Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org

Volume 25, Issues 6, pp. 4669–4682, DOI 10.12694/scpe.v25i6.3386

AN ENHANCED RSA ALGORITHM TO COUNTER REPETITIVE CIPHERTEXT THREATS EMPOWERING USER-CENTRIC SECURITY

VISHAL BALANI*, CHAITANYA KHARYA †, SHIV NARESH SHIVHARE‡, AND THIPENDRA P. SINGH §

Abstract. The technology-driven modern age emphasizes the security and privacy of communication. Through this paper, we delve deep into the need for user-centric security within cloud-based environments. The need for enhancement in encryption arises due to the increasing cases of data breaches and insider threats in cloud-based environments recently. The focus is laid upon the use of RSA encryption, end-to-end encryption, and anonymous messaging to address security-based concerns. The primary focus of this research is to develop a comprehensive security system to ensure the confidentiality and authenticity of text-based messages shared in the cloud. The proposed improved RSA algorithm, as suggested, incorporates three prime numbers in the key generation process. To address repetitive ciphertext, the proposed algorithm involves adding the index of each character in the plaintext string to the character's integer value before encryption. Conversely, during decryption, the same index is subtracted. This proposed algorithm has been utilized in a practical scenario, specifically in the implementation of a chat application. This paper presents a proof-of-concept for the proposed enhanced version of the RSA algorithm, accompanied by a thorough comparison and analysis of computational times across various bit lengths. Increase in data security at a cost of minor increase in computation time was observed through this research.

Key words: Enhanced RSA Algorithm, End-to-End Encryption (E2EE), Data Privacy, Confidentiality, Privacy Protection.

1. Introduction. In the ever-evolving landscape of digital communication, ensuring the security and privacy of sensitive information has become paramount. User authentication plays a crucial role in ensuring the security of a system [29, 30]. One of the cornerstones of secure communication is the use of cryptographic techniques, which have witnessed significant advancements in recent years [20]. The RSA cryptosystem is one of the most generally utilized public-key cryptosystems. RSA algorithm utilizes mathematical operations, including modular multiplication and exponentiation, making it an algorithm suitable for encryption and decryption using asymmetric key pairs[16]. In this process, two keys are utilized: one public key and one private key. Producing these keys includes complex calculations with large prime numbers, and the security of the RSA cryptosystem depends on the difficulty of factoring these large prime numbers.

Though RSA encryption and decryption are acknowledged for their security, they come with inherent performance limitations. Techniques such as fast modular multiplication, fast modular exponentiation, and the use of the Chinese remainder theorem (CRT) [1] have been developed to accelerate RSA operations, but they still lag behind symmetric-key encryption algorithms in terms of speed. Consequently, RSA encryption is often employed for secure key transport and the encryption of smaller data elements [2].

Traditional multiplication methods have a time complexity proportional to the square of the operand bit length. However, algorithms such as Karatsuba and the Toom-Cook method [5], which exploit recursive and divide-and-conquer strategies, respectively, enable faster modular multiplication by reducing the number of basic multiplication operations required. Exponentiation involves repeated modular multiplications, resulting in a time complexity proportional to the exponent's bit length. To expedite this process, techniques like square-and-multiply and Montgomery exponentiation provide more efficient algorithms. Instead of performing modular

*School of Computer Science Engineering and Technology, Bennett University Greater Noida, Uttar Pradesh-201310, India (vishal.balani@outlook.com)

†School of Computer Science Engineering and Technology, Bennett University Greater Noida, Uttar Pradesh-201310, India (kharyachaitanya@gmail.com)

‡School of Computer Science Engineering and Technology, Bennett University Greater Noida, Uttar Pradesh-201310, India (Corresponding Author, shiv827@gmail.com)

§School of Computer Science Engineering and Technology, Bennett University Greater Noida, Uttar Pradesh-201310, India (thipendra@gmail.com)

exponentiation with the modulus N directly, CRT allows breaking the computation into smaller, independent components, each modulo a prime factor of N. By performing these computations separately and combining the results using the CRT, the overall execution time is significantly reduced.

Numerous research projects have been conducted in an effort to modify the RSA cryptosystem [17]. These included both hardware and software solutions, as well as solutions optimized for specific platforms such as .NET and Java. Hardware implementations include RNS Montgomery multiplication, use of the TMS320C54X signal processor, and designing custom hardware circuits using field-programmable gate arrays (FPGA) to perform the mathematical computations involved. Software implementations include salting, use of multiple prime numbers for key generation, use of mixed-base representation for depicting encoded messages, use of randomized exponentiation, RSA with elliptic curve cryptography, etc. These techniques have been observed to enhance security in real-world environments but may compromise the time needed for computation. Ongoing investigations are centered on addressing the diverse vulnerabilities present in the original RSA algorithm, aiming to enhance its resistance to known deterministic encryption. In this context, we propose an enhanced RSA algorithm to overcome such types of issues. The major contribution of this paper is threefold:

1. The proposed algorithm prevents repetitive ciphertext threats, such as frequency-based attacks and dictionary attacks, which are significant vulnerabilities in traditional RSA by adding a buffer to each character in the process of encryption.
2. The performance of the proposed algorithm is compared with that of traditional RSA, especially for the total computation time required for varying key bit lengths.
3. The implementation and integration of the proposed enhanced algorithm with a chat application and evaluated its calculation consistency.

2. Related Work. In present-day cryptography, the journey toward strengthening safety efforts while enhancing computational productivity has prompted different investigations and variations of the conventional RSA calculation. The purpose of the enhanced versions of RSA is to improve performance and security while demonstrating novel approaches to the cryptographic landscape. In this direction, Nivetha et al. [11] modified the RSA algorithm with multiple primes and four indivisible numbers inside the encryption system to reinforce the modulus size, apparently increasing security by muddling factorization processes. The expanded intricacy of key age and the board in frameworks utilizing numerous primes presents critical difficulties in true organization, frequently offsetting the potential security upgrades. The modified RSA with Mixed-Base Representation (MBR) computation streamlines execution by involving a mixed-base depiction for encoded messages [12]. Despite the fact that efforts have been made to reduce the time it takes to encrypt and decrypt as compared to traditional RSA, there are still concerns about the chance of safety degradation brought about by this streamlining. The focus of both research projects is on improving performance without jeopardizing security integrity.

