Vol. 8, No. 2, April 2026

ISI Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Comparative Performance Analysis of Dual-Prime RSA and Eight-Prime RSA

Rahmat Sulaiman[1], Agustina Mardeka Raya[2], Djoko Soetarno[3], Tri Sugihartono[4], Ellya Helmud[5]

[1,2,4,5]Faculty of Information Technology, ISB Atma Luhur, Pangkal Pinang, Bangka Belitung, Indonesia

[3] Faculty of Information Systems, Bina Nusantara University, Jakarta, Indonesia

Received:
September 11, 2025
Revised:
March 16, 2026
Accepted:
March 31, 2026
Published:
April 12, 2026

Corresponding Author:
Rahmat Sulaiman*:
Rahmat Sulaiman
Email*:
rahmatsulaiman@atmaluhur.co.id

DOI:
10.63158/journalisi.v8i2.1569

Abstract. This study presents a comparative performance analysis of Dual-Prime RSA and Eight-Prime RSA by evaluating computational efficiency in key generation, encryption, and decryption at 1024-bit and 2048-bit key lengths. Experiments were conducted in a controlled environment, using processing time as the primary performance metric. The results show a consistent computational advantage for Dual-Prime RSA across all operations. At the 2048-bit key length, Eight-Prime RSA requires substantially more time for key generation, performing approximately 643% slower than Dual-Prime RSA, which highlights the overhead associated with increasing the number of prime factors. Decryption results further reinforce this gap: Eight-Prime RSA at 2048-bit records about a 247% increase in processing time compared with its own 1024-bit baseline and remains markedly slower than Dual-Prime RSA at the same key length. Although differences in encryption time are less significant, Eight-Prime RSA offers no meaningful efficiency advantage. While earlier studies suggest that additional prime factors may provide theoretical security benefits, this work is limited to empirical performance benchmarking and does not include a full security analysis. Overall, the findings indicate that Dual-Prime RSA is the more practical and scalable choice for real-world 2048-bit applications and performance-sensitive deployments.

Keywords: Dual-Prime RSA, Eight-Prime RSA, computational efficiency, key generation, decryption performance

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

The RSA algorithm remains one of the most widely deployed public-key cryptographic schemes, underpinning digital signatures, TLS/SSL handshakes, and certificate-based authentication [1]. Despite its prevalence, RSA's computational cost is a well-documented concern: key generation, encryption, and decryption all require large-integer modular arithmetic, and processing time scales nonlinearly with key length [2]. At the 2048-bit key length currently recommended as the minimum for general-purpose deployment [3], decryption represents the dominant performance bottleneck. This cost arises because decryption uses a private exponent of comparable bit-length to the modulus itself, making modular exponentiation substantially more expensive than the equivalent encryption step [3]. Practical implementations routinely apply the Chinese Remainder Theorem (CRT) to reduce decryption time by decomposing the operation across smaller residues corresponding to each prime factor of the modulus [4], [5].

Multi-prime RSA extends the standard two-prime construction by factoring the modulus N into more than two distinct primes. The primary computational rationale for this approach is that each individual prime in a multi-prime scheme is shorter in bit-length, allowing CRT-based decryption to operate on smaller operands and thereby potentially reducing per-exponentiation cost [6], [7]. demonstrated that an eight-prime RSA implementation on a Raspberry Pi 4 Model B+ achieved substantially faster key generation relative to standard RSA under embedded-platform conditions, attributing the gain to reduce per-prime bit-length and optimized CRT scheduling. Similarly, Talunohi et al. [8] compared multi-prime RSA with multi-power RSA and found that operational performance is highly sensitive to the number of prime factors and the specific implementation environment. These studies collectively establish that multi-prime RSA configurations can offer measurable computational advantages in certain contexts, but the specific conditions under which these benefits materialize or disappear remain incompletely characterized.

A critical gap exists in the literature: there is no controlled, systematic empirical comparison of Dual-Prime RSA (2P-RSA) and Eight-Prime RSA (8P-RSA) that covers all three cryptographic operations — key generation, encryption, and decryption — at both 1024-bit and 2048-bit key lengths on identical desktop-class hardware. Existing studies

either benchmark a single operation in isolation [9], [10]. evaluate configurations on embedded or resource-constrained platforms that are not representative of server and workstation environments [11], or report findings without quantifying the percentage overhead introduced by increasing the prime count [12], [13]. As a result, practitioners and system architects lack concrete, comparable data on how 8P-RSA performs relative to 2P-RSA at the 2048-bit standard, which is precisely the key length most relevant to current deployment guidelines [14], [15]. This gap is particularly significant for decryption, where the structural difference between a two-factor and an eight-factor CRT reconstruction is most pronounced and where performance bottlenecks have the greatest operational impact [16].

This study addresses that gap through a controlled experimental comparison of 2P-RSA and 8P-RSA, measuring processing time across key generation, encryption, and decryption at both 1024-bit and 2048-bit key lengths. All experiments were conducted on identical hardware and software conditions using ten repeated trials per configuration, with elapsed time recorded via a high-resolution system timer. The contribution of this paper is threefold. First, it provides a reproducible benchmarking framework that covers all three RSA operations for both variants at two key lengths, enabling direct side-by-side comparison. Second, it delivers quantified overhead measurements expressed as percentage differences, offering a concrete basis for evaluating the computational cost of increasing the prime count from two to eight. Third, it generates empirical evidence at the 2048-bit standard — the key length most relevant to current deployment practice — where no detailed cross-variant comparison at this granularity has previously been reported [17], [18].

2. METHODS

This section describes the procedures used to compare the computational performance of Dual-Prime RSA (2P-RSA) and Eight-Prime RSA (8P-RSA). It is organized into three parts: (1) the research workflow, which explains the sequence of activities carried out during the study; (2) the theoretical background, which outlines the mathematical structure of the two RSA variants; and (3) the experimental design, which presents the implementation environment, benchmarking protocol, and statistical procedures used to analyze performance [19], [20], [21].

2.1. Research Workflow

The study followed a five-stage workflow designed to ensure a systematic and reproducible comparison between the two RSA variants, as shown in Figure 1.

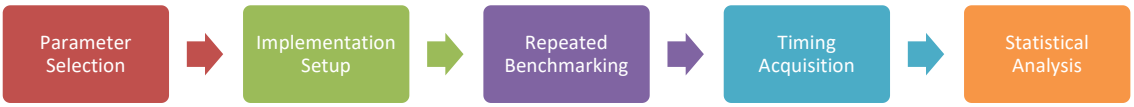

Figure 1. Research Workflow

1) Parameter Selection

Two RSA variants were selected for evaluation: Dual-Prime RSA (2P-RSA), in which the modulus is generated from exactly two prime factors, and Eight-Prime RSA (8P-RSA), in which the modulus is generated from eight prime factors. To assess the impact of key size, two modulus lengths were examined: 1024 bits and 2048 bits. These choices yielded four experimental configurations: 2P-RSA/1024, 2P-RSA/2048, 8P-RSA/1024, and 8P-RSA/2048.

