ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

Optimized Hybrid Homomorphic Signed-Integers Encryption: Addressing Challenges and Enhancing Capabilities

Abdulrahman Tawfeeq Jalal and Mohammed Anwar Mohammed

Department of Computer Science, College of Science, University of Sulaimani, Sulaymaniyah, Iraq

ABSTRACT

Homomorphic cryptography produces encrypted data that supports computation, but current exponential techniques face critical limitations: they leak plaintext information for negative values and fail on trivial values (0, ±1). Additional challenges include private key disclosure, encryption failure under specific conditions, and poor performance and storage efficiency. This paper addresses these limitations through comprehensive analyses, identifying the mathematical conditions that cause encryption failure, and examining their relationship to the modulus's randomness. A hybrid encryption approach is proposed in which the linear technique complements the exponential technique, particularly for negative and trivial values. The scheme uses the Carmichael function $\lambda(n)$ to reduce computational costs and provides a unified decryption algorithm that supports signed-integer operations. The unified decryption formula with rounding operations successfully handles positive and negative values, enabling true homomorphic subtraction alongside existing addition and multiplication properties. Decryption is optimized using a single prime key, enhancing security and reducing complexity. Experimental verification shows that the proposed technique passes all 15 National Institute of Standards and Technology tests, with probability values ranging from 0.122325 to 0.991468. Improvements include a 71.8% reduction in exponent size, 89.5% faster encryption, 67.2% faster decryption, and a 33.4% reduction in storage requirements, making the proposed hybrid technique suitable for resource-constrained secure computation applications.

Index Terms: Homomorphic Encryption, Signed Integer Encryption, Subtractive Homomorphic Property, Carmichael Function, National Institute of Standards and Technology Statistical Test Suite

1. INTRODUCTION

Online data privacy is of paramount importance, especially with the increasing demand for third-party applications. While customers benefit from these services, their uploaded data are exposed to third-party services [1]. Data can only be considered secure if it possesses three characteristics: confidentiality, integrity, and availability [2]. In multiparty environments such as the cloud, all parties – providers and users – are vulnerable to threats [3]. Encryption is an effective component of information security systems, used to ensure integrity against data breaches and protect against unauthorized access [4]. Therefore, storing data in encrypted form is a minimum requirement for data security [3], and complete privacy is achieved only when confidentiality is guaranteed during transmission, storage, and processing. However, traditional encryption methods require decryption before any computation can be performed, exposing plaintext during processing and creating a fundamental security vulnerability [1].

Homomorphic encryption addresses this critical limitation by enabling computation directly on encrypted data without

Access this article online

DOI:10.21928/uhdjst.v10n1y2026.pp116-129

E-ISSN: 2521-4217

P-ISSN: 2521-4209

Corresponding author's e-mail: Abdulrahman Tawfeeq Jalal, Department of Computer Science, College of Science, University of Sulaimani, Sulaymaniyah, Iraq. E-mail: abdulrahman.jalal@univsul.edu.iq

Received: 24-12-2025 Accepted: 23-03-2026 Published: 27-04-2026

decryption, thereby maintaining data confidentiality even during processing [5]. This represents a transformative paradigm in encryption systems, enabling secure operations on sensitive information in cloud computing and data-sharing applications [6]. In contrast, traditional approaches – while effective for protecting data at rest and in transit – are not designed for computation on encrypted data. Symmetric encryption, such as Advanced Encryption Standard (AES), offers high-speed performance for large data [1], and asymmetric methods such as Rivest–Shamir–Adleman (RSA) provide secure key-exchange mechanisms [1], but neither supports homomorphic computation. Exact homomorphic encryption preserves the algebraic structure of plaintext operations within the ciphertext, allowing external parties to perform calculations without accessing the underlying plaintext [6].

This paper examines current homomorphic cryptography systems, exploring their theoretical potential and practical limitations in security, functionality, and performance. It addresses these limitations by identifying weaknesses, expanding capabilities, and improving efficiency through mathematical analyses and practical experiments. Section 2 reviews the literature on symmetric, asymmetric, hybrid, advanced, and homomorphic cryptographic techniques, building the theoretical foundation that exposes the vulnerabilities driving this work. Section 3 details the problem statement. Section 4 presents materials and methods. Section 5 presents experimental results, with further implications in Section 6.

2. LITERATURE REVIEW

2.1. Symmetric Encryption

The strength of symmetric encryption lies in its efficiency, whereas using the same key for both encryption and decryption is a security weakness. Therefore, secret keys are encrypted asymmetrically as an effective, more secure strategy, balancing the speed of symmetric encryption with the security of asymmetric encryption [7].

The AES is a symmetric encryption standard known for strong security and high-speed performance, capable of processing large amounts of data without significant computing resources [8]. It has proven to be more secure than the data encryption standard (DES) or 3-DES (Triple-DES), offering faster encryption with larger security keys [9], with execution times of 0.01363 ms/0.01020 ms [10]. It protects data stored on disks and transferred between servers in cloud environments [8]. However, it has key management challenges related to distribution and storage [8], and despite its high performance, it is less secure than asymmetric encryption [9].

DES and 3-DES are both symmetric. DES is preferred for decryption due to lower complexity [8], whereas 3-DES suits use cases such as client-server communications [11]. Both face increased computational overhead as data size grows and performance concerns when combined with RSA [8].