The Modified RSA with Randomized Exponentiation (MRE) estimation carries randomized exponentiation into the encryption cooperation [14]. Even though this complexity is intended to deter adversaries from attempting to separate sensitive data through known-plaintext situations, it results in computational overhead, particularly in asset-obligated circumstances. Investigations consolidating RSA with elliptic curve cryptography (ECC) look for improved security while diminishing key sizes [13]. Essentially, coordinating two particular cryptographic frameworks presents significant execution intricacies and raises interoperability concerns among the RSA and ECC conventions. Kapoor et al. [18] proposed a modified RSA method that was based on multiple public keys and n prime integers. This method aimed to provide efficiency and enhances data sharing security across networks. However, the authors observed that as the prime numbers increase, key generation time also increases exponentially.

Moreover, Anagaw and Vuda [15] efficiently implemented of the RSA algorithm using two public key pairs and mathematical logic. Separately delivering two public keys prevents attackers from learning about the key and the message. A similar method was proposed by Jahan et al. [19] that utilizes two public key pairs and mathematical logic instead of delivering one public key directly. This approach aims to enhance security by making it more difficult for attackers to obtain the private key. Imam et al. [20] modified the RSA algorithm for encryption using two public keys derived from four prime integers. This method aims to enhance security by using dual modulus to eliminate flaws and improves the system's security. Furthermore, Mezher [21] devised a method that employs multiple public and private keys, making the algorithm more secure and immune to brute-force attacks. This modification technique takes nearly nine times more time to break the traditional method when using alternative key sizes. The use of modified RSA with salt in cloud data encryption aims to enhance security by introducing randomness. It addressed the fundamental aspects of cloud security, focusing on data encryption and its pivotal role in safeguarding data within the cloud [17]. While investigating the aforementioned modification in the traditional RSA algorithm, we observed that encryption has certain limitations, especially in the context of cloud computing, which are highlighted as follows:

- Performance Overhead: Traditional RSA encryption can introduce performance overhead due to its computational complexity, especially when dealing with large volumes of data in cloud environments.
- Key Management: The management of encryption keys in RSA encryption can be challenging, particularly in shared cloud environments where multiple users and organizations coexist. Ensuring secure key exchange and management is crucial for maintaining data confidentiality and integrity.
- Vulnerability to Attacks: Traditional RSA encryption may be vulnerable to certain attacks, such as brute-force attacks, especially if the encryption keys are not sufficiently random or complex. This vulnerability can pose a significant risk to data security within the cloud.

To address these drawbacks and enhance cloud data security, the use of modified RSA with salting was proposed. The modified approach incorporated salting (password-based encryption schemes) to add an extra layer of randomness and complexity to the encryption process, making it more resilient against brute-force attacks and other security threats [17]. A few hardware implementations of the modified RSA algorithms were introduced by several researchers recently. The details are as follows:

- Use of RNS Montgomery multiplication for implementing RSA encryption involves converting the large integers used in RSA encryption into a residue number system (RNS) and performing modular multiplication using the Montgomery algorithm. This approach can improve the efficiency of RSA encryption by reducing the number of operations required for modular multiplication [22, 23, 27].
- Use of Texas Instruments TMS320C54X signal processors for implementing RSA encryption involves optimizing the RSA algorithm for the architecture of the TMS320C54X family of signal processors, which can improve the performance of RSA encryption in hardware environments [22, 23].
- VLSI design using FPGA for implementing RSA encryption involves designing custom hardware circuits using field-programmable gate arrays (FPGAs) to perform the modular arithmetic operations required for RSA encryption. This approach can improve the performance of RSA encryption in hardware and reduce power consumption [22, 24].

On the other hand, various software implementations of RSA encryption have been proposed, including .NET [25, 26] and Java [28]. These software implementations involve optimizing the RSA algorithm for specific software platforms, which can improve the performance of RSA encryption in software [22]. Bonde and Bhadade [22] provide insights into the advantages and limitations of each implementation method, highlighting the importance of selecting the appropriate implementation method based on the specific requirements of the application. For example, hardware-based approaches such as VLSI design using FPGA and Texas Instruments TMS320C54X signal processors can improve the performance of RSA encryption in hardware, while software-based approaches such as .NET and Java can improve the performance of RSA encryption in software. Topics closely related to modified RSA algorithms have been the subject of recent research, which has made a significant contribution to the field of cryptography. Encrypted chat applications using RSA encryption plans [7, 8, 9, 10] feature the pragmatic parts of cryptography methods in real-time applications. In addition, comprehensive literature reviews on big data management techniques in the Internet of Things (IoT) [6] provide insights into managing massive amounts of data generated by interconnected devices, highlight obstacles, and suggest directions for future research.

Thus, the range of enhanced RSA algorithms offers novel strategies for enhancing performance or strengthening security. However, their practical implementation faces challenges due to complexities, potential vulnerabilities, and interoperability concerns. Comprehensive evaluations and standardization efforts are imperative to advance cryptography techniques. Table 2.1 summarizes and highlights several recent and relevant research work based on the RSA algorithm.