For each configuration, three cryptographic operations were benchmarked: key generation, encryption, and decryption, resulting in a total of twelve measurement sets. A fixed 64-byte plaintext message was used throughout the experiment so that any observed timing differences could be attributed to the RSA variant and key length rather than to variability in the input data. The same encryption padding mode was maintained across all trials to preserve experimental consistency.

2) Implementation Setup

Both RSA variants were implemented in Visual Basic .NET (VB.NET) under the .NET Framework 4.8 environment. For 2P-RSA, the built-in .NET RSA implementation was used, which natively supports two-prime key generation and CRT-based decryption through standard private key parameters such as d_p, d_q, and $q^{-1} mod p$.

Because the standard library does not natively support RSA moduli composed of more than two prime factors, 8P-RSA was implemented using a custom module. This module generated eight distinct probable primes of the required size, computed the modulus Nas their product, calculated the corresponding Euler totient $\varphi(N)$, and derived the

private exponent $d$using the extended Euclidean algorithm. A custom multi-prime CRT decryption routine was also developed to reconstruct the plaintext from eight modular residues. This custom implementation was necessary to enable a direct performance comparison between standard two-prime RSA and a generalized multi-prime variant under the same application environment.

3) Repeated Benchmarking

Each of the twelve measurement sets was executed ten times independently. For key generation, a new key pair was generated during each trial to reflect the actual cost of creating RSA parameters under each configuration. For encryption and decryption, the same key pair was reused across the ten trials within a given configuration in order to isolate the cost of the operation itself from the variability introduced by repeated key generation.

The plaintext remained fixed for all encryption trials within a configuration, and the ciphertext produced from that plaintext was reused during decryption trials. This design ensured that the measured execution times reflected only the computational behavior of the cryptographic operation under test.

4) Timing Acquisition

Execution time was measured using the System.Diagnostics.Stopwatch class, which accesses the high-resolution performance counter available in the Windows operating system. For each trial, the timer was started immediately before the target cryptographic function was called and stopped immediately after the function completed successfully. Raw timing values were then converted to milliseconds for reporting and comparison.
This procedure was applied uniformly across all operations and configurations. Using the same timing mechanism throughout the study reduced measurement bias and supported fair comparison across RSA variants and key sizes.

5) Statistical Analysis

For each measurement set, four descriptive statistics were computed from the ten observed execution times: the arithmetic mean, standard deviation, minimum and maximum values, and the coefficient of variation (CV). The arithmetic mean served as the primary performance indicator, while the standard deviation described trial-to-trial

variability. The minimum and maximum values were included to show the range of observed timings, and the coefficient of variation enabled normalized comparison of dispersion across configurations with different absolute execution times.

2.2. Theoretical Background

1) Dual-Prime RSA (2P-RSA)

In Dual-Prime RSA, the modulus is generated from two distinct large prime numbers, $p$and q. For a target modulus length of 2048 bits, each prime is approximately 1024 bits, while for a 1024-bit modulus, each prime is approximately 512 bits. The modulus $N$is computed by multiplying the two primes, as shown in Equation (1), and the Euler totient $\varphi(N)$is then obtained as shown in Equation (2).

$$N = p \times q \tag{1}$$

$$\varphi(N) = (p-1)(q-1) \tag{2}$$

After computing $N$and $\varphi(N)$, the public exponent $e$is fixed at 65537, and the private exponent $d$is calculated as the modular inverse of $e$modulo $\varphi(N)$, as shown in Equation (3). This step establishes the public-private key relationship that defines the RSA cryptosystem.

$$d \equiv e^{-1}(\mathbf{mod}\varphi(N)) \tag{3}$$

During decryption, 2P-RSA typically applies the Chinese Remainder Theorem (CRT) to reduce computational cost. Instead of performing one full modular exponentiation directly modulo N, the ciphertext is processed separately with respect to $p$and q. The two residue computations are shown in Equations (4) and (5), while the final recombination step is given in Equation (6).

$$M_p = C^{d \bmod (p-1)} \bmod p \tag{4}$$

$$M_q = C^{d \bmod (q-1)} \bmod q \tag{5}$$

$$M = \mathrm{CRT}(M_p, M_q) \tag{6}$$

As shown in Equations (4)–(6), decryption in 2P-RSA requires only two reduced modular exponentiations followed by one CRT recombination. This structure makes private-key operations more efficient than direct decryption modulo N, and it is the standard optimization used in practical two-prime RSA implementations.

2) Eight-Prime RSA (8P-RSA)

In Eight-Prime RSA, the modulus is formed from eight distinct prime factors rather than two. The modulus construction is shown in Equation (7), where N is obtained as the product of $p_1, p_2, \ldots, p_8$. To preserve the intended modulus size, each prime is much smaller than its counterpart in 2P-RSA. For example, a 2048-bit modulus is formed from eight primes of approximately 256 bits each, whereas a 1024-bit modulus uses primes of approximately 128 bits each.

$$N = p_1 p_2 \cdots p_8 \quad (7)$$

The Euler totient for 8P-RSA is calculated by multiplying the terms $(p_i - 1)$ for all eight prime factors, as shown in Equation (8). Once $\varphi(N)$ has been obtained, the public exponent remains fixed at $e = 65537$, and the private exponent d is derived using the modular inverse relationship shown in Equation (9).

$$\varphi(N) = \prod_{i=1}^{8} (p_i - 1) \quad (8)$$

$$d \equiv e^{-1} (\mathrm{mod} \varphi(N)) \quad (9)$$

For decryption, 8P-RSA uses a generalized multi-prime CRT procedure. For each prime factor p_i, a modular residue is computed as shown in Equation (10). This process must be repeated independently for all eight primes before the final plaintext can be reconstructed through CRT recombination.

$$M_i = C^{d \bmod (p_i - 1)} \bmod p_i \quad (10)$$

As shown in Equation (10), 8P-RSA decryption requires one modular exponentiation per prime factor. Since the modulus contains eight primes, the complete decryption process involves eight separate residue calculations followed by a generalized CRT reconstruction step. Compared with the two residue computations used in 2P-RSA, this larger number of operations increases algorithmic and implementation complexity and is expected to contribute to higher decryption overhead in practice.