2.2. Asymmetric Encryption

In asymmetric encryption, a public key is used for encryption and a private key for decryption, with the private key kept secure and not shared [7]. Compared to symmetric encryption algorithms, asymmetric encryption algorithms are more secure, as the public key cannot be used for decryption and is also used for digital signatures. However, they are considered relatively complex and slow-performing, as encryption and decryption take longer, making them suitable for encrypting small amounts of data at a time [7].

RSA's main strength is that the public key can be shared freely without risk [8]. It is used primarily for encrypting small data volumes such as encryption keys in hybrid systems [8], with speeds of 0.01817 ms/0.02310 ms [10], and is not preferred for large data due to complexity [8]. Resource-intensive pair-based encryption operations exploit the limited power of mobile devices [12].

Elliptic curve cryptography (ECC) is characterized by its small key size [9], providing greater security for small keys than other public key techniques [10], and is suited for mutual authentication in radio frequency identification systems [8]. However, it is slower and takes longer to complete [9], as its elliptic curve operations are slow and negatively impact performance [8].

Kyber (quantum-resistant) is an asymmetric method transformed into a secure key encapsulation mechanism for Indistinguishability Under Adaptive Chosen-Ciphertext Attack (IND-CCA2) [10]. One of its use cases is in the initial login phase for secure transactions in healthcare systems. However, as with other asymmetric algorithms, it has higher computational complexity than symmetric methods like AES-GCM, which is why it is used primarily for the initial secure key encapsulation, while symmetric methods handle subsequent data transfers [10].

2.3. Hybrid Encryption

Kyber can also be used with AES-GCM as a hybrid (symmetric + asymmetric) encryption method, where

Kyber's key-encapsulation mechanism algorithm is used in the login phase, and AES-GCM handles subsequent session transactions [10]. This achieves computational costs of 43.69–118.99% lower than comparable protocols, but remains complex in implementation [10].

ECC with AES is another hybrid type that reduces encryption and decryption time by approximately 74% [9], with a 100 MB file taking En: 0.957s, De: 0.898s versus En: 1.428s, De: 1.414s [9]. It is used in blockchain-based electronic health record (EHR) systems.

2.4. Advanced Encryption

Attribute-based encryption (ABE) enables precise sharing of ciphertexts and data access [13]. It includes schemes such as ciphertext-policy ABE, key-policy ABE, and ciphertext-policy weighted ABE [13]. It is used for multiuser data sharing in healthcare and cloud-based EHR storage. However, it lacks fine-grained access control for sovereign patient-centric [14], and patient-side key generation imposes a significant burden [15].

Proxy re-encryption converts ciphertext to a recipient's key without revealing plaintext to the intermediary [14], used in data sharing across multiple domains and delegated access scenarios [15]. However, its reliance on a single agent creates a single point of failure, making the entire mechanism vulnerable if the agent is compromised [8].

2.5. Homomorphic Encryption

Cheon *et al.* proposed a homomorphic scheme for approximate arithmetic operations, enabling addition and multiplication of encrypted messages [16]. The CKKS asymmetric homomorphic scheme supports approximate arithmetic on complex and real numbers [16, 17]. It operates on integer polynomial rings, encoding input $z \in C^{\{N/2\}}$ into a polynomial $m(X) \in Z[X]/(X^N+1)$, making the plaintext space $R = Z[X]/(X^N+1)$ [18]. The robustness of CKKS depends on the ratio N/q, where N is the degree of the polynomial and q is the coefficient modulus [19]; choosing wrong parameters reduces security and requires increasing N, raising computational cost [19].

Omar and Abed [20] proposed an exponential additive and multiplicative homomorphic asymmetric encryption scheme for secure cloud storage, relying on Fermat's Little Theorem and Euler's Theorem. The private key consists of two large primes (p, q), with encryption $C = Pubey + M^{(ek+1)} \bmod pk$ and decryption $M = C \bmod n$, where $n = p \times q$. The constraint requires that messages and their operations remain smaller than Pk [20].

Jalal and Mohammed [21] investigated critical mathematical vulnerabilities in linear homomorphic cryptography, which allow unauthorized decryption using simple factors beyond the message value. They proposed an enhanced linear technique with a windowing mechanism (W: Prime Set windows), demonstrating that it ensures decryption relies solely on the assigned secret key with higher randomness, while preserving additive and multiplicative homomorphic properties [21].

3. PROBLEM STATEMENT

3.1. Key Disclosure

In homomorphic encryption techniques, which rely on private key parameters (*mod n*, *mod p*, or *mod q*) for decryption, any mistakenly accepted out-of-range input messages result in private key disclosure. This is a critical vulnerability: a single incorrectly decrypted message is sufficient to expose the secret key. The attacker only needs to observe the incorrect decryption result and compare it with the original message.

3.2. Handling Negative Values

These homomorphic techniques, which rely on modulo operations in decryption, inherently do not handle negative values, as all modulo results are positive when the input is restricted to positive numbers [22]. This means they lack the subtractive homomorphic property, since subtracting large values from small ones produces negative results.