Table 2.1: Description of the referenced research papers in Related Work Section

Author(s)	Methodology	Advantages	Limitations
Nivetha et al. [11]	Modified RSA with Multiple primes, 4 Keys	Increased complexity of modulus factorization	Computational overhead
Guo and Zhang [12]	Mixed-Base representation for encoded messages	Increased decryption time in brute-force attacks	Computational overhead
Wang and Tang [13]	RSA with Elliptic Curve Cryptography	Improved security with smaller key sizes	Complex execution in real world use cases and interoperability concerns
Lee and Kim [14]	Modified RSA with Randomized Exponentiation	Increased decryption time in known-plaintext attacks	Computational overhead
Anagaw and Vuda [15]	Modified RSA with 2 public keys	Prevents attackers from decoding key and message	Minor security enhancement
Kaur and Aarju [17]	Modified RSA with Salt	Enhance security due to randomness	Computational overhead
Kapoor [18]	Modified RSA with n primes, multiple public keys	Enhanced data sharing security across networks	Exponential computation overhead in key generation
Jahan et al. [19]	Modified RSA with 2 public keys	Prevents attackers from decoding key and message	Minor security enhancement
Imam et al. [20]	Modified RSA with 2 public keys, 4 primes	Enhance Security using dual modulus	Computational overhead
Mezher [21]	Modified RSA with multiple public and private keys	Immune to brute-force attacks	Computational overhead
Bonde and Bhadade [22]	VLSI design using FPGA for implementing RSA	Improved performance in hardware and reduced power consumption	Complex Hardware and platform dependent
Nozaki et al. [23]	RSA using RNS Montgomery Multiplication	Computational Efficient	Vulnerable to known-plaintext attacks
Markovic et al. [24]	RSA optimization for TMS320C54X Signal processor	Improved performance in hardware environments	Hardware dependent
Kumar and Chaudhary [25]	Modified RSA using n primes and bit stuffing	Enhanced security due to randomness	Computational overhead
Sharma et al. [28]	RSA using modified Subset Sum cryptosystem	Resistant against modulus factorization, brute-force and Shamir attacks	Computational overhead

3. Proposed Methodology. In existing RSA cryposystem, a character generates the same ciphertext each time it is encrypted in a message. This leads to vulnerability and the threat of frequency-based attacks, which can compromise the application. The enhanced RSA discussed below deals with this vulnerability, hence enhancing security while minimizing computation time differences as compared to traditional RSA. The following steps delineate the methods employed to modify the traditional RSA encryption and decryption algorithms.

3.1. Selection of Prime Numbers (Key Generation).

- *Traditional RSA:* The conventional RSA algorithm utilizes two prime numbers, p and q, to generate the public (e, n) and private (d, n) keys. However, to fortify the encryption system, larger prime numbers significantly contribute to the increased complexity of $n = p \times q$.
- *Enhanced RSA:* To substantially increase the value of n, the modification incorporates three large prime numbers. Hence, the value of n becomes $n = p \times q \times r$. While this increases computational complexity, it significantly fortifies the encryption. The optimal balance in the number of prime numbers used to

generate n is under study, aiming for lower computational complexity and higher protection against common brute-force attacks against RSA.

3.2. Index Increment in Each Character Before Encryption.

- *Traditional RSA:* Plaintext is encrypted using the equation:

$$C = (m^e) \mod n \tag{3.1}$$

The vulnerability in this method arises from the generation of the same ciphertext for repeated characters in the message, making RSA encryption susceptible to attacks.

- *Enhanced RSA:* To mitigate this vulnerability, the enhanced RSA method increments each character by its index in the original message before encryption, transforming the encryption equation to:

$$C = ((m + index)^e) \mod n \tag{3.2}$$

This slight modification significantly enhances the security of the RSA encryption algorithm, making it arduous for attackers to decrypt the message, even with access to the private key.

3.3. Index Decrement in Each Character After Decryption.

- *Traditional RSA:* Ciphertext is decrypted using the equation:

$$m = (C^d) \mod n \tag{3.3}$$

If the private key is leaked, it may lead to the release of sensitive information in the message.

- *Enhanced RSA:* To ensure seamless decryption of ciphertext encrypted using enhanced RSA, the inverse of the steps performed during encryption are applied to the decryption algorithm. The decryption algorithm is as follows:

$$m = ((C^d) \mod n) - index \tag{3.4}$$

In the event that the private key is leaked, attackers will be unable to obtain sensible information from the decrypted text, producing gibberish as the message. The application of the Chinese Remainder Theorem in the modified decryption algorithm leads to reduced computation time.

The following are the detailed implementation steps for the changes explained above:

1. *Choose 3 Prime Numbers (p, q, and r):* Generate three distinct, large prime numbers: p, q, and r. Random numbers of bits (ranging from 45 to 52 due to system computation limitations) are used to generate large prime numbers.
2. *Compute Modulus (n):* After selecting p, q, and r, multiply them to obtain the modulus, n.

$$n = p \times q \times r \tag{3.5}$$

This modulus is a fundamental component of both the public and private keys, serving as the basis for encryption and decryption.

3. *Computation of Euler's Totient Function ($\phi(n)$):* Compute the Euler's Totient Function using the prime numbers, p, q, and r, required to derive the value of d, a part of the private key.

$$\phi(n) = (p - 1) \times (q - 1) \times (r - 1) \tag{3.6}$$

The totient function is crucial in selecting the public exponent to ensure the existence of a unique modular multiplicative inverse in the decryption process.

4. *Select Public Key Exponent (e):* Choose the public key exponent denoted as e. It should be a positive integer that is relatively prime to $\phi(n)$ (i.e., $\text{GCD}(e, \phi(n)) = 1$).
5. *Generate Public Key (e, n):* The formation of the public key comprises the concatenation of 'e' and 'n', serving the purpose of encrypting plaintext messages. The encryption process involves incrementing each plaintext character 'm' by its corresponding index within the message. The equation representing this process is:

$$C = ((m + index)^e) \mod n \tag{3.7}$$

6. *Compute Private Key (d):* Obtain the private key exponent, d, by finding an integer that satisfies the equation:

$$(d \times e) \mod \phi(n) = 1 \tag{3.8}$$

7. *Generate Private Key (d, n):* The private key is formulated by combining 'd' and 'n'. The decryption process involves decrementing the index of the decrypted text to obtain the original message. The decryption formula can be expressed as follows:

$$m = (C^d \mod n) - \text{index} \tag{3.9}$$

The key pairs, the public key (e, n) and the private key (d, n), are the basis of RSA encryption. The public key allows anyone to encrypt messages, while only the holder of the private key can decrypt them. The security of RSA relies on the computational complexity of factoring the modulus n into its prime factors p, q, and r. Our proposed system utilizes the enhanced RSA algorithm for secure communication within the chat application. On the sender's side, text messages are encrypted using the recipient's public key, retrieved from the cloud database. This public key encryption ensures that the message remains unreadable while stored on the cloud server, mitigating potential security risks during data transport. The encrypted message is then transmitted to the receiver, who possesses the corresponding private key stored locally on their device. This private key is used for decryption of the message using the enhanced RSA algorithm, enabling retrieval of the original content.