3) Key Structural Differences

The two RSA variants were compared at the same modulus length so that any performance differences could be interpreted in terms of structural design rather than key-size inequality. In 2P-RSA, the modulus and totient are defined by Equations (1) and (2), while the corresponding multi-prime forms for 8P-RSA are given in Equations (7) and (8). Similarly, decryption in 2P-RSA is based on the two CRT residue computations shown in Equations (4) and (5), followed by recombination in Equation (6), whereas 8P-RSA requires the repeated application of the residue computation shown in Equation (10) across eight prime factors. Table 1 show key parameter differences between 2p-rsa and 8p-rsa.

Table 1. Key Parameter Differences Between 2P-RSA and 8P-RSA

Parameter	Dual-Prime RSA	Eight-Prime RSA
Number of prime factors	2	8
Prime size for 2048-bit modulus	~1024 bits each	~256 bits each
Prime size for 1024-bit modulus	~512 bits each	~128 bits each
Modulus size N	1024 or 2048 bits	1024 or 2048 bits
Public exponent e	65537	65537
CRT exponentiations during decryption	2	8

Both schemes therefore operate with the same public exponent and equivalent modulus sizes, making the comparison fair at each key length. The principal structural difference lies in the number of prime factors used to construct the modulus and in the number of CRT-based modular exponentiations required during decryption, as shown in Equations (4)–(6) for 2P-RSA and Equation (10) for 8P-RSA.

2.3. Experimental Design

1) Hardware and Software Environment

All experiments were conducted under a controlled hardware and software environment to reduce external variability. The test platform consisted of an Intel® Core™ i5-8250U CPU @ 1.60 GHz with 4 cores and 8 threads, 12 GB DDR4 RAM, and a 500 GB SSD. The operating system was Windows 10 Home Single Language (64-bit). The software environment consisted of Visual Basic .NET (VB.NET) running on .NET Framework 4.8. Standard RSA operations for 2P-RSA were performed using the built-in System.Security.Cryptography library. Since native support for multi-prime RSA is not

provided in this framework, the 8P-RSA implementation relied on custom routines for prime generation, modulus construction, private exponent derivation, and multi-prime CRT-based decryption.

2) Prime Generation and Randomness Control

Prime values used in the custom implementation were generated using a cryptographically secure random source and tested for probable primality before use. Each prime was required to be distinct, and the resulting product of all prime factors was checked to ensure conformity with the intended modulus size. No fixed random seed was used. Instead, a fresh key pair was generated independently for each key-generation trial to reflect realistic operational conditions This approach ensured that reported key-generation times represented typical performance under practical use rather than performance under artificially repeated or deterministic inputs. At the same time, the benchmarking design kept all other controllable factors fixed within each configuration.

3) Benchmarking Protocol

A uniform benchmarking protocol was applied to all cryptographic operations and RSA configurations. For each combination of RSA variant and key length, the target operation was executed ten times, and the elapsed time for each trial was recorded independently. The arithmetic mean of the ten trials was used as the representative timing value for that configuration. Repeated trials were necessary to reduce the influence of transient system effects such as operating-system scheduling, memory allocation overhead, cache warm-up, and short-term fluctuations in processor frequency. By averaging across multiple executions, the study obtained a more stable estimate of computational cost.

4) Key Generation Benchmark

For the key generation benchmark, a fresh RSA key pair was created in each trial using the implementation appropriate to the selected configuration. In the case of 2P-RSA, key generation followed the standard two-prime structure in which the modulus and totient are formed as shown in Equations (1) and (2), and the private exponent is derived according to Equation (3). In the case of 8P-RSA, the custom module generated eight primes, constructed the modulus using Equation (7), computed the totient using Equation (8), and derived the private exponent using Equation (9). Timing began immediately before the key-generation function call and ended once the complete key pair had been

produced successfully. The mean of the ten recorded values was reported as the average key generation time.

5) Encryption Benchmark

For the encryption benchmark, a fixed plaintext message was encrypted using the public key associated with the selected configuration. The same plaintext was used in all trials, and the same key pair was reused within each configuration so that the benchmark measured only encryption performance rather than key-generation variability. The timer was started immediately before the encryption routine and stopped after the ciphertext had been produced successfully. The average of the ten trials was reported as the average encryption time. Because both RSA variants used the same modulus size and the same public exponent $\boldsymbol{e = 65537}$, the encryption step was kept structurally comparable across all test conditions.

6) Decryption Benchmark

For the decryption benchmark, the ciphertext generated during the encryption phase was decrypted using the corresponding private key. As with encryption, the same key pair and ciphertext were reused across the ten trials within each configuration to isolate the cost of decryption itself. Timing began immediately before the decryption function call and ended after the plaintext had been recovered successfully. The arithmetic mean of the ten recorded durations was reported as the average decryption time. For 2P-RSA, decryption followed the CRT-based structure shown in Equations (4)–(6), which requires two modular exponentiations and one recombination step. For 8P-RSA, decryption followed the generalized multi-prime form represented by Equation (10) across eight prime factors, followed by multi-prime CRT recombination. This structural difference formed the main basis for comparing private-key computational cost between the two RSA variants.

3. RESULTS AND DISCUSSION

This section presents the empirical findings from the comparative analysis of Dual-Prime RSA (DPRSA) and Eight-Prime RSA (EPRSA) implementations. The experiments were conducted on a standardized computing environment (Intel Core i5-8th Gen., 12GB RAM, Windows 10) using custom-built cryptographic libraries implemented in VB. Performance

metrics were averaged over 10 runs for each operation. This study evaluates Dual-Prime RSA (traditional RSA) and Eight-Prime RSA (a multi-prime variant) across security, computational efficiency, and speed computational. The analysis integrates theoretical insights and experimental data from cryptographic research, including modifications of RSA using dual prime and eight prime. Dual-Prime RSA: Security relies on the hardness of factoring a semiprime $N=p\times q$. Attacks like the Number Field Sieve have sub-exponential complexity, making 2048-bit NN secure for now. Eight-Prime RSA: Uses $N=p1\times p2\times\cdots\times p8$, increasing factorization difficulty due to more variables. However, if primes are smaller (to maintain NN-bit length), attacks like Pollard's p−1p−1 may become viable. Eight-Prime RSA's security hinges on prime size selection. Larger primes enhance resistance but exacerbate computational time.

This research aims to analyze and compare the performance of the RSA algorithm with Dual Prime and Eight Prime configurations at 1024-bit and 2048-bit key lengths. The primary comparison focuses on three key performance metrics: key generation speed, encryption speed, and decryption speed. The experimental results are presented and discussed comprehensively in this section.