3.3. Encryption Failure

Exponential homomorphic encryption techniques, particularly those that use a modular exponentiation structure, may fail to encrypt—resulting in a ciphertext identical to the plaintext—when specific mathematical relationships exist between the message M and the random modulus parameter [23]. Clear constraints on random values must be established through a thorough mathematical analysis of the failure conditions and mitigation strategies.

3.4. Multiplication Function Selection

Exponential homomorphic encryption techniques suffer from performance and storage issues, as large exponents result in excessively long encryption times and larger ciphertext sizes. The key question is: can we use a smaller exponent while

maintaining cryptographic correctness? Techniques based on Euler's $\varphi(n)$ as an exponent can be improved using a more efficient function when a mathematical basis is established that yields equivalent results [24].

4. MATERIALS AND METHODS

4.1. Analysis

Before diving into the mathematical details, it is important to understand what this encryption scheme accomplishes and why certain design choices create vulnerabilities. The exponential homomorphic technique accepts as input a Message $M < pk$, primes p and q, and large random values z and w [20]. Critically, this design allows "M" to exceed n (since $pk = n \times z$ and $M < pk$), which leads to key exposure risks and encryption failures detailed in the following section

4.1.1. It outputs a ciphertext C through the following process

- Key generation:
  - $n \leftarrow p \times q$
  - $\varphi(n) \leftarrow (p-1)\,(q-1)$, where φ is Euler's Theorem
  - $ek \leftarrow w \times \varphi(n)$
  - $pk \leftarrow n \times z$
  - $Pubkey = pk \times ek$
- Encryption: $C \leftarrow Pubkey + M^{(ek+1)} \bmod pk$
- Decryption: $M \leftarrow C \bmod n$

To understand why decryption works, the correctness of this scheme relies on a fundamental property of number theory called Euler's theorem, φ, and the greatest common divisor (*gcd*):

$$M^{\varphi(n)} \equiv 1 (\bmod n), if\ gcd\,(M, n) = 1 \tag{1}$$

We now show how the decryption process recovers the original message. Starting with $C \bmod n$:

$$M^{(\varphi(n)\,w+1)} \bmod n \tag{2}$$

This can be separated into two parts:

$$M^{(\varphi(n)\,w)} \bullet M^{1} \bmod n \tag{3}$$

$$(M^{\varphi(n)})^{w} \bullet M \bmod n \tag{4}$$

The key insight here is that $M^{\varphi}(n) \equiv 1 \bmod n$ based on Equation (1), which simplifies to:

$$1^{w} \bullet M \bmod n \tag{5}$$

$$M \bmod n = M \tag{6}$$

The main implication is that encryption and decryption are reversible, ensuring that the original message is correctly recovered when the input satisfies certain conditions. However, as we show in the following sections, this design has several critical flaws.

4.1.2. Risk behind out-of-range messages

To understand how this exposure occurs, consider what happens when an out-of-range message is processed. The current technique accepts $M < pk$, meaning M can be greater than n. Examining a negative out-of-range input $M \in [-n+1, -1]$:

$$M = -k,\ where\ k \in [1,\ n]$$

$$-k \bmod n = n-k = m_{decrypted} \tag{7}$$

$$|M-m_{\text{decrypted}}| = |-k-(n-k)| = |-n| = n \tag{8}$$

Hence, the mathematical relationship directly exposes the secret key using the original input and incorrect decryption result because of the wrap-around mod property [22].

We now demonstrate this with a concrete example. Given: $p = 19259$, $q = 54101$, $z = 472938476$, $w = 835297183$, $n = 1041931159$, $pk = 492769334434373684$, $ek = 870260885426577400$, $M = 1041931999 < Pk$, out of $[0, n\text{-}1]$:

$$C = M^{ek+1} \bmod pk = 359808971229199551$$

$$M = C \bmod n = 359808971229199551 \bmod 1041931159 = 840$$

$$n = |M-m_{\text{decrypted}}| = |1041931999-840| = 1041931159$$

The attacker recovers $n = 1041931159$, the secret key, from a single incorrect decryption. Based on Equation (6), when $M \geq n$, then M is outside this range, and no calculation $\bmod n$ can ever equal M. This mismatch becomes a vulnerability: The difference between input and output leaks the key, completely compromising the encryption scheme.

4.1.3. Handling negative values

Since the exponent is part of the encryption, dealing with negative values depends on whether the exponent is odd or even. We first determine whether $ek+1$ is even or odd. Since p and q are odd primes, both $(p-1)$ and $(q-1)$ are even:

$$\varphi(n) \leftarrow (p\text{-}1)\,(q-1) = even \times even = even$$

$$ek \leftarrow w \times \varphi(n) = w \times even = even \ (regardless\ of\ w)$$

$$ek+1 = even + 1 = odd$$

$$Exponent\ part = M^{(ek+1)} = -M^{(odd)} = -(M^{(odd)}) = Negative\ result$$

The critical flaw is that negative ciphertext results from negative plaintext encryption, leaking information about the plaintext, and making the exponential technique unsuitable for encrypting negative values. In contrast, the linear homomorphic encryption [21] handles negative values correctly: When random values w and z are non-negative integers with $w + z \geq 1$, the cipher C is always positive even if M is negative.

$$C = M+p\ (w+z \times W\ (Prime\ Set)) \tag{9}$$

Given $M \in [-p/2, p/2]$ and big prime private key $p > 0$

$$C \geq p\left(randomness - \frac{1}{2}\right) \tag{10}$$

However, decryption using modular p maps all integers to the range $[0, p-1]$. Therefore, this representation cannot distinguish between positive and negative values, as shown in Table 1.