4. Experimental Results. In this segment, we present the outcomes we got from the assessment of the enhanced RSA calculation. Centered around surveying the calculation's adequacy by analyzing the distinctions in encryption results compared with the conventional RSA approach. Furthermore, we investigate the computational times expected for encryption, decryption, and key decryption processes. The essential goal is to exhibit the effect of the adjustments on encryption quality and computational proficiency.

4.1. Algorithm for Enhanced RSA. To demonstrate the effectiveness of the proposed enhanced RSA we have successfully implemented it in chat application. For detailed understanding of the algorithm, please refer to Algorithm 1 where a detailed pseudo-algorithm is provided.

4.2. Proof of Algorithm. Sample Text: "HELLO"

Step 1: Prime Number Generation. Generate three large prime numbers, p, q, and r, each of length bits: $p = 61$, $q = 53$, $r = 67$ (arbitrary values for demonstration purposes)

Step 2: Modulus Calculation. $n = p \times q \times r = 61 \times 53 \times 67 = 216611$

Step 3: Euler's Totient Function Calculation. $\phi(n) = (p-1) \times (q-1) \times (r-1) = 60 \times 52 \times 66 = 205920$

Step 4: Public Key Generation. Choose an integer e such that: $1 < e < \phi(n)$, $\gcd(e, \phi(n)) = 1$. Let $e = 65537$, the public key is represented as $< 65537, 216611 >$

Step 5: Private Key Calculation. Calculate the private key component d using the equation: $(d \times e) \mod \phi(n) = 1$ Let $d = 187,217$ The private key is represented as $< 40193, 216611 >$

Step 6: Encryption Process. Encrypt the message "HELLO":

$$C = (m + \text{index})^e \mod n$$

Assuming ASCII values for each character and index are starting from 0:

$$C_H = (72 + 0)^{65537} \mod 216611 = 112922$$

$$C_E = (69 + 1)^{65537} \mod 216611 = 61752$$

$$C_L = (76 + 2)^{65537} \mod 216611 = 151883$$

$$C_L = (76 + 3)^{65537} \mod 216611 = 140326$$

$$C_O = (79 + 4)^{65537} \mod 216611 = 57641$$

Encrypted message: $< 112922, 61752, 151883, 140326, 57641 >$

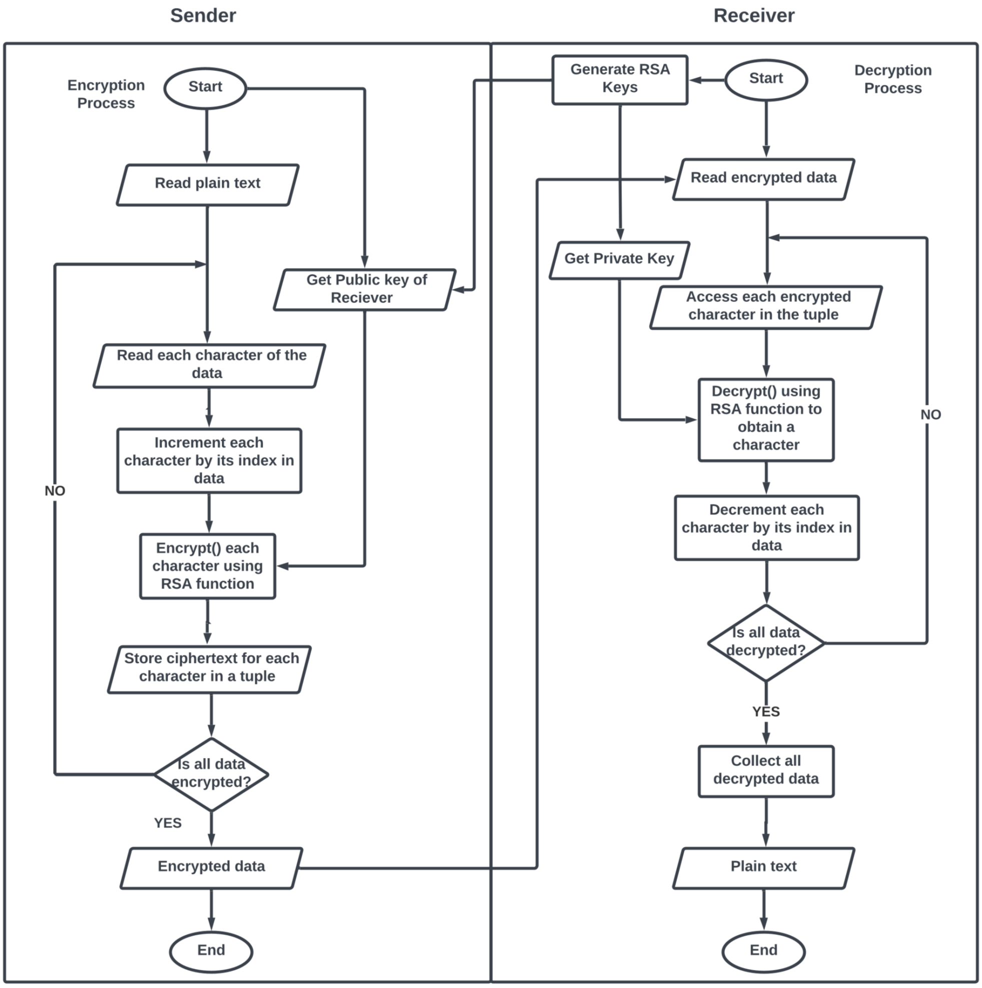

Fig. 3.1: Flow diagram illustrating the Encryption and Decryption processes in Enhanced RSA

Step 7: Decryption Process. Decrypt the ciphertext:

$$m = (C^d \mod n) - \text{index}$$

$$m_H = (112922^{40193} \mod 216611) - 0 = 72$$

$$m_E = (61752^{40193} \mod 216611) - 1 = 69$$

$$m_L = (151883^{40193} \mod 216611) - 2 = 76$$

$$m_L = (140326^{40193} \mod 216611) - 3 = 76$$

$$m_O = (57641^{40193} \mod 216611) - 4 = 79$$

Decrypted message: ”HELLO”

The algorithm successfully encrypted the message ”HELLO” and decrypted it back to the original text. This demonstrates the correctness of the RSA encryption and decryption processes for the given sample text.