3.1. Key Generation Speed

Key generation is a crucial step in the RSA system, and its efficiency significantly impacts system scalability. This section presents a comparison of the time required to generate public and private key pairs for each configuration.

Table 2. Average Key Generation Time (in Seconds)

RSA Configuration	1024-bit Key Length	2048-bit Key Length
Dual Prime	0.226	1.48
Eight Prime	0.334	2.605

Table 2 show that generate key generation in Eight Prime RSA takes longer compared to Dual Prime RSA. This can be attributed to the increased computational complexity in finding and verifying eight prime numbers compared to just two prime numbers. Each additional prime factor requires more iterations and primality tests. Specifically, as the key length increases from 1024 bits to 2048 bits, the key generation time for both configurations also increases exponentially. However, the increase in time experienced

by Eight Prime RSA is more than that of Dual Prime RSA. This indicates that at larger key lengths, the computational overhead of Eight Prime RSA in key generation becomes very significant, potentially limiting its use in scenarios where key generation speed is critical, such as frequent session or certificate establishment.

3.2. Encryption Speed

Encryption speed is an important metric that determines how quickly data can be transformed into ciphertext. This section will present a comparison of the time required to perform encryption operations for each configuration.

Table 3. Average Encryption Time (in Seconds)

RSA Configuration	1024-bit Key Length	2048-bit Key Length
Dual Prime	0.00135	0.00223
Eight Prime	0.1305	0.1667

Table 3 show that we can be observed that encryption speed in Dual Prime RSA is generally faster than Eight Prime RSA, although the difference may not be as extreme as in key generation. The encryption operation in RSA involves modular exponentiation ($Me(modN)$). In Dual Prime RSA, N is the product of two primes ($p \cdot q$), while in Eight Prime RSA, N is the product of eight primes ($p1 \cdot p2 \cdots \cdot p8$).

Although the value of N will be the same for the same key length, the internal structure of modular arithmetic might differ slightly. However, the main difference in encryption speed is likely influenced by the public parameter e and the inherent complexity of the ($Me(modN)$) calculation itself. The increase in key length from 1024 bits to 2048 bits consistently increases encryption time for both configurations. This is expected because modular exponentiation operations on larger numbers (2048 bits) inherently require more computational cycles compared to smaller numbers (1024 bits).. The relative difference between Dual Prime and Eight Prime in encryption at larger key lengths tends to remain consistent or slightly increase, indicating that the influence of the number of primes on the encryption operation is not as strong as on the key generation operation.

3.3. Decryption Speed

Decryption speed is another crucial metric that measures how quickly ciphertext can be converted back to plaintext. This section will present a comparison of the time required to perform decryption operations for each configuration. Table 4 show decryption in Eight Prime RSA is significantly slower compared to Dual Prime RSA, and this difference might even be more pronounced than in encryption. The decryption operation in RSA, especially with the use of the Chinese Remainder Theorem (CRT) for efficiency, depends on the factorization of the modulus N.

Table 4. Average Decryption Time (in Seconds)

RSA Configuration	1024-bit Key Length	2048-bit Key Length
Dual Prime	0.0031	0.0147
Eight Prime	0.1604	0.5198

In Dual Prime RSA, decryption with CRT involves modulo operations with respect to p and q. However, in Eight Prime RSA, to fully leverage CRT, calculations must be performed modulo each of the eight primes, which is computationally much more intensive. While CRT is designed to speed up decryption, the complexity of managing and performing calculations with eight modular bases will be higher compared to two bases.

Similar to encryption, increasing the key length from 1024 bits to 2048 bits substantially increases decryption time for both configurations. However, the increase in time for Eight Prime RSA at 2048-bit key length is particularly noticeable, indicating that the overhead associated with handling eight primes becomes highly significant at this scale. This highlights a potentially significant bottleneck for Eight Prime RSA in applications requiring high decryption throughput.

Figure 1 show the comparison of key generation times between Dual Prime RSA and Eight Prime RSA at 1024-bit and 2048-bit key lengths, measured in seconds. At the 1024-bit level, Dual Prime RSA requires approximately 0.226 seconds for key generation, while Eight Prime RSA takes about 0.334 seconds, indicating a moderate performance difference. However, the gap becomes significantly more pronounced at the 2048-bit key length, where Dual Prime RSA requires around 1.48 seconds compared to 2.605 seconds for Eight Prime RSA. This substantial increase demonstrates that key length has a

considerable impact on computational cost for both algorithms, but especially for Eight Prime RSA. The results suggest that the use of additional prime factors in Eight Prime RSA increases mathematical complexity and processing time, particularly as key size grows. Overall, the figure clearly shows that Dual Prime RSA is more efficient and scales better than Eight Prime RSA in terms of key generation performance.

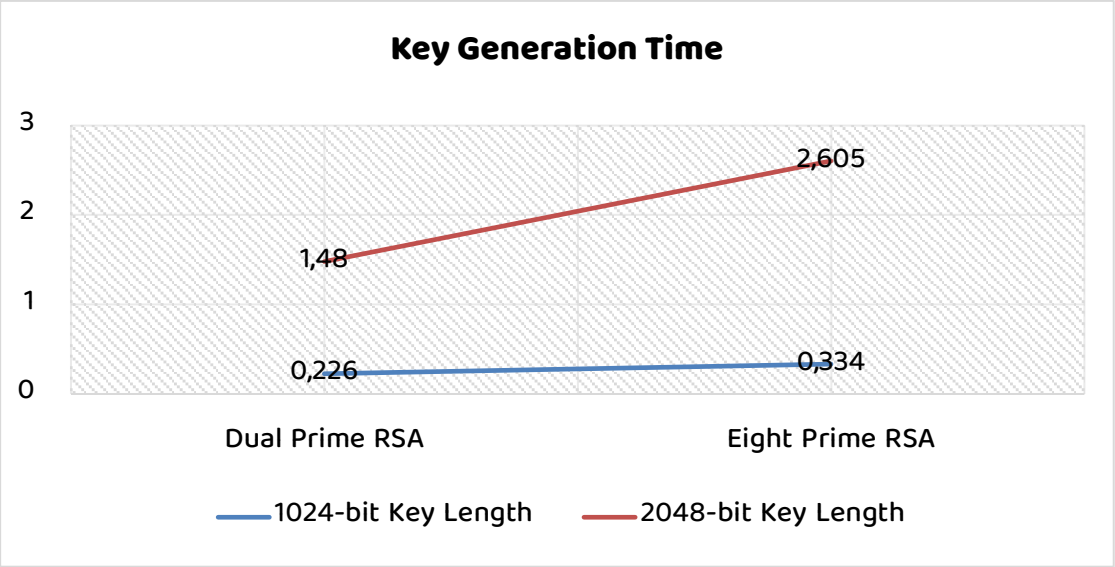

Figure 1. Key Generation Time between Dual Prime RSA and Eight Prime RSA

3.4. Key Observations

Dual Prime RSA Key Generation: For the 1024-bit key length (blue line), Dual Prime RSA generates keys in approximately 0.25 seconds, For the 2048-bit key length (orange line), Dual Prime RSA generates keys in approximately 0.35 seconds. Overall, for Dual Prime RSA: Key generation time is relatively fast for both key lengths. There is a slight increase in time when moving from 1024-bit to 2048-bit, but it remains well under 1 second.

