classical Encryption to Post-Quantum Strategies

Bhanu Prakash*, Sachin Srivastava, Sachin Prajapati, Rupesh Kumar and Gagandeep Kaur

Computer Science and Engineering Lovely Professional University Phagwara, Punjab, India

*Corresponding Author
Bhanu Prakash, Computer Science and Engineering Lovely Professional University Phagwara, Punjab, India.

Submitted: 2025, Apr 30; **Accepted:** 2025, Jun 02; **Published:** 2025, Jun 05

Citation: Prakash, B., Srivastava, S., Prajapati, S., Kumar, R., Kaur, G. (2025). A Numerical and Security Analysis of RSA: From Classical Encryption to Post-Quantum Strategies. *AI Intell Sys Eng Med Society, 1*(1), 01-07.

Abstract
This paper provides a detailed analysis of the RSA algorithm, an extensively used asymmetric encryption system forming the foundation of ultramodern cryptographic security, the most prominent asymmetric encryption algorithm, which now forms the base of ultramodern cryptographic security in the information period. In fact, RSA uses the fine complexity of large high number factorization to insure both trustability and security of communication by using public- crucial cryptography. The paper addresses the abecedarian principles of RSA concerning crucial generation, encryption, and decryption; discusses counteraccusations of RSA concerning security, vulnerabilities, and adaptability to a variety of cryptographic attacks; and identifies veritable operations of RSA in real- life situations similar as secure dispatches protocols, authentication systems, and digital autographs. This paper also discusses performance considerations, crucial length recommendations, and openings for perfecting RSA encryption.

Keywords: RSA, Encryption, Cryptography, Key Management, Public-Key, Asymmetric Encryption, Secure Communication, Digital Signatures

1. Introduction

The rapid development of digital technology has fundamentally transformed the way information is stored and communicated. This transformation has brought about significant benefits but has also introduced critical challenges in ensuring the security and integrity of sensitive information. In today's world, online transactions, financial exchanges, healthcare data management, and government communications are predominantly digital, making the protection of this information paramount [1]. The increasing prevalence of cyberattacks, including man-in-the-middle attacks, data breaches, and identity theft, poses serious threats to both organizations and individuals [2,3].

To mitigate these risks, cryptographic methods play a crucial role in safeguarding data integrity, authenticity, and confidentiality [4].

Cryptography, the art of secret writing, has evolved significantly to address these challenges. Traditional symmetric encryption protocols, such as the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES), have been widely used for securing communications for many years [5]. These protocols utilize a single key for both encryption and decryption, which presents a significant challenge: secure key distribution. The secure transmission of the key between communicating parties is a fundamental drawback of symmetric encryption. If an attacker intercepts this key during transmission, the security of the entire communication is compromised, as all encrypted data can be decrypted [6].

Asymmetric cryptography was introduced to overcome the key distribution problem inherent in symmetric systems. Asymmetric cryptography, also known as public-key cryptography, employs two distinct keys: a public key for encryption and a private key for decryption [7]. The public key can be freely shared, while the private key must be kept secret by the owner. This fundamental difference allows for secure communication without the need for a secure channel to exchange keys. One of the most widely used asymmetric encryption algorithms is the Rivest-Shamir-Adleman

(RSA) algorithm, named after its creators, Ron Rivest, Adi Shamir, and Leonard Adleman, who developed it in 1977 [8]. The RSA algorithm's security relies on the computational complexity of prime factorization, a mathematical problem that is considered infeasible for classical computers when dealing with sufficiently large integers [9]. This computational hardness makes RSA a robust solution for many security applications [10].

1.1. Significance and Uses of RSA

The RSA algorithm has become a crucial component of modern cryptographic infrastructure and is widely employed in various applications, including:

- Secure Web Browsing (HTTPS): RSA is commonly used in TLS/SSL certificates to establish secure connections between clients and servers [2].
- Email Encryption (PGP & S/MIME): RSA encrypts email content, ensuring that only intended recipients can access it [5].
- Digital Signatures: RSA enables users to digitally sign documents, providing authentication and protection against tampering [4].
- Blockchain Technology: Many blockchain platforms use RSA-based cryptographic methods for wallet security and transaction verification [6].
- VPN & Network Security: RSA is used in virtual private networks (VPNs) and encrypted communication protocols for secure authentication [8].