The value 4 is ambiguous, as it may represent either 4 or −3 when negative values are allowed. A signed interpretation is therefore required, reinterpreting the standard range $[0, p-1]$ as a signed range centered around zero, where $[0, \lfloor p/2 \rfloor]$ represents positive values and $[\lfloor p/2 \rfloor+1, p-1]$ represents negative values [25]. To convert a modular result $M \in [0, p-1]$ to the signed range $[-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor]$:

$$\text{Shift up: } M' = M+\lfloor p/2 \rfloor \Rightarrow 3, 4, 5, 6, 7, 8, 9$$

$$\text{Wrap around: } M'' = M' \bmod p \Rightarrow 3, 4, 5, 6, 0, 1, 2$$

$$\text{Shift down: } S = M''-\lfloor p/2 \rfloor \Rightarrow 0, 1, 2, 3, -3, -2, -1$$

Therefore, a hybrid technique can be implemented using the following selection criteria:

- Linear homomorphic encryption for: $M \in (-\lfloor p/2 \rfloor, ..., -1)$ (negative values)
- Exponential homomorphic encryption for: $M \in (2, 3, ..., \lfloor p/2 \rfloor)$ (positive values ≥ 2)

Note: The handling of zero and one is discussed in Section 4.1.3, Case 3.

4.1.4. Encryption failure

We identified a critical flaw: under certain conditions, encryption completely fails. In some experiments, $M^{(ek+1)} \equiv M \ (mod\ pk)$, meaning no encryption is performed – the output is simply M again [20]. To understand when and why this occurs, we must examine the structure of pk. The critical part is $pk = n \times z$, by Euler's theorem, as shown in Equations (2), (3), (4), (5), and (6), $M^{(ek+1)} \equiv M \ (mod\ n)$ is always satisfied. Therefore, security depends entirely on the modulus z.

The failure condition depends on the relationship between M and z, which we analyze in Cases 1 and 2. In addition, Case 3 identifies specific trivial values of M that always fail regardless of z.

- Case 1, when gcd (M, z) = 1: Failure occurs only when the exponent aligns with the multiplicative order. Given integers M, a, and b where $gcd\ (M, b) = 1$, $M^{(a+1)} \equiv M \ (mod\ b)$ if and only if "a" equals $ord\_b(M)$ or a multiple of it, where $ord\_b(M)$ is the smallest positive integer k such that $M^k \equiv 1 \ (mod\ b)$ [26].

We now show why this relationship holds:

$$M^{(a+1)} \equiv M \ (mod\ b) \tag{11}$$

$$M^{(a+1)}-M \equiv 0 \ (mod\ b) \tag{12}$$

Since $gcd\ (M, b) = 1$, we can divide by M:

$$M(M^a-1) \equiv 0 \ (mod\ b) \tag{13}$$

$$M^a \equiv 1 \ (mod\ b) \tag{14}$$

where $a = ord\_b(M)$ or $k \times ord\_b(M)$ for any positive integer k. In our case, if $gcd\ (M, z) = 1$, then $M^{(ek+1)} \equiv M \ (mod\ z)$ if and only if $ord\_z(M) \mid \varphi(n) \times w$:

$$\varphi(n)w+1 \equiv 1 (mod\ ord_z\ (M)) \tag{15}$$

TABLE 1: Standard modular reduction (e.g., mod 7)

Input value	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5
Result (mod 7)	0	1	2	3	4	5	6	0	1	2	3	4	5

$$\varphi(n)w \equiv 0(mod\ ord_z(M)) \quad (16)$$

$$ord\_z(M) \mid \varphi(n)w \quad (17)$$

Example demonstrating this: $M = 8$, $b = 15580052599$, $a = 25610241600$

$$M^{(a+1)}\ mod\ b = 8^{(25610241600+1)}\ mod\ 15580052599 = 8$$

Check if $ord_b(M)$ divides "a":

$$ord_b(M) = ord_{15580052599}(8) = 7129800$$

$$\frac{a}{ord_b(M)} = \frac{25610241600}{7129800} = 3592\ (exact\ division)$$

This confirms the failure condition: because "a" is a multiple of $ord\_b(M)$, specifically $a = 3592 \times 7129800$, then for $M = 8$, all values:

$8^{(k*7129800+1)} = 8\ (mod\ 15580052599)$, for $k = 1,2,3,\ldots,3592,\ldots$,etc.

- Case 2, when gcd (M, z) > 1: The situation becomes more complex when M and z share common factors. $M^{(ek+1)} \equiv M\ (mod\ z)$ requires every prime factor in z to satisfy a "bad" condition; if even one prime in z prevents this, encryption succeeds.

For common primes (primes that appear in both M and z), when M contains p^a (p^a divides M), z contains p^b (p^b divides z), and $gcd\ (M, z) = p$, then:

- If $a \geq b$: This prime p allows the failure $M^{(ek+1)} \equiv M\ (mod\ z)$, as both M and $M^{(ek+1)}$ are divisible by p^b
- If $a < b$: This prime p prevents the failure, as M is NOT divisible by p^b, but $M^{(ek+1)}$ is divisible by p^b. However, the "w" random value is not effective in this case.