Algorithm 1 The proposed Enhanced RSA Algorithm

```
procedure GeneratePrimes(bits)
    p ← GeneratePrime(bits)
    q ← GeneratePrime(bits)
    r ← GeneratePrime(bits)
end procedure
procedure ModulusCalculation(p, q, r)
    n ← p × q × r
end procedure
procedure EulersTotientFunction(p, q, r)
    φ_n ← (p − 1) × (q − 1) × (r − 1)
end procedure
procedure GeneratePublicKey(φ_n, n)
    e ← ChooseRandomInteger(1 < e < φ_n)
    while GCD(e, φ_n) ≠ 1:
        e ← ChooseRandomInteger(1 < e < φ_n)
    end while
    return (e, n)
end procedure
procedure Encrypt(m, e, n)
    C ← (m + index)^e mod n
end procedure
procedure PrivatekeyCalculation(e, φ_n)
    d ← ModInverse(e, φ_n)
    return (d, n)
end procedure
procedure Decrypt(c, d, n)
    m ← (c^d mod n) − index
end procedure
```

4.3. Analysis: Differences in Encryption. A comprehensive comparison was conducted between the encryption mechanisms of enhanced RSA and traditional RSA, implemented without the use of predefined cryptographic libraries. This analysis aimed to demonstrate the encryption disparities, particularly focusing on the handling of sample alphanumeric text such as "tt," "11," and "@@". 50-bit long prime numbers were used to generate the public and private keys in both traditional RSA and enhanced RSA.

The enhanced RSA encryption notably reveals that the repetition of a character generates distinct ciphertexts for each instance of the repeated character, enhancing its resistance against certain attacks.

From Table 4.1, we can observe that, in the process of encrypting plain text, any instances of repeated alphanumeric characters do not show repetitions in the resulting cipher text. This intriguing phenomenon suggests that the encryption algorithm employed successfully obfuscates patterns associated with repeated characters, adding an extra layer of complexity and security to the encrypted data.

4.4. Computational Time Analysis. In order to conduct a thorough analysis of the computational performance of both traditional RSA and enhanced RSA, an experimental setup was established. The computational performance was measured on an Apple MacBook Air equipped with an Apple M1 Chip, featuring an 8-core CPU and 256 GB storage. The analysis was conducted consistently in the same environment for varying bit lengths. The encryption process was applied to an alphanumeric text message of 2150 characters in length to gauge the encryption time for both algorithms. The aim was to evaluate and compare the computational efficiency of the encryption process.

To visually depict the variation in total execution times for different bit lengths, a graphical representation illustrating the relationship between the number of bits and total execution time is presented below.

Fig. 4.1 is plotted over the total computation time analysis between RSA and enhanced RSA. It was observed that enhanced RSA takes nearly the same amount of time as compared to RSA when the number of

Table 4.1: Comparative analysis of encrypted text using traditional RSA implementation against enhanced RSA

Message	Traditional RSA	Enhanced RSA
"tt"	[181844379188961449504 845017918, 181844379188961449504 845017918]	[180023598929568362674 11631437, 690275116192347687 130973236]
"11"	[220333787195340168864 840524335, 220333787195340168864 840524335]	[555164721889402328310 3454432, 513948457946383623519 8996570]
"@@"	[949363962348670891302 47824364, 949363962348670891302 47824364]	[461276186623075248004 7847462, 889961136969263822271 1964142]

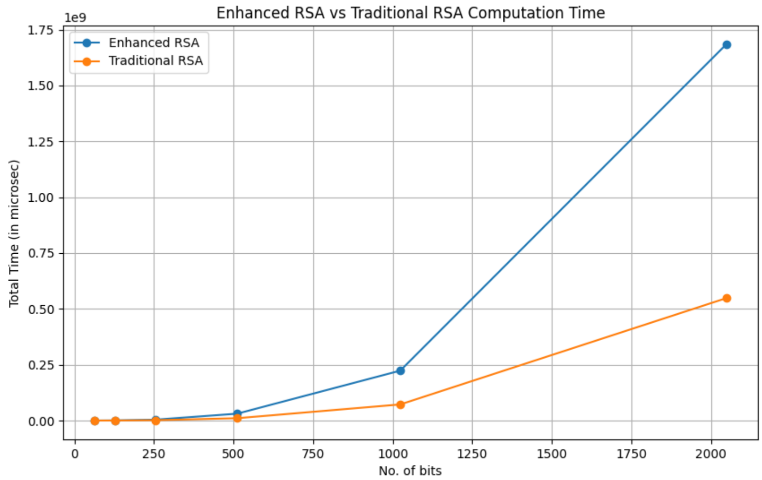

Fig. 4.1: Graph for comparative analysis of Total Execution Time for varying Bit Length of Keys

bits is less than 1024. But when the number of bits is over 1024, we see a drastic increase in computation time for enhanced RSA over traditional RSA. The key generation time, encryption time, and decryption time are measured in milliseconds.

Tables 4.2 and 4.3 provide a detailed analysis of the computational time for various bit lengths using both traditional RSA and enhanced RSA.

5. Real World Implementation. To validate the practical applicability of the enhanced RSA algorithm, we integrated it into a chat application developed using the Flutter SDK and utilizing Firebase for database functionality. The primary aim was to fortify the security of communication within the application while ensuring seamless usability.

5.1. Algorithm Integration. The adjusted RSA encryption procedure was flawlessly embedded inside the chat application's messaging functionality. This joining took into account the encryption and decryption of messages traded between clients, improving the general security of correspondence channels.