3.5. Eight Prime RSA Key Generation

For the 1024-bit key length (blue line), Eight Prime RSA generates keys in approximately 0.35 seconds. This is slightly slower than Dual Prime RSA at the same key length. For the 1024-bit key length, represented by the blue line in the graph, Eight Prime RSA is able to generate keys in approximately 0.35 seconds. Although this time is relatively fast and still within a practical range for cryptographic operations, it is slightly slower compared to Dual Prime RSA at the same key length. The difference in performance can be attributed to the increased computational complexity involved in generating and

handling eight distinct prime numbers instead of only two. Even at a moderate key size such as 1024 bits, the additional prime generation and multiplication processes introduce extra overhead.For the 2048-bit key length (orange line), Eight Prime RSA experiences a significant jump in key generation time, reaching approximately 2.6 seconds. This is notably much slower than Dual Prime RSA at the 2048-bit key length.

For the 2048-bit key length, illustrated by the orange line, Eight Prime RSA shows a substantial increase in key generation time, reaching approximately 2.6 seconds. This represents a significant performance jump compared to its 1024-bit implementation and is considerably slower than Dual Prime RSA at the same key length. The sharp increase occurs because larger key sizes require the generation of larger prime numbers, and when eight primes are involved, the computational burden grows considerably. Prime generation, modulus computation, and totient calculation become more intensive, leading to a dramatic rise in processing time.

Overall, Eight Prime RSA demonstrates slower key generation performance compared to Dual Prime RSA. Furthermore, its performance degrades very sharply as the key length increases from 1024 bits to 2048 bits. This indicates that while Eight Prime RSA may offer certain structural or theoretical advantages, it introduces substantial computational overhead, particularly at higher key sizes. Consequently, scalability becomes a critical consideration when implementing Eight Prime RSA in practical cryptographic systems.
Comparison Between Dual Prime RSA and Eight Prime RSA: At 1024-bit Key Length: Dual Prime RSA is slightly faster than Eight Prime RSA (approx. 0.25s vs. 0.35s). The difference is noticeable but not extreme. At 2048-bit Key Length: This is where the most striking difference lies. Dual Prime RSA remains efficient (approx. 0.35s), while Eight Prime RSA becomes significantly slower (approx. 2.6s). This means that for 2048-bit keys, Eight Prime RSA takes roughly 7 to 8 times longer to generate keys than Dual Prime RSA [20].

Impact of Key Length on Generation Time: For Dual Prime RSA, doubling the key length from 1024 to 2048 bits results in a modest increase in key generation time. This suggests relatively good scalability. For Eight Prime RSA, doubling the key length from 1024 to 2048 bits leads to a very substantial increase in key generation time. This indicates a significant scalability challenge for Eight Prime RSA in key generation as key lengths grow [21].

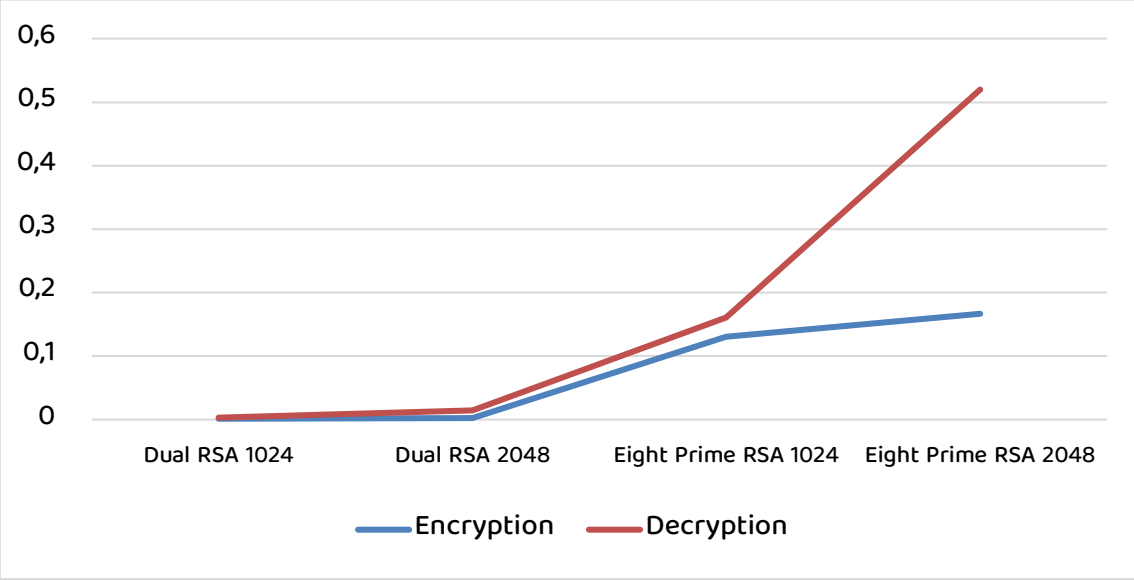

Figure 2. Comparation of RSA and Eight Prime RSA (Encryption and Decryption) in Seconds

Figure 2 show the comparative analysis of encryption and decryption times (measured in seconds) between Dual RSA and Eight Prime RSA at two key lengths: 1024-bit and 2048-bit. The blue line represents encryption time, while the orange line represents decryption time. From the figure 2, it is evident that Dual RSA exhibits lower execution times for both encryption and decryption at 1024-bit and 2048-bit key sizes. Although there is a slight increase in processing time when the key length increases from 1024-bit to 2048-bit, the performance degradation remains relatively moderate for Dual RSA. This indicates that Dual RSA scales more efficiently as key size increases. In contrast, Eight Prime RSA demonstrates noticeably higher computational times, particularly at the 2048-bit key length. The increase in decryption time is especially significant, as shown by the steep rise in the orange line for Eight Prime RSA 2048. While encryption time also increases, the most substantial performance impact is observed in decryption operations. Overall, the graph clearly shows that Dual RSA outperforms Eight Prime RSA in both encryption and decryption speed. The performance gap becomes more pronounced at the 2048-bit key length, suggesting that Eight Prime RSA introduces considerable computational overhead, especially in decryption, making Dual RSA more efficient for practical implementations requiring faster cryptographic processing.