For non-common primes (primes only in z, not in M), the behavior mirrors Case 1:

- If $ord\_p(M)$ divides $ek = \varphi(n)w$: This prime allows the failure $M^{(ek+1)} \equiv M\ (mod\ z)$
- If $ord\_p(M)$ does NOT divide ek: This prime prevents the failure

We illustrate this with an example where all factors satisfy the failure condition: $M = 502332$ (492×1021) $< n$, $n = 1022117$, $\varphi(n) = 1020096$, $z = 23483$ (1021×23).

For common primes (prime 1021^1, a ≥ b):

$$\frac{M}{gcd(M,z)} = \frac{502332}{1021} = 492,\ divisable$$

$$\frac{M^{(ek+1)}}{gcd(M,z)} = \frac{502332^{(74467008+1)}}{1021} = 0,\ divisable$$

For non-common primes (prime 23):

$$ord_{23}(M) = 11$$

$$\frac{ek}{ord_{23}(M)} = \frac{74467008}{11} = 6769728,\ divisable$$

Since all z factors satisfy the failure condition, then:

$$M^{(ek+1)}\ mod\ pk = 502332^{(74467008+1)}\ mod\ 24002373511 = 502332 = M$$

However, changing z's structure to $z = 1042441$ (1021^2, $a < b$):

$$M^{(\varphi(n)*w+1)}\ mod\ (n * z)$$

$$502332^{(1020096*w+1)}\ \text{mod}\ (1022117 * 1021^2) = 660587564613 \neq M$$

Nevertheless, different "w" values lead to the same output; "w" is not effective in this case.

- Case 3, zero and one encryption: This technique [20] is also unable to correctly encrypt the trivial values −1, 0, and 1. The result of $M^{(ek+1)}$ mod pk is simply M, and adding a public key is useless since it is a simple linear transformation. This mirrors a known limitation in textbook RSA, which fails to encrypt trivial values (0 and 1) due to its modular exponentiation structure. Standard RSA implementations mitigate these vulnerabilities by introducing randomness via padding mechanisms [27]. However, padding or hashing methods are incompatible with homomorphic encryption, as they destroy the algebraic properties necessary for homomorphic operations.

Our proposed solution uses the observation that the linear homomorphic encryption in [21] shares the same decryption algorithm as the exponential technique. A hybrid system is implemented in which the encryption method is selected based on the input: if the input is negative, zero, or one, the linear method is used; otherwise, the exponential method

is used with the same private key. Decryption succeeds regardless of which method was used.

- Summary of encryption failure conditions: The analysis shows that encryption fails under three distinct conditions: (i) when $gcd\ (M, z) = 1$ and ord_z(M) divides $\varphi(n)$w (Case 1) – mathematically unavoidable yet empirically rare; (ii) when $gcd\ (M, z) > 1$ and all prime factors of z satisfy specific alignment conditions (Case 2) – diminishing rapidly as z's size and randomness increase; and (iii) for trivial inputs $M \in \{-1, 0, 1\}$ (Case 3) – always failing and requiring hybrid linear-exponential handling. Among these, Case 1 is the dominant residual failure mode in practice, as it can occur even with secure parameters due to multiplicative-order alignment, motivating the check-and-retry strategy adopted in this work.

4.1.5. Carmichael's function versus Euler's totient

We now compare two candidate functions to identify which provides better efficiency. Carmichael's function $\lambda(n)$ is defined as the least common multiple (lcm):

$$\lambda(n) = lcm(p-1, q-1) \tag{18}$$

This function satisfies the fundamental property:

$$M^{\lambda(n)} \equiv 1 (mod\ n), for\ gcd(M,n) = 1 \tag{19}$$

The main implication here is that both $\varphi(n)$ and $\lambda(n)$ meet our encryption requirements. Based on Equation (1), (18), encryption using $\lambda(n)$ produces a valid decryption [24].

To understand the efficiency gain, consider their mathematical relationship. The connection between $\varphi(n)$ and $\lambda(n)$ is:

$$\lambda(n) = \frac{(p-1)(q-1)}{gcd\ (p-1, q-1)}, \varphi(n) \leftarrow (p-1)(q-1) \tag{20}$$

$$\varphi(n) = \lambda(n) \bullet gcd(p-1, q-1) \tag{21}$$

This reveals that $\lambda(n) \leq \varphi(n)$, with equality when $gcd\ (p-1, q-1) = 1$. Since $\lambda(n) \mid \varphi(n)$, using Carmichael's function as the exponent yields improvements in both performance and storage.

4.1.6. Complexity analysis

Relying on p and q values in encryption to mimic RSA can be counterproductive. In RSA, decryption requires knowing both p and q values [28]. Whereas in [20] technique, decryption is possible with either p or q. This introduces unnecessary complexity without providing additional security benefits. Since security is based on the weakest link, the algorithm's strength then depends on the length of the shortest key [29]. Therefore, it is preferable to optimize such algorithms and eliminate the additional complexity by restricting decryption to a single value.