Table 4.2: Traditional RSA Computation Time Data

No. of bits	Prime 1	Prime 2	Key Generation Time (μs)	Encryption Time (μs)	Decryption Time (μs)	Total Time (μs)
64	15808577391726329629	14260400995176596597	385	7952	70754	79091
128	16280253345026454025 1595760900762820863	12363353070787334987 7909917480968910557	1951	12207	252890	267049
256	54915321636266941842 71320098812504412225 53033631960833420736 04232952887679949	68311923525130379525 39481945040563280386 58197328192474348705 21880628413936897	76224	26992	1690030	1793246
512	98356773358620330950 93005857861743182529 60197054195808932527 50743731019041094821 65450541927319703 06555870732303586737 66064905148767772 85579416228325331729	23421960722030594014 83227193370002797353 48912721639030161663 22273766147682444332 74630955985944367939 99121593656536479919 06746104265782978716 52975323707519	365257	72662	10109813	10547733
1024	10355998648560940987 27092965761783229667 20395857109009322685 61659118546269093253 85088799470693958881 77329807438352949386 60060463518448572052 76452305136483739217 57590342129819059764 37808011906700354151 94726256048803078161 22108815497814938714 62924911682402363706 05505558781893824713 86767184600957743663 7112030671	39148147267121377819 10286948821470529041 78158252062897094683 47314743134675306148 91593869246492872524 96091173158070672912 56583249341876986423 52650181325432493304 66741436724948960560 67863791743752062517 64230066013725583353 70355522320760408466 75130880056359126227 96300759283619254300 63129887075536565596 57669171	2828727	217127	69200421	72246275
2048	12563266109853096901 99916653303456054288 30509139373717777485 46384772136585991102 58671022761918320299 15276125355827220880 57194105006759205680 73624603565786467987 61534876686066076067 62924576040597505747 55457558959628261583 45723667311286588641 24743030762050179463 48847378760151827073 20885866172142137134 80960066029817857140 93574578628051666551 26659509067815295005 58181814171883616694 23832142054945279729 30659285531433751503 03670915018084158180 78232492595748126726 88336158766204060292 96794457629898452211 53452424974327557126 58974579713235175862 66491755660662823381 83847474806973184804 06473303734232627139 26481784020868677	90763046492552192551 57329503811314928313 34091228495335408774 49474919009579699525 84770786322442195699 80529915778506240260 59132540888865403117 79984473711383271992 62359764153428497501 11908292855266770526 06335733980909969844 49444013204608169716 97000225898240808434 94118538561055060744 49970650506877411735 97459125721859425160 44442539843032920826 20780605737002769854 30981333442061625374 72893541401747304441 93645979710675974605 42060367631145539167 72233811111118118302 69463112145077209176 26493666685841294323 39092568175680607842 54954952468574266196 47953070099623707704 68556144526290460555 92442632082209095287 220790518451	23165632	726211	524119375	548011218

Table 4.3: Enhanced RSA Computation Time Data

No. of bits	Prime 1	Prime 2	Prime 3	Key Generation Time (μs)	Encryption Time (μs)	Decryption Time (μs)	Total Time (μs)
64	10951074626344098541	9168151004544505097	2250792364417928861	1217	12204	148285	161706
128	25970864924525562003 2862220217273076359	29996469542284176256 9803344431246112881	13950203453132120552 7998564580848000771	2707	24711	705942	733361
256	72481285327005353993 74662317306967758874 65015687063623652353 50842066777319911	94444064820623919173 45609078643514822194 95824232929798252462 12261296197524349	46465316631097120841 06046925878081484752 66299596424889339420 98803130755444939	77715	42810	3977039	4097564
512	13247934241092971526 20453797975448190132 63225234721427513853 14183401981757094736 87836876776108493786 06889993694706923129 89686442751917263133 444136575853711	18728837919366120687 26293450120394268619 25925981009450358690 97051957272886870979 59367854682668720879 86157581391777329813 74422646569422304944 98376010335381	11606870030724497024 33665505726456389181 76560707306416264115 32810654165823635755 16339542782451752872 53065804089691050732 10952786552922870908 128126741214899	314509	146666	30507855	30969030
1024	99638769104583257053 90585286325058921923 35440940357303185774 02541274600676873562 12995362505851826491 02337003621392051724 39693860412757617797 26035738387070481210 11266668900033394669 58175467257712486494 32897275708355354183 42201870065714914226 70142495510591583507 98248834374558596979 99187456623300017272 897589	67517690780836936173 74286485902384283063 91110079395884759077 35894301492075920260 60821209527712737243 25106373938946831699 77377018397456431008 34441211666489538652 07581745806818137074 68314347988465332310 70895906874078208216 61574293804683660508 90166896486415700074 52687750240014746183 57103490720789495327 37244017	15046623347357680613 07794130623702155040 29219869549417470587 27852925025128085122 14864796289846292504 10202012456420659503 76678237272754573535 66845863746727485847 41552959420245712549 01555153693986033250 75930161305381341388 63275845190260875779 57090075352877977504 06445315609864336007 62370779605060462732 963691467	2715002	472767	220340486	223528255
2048	19631991671866701083 64582752465255119955 13813017470752829730 38502983025596559323 30807295752724975737 35948890915305729781 57269964843949614815 64858609917271707653 25057277925222402758 43079064200597759861 90439804902875939744 94760995909755595494 86894023062663886189 73742797465346055131 05714223972408051965 61223047520274153447 61537088074647250578 35466115298803023066 28548926762660092973 53430397217406810571 11297758815307065790 68666702416393023343 39617376633890886287 83455148793826595748 55670181255177666953 67294352922541701305 78909097431456438968 93611070915536243802 24569056227110477656 11056662215296497456 901483687589	72048155387323709166 60848707930336074888 91566798644692843080 67794585306592790502 81367373094090014810 68667418538424923895 35546783227570110037 39554233344776413617 24440179822776849936 96123730683833200096 06075968434171131935 37191614338757283716 76468281405320268072 65426614226829887130 92783537647339329695 30366247098854944951 84165933602502430046 76482920541359418804 85164085567795469978 81107062399963138470 50847264409549977811 51796286958504880002 98402434762454804739 92614034485308847645 98758643667639502489 40852372158380574099 89610932125158799787 31162299442144629181 31673891835568409149 00281829062685080531 1594344303	29630662596132586125 10722504963395548258 61233355569709440363 10973584073054997481 32713366742526667518 21036572790927651729 93760836300359435459 65634272701189539927 10112526004080759815 92415385162991240552 22035670674309040232 70037325856084145438 59496762731252232496 64406901498206980230 38175506355495692251 66427796755397154788 58219604169042434990 61096173789247159594 74874776568633375546 82325250872944412496 35383241018669613903 77755965918617524349 15893246017175644700 59815188744442483658 36933345250450555878 18578570002746785166 01871197021747517027 03527642345159922750 27242850417002297606 58084590689586451062 8309	57348009	1519540	1625294097	1684161647