1.2. Security Challenges and Emerging Threats

Despite its widespread adoption, the RSA algorithm faces several challenges:

- Computational Overhead: RSA requires high computational resources for encryption and decryption, particularly when using 2048-bit or 4096-bit keys to maintain strong security [9].
- Key Size Growth: As computing power increases, larger RSA key sizes are needed for security, which can lead to performance inefficiencies in real-time applications [10].
- Quantum Computing Threat: The advancement of quantum computers poses a major risk to RSA security. Shor's algorithm allows quantum computers to efficiently factor large numbers, potentially breaking RSA encryption in the near future [8].
- Hybrid Encryption Models: Due to RSA's computational complexity, it is often combined with symmetric encryption (e.g., AES + RSA) to balance security and performance [12].
- *C) Objectives of this Paper*
- This paper presents a comprehensive study of the RSA algorithm, covering:
- Key generation, encryption, and decryption mechanisms to understand the fundamental workings of RSA.
- Numerical implementation and mathematical foundation behind RSA security.
- Historical evolution of RSA key sizes and cryptographic advancements addressing security challenges.
- Potential improvements, including hybrid encryption techniques (e.g., AES + RSA) and post-quantum cryptographic solutions.
- Real-world implementation of RSA in secure messaging applications, demonstrating its practical use in encrypted communication.

2. Methodology

The RSA encryption technique employs a standardized stepwise procedure, fundamentally comprising key generation, encryption, and decryption. This methodology is detailed in Section III. The following subsections elaborate on these steps:

2.1. Public Key Cryptography

RSA is an asymmetric encryption algorithm, also known as public-key cryptography, which utilizes a pair of related keys:

- A public key, which can be shared with anyone, used for encryption.
- A private key, which must be kept secret by the owner, used for decryption.

This key separation ensures that only the intended recipient, possessing the corresponding private key, can decrypt the message, thereby eliminating the need for a secure key exchange mechanism. This is a core advantage of asymmetric cryptography over symmetric cryptography [7].

2.2. Key Generation Process

The generation of the RSA key pair involves the following steps:

- Selecting two distinct large prime numbers, typically denoted as p and q. The security of RSA heavily relies on the size of these primes [9].
- Computing the modulus n, which is the product of p and q ($n = p \times q$).
- Computing Euler's totient function, denoted as $\phi(n)$, which is crucial for key derivation. For RSA, $\phi(n)$ is calculated as (p-1)(q-1) [11].

2.3. Encryption and Decryption Process

- Encryption: The sender encrypts the plaintext message using the recipient's public key. This process ensures that only the intended recipient can access the data.
- Decryption: The recipient decrypts the ciphertext using their corresponding private key, thus maintaining the confidentiality and security of the communication [13].

2.4. Security Requirements

To ensure robust cryptographic security, the RSA algorithm must adhere the following requirements:

- Computational Infeasibility: Deriving the private key from the public key should be computationally infeasible. This robustness is essential to protect against brute-force attacks and factorization attacks [14].
- Efficiency: The encryption and decryption processes should be optimized to balance security and performance for practical applications.
- Key Size Selection: RSA keys, determined by the size of the primes p and q, must be sufficiently large (e.g., 2048-bit or

4096-bit) to resist factorization attacks and ensure long-term security [15].

3. RSA Algorithm

The RSA algorithm is a structured process that comprises three main phases: key generation, encryption, and decryption. The steps involved in each phase are detailed below

3.1. Key Generation

The key generation process is fundamental to RSA security and involves the following steps:

1. Select Two Distinct Large Prime Numbers: Choose two large prime numbers, denoted as p and q. These primes must be kept secret. The security of RSA relies heavily on the difficulty of factoring the product of these large primes [6].