4.1.7. Common factor attack

In the linear homomorphic encryption technique, if z and w share common factors, those factors can be exploited in decryption [21], reintroducing the same vulnerability the technique was designed to prevent.

When common factor $f = gcd\ (z, w)$, then $z = f \times a$, and $w = f \times b$ for some integers a and b where $gcd\ (a, z/f) = 1$ and $gcd\ (b, w/f) = 1$. Applying this to Equation (8), as follows:

$$\mathrm{M} + p\ (f \times b + f \times a \times W\ (Prime\ Set))$$

$$M + f \times p\ (b + a \times W\ (Prime\ Set))$$

$$C \equiv \mathrm{M}\ (mod\ f \times p)$$

This means that any divisor of $(f \times p)$ greater than M can be used in decryption instead of the private key "p". Therefore, an additional constraint should be added: $gcd\ (z, w) = 1$, to ensure that no value other than "p" can be used to reduce C and recover M.

4.2. Design

4.2.1. Exponential technique

Regarding the exponential homomorphic technique, we redesigned it to have the input message $M \in [2, \lfloor p/2 \rfloor]$:

- Key generation:
  - $\lambda(n) \leftarrow$ lcm $(p-1, q-1)$, where λ is Carmichael's function
  - $w, z \leftarrow$ secure randoms
  - $ek \leftarrow w \times \lambda(n)$
- Encryption:
  - $C \leftarrow M^{(ek+1)}\ mod\ (p \times z)$
  - if $C = M$, then regenerate z and retry (validation)
  - return C
- New decryption: $M \leftarrow \mathrm{C} - p \times \lfloor C/p \rfloor$

4.2.2. Parameter constraints

The following constraints can be enforced to guarantee scrambling in the exponential technique: "z" should be a cryptographically large secure prime; $gcd\ (M, z) = 1$; and $ord\_z(M) \nmid \varphi(n)w$. However, these restrictions add overhead

from GCD computations and prime factorization checks, affecting performance especially at large bit parameters.

Rather than enforcing strict constraints, we propose a hypothesis: Modulus z-value characteristics, such as length and secure randomness, are inversely related to the encryption failure rate. Given the low failure rate with a large, secure random z-value, the cost of occasional retries is much lower than the cost of validating the new fixed constraints. Hence, when a failure occurs ($M^{(ek+1)} \bmod pk = M$), regenerate z and retry. This hypothesis is directly motivated by the failure analysis in Section 4.1.3, which shows that only Case 1 remains theoretically possible under large random parameters, whereas Case 2 failures are expected to diminish as the entropy and size of z increase.

Furthermore, as discussed in Section 4.1.5, limiting decryption to a single prime eliminates unnecessary complexity. This is implemented by computing modulo ($p \times z$) rather than ($n \times z$), so decryption requires only the prime p. The other prime q is still used to compute ek for the $M^{(ek+1)}$ term.

Regarding the linear technique, as established in Section 4.1.6, the only necessary constraint is that $gcd\ (n,\ z \times W\ (Prime\ Set)) = 1$. This ensures that no divisor other than p is used in decryption.

4.2.3. Unified decryption algorithm

$$(C \bmod p) + \lfloor p/2 \rfloor,\ Shifting\ up \tag{22}$$

$$((C \bmod p) + \lfloor p/2 \rfloor) \bmod p \tag{23}$$

$$(((C \bmod p) + \lfloor p/2 \rfloor) \bmod p) - \lfloor p/2 \rfloor,\ Shifting\ down \tag{24}$$

Definition of modulo:

$$(r = C \bmod p = C - p(\lfloor C/p \rfloor), remainder{:}0 \le r < p \tag{25}$$

$$q` = \lfloor C/p \rfloor,\ quotient \tag{26}$$

$$r = C - q` \bullet p \tag{27}$$

$$C = q` \bullet p + r \tag{28}$$

Return to Equation (23):

$$((r + \lfloor p/2 \rfloor) \bmod p) - \lfloor p/2 \rfloor \tag{29}$$

$$((C - q` \bullet p + \lfloor p/2 \rfloor) \bmod p) - \lfloor p/2 \rfloor \tag{30}$$

$$((C + \lfloor p/2 \rfloor - q` \bullet p) \bmod p) - \lfloor p/2 \rfloor \tag{31}$$

Apply the Modular Arithmetic Property:

$$(a - b * x) \bmod x = a \bmod x \tag{32}$$

Return to Equation (30):

$$((C + \lfloor p/2 \rfloor - 0) \bmod p) - \lfloor p/2 \rfloor \tag{33}$$

$$(C + \lfloor p/2 \rfloor) \bmod p) - \lfloor p/2 \rfloor \tag{34}$$

Applying Equation (24):

$$((C + \lfloor p/2 \rfloor) - p\lfloor (C + \lfloor p/2 \rfloor)/p \rfloor) - \lfloor p/2 \rfloor \tag{35}$$

$$(C + \lfloor p/2 \rfloor - \lfloor p/2 \rfloor - p\lfloor (C + \lfloor p/2 \rfloor)/p \rfloor \tag{36}$$

$$(C - p\lfloor (C + \lfloor p/2 \rfloor)/p \rfloor \tag{37}$$

$$(C - p\lfloor C/p + \lfloor p/2 \rfloor/p \rfloor \tag{38}$$

$$(C - p\lfloor C/p + 0.5 \tag{39}$$

Using the math property:

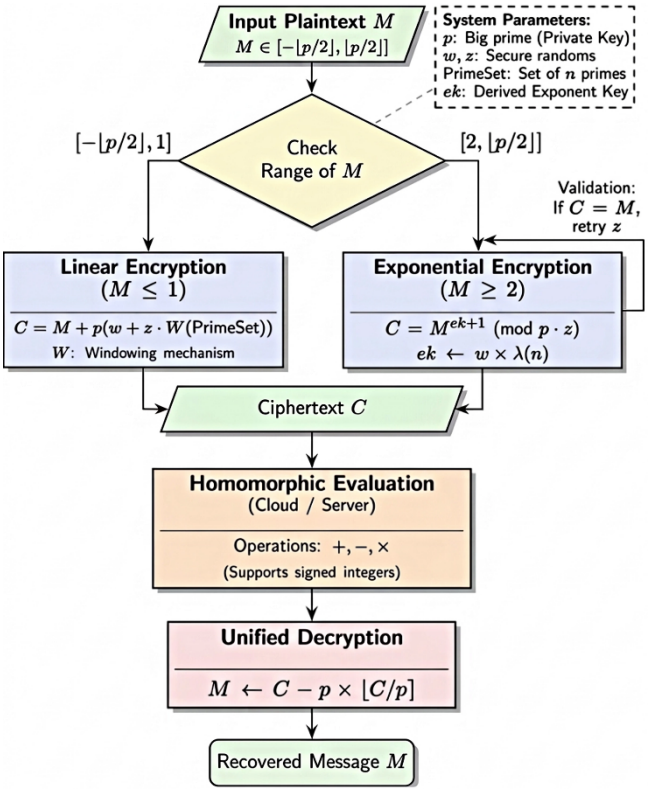

Fig. 1. The hybrid technique data flow with dual-path encryption determined based on the input plaintext *M*, with unified decryption that retrieves signed integers.

$$\lfloor x + k \rfloor = round(x) \text{ when } k \approx 0.5 \quad (40)$$

$$C - p \times round\,(C/p) = M \quad (41)$$

This decryption formula correctly interprets negative results when M stays within $[-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor]$ (Fig. 1). The formula has structural similarity to $M = C \bmod p = C - p \times floor\,(C/p)$, differing only in the rounding function. This does not imply equal efficiency to the built-in mod: modern Intel processors implement mod as a single division instruction returning quotient and remainder in separate registers [30]. However, calculating $C - p \times floor\,(C/p)$ requires separate division, multiplication, and subtraction operations.

The connection to mod enables further optimization. When the fractional part of C/p is <0.5, rounding and flooring are equal, verified as follows:

- Input: ciphertext C, modulus p, precomputed $pHalf = p/2$
- $r \leftarrow C \bmod p$ //Fast hardware operation
- if $r < pHalf$ then
  - return r //Round = floor
- else
  - return $r - p$ //Adjust for rounding up
- Output: plaintext M

In this case, we use fast modulo operations to reduce the complexity to a single comparison and/or subtraction.

4.3. Proposed Hybrid Technique

A hybrid technique that combines linear and exponential encryption paths to handle signed integers. The encryption path is selected based on the input value, ensuring the security and integrity of all valid plaintexts. As shown in Figure 1, both paths converge to a single ciphertext that supports homomorphic properties. At the same time, the unified decryption consistently recovers the original signed integer, regardless of the chosen encryption path.

4.4. Testbed

- Hardware configuration
  - Operating System: Windows 10
  - Architecture: 64-bit operating system, ×64-based processor
  - Processor: Intel(R) Core (TM) i7-6600U CPU @ 2.60GHz 2.81 GHz
  - Memory: 8.00 GB installed RAM (7.88 GB usable)
- Software environment
  - Programming language: Java (version 20.0.2)
  - Cryptographic libraries: Java Security API for Secure Random Implementation
- Analysis tools:
  - Cygwin64 terminal environment for running the National Institute of Standards and Technology Statistical Test Suite (NIST STS)

5. RESULTS

5.1. Experiment 1

5.1.1. Parameter configurations

- Primes: $p = 4062405317$ (32-bit), $q = 3955932973$ (32-bit)
- Carmichael function: $\lambda(n) = 4017650783798119788$ (62-bit)
- Plaintext space: $M \in [-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor] = [-2031202658, 2031202658]$

5.1.2. Randoms

- $w_1 = 4260346636$ (32-bit), $z_1 = 4068323483$ (32-bit), $ek_1 = 171165850013770829419308331 68$ (94-bit)
- $w_2 = 4215194085$ (32-bit), $z_2 = 4210434652$ (32-bit), $gcd\,(w_2, z_2) = 1$
- $w_3 = 4021833325$ (32-bit), $z_3 = 4061806978$ (32-bit), $ek_3 = 161583218104916482357203351 00$ (94-bit)
- $w_4 = 4248854822$ (32-bit), $z_4 = 4255573487$ (32-bit), $ek_4 = 170704149058527207357774177 36$ (94-bit)