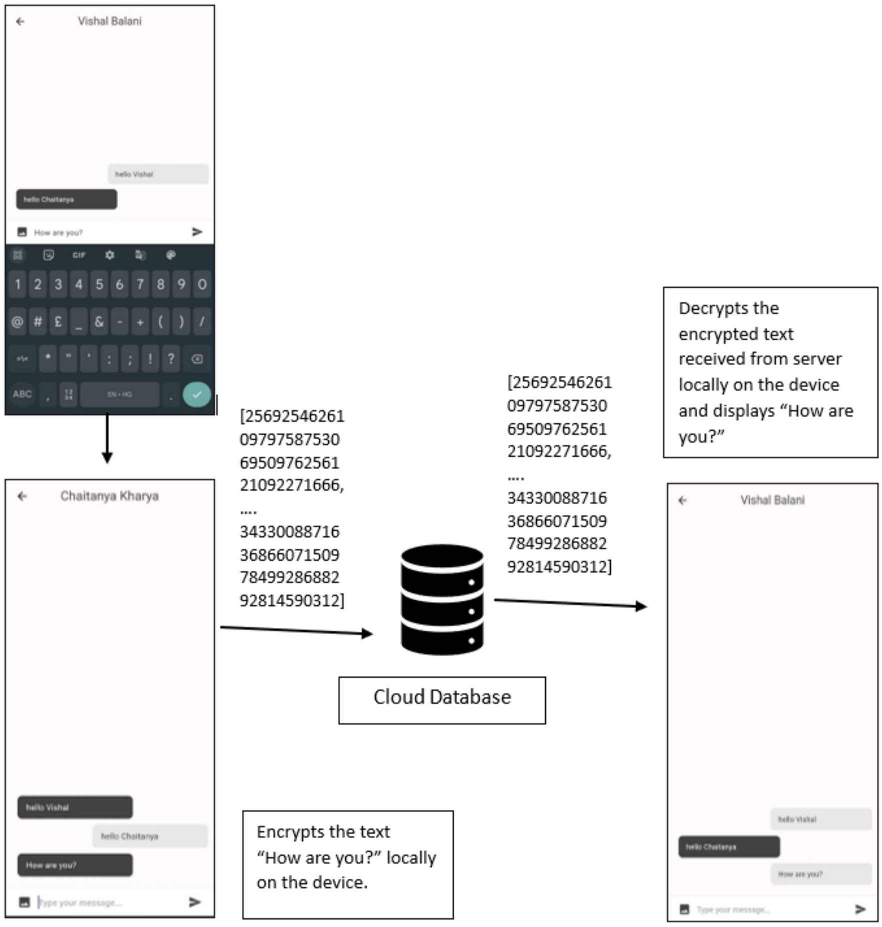

Fig. 5.1: Encryption and decryption process flow within the app

5.2. Platform Testing. Extensive testing across different Android versions (going from 10 to 14) was led to guarantee the enhanced RSA execution's reliable presentation and functionality across various platforms. The thorough testing was meant to identify and address any compatibility or operational issues that could arise across various Android versions.

5.3. Enhanced Functionality and Security. The incorporation of the enhanced RSA algorithm brought forth a significant enhancement in the security aspect of the chat application. This upgrade assured users of secure encryption methods, providing a secure environment for sharing sensitive information through the messaging application.

5.4. Process Visualization. For a more clear comprehension of the encryption and decryption processes inside the application, a definite interaction flowchart is shown in Fig. 5.1. This visual guide effectively clarifies how the adjusted RSA calculation is utilized inside the application, showing the means engaged with the encryption and decryption of messages.

5.5. Accessibility and Further Exploration. The complete process flow diagram for exploring and understanding the enhanced RSA feature within the chat application is available. Feel free to access the application available on Google Play named **EncryptoSafe Private Messaging**.

6. Conclusion. The development of this enhanced RSA and secure chat application has showcased promising results in the realm of digital messaging encryption. Employing three integers within the enhanced RSA algorithm has introduced an additional layer of complexity to the encryption procedure. Consequently, the resultant chat application demonstrates resilience against potential threats such as eavesdropping and unauthorized access to chats, thereby significantly contributing to the domains of cryptography and cybersecurity by ensuring the protection of sensitive information exchanged between users. During app development, several challenges were faced. These included ensuring secure key management, maintaining real-time data transfer capabilities, and implementing user authentication protocols. Additionally, limitations in the scope of the project prevented the inclusion of file encryption functionalities for multimedia content such as audio, video, and images. The collaborative effort and advancements made in this work signify a substantial leap forward in establishing secure communication frameworks, laying a foundation for continued research and innovation in securing digital interactions.

The app was provided to multiple users for user testing, the reviews received have been summarised as follows. The chatting was seamless, with messages being sent and received in real-time. But issues were faced in sending media files like photos, videos and audios, only text and emoticons were supported by the application. User can initiate chat with any other user of the app, regardless of whether they are in the user's contact list or not. While the implementation has demonstrated considerable success, there are several avenues for future enhancements and improvements in the chat application. Key generation processes can benefit from leveraging parallel processing and hardware acceleration techniques to optimize performance. As user numbers increase, scaling the performance and security aspects becomes crucial. Moreover, ensuring resistance against modern quantum attacks is imperative for the enhanced RSA algorithm.

REFERENCES

[1] Shand, M. and Vuillemin, J., *Fast implementations of RSA cryptography*, *Proceedings of IEEE 11th Symposium on Computer Arithmetic*, pp. 252-259, 1993.

[2] R. Abdeldaym, H. Mohamed, H. Abd elkader, and R. Hussien, *Modified RSA Algorithm Using Two Public Key and Chinese Remainder Theorem*, *International Journal of Electronics and Information Engineering*, vol. 10, no. 1, pp. 51-64, 2019.

[3] P. Karanam, V. Varadarajan, and V. Subramaniyaswamy, *An Efficient Framework for Sharing a File in a Secure Manner Using Asymmetric Key Distribution Management in Cloud Environment*, *Journal of Computer Networks and Communications*, vol. 2019, pp. 1-8, 2019.