2. Compute the Modulus (n): Calculate the modulus n by multiplying the two primes:

$$n = p \times q$$

The modulus n forms part of both the public and private keys [9].

3. Compute Euler's Totient Function ($\phi(n)$): Compute Euler's totient function, denoted as ϕ(n). For RSA, this is calculated as:

i.$\phi(n) = (p - 1)(q - 1)$

Euler's totient function is crucial for determining the encryption and decryption keys [12].

4. Select the Public Exponent (e): Choose an integer e as the public exponent such that:

$$1 < e < \phi(n) \text{ and } \gcd(e, \phi(n)) = 1$$

Where gcd represents the greatest common divisor. The public exponent e must be coprime with $\phi(n)$. Common choices for e include 3, 17, or 65537, with 65537 being widely used due to its security and efficiency advantages [3], [8].

5. Compute the Private Key (d): Calculate the private key d using the modular multiplicative inverse of e modulo $\phi(n)$:

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

The private key d must be kept secret [14].

3.2. Encryption

The encryption process transforms the plaintext message into ciphertext using the recipient's public key:

- Convert Plaintext to Numerical Value: Convert the plaintext message M into a numerical value. This can be done using various encoding schemes [10].
- Compute the Ciphertext (C): Calculate the ciphertext C using the public key (e, n):
  $C \equiv M^e \bmod n$
- Secure Transmission: The encrypted message C is then sent securely to the recipient [13].

3.3. Decryption

The decryption process recovers the original plaintext message M from the ciphertext C using the recipient's private key (d, n):

- Decrypt the Ciphertext (C): The recipient decrypts the ciphertext C using their private key (d, n):
  $M \equiv C^d \bmod n$
- Retrieve Plaintext: This retrieves the original plaintext message M, ensuring secure and authenticated communication [16].

4. Numerical Study on RSA Algorithm

Example: RSA Key Generation And Encryption Process

The RSA algorithm uses two large prime numbers to generate public and private keys. The step-by-step process is described below:

- Select two distinct primes numbers
  $p = 17, \quad q = 23$
  In practical RSA implementations, much larger prime numbers, typically 2048-bit or 4096-bit, are used to provide robust security against factorization attacks [9]. This is a crucial point to emphasize, as the security of RSA is directly related to the size of the primes [15].
- Compute modulus n, which forms part of the RSA key pair
  $n = p \times q = 17 \times 23 = 391$.
  The modulus n is a public value and is part of both the public and private keys [10].
- Compute Euler's totient function (ϕ(n)):
  $\phi(n) = (p - 1) \times (q - 1) = 16 \times 22 = 352$.
  This value remains secret and is essential for deriving the private key [11].
- Select a public exponent e such that satisfies the following conditions:
  $1 < e < \phi(n)$, $\gcd(e, \phi(n)) = 1$.
  A common choice is e = 13, ensuring it is coprime with 352 for efficient encryption [6].

4.1. Compute Private Key *d*

- The private key d is calculated using the modular inverse of e modulo ϕ(n):

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

- Using the Extended Euclidean Algorithm, we find:

  $$13 \times 325 = 4225 \equiv 1 \bmod 352 \Longrightarrow d = 325.$$

- The final key pair is:
  Public Key (e, n) = (13, 391)
  Private Key (d, n) = (325, 391)

4.2. Encrypt plaintext (M)

- Assume the plaintext message M = 42.

- Compute the ciphertext C using the public key (e, n):
  $C \equiv M^e \bmod n$
  $C = 42^{13} \bmod 391$

- Breakdown of Computation:
  $42^2 \bmod 391 = 200$
  $42^4 \bmod 391 = 200^2 \bmod 391 = 118$
  $42^{13} \equiv 118 \times 200 \times 42 \bmod 391 = 145$

- Final ciphertext:
  $C = 145$ [5].