5.1.3. Encryption

- $M_1 = 12 \rightarrow$ Exponential, $C_1 = 7615210262883489257$ (63-bit)
- $M_2 = -10 \rightarrow$ Linear, $C_2 = 34228318980256794619$ (62-bit)
- $M_3 = 3 \rightarrow$ Exponential, $C_3 = 12093672008115634057$ (64-bit)
- $M_4 = 100 \rightarrow$ Exponential, $C_4 = 12922534408151227245$ (64-bit)

5.1.4. Homomorphic evaluation

- Expected plaintext result: $M_1 \times M_2 + M_3 - M_4 = 12 \times (-10) + 3 - 100 = -217$
- Homomorphic operation: $C_1 \times C_2 + C_3 - C_4 = 260655845979701269881232390925106314895$

5.1.5. Decryption

- Using prime p
  - $round(C/p) = 64162934429248451547660376136$
  - $p \times round(C/p) = 260655845979701269881232390925106315112$
  - $M = C - p \times round(C/p) = -217$ ✓ (Correct)

- Using prime q
  - $round(C/q)$ = 6588985398861086059185674451 6
  - $q \times round(C/q)$ = 2606558459797012698812323909 23281326068
  - $M = C - q \times round(C/q)$ = 1824988827✗ (Incorrect)
- Using modulus n = $p \times q$ = 16070603143210817441
  - $round(C/n)$ = 16219419001073265037
  - $n \times round(C/n)$ = 2606558459797012698851972532 79615110317
  - $M = C - n \times round(C/n)$ = −396486235450879542 2✗ (Incorrect)

5.2. Experiment 2

This experiment validates the hypothesis and analyzes the effect of z-value characteristics on the encryption failure rate through a systematic test.

5.2.1. Configurations

- Prime sizes: $p, q \in [10^{17}, 10^{18}]$ (590-bit)
- Secure random bit-length: w-value $\in [10^{12}, 10^{14}]$ across all tests, whereas z-value is generated randomly using different bit-lengths
- Samples: 100,000 M samples chosen randomly per test, $M \in [2, 10^{17}]$
- Three independent tests with increasing z bit-lengths

5.2.2. Methodology

- Secure random generation
- Encryption: Compute $C = M^{(ek+1)} \bmod pk$
- Verification: Check if $C = M$ (failure, either Case 1 or Case 2 occurred) or $C \neq M$ (success)

5.2.3. Results

To verify the behavior at larger z-values, we tested another 300,000 samples with z-values $\in [10^{17}, 10^{18}]$. Result: 1 failure – specifically, M = 11, z = 530969872552609280, w=150384577100847752, n = 821635022676689453093884 34558577857, $\varphi(n)$ = 821635022676689446738587673873 61280, confirmed as Case 1.

5.3. Experiment 3

Significant changes were made to the exponential technique; a NIST statistical test must be performed to detect non-randomness that could compromise cryptographic security. The NIST STS framework evaluates the randomness of binary sequences through 15 different tests [31].

For statistically valid results across all 15 tests, the minimum bitstream length must be at least 1 million bits, as recommended by NIST. Tested samples are converted to binary representation and compiled into a single continuous bitstream. This can be prepared as a text file in ASCII format, where each bit is stored as a "0" or "1," or as a binary file that packs every eight bits into one byte [31].

The assess command requires two parameters: the bitstream length n (size of each test sequence) and the number of sequences m. The total tested bits $n \times m$ are divided into m separate sequences for each statistical evaluation [31].

Each test produces a P-value indicating sequence randomness, with significance level $\alpha = 0.01$. $P \geq 0.01$ indicates randomness with 99% confidence; a $P < 0.01$ suggests non-randomness [31].

5.3.1. Cryptographic parameters

- Prime p, q: 512 bits
- Modulus $n = p \times q$: 1024 bits
- $\lambda(n)$: 1023 bits
- Random parameters w, z: Generated fresh per encryption

5.3.2. Test configuration

- Sample size: 10,000 ciphertexts
- Message bit size: Up to 64 bits per plaintext
- Message range: $[-p/2, p/2]$ excluding $\{-1, 0, 1\}$
- Total bits tested: 10,240,000 bits
- Bitstream format: 10 bitstreams of 1,024,000 bits each
- Normalization: Each ciphertext is normalized to 1024 bits with zero-padding

5.3.3. Implementation environment

- Encryption implementation: Java
- NIST STS execution: Cygwin64 terminal environment
- Test command: "./assess 1024000" with 10 bitstreams in ASCII format

Table 2 presents aggregated results for all 188 test variants: 8 primary tests, 2 cumulative sums variants, 148 non-overlapping template variants, 8 random excursions variants, 18 random excursions variant variants, 2 serial variants, 1 approximate entropy, and 1 linear complexity test.

5.4. Experiment 4

This experiment compares performance and storage between two exponential technique implementations: the

TABLE 2: Summary of aggregated outcomes for all 188 *National Institute of Standards and Technology* statistical test suite variants

Test category	Variants	Min *P*-value	Max *P*-value	Pass rate	Status
Frequency	1	0.350485	0.350485	10/10	PASS
Block frequency	1	0.213309	0.213309	10/10	PASS
Cumulative sums	2	0.534146	0.911413	10/10	PASS
Runs	1	0.534146	0.534146	10/10	PASS