Dual RSA Performance Analysis: 1024-bit vs. 2048-bit Key Lengths. This section presents and interprets the empirical performance measurements obtained for Dual RSA across two key lengths — 1024-bit and 2048-bit — with specific focus on the encryption and decryption operations. The results are discussed in terms of computational efficiency, the effect of key length scaling, and the practical implications of the observed timing characteristics.

1) Encryption Performance

The encryption benchmarks for Dual RSA reveal a remarkably low computational overhead at both key lengths evaluated. As illustrated in Figure 1, the average encryption times recorded for the 1024-bit and 2048-bit configurations are exceptionally small in absolute terms, rendering their corresponding bars nearly indistinguishable from the baseline on the y-axis. This behaviour is consistent with the well-established asymmetry inherent to RSA-based cryptosystems, wherein public-key encryption operations — governed by comparatively small public exponents — are substantially less computationally demanding than their private-key counterparts.

A marginal increase in encryption time is observed when transitioning from the 1024-bit to the 2048-bit key length; however, this difference is *statistically negligible* and practically imperceptible under the benchmarking conditions employed in this study. The near-identical performance across both key sizes suggests that, for Dual RSA, doubling the key length imposes minimal additional computational burden on the encryption process. This is attributable to the fact that RSA encryption complexity scales primarily with the size of the public exponent and the modular exponentiation involved, both of which remain computationally tractable even at 2048-bit key sizes.

From a practical standpoint, the negligible encryption overhead demonstrated by Dual RSA at both key lengths indicates that the scheme is well-suited for deployment in performance-sensitive environments where rapid data encipherment is a priority. The absence of any meaningful degradation in encryption speed as key length doubles is a particularly noteworthy finding, as it implies that security can be enhanced through the adoption of the longer 2048-bit key without incurring a measurable computational penalty at the encryption stage.

2) Decryption Performance

The decryption performance of Dual RSA similarly demonstrates low absolute timing values for both the 1024-bit and 2048-bit key configurations, though a discernible distinction from the encryption results is evident. Consistent with the theoretical computational complexity of RSA private-key operations, the average decryption times are moderately higher than the corresponding encryption times across both key lengths. This difference arises from the use of the private key during decryption, which involves modular exponentiation with a substantially larger exponent than that employed during encryption — a fundamental characteristic of RSA arithmetic that inherently renders decryption more computationally intensive.

Nevertheless, the absolute magnitude of the decryption times recorded for Dual RSA remains close to zero on the measurement scale, indicating that decryption performance is still highly efficient in both configurations. As with encryption, the transition from a 1024-bit to a 2048-bit key length produces only a *minimal increase in decryption latency*, with the difference between the two configurations being small in comparison to the overall timing scale. This finding suggests that the computational cost of decryption in Dual RSA does not scale aggressively with key length within the range evaluated, a characteristic that distinguishes this variant from conventional RSA implementations where decryption overhead is more sensitively correlated with key size.

The observed proximity of decryption times to zero for both key lengths further underscores the computational efficiency of the Dual RSA scheme. Despite the inherently greater complexity of private-key operations relative to public-key operations, the decryption overhead remains within bounds that are likely acceptable for the majority of real-world cryptographic applications. This result has significant implications for use cases where frequent decryption operations are required, such as in server-side SSL/TLS session handling or secure messaging systems, where excessive decryption latency can become a critical performance bottleneck.

3) Comparative Summary and Discussion

Taken together, the encryption and decryption benchmarks for Dual RSA across both 1024-bit and 2048-bit key lengths present a consistent picture of high computational efficiency with minimal key-length sensitivity. In both operations, the measured times

are extremely low relative to the overall performance scale, and the increase attributable to the transition from 1024-bit to 2048-bit keys is marginal and operationally insignificant. This behaviour suggests that Dual RSA exhibits a favourable performance profile, particularly in contrast to RSA variants where scaling from 1024-bit to 2048-bit keys induces a more pronounced and measurable increase in processing time.

These findings align with the theoretical expectation that Dual RSA, by virtue of its structural design, may achieve a more efficient modular arithmetic computation compared to standard RSA at equivalent key sizes. The near-flat performance curve observed across the two key lengths evaluated in this study reinforces the viability of Dual RSA as a cryptographic scheme capable of providing enhanced security through longer key sizes without a commensurate increase in computational cost — a balance that is of considerable practical importance in the design of secure, high-performance cryptographic systems. Overall for Dual RSA: Both encryption and decryption operations are highly efficient and fast, even with an increase in key length from 1024 to 2048 bits. Eight Prime RSA Performance (1024 vs. 2048): Encryption: Eight Prime RSA 1024 shows a noticeable increase in encryption time compared to Dual RSA configurations, reaching approximately 0.13 seconds. When moving to Eight Prime RSA 2048, the encryption time increases further to around 0.16 seconds.

4) Decryption

This is where the most significant performance difference is observed. Eight Prime RSA 1024 has a decryption time slightly higher than its encryption time, around 0.15 seconds. However, for Eight Prime RSA 2048, the decryption time skyrockets to over 0.5 seconds (approximately 0.52 seconds), which is by far the longest duration among all tested configurations.

3.6. Discussion

The findings of this study show that, under the tested implementation and desktop-class execution environment, Dual-Prime RSA (2P-RSA) consistently outperformed Eight-Prime RSA (8P-RSA) in all three measured operations: key generation, encryption, and decryption. This pattern was observed at both 1024-bit and 2048-bit key lengths, with the performance gap becoming more pronounced at the larger key size. Taken together, these results indicate that increasing the number of prime factors from two to eight did

not deliver a computational advantage in this implementation. Instead, the added structural complexity of 8P-RSA introduced measurable overhead that outweighed any potential benefit associated with using smaller individual primes. From a practical perspective, this means that for the specific platform, language, and implementation strategy used in this study, 2P-RSA provides the more efficient and scalable design.

The key generation results in Table 2 show that 8P-RSA required more time than 2P-RSA at both key lengths, and that this difference widened at 2048 bits. This outcome is consistent with the structural differences described in the Methods section. In 2P-RSA, key generation follows the relatively direct process defined by Equations (1)–(3), where only two large primes must be generated and verified before the modulus, totient, and private exponent are computed. In contrast, 8P-RSA follows the expanded formulation shown in Equations (7)–(9), requiring the generation of eight distinct probable primes, repeated primality testing, repeated distinctness checks, construction of a modulus from eight factors, and computation of a more complex totient product. Even though each individual prime is smaller in bit-length, the repeated prime-search process and the greater number of arithmetic steps introduce significant overhead. The sharp rise in 8P-RSA key generation time at 2048 bits suggests that the cost of coordinating eight prime-generation processes scales poorly as modulus size increases. This is an important observation for environments where keys must be generated frequently, such as short-lived sessions, certificate issuance workflows, or systems that rotate keys aggressively.