4.3. Decrypt ciphertext (C)

- To decrypt the ciphertext C = 145, use the private key (d, n) = (325, 391):
  $M \equiv C^d \bmod n$
  $M = 145^{325} \bmod 391 = 42$

- Applying modular exponentiation, we obtain:
  $M = 42$

- Decryption is successful, and the original message is recovered [7].

4.4. Security Considerations

- Prime Size Importance: It is crucial to reiterate that the small prime numbers used in this example (17, 23) are for illustrative purposes only. Real-world RSA implementations employ much larger primes (e.g., 2048-bit, 4096-bit) to ensure resistance against factorization attacks [9,15].
- Hybrid Encryption: RSA is computationally intensive for encrypting large amounts of data. In practice, RSA is often used in conjunction with symmetric encryption algorithms (e.g., AES + RSA) to achieve a balance between security and efficiency. RSA is used for key exchange, while AES is used for data encryption [12].
- Quantum Computing Threat: The potential threat posed by quantum computers to RSA security due to Shor's algorithm is a significant concern. The development of post-quantum cryptography is essential to address this vulnerability [8].

5. Discussion

This section provides an in-depth discussion of the RSA algorithm, examining its historical context, security implications, performance trade-offs, and real-world applications.

5.1. Historical Progression of key Sizes

The evolution of RSA key sizes is a critical aspect of the algorithm's ongoing adaptation to evolving security threats. As computational power has increased and cryptographic research has advanced, it has become necessary to progressively increase key lengths to maintain adequate security levels

- 512-bit Keys (Pre-2000s): Early RSA implementations commonly employed 512-bit keys, which were considered sufficient to withstand the attacks prevalent at the time. However, the development of attacks like the General Number Field Sieve (GNFS) rendered these keys vulnerable, and they were deprecated by 2013 [1].
- 1024-bit Keys (2000s–2013): Throughout the early 2000s, 1024-bit RSA keys became the industry standard, striking a balance between security and computational efficiency. Nevertheless, the rapid increase in computational capabilities made these keys insufficient by 2013, prompting cryptographic standards to recommend the adoption of larger key sizes [9,14].
- 2048-bit Keys (2013–Present): In response to recommendations from organizations like the National Institute of Standards and Technology (NIST), 2048-bit RSA keys became the new baseline for cryptographic security, providing approximately 112-bit equivalent symmetric security. It is projected that this key length will remain secure until at least 2030 [10,11].
- 4096-bit Keys (Emerging Trend): With the continued advancement of computing technology, 4096-bit keys are increasingly being adopted for applications requiring the highest levels of security. These keys offer approximately 150-bit equivalent symmetric security but introduce increased computational overhead, affecting encryption and decryption speeds [11].

5.2. Security Implications

5.2.1. Factorization Resistance and Quantum Threats

A fundamental strength of RSA lies in its reliance on the computational difficulty of factoring large composite numbers. However, the increasing availability of powerful computing resources and the potential emergence of quantum computers pose significant threats to RSA's security [8,15].

Table 1: Provides A Comparative Analysis of The Feasibility of Classical and Quantum Attacks Against Different RSA Key Sizes

Key Size	**Classical Attack Feasibility**	**Quantum Attack Risk (Shor's Algorithm) [8]**
1024-bit	Broken With $1M cluster in 2-3 years	Vulnerable
2048-bit	Requires ~$10B and 1B years	At risk post-2030
4096-bit	Infeasible with current methods	Resistant to near-term quantum threats

Table 1: RSA Key Size Security Analysis

5.3. Identified Vulnerabilities

- Key Size Limitation: The need to increase RSA key sizes to maintain security in the face of growing computing power leads to performance inefficiencies, particularly in real-time applications [13].
- Side-Channel Attacks: RSA implementations are vulnerable to side-channel attacks, such as timing attacks, power analysis, and fault attacks, which can be exploited to extract private keys by analyzing physical characteristics of the cryptographic operations [12].
- Hybrid Cryptography Risks: Many systems employ hybrid encryption schemes (e.g., RSA for key exchange and AES for bulk encryption) to achieve a balance between performance and security. However, if the key exchange mechanism is compromised, the security of the entire system is jeopardized [14].