[4] A. I. Khyoon, *Modification on the Algorithm of RSA Cryptography System*, *Al-Fatih Journal*, vol. 1, no. 24, pp. 80-89, 2005.

[5] Bodrato, Marco and Zanoni, Alberto, *Karatsuba and Toom-Cook methods for multivariate polynomials*, *Acta Universitatis Apulensis*, 2011.

[6] A. Naghib, N. Jafari Navimipour, M. Hosseinzadeh, and A. Sharifi, *A comprehensive and systematic literature review on the big data management techniques in the internet of things*, *Wireless Networks*, vol. 29, no. 3, pp. 1085-1144, 2023.

[7] N. Namassivaya, S. Nithigna, S. Kovilala, and MD S. Hussain, *Encrypted Chat Application Using RSA Algorithm*, *International Journal of Engineering Technology and Management Sciences*, vol. 7, no. 2, pp. 854-859, 2023.

[8] X. Zhou and X. Tang, *Research and Implementation of RSA Algorithm for Encryption and Decryption*, in *Proceedings of 2011 6th International Forum on Strategic Technology*, vol. 2, pp. 1118-1121, 2011.

[9] B. Aniket et al., *RSA Algorithm in Cryptography*, *https://www.geeksforgeeks.org/rsa-algorithm-cryptography/*, *Accessed on 18-02-2024*.

[10] N. Y. Goshwe, *Data Encryption and Decryption Using RSA Algorithm in a Network Environment*, *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 13, no. 7, pp. 9, 2013.

[11] A. Nivetha, S. Preethy Mary, and J. Santosh Kumar, *Modified RSA Encryption Algorithm using Four Keys*, *International Journal of Engineering Research & Technology (IJERT)*, vol. 3, no. 7, pp. 1-5, 2015.

[12] Y. Guo and X. Zhang, *A Modified RSA Algorithm with Mixed-Base Representation*, in *Proceedings of the 2011 International Conference on Communication Technology and Information Processing (ICCTIP 2011)*, pp. 368-371, 2011.

[13] D. Wang and L. Tang, *A modified RSA cryptosystem with elliptic curve cryptography*, *Computers and Security*, vol. 22, no. 1, pp. 41-45, 2003.

[14] J. Lee and J. Kim, *A modified RSA cryptosystem with randomized exponentiation*, *Information Processing Letters*, vol. 92, no. 1, pp. 1-5, 2004.

[15] A. Amare Anagaw, S. Vuda, *A Modified RSA Encryption Technique Based on Multiple public keys*, *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 4, pp. 2320 – 9801, 2013.

[16] S. Gunasekaran and M. P. Lavanya, *A Review on Enhancing Data Security in Cloud Computing using RSA and AES Algorithms*, *International Journal of Advances in Engineering Research*, vol. 9, no. 4, pp. 1-7, 2015.

[17] H. Kaur and Aarju, *Enhancing Cloud Data Security with Modified RSA Encryption and Salt: A Comprehensive Review*, *International Journal of Emerging Technologies and Innovative Research*, vol. 10, no. 11, pp. 298-306, 2023.

[18] V. Kapoor, *Data Encryption and Decryption Using Modified RSA Cryptography Based on Multiple Public Keys and 'n' prime*

Number, *International Journal of Scientific Research in Network Security and Communication*, vol. 1, no. 2, pp. 35-38, 2013.
[19] I. Jahan, M. Asif, and L. J. Rozario, *Improved RSA cryptosystem based on the study of number theory and public key cryptosystems*, *American Journal of Engineering Research (AJER)*, vol. 4, no. 1, pp. 143-149, 2013.
[20] R. Imam, Q. M. Areeb, A. Alturki, and F. Anwer, *Systematic and Critical Review of RSA Based Public Key Cryptographic Schemes: Past and Present Status*, *IEEE Access*, vol. 9, pp. 155949-155976, 2021.
[21] A. E. Mezher, *Enhanced RSA Cryptosystem based on Multiplicity of Public and Private Keys*, *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 5, pp. 3949-3953, 2018.
[22] S. Y. Bonde and U. S. Bhadade, *Implementation of RSA Algorithm and Modified RSA Algorithm Methods: A Review*, *International Journal of Advanced Technology in Engineering and Science*, vol. 5, no. 5, 2017, pp. 1-6.
[23] H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, *Implementation of RSA Algorithm Based on RNS Montgomery Multiplication*, *Springer-Verlag Berlin Heidelberg*, 2001, pp. 364-376.
[24] M. Markovic, T. Unkasevic, and G. Dordevic, *RSA Algorithm Optimization on Assembler of TI TMS320C54X Signal Processors*, *IEEE 11th European Signal Processing Conference*, 2002, pp. 1-4.
[25] N. Kumar and P. Chaudhary, *Implementation of Modified RSA Cryptosystem for Data Encryption and Decryption based on n Prime number and Bit Stuffing*, *International Conference on Information and Communication Technology for Competitive Strategies (ICTCS-2016)*, 2016, pp. 1-6.
[26] V. Yakovyna, D. Fedasyuk, M. Seniv, and O. Bila, *The Performance Testing of RSA Algorithm Software Realization*, 2007, pp. 390-392.
[27] J. C. Bajard and L. Imbert, *A Full RNS Implementation of RSA*, *IEEE Transactions on Computers*, vol. 53, no. 6, 2004, pp. 769 - 774.
[28] S. Sharma, P. Sharma, and R. S. Dhakar, *RSA Algorithm Using Modified Subset Sum Cryptosystem*, *IEEE International Conference on Computer and Communication Technology (ICCCT)*, 2011, pp. 457-461.
[29] G. Shrivastava, P. Kumar, *SensDroid: analysis for malicious activity risk of Android application*, *Multimedia Tools and Applications*, vol. 78, no. 24, 2019, pp. 35713-35731.
[30] A. Sinha, G. Shrivastava, P.Kumar, *A pattern-based multi-factor authentication system*, *Scalable Computing: Practice and Experience*, vol. 20, no. 1, 2019, pp. 101-112.

Edited by: Kavita Sharma
Special issue on: Recent Advance Secure Solutions for Network in Scalable Computing
Received: Feb 19, 2024
Accepted: Apr 28, 2024