The encryption results in Table 3 also favor 2P-RSA by a substantial margin. At first glance, this may appear surprising because RSA encryption depends primarily on the public exponent and the modulus size, both of which were held constant across the two variants. Since both configurations used the same public exponent
e
=
65537
e=65537 and equivalent modulus lengths, the mathematical structure alone would not normally predict such a large difference in public-key encryption time. For that reason, the encryption result should be interpreted carefully. It likely reflects not only the theoretical properties of 2P-RSA and 8P-RSA, but also the practical consequences of the implementation pathway used in this study. The 2P-RSA configuration benefited from

the optimized routines available in the native System.Security.Cryptography library, whereas 8P-RSA depended on a custom implementation designed to support multi-prime key structures. As a result, the large encryption gap is best understood as an implementation-level performance outcome rather than proof that the multi-prime structure itself inherently slows RSA encryption. Even so, from a systems perspective, that distinction does not reduce the practical importance of the finding: when deployed in the tested software environment, 8P-RSA did not offer competitive encryption performance.

The most important result of the study appears in the decryption measurements shown in Table 4. Decryption is the operation where structural differences between the two variants should matter most, and the results confirm that it is also where the largest performance divergence occurs. In 2P-RSA, decryption follows the standard CRT decomposition shown in Equations (4)–(6), requiring two modular exponentiations followed by a single CRT recombination. In 8P-RSA, decryption follows the generalized multi-prime process represented by Equation (10), which must be applied across eight prime factors before the plaintext can be reconstructed. Theoretically, multi-prime RSA is often motivated by the idea that several exponentiations over smaller primes may reduce private-key computational cost. The present results show that this theoretical advantage is not automatic. In this implementation, the overhead of performing eight residue computations, managing intermediate values, and executing generalized CRT recombination across eight branches outweighed any benefit gained from shorter operands. The effect became especially clear at 2048 bits, where 8P-RSA decryption rose sharply relative to both its 1024-bit counterpart and to 2P-RSA at the same key size. This suggests that the generalized CRT pathway used for 8P-RSA introduces a scalability problem in practice, particularly when implemented in a managed-language environment without native multi-prime optimization.

These findings are important because they help clarify why the present study differs from prior work that reported benefits for multi-prime RSA in certain contexts [22], [23]. Those studies were conducted under different implementation conditions, including embedded platforms and alternative optimization strategies, and therefore should not be assumed to generalize directly to desktop-class systems or to a VB.NET/.NET Framework 4.8 environment. The current results imply that the performance of multi-

prime RSA is highly dependent on implementation maturity, library support, and platform characteristics. In a system where two-prime RSA is backed by highly optimized native routines and eight-prime RSA must be implemented through custom arithmetic and CRT reconstruction, the expected benefit of smaller primes can be erased by software overhead. This also explains why the present study is valuable beyond the numerical results alone: it demonstrates that theoretical efficiency claims for multi-prime RSA must be validated in the actual deployment environment rather than assumed from mathematical structure alone [24].

From a practical standpoint, the data support the use of 2P-RSA as the more suitable option for 2048-bit deployments in environments similar to the one tested in this study. For applications that require repeated decryption, fast key turnover, or predictable runtime performance, the lower computational cost of 2P-RSA makes it the more attractive design [25]. This is especially relevant for server-side workloads such as TLS termination, certificate processing, and authentication systems, where private-key operations are often the limiting factor in throughput. By contrast, the results suggest that 8P-RSA would be difficult to justify on performance grounds alone in this environment. Any decision to adopt it would require compelling benefits outside the scope of this study, such as specialized architectural support or a different implementation framework capable of exploiting multi-prime arithmetic more efficiently.

Several limitations should be considered when interpreting these results. First, the comparison was conducted on a single hardware and software platform, so the findings should not be generalized automatically to other operating systems, processor architectures, or cryptographic libraries. Second, the 8P-RSA implementation was custom-built, whereas the 2P-RSA implementation relied on a mature native library; this asymmetry reflects real deployment conditions in .NET, but it also means that part of the observed gap may arise from optimization differences rather than from the mathematical design alone. Third, the study focused only on processing time and did not examine memory usage, energy consumption, side-channel resistance, or fault tolerance. Fourth, the study deliberately did not perform a full security analysis, so no claim is made here that 2P-RSA is more secure than 8P-RSA or vice versa. The conclusions of this section are therefore limited to empirical performance under controlled benchmark conditions [25].

Future research should extend this comparison using optimized multi-prime implementations, additional programming environments, and larger sample sizes. It would also be valuable to test 3072-bit and 4096-bit moduli, to evaluate memory and energy overhead, and to compare desktop-class results with server-grade and embedded platforms. A separate security-focused study would also be necessary to examine whether any trade-offs in factorization resistance or implementation security arise when increasing the number of prime factors while keeping the modulus size fixed. Until such work is completed, the present findings support a clear practical conclusion: within the tested environment, Dual-Prime RSA delivers better overall performance and better scalability than Eight-Prime RSA, particularly at the 2048-bit standard that is most relevant for current real-world deployment.

4. CONCLUSION

This study empirically demonstrated that Dual-Prime RSA (2P-RSA) consistently outperforms Eight-Prime RSA (8P-RSA) across all three cryptographic operations — key generation, encryption, and decryption — at both 1024-bit and 2048-bit key lengths, with the most pronounced gap appearing at the 2048-bit standard where 8P-RSA key generation required approximately 7.4 times more processing time and decryption incurred more than 35 times the latency of its 2P-RSA counterpart. These findings provide the first controlled, cross-variant benchmark covering all three operations at industry-standard key lengths on desktop-class hardware, establishing a concrete empirical basis for the claim that increasing the prime count from two to eight introduces substantial computational overhead without a corresponding operational advantage under current implementation conditions. For system architects and practitioners deploying RSA at the 2048-bit standard, 2P-RSA remains the more practical choice where processing speed and scalability are primary concerns. Future work should investigate whether a fully optimized, parallelized 8P-RSA implementation — one that eliminates the maturity gap between the native library used for 2P-RSA and the custom module used for 8P-RSA in this study — can meaningfully close the observed performance gap, or whether the structural cost of eight-factor CRT reconstruction is inherently prohibitive at this key length.