5.4. Performance Trade-offs

RSA encryption and decryption involve modular exponentiation, a computationally intensive operation, especially with larger key sizes. Consequently, there are inherent trade-offs between encryption speed, memory consumption, and the level of security provided by RSA [11,13].

Table II summarizes these trade-offs:

Key Size	Encryption/ Decryption Speed	Memory/ Bandwidth	Use Cases
512-bit	Fast (~ms)	Low	Obsolete [5]
1024-bit	Moderate	Moderate	Legacy systems (Phased out) [9]
2048-bit	2-3x slower than 1024-bit	High	Modern SSL/TLS, digital signatures [11]
4096-bit	10x slower than 2048-bit	Very High	Long-term storage, high security sectors [5]

Table 2: RSA Key Size vs. Performance

5.5. Computational Overhead Considerations

- RSA's computational complexity is $O(n^3)$, indicating that decryption time increases exponentially with the key size. For example, decrypting a message with a 4096-bit key takes approximately four times longer than decrypting the same message with a 2048-bit key [14].
- Hybrid cryptographic approaches, which combine RSA for key exchange with symmetric encryption algorithms like AES for data encryption, are commonly used to mitigate the performance bottlenecks associated with RSA [13].

5.6. Standards and Cryptographic Recommendations

5.6.1. NIST Guidelines

The National Institute of Standards and Technology (NIST) plays a crucial role in providing guidelines and recommendations for cryptographic security, including RSA key sizes [10]:

- 2013: NIST mandated the use of 2048-bit keys for federal government systems [11].
- 2015: NIST disallowed security levels below 112-bit, which is equivalent to the security provided by 2048-bit RSA keys [10].

5.7. Real-time Implementation of RSA in Secure Communication

RSA is widely implemented in various real-world applications, particularly in secure messaging systems and real-time communication platforms [7].

- WebSockets-Based Secure Messaging: RSA has been integrated with technologies like Node.js, Express.js, and Socket.io to enable secure real-time communication. Event-driven processing techniques are employed to optimize performance and minimize latency in these systems [6].
- Dynamic Key Management: In secure communication systems, each user typically generates an RSA key pair upon joining the system, with the private key stored locally to ensure end-to-end encryption. Public keys are then dynamically shared to facilitate secure message transmission [7].
- Secure Key Transfer Protocol: RSA-based systems often implement protocols that require mutual approval of public key exchanges before initiating encrypted communication. This approach helps to prevent man-in-the-middle (MITM) attacks [15].
- Risk Mitigation Strategies: To enhance security, RSA-based systems are designed to never transmit private keys over the network, mitigating the risk of replay attacks and key exposure [16].
- Performance Optimization: Hybrid encryption schemes, such as combining RSA for key exchange and AES for bulk data encryption, are employed to optimize performance. Additionally, WebSocket persistence is used to maintain session integrity and reduce the overhead associated with repeated key exchanges [13].

5.8. Future of RSA and Post-Quantum Cryptography

The RSA algorithm has been a cornerstone of cryptographic security, providing robust encryption and authentication mechanisms for a wide range of applications. However, the rapid advancements in quantum computing pose significant challenges to RSA's continued security [3,12].

5.9. Quantum Computing and Its Impact on RSA

Quantum computers, leveraging principles of quantum mechanics, have the potential to break RSA encryption by efficiently solving mathematical problems that are infeasible for classical computers. Shor's Algorithm, a quantum algorithm developed by Peter Shor, can factorize large numbers exponentially faster than the bestknown classical algorithms. Since the security of RSA relies on the difficulty of prime factorization, a sufficiently powerful quantum computer could render RSA encryption obsolete [6,14].

Table III illustrates the projected impact of quantum computing on the security of different RSA key sizes [8,14]:

RSA Key Size	Classical Security	Quantum Security (Shor's Algorithm)
1024-bit	Insecure (breakable in months)	Easily broken
2048-bit	Secure until 2030	At risk post-2030
4096-bit	Secure against classical attacks	Temporary resistance, but breakable in the long term

Table 3: Projected Impact of Quantum Computing on RSA Key Sizes

5.10. Post-Quantum Cryptographic Alternatives

To address the vulnerabilities presented by quantum computing, researchers are actively developing post-quantum cryptographic (PQC) algorithms that are designed to remain secure even against quantum attacks [4,10,16]. Several promising PQC alternatives are being explored:

- Lattice-Based Cryptography: This approach relies on the hardness of solving lattice problems, which are believed to be resistant to quantum attacks. Examples of lattice-based cryptographic algorithms include the NTRU encryption algorithm and Learning-With-Errors (LWE)-based schemes [7].
- Hash-Based Cryptography: Hash-based cryptography utilizes cryptographic hash functions to construct secure digital signatures that maintain quantum resistance. Examples of hash-based signature schemes include the XMSS (eXtended Merkle Signature Scheme) and SPHINCS+ [9].
- Code-Based Cryptography: This method employs error-correcting codes to provide security. The McEliece cryptosystem is a notable example of code-based cryptography [11].
- Multivariate Polynomial Cryptography: This approach uses multivariate polynomial equations to create public-key cryptosystems that are designed to resist quantum attacks. The Rainbow Signature Scheme is an example of multivariate polynomial cryptography [13,15].
- Table IV provides a comparative analysis of these post-quantum cryptographic alternatives [2,8].

Cryptographic Approach	Security Basis	Quantum Resistance	Adoption Readiness
Lattice-Based	Hardness of lattice problems	High	Actively researched
Hash-Based	Cryptographic hash functions	High	Ready for implementation
Code-Based	Error-correcting codes	Moderate	Still being evaluated
Multivariate Polynomial	Multivariate equations	Moderate	Experimental stage

Table 4: Comparative Analysis of Post-Quantum Cryptographic Alternatives

5.11. The Transition to Post-Quantum Security

Recognizing the need to transition away from RSA and other quantum-vulnerable algorithms, organizations and governments worldwide are actively engaged in research and standardization efforts for post-quantum cryptography [1,12]. The National Institute of Standards and Technology (NIST) is leading a significant initiative to standardize quantum-resistant algorithms that can replace existing vulnerable encryption methods [10,14]. The transition to post-quantum cryptographic protocols will necessitate updates across various aspects of digital infrastructure, including software, hardware, and security frameworks. Key steps in this transition include:

- Algorithm Standardization: NIST's Post-Quantum Cryptography Standardization Project aims to establish secure and efficient PQC algorithms by 2024-2025 [6].
- Software and Hardware Adaptation: Updating encryption libraries, protocols (TLS, SSH, VPNs), and hardware security modules (HSMs) to support PQC [3].
- Hybrid Cryptography Approaches: Implementing transitional systems where both RSA and PQC algorithms coexist to ensure backward compatibility [5,8].
- Risk Mitigation Strategies: Organizations must proactively conduct security audits and risk assessments to identify potential vulnerabilities in their cryptographic infrastructure and plan for the adoption of PQC [9,16].

6. Conclusion

The RSA cryptosystem remains a fundamental component of modern cryptographic security, ensuring secure communication, authentication, and data integrity across digital platforms [3,8]. This paper has provided an in-depth exploration of RSA, covering its mathematical foundations, key generation process, encryption and decryption mechanisms, security considerations, and performance trade-offs. A numerical example demonstrated RSA's practical implementation, emphasizing its operational efficiency and computational complexity [6]. Additionally, the security analysis highlighted the significance of key size in determining cryptographic strength, reinforcing the need for larger key sizes to counter evolving cyber threats [9].