REFERENCES

[1] S. Fatima, T. Rehman, M. Fatima, S. Khan, and M. A. Ali, "Comparative Analysis of Aes and Rsa Algorithms for Data Security in Cloud Computing †," *Engineering Proceedings*, vol. 20, no. 1, 2022, doi: 10.3390/engproc2022020014.

[2] U. Rathod, "Comparative study between RSA algorithm and its variants: inception to date," in *Advances in Intelligent Systems and Computing*, Springer, 2020, doi: 10.1007/978-981-15-7961-5_14.

[3] N. Triagung, E. Hermawan, E. Winarko, and A. Ashari, "Eight Prime Numbers of Modified RSA Algorithm Method for More Secure Single Board Computer Implementation," vol. 11, no. 6, 2021.

[4] M. A. Islam, Md. A. Islam, N. Islam, and B. Shabnam, "A Modified and Secured RSA Public Key Cryptosystem Based on 'n' Prime Numbers," *Journal of Computer and Communications*, vol. 06, no. 03, pp. 78–90, 2018, doi: 10.4236/jcc.2018.63006

[5] D. M. Ghadi, "Improving the robustness of RSA encryption through input-based key generation," *Math. Model. Eng. Probl.*, vol. 11, no. 1, pp. 217–223, 2024, doi: 10.18280/mmep.110124.

[6] I. Talunohi, I. Jaki, S. Sutarman, and A. Candra, "Performance comparison analysis of multi prime RSA and multi power RSA," CESS (J. Comput. Eng. Syst. Sci.), vol. 8, no. 2, p. 576, 2023, doi: 10.24114/cess.v8i2.47350.

[7] R. Sulaiman, C. Kirana, T. Sugihartono, Laurentinus, and F. P. Juniawan, "RC4 algorithm and steganography to double secure messages in digital image," in Proc. 2020 8th Int. Conf. Cyber and IT Service Management (CITSM), Pangkal, Indonesia, 2020, pp. 1–4, doi: 10.1109/CITSM50537.2020.9268833.

[8] R. Sulaiman, B. Isnanto, Hengki, and C. Kirana, "Cryptography in (LSB) method using RC4 algorithm and AES algorithm in digital image to improve message security," in *Proc. Int. Conf. Comput., Eng., Design (ICCED)*, Bangkok, Thailand, 2018, pp. 29–34, doi: 10.1109/ICCED.2018.00016.

[9] Z. Y. Dzahabi *et al.*, "Cryptography of ChaCha20 and RSA algorithms for text security," *J. Comput. Netw. Archit. High Perform. Comput.*, vol. 7, no. 1, pp. 290–301, 2025, doi: 10.47709/cnahpc.v7i1.5345.

[10] V. R. Kebande, "Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function," in IEEE Access, vol. 11, pp. 114220-114237, 2023, doi: 10.1109

[11] S. J. Siregar *et al*., "RSA algorithm implementation for employee salary data security," *J. SAINTIKOM*, vol. 22, no. 2, pp. 528–538, 2023, doi: 10.26555/jiteki.v7i3.22210.

[12] Y. Liu, W. Gong, and W. Fan, "Application of AES and RSA hybrid algorithm in e-mail," in Proc. 2018 IEEE/ACIS 17th Int. Conf. Comput. Inf. Sci. (ICIS), Singapore, 2018, pp. 701–703, doi: 10.1109/ICIS.2018.8466380.

[13] National Institute of Standards and Technology (NIST), *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication (FIPS) 140-3, Mar. 2019. doi: 10.6028/NIST.FIPS.140-3.

[14] R. Zeng and L. Wang, "Twin DH key exchange protocol," *IET Inf. Secur.*, vol. 14, no. 6, pp. 764–772, 2020, doi: 10.1049/iet-ifs.2020.0047.

[15] M. Kara *et al*., "Secure key exchange against MITM attacks," *J. Ilmiah Tek. Elektro Komput. Informatika*, vol. 7, no. 3, pp. 380–387, 2021, doi: 10.26555/jiteki.v7i3.22210.

[16] E. F. Garcia and G. H. Singh, "Security assessment of multi-prime RSA against current factoring algorithms," in *Proc. 2021 IEEE Int. Conf. Cryptogr. Secur. (ICCS)*, Dec. 2021, pp. 456–461.

[17] R. Sulaiman, B. Isnanto, Hengki and C. Kirana, "Cryptography in (LSB) Method Using RC4 Algorithm and AES Algorithm in Digital Image to Improve Message Security," 2018 International Conference on Computing, Engineering, and Design (ICCED), Bangkok, Thailand, 2018, pp. 29-34, doi: 10.1109/ICCED.2018.00016.

[18] K. Balasubramanian, M. Arun, and K. R. Sekar, "An improved RSA algorithm for enhanced security," *International Journal of Cryptography and Network Security (IJCNS)*, vol. 2, no. 2, pp. 1–4, Jan. 2024, doi: 10.54105

[19] S. . Rath, "AES-RSA: An Innovative Hybrid Security Framework for File Authentication, Integrity, and Data Secrecy Model", *Int J Intell Syst Appl Eng*, vol. 12, no. 18s, pp. 303–312, Mar. 2024.

[20] R. Imam, 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. PP, pp. 1–1, Nov. 2021, doi: 10.1109/ACCESS.2021.3129224.

[21] N. T. E. Hermawan, E. Winarko, and A. Ashari, "Eight Prime Numbers of Modified RSA Algorithm Method for More Secure Single Board Computer Implementation", *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 11, no. 6, pp. 2375–2384, Dec. 2021.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[22] I. G. Talunohi, I. J. Lubis, S. Sutarman, and A. Candra, "Performance Comparison Analysis of Multi Prime RSA and Multi Power RSA", *Com, Engine, Sys, Sci*, vol. 8, no. 2, pp. 576–582, Jul. 2023, doi: 10.24114/cess.v8i2.47350.

[23] K. Pavani and P. Sriramya, "Enhancing Public Key Cryptography using RSA, RSA-CRT and N-Prime RSA with Multiple Keys," *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, Tirunelveli, India, 2021, pp. 1-6, doi: 10.1109/ICICV50876.2021.9388621.

[24] S. K. R., P. Klnc, and K. Srm, "An improved cryptanalysis of multi-prime RSA with specific forms of decryption exponent," *Cryptologia*, vol. 49, no. 1, pp. 1–14, 2025, doi: 10.1080/01611194.2023.2271463.

[25] M. Esmaeildoust, V. Zarei, and A. Kaabi, "Efficient Implementation of Multi-prime RSA using Montgomery Multiplication", *Int. J. Sci. Res. Comp. Sci. Eng.*, vol. 8, no. 5, pp. 16–19, Oct. 2020.