While 2048-bit and 4096-bit RSA keys offer enhanced security, they introduce computational overhead, necessitating hybrid encryption approaches such as AES-RSA integration for improved efficiency [12]. Furthermore, the rise of quantum computing presents a formidable challenge to RSA's long-term viability. The potential application of Shor's algorithm to factor large integers efficiently underscores the urgency of transitioning to post-quantum cryptographic solutions [14]. Ongoing research aims to develop quantum-resistant algorithms capable of securing data in the post-quantum era. Despite these challenges, RSA continues to play a crucial role in securing web applications and other digital infrastructures [7]. Future research should focus on enhancing RSA's scalability in large-scale distributed environments, optimizing its real-time performance, and integrating post-quantum cryptographic methods to address emerging threats.

References

1. Saranya, V. (2014). Vasumathi,"A study on RSA algorithm for cryptography,". *International journal of Computer Science and Information technologies, 5*(4), 5708-5709.
2. Phatangare, S., Jadhav, S., Kawane, S., Holkar, P., & Gaikwad, P. (2024). Multi-Level Encryption System using AES and RSA Algorithms. *International Journal for Research in Applied Science and Engineering Technology, 12*(5), 4043-4051.
3. Saini, A., & Vandana, D. A. (2022). Study on Modified RSA Algorithm in Network Security. *International Research Journal of Modernization in Engineering Technology Science, 4*(4), 1461-1465.
4. Chaitanya, J., & Solanki, A. (2024). DATA ENCYPTION AND DATA DECRYPTION BY AN IMPROVED RSA ALGORITHM. *Global Journal of Advanced Engineering Technologies and Sciences, 11*(12), 1-4.
5. Ranasinghe, R., Chathurangi, M., & Athukorala, P. (2024). A novel improvement in RSA algorithm. *Journal of Discrete Mathematical Sciences and Cryptography, 27*(1), 143-150.
6. Sihotang, H. T., Efendi, S., Zamzami, E. M., & Mawengkang, H. (2020, November). Design and implementation of Rivest Shamir Adleman's (RSA) cryptography algorithm in text file data security. In *Journal of Physics: Conference Series* (Vol. 1641, No. 1, p. 012042). IOP Publishing.
7. 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.
8. Zhang, C., Liang, Y., Tavares, A., Wang, L., Gomes, T., & Pinto, S. (2024). An improved public key cryptographic algorithm based on chebyshev polynomials and RSA. *Symmetry, 16*(3), 263.
9. Preetha, M., & Nithya, M. (2013). A study and performance analysis of RSA algorithm. *International Journal of Computer Science and Mobile Computing, 2*(6), 126-139.
10. Jaju, S. A., & Chowhan, S. S. (2015, October). A Modified RSA algorithm to enhance security for digital signature. In 2015 international conference and workshop on computing and communication (IEMCON) (pp. 1-5). IEEE.
11. Obaid, T. S. (2020). Study a public key in RSA algorithm. *European Journal of Engineering and Technology Research, 5*(4), 395-398.
12. Anwar, M., Ismail, M., & Bahig, H. M. (2024). Cryptoanalysis of RSA variants with special structure of RSA primes. *arXiv preprint arXiv:2403.06184*.
13. Pelofske, E. (2024). An Efficient All-to-All GCD Algorithm for Low Entropy RSA Key Factorization. *arXiv preprint arXiv:2405.03166*.
14. Mobin, M. A., & Kamrujjaman, M. (2024). Cryptanalysis of RSA Cryptosystem: Prime Factorization using Genetic Algorithm. *arXiv preprint arXiv:2407.05944*.
15. Bahig, H. M., Mahdi, M. A., Alutaibi, K. A., AlGhadhban, A., & Bahig, H. M. (2020). Performance analysis of fermat factorization algorithms. *International Journal of Advanced Computer Science and Applications, 11*(12).
16. P. DaoThi, H. BuiTa, and V. Thai, "Application of Encryption and Decryption in Digital Signatures Using RSA Algorithm," *International Journal of Research in Engineering and Science* (*IJRES*), vol. 12, no. 7, pp. 50–54, July 2024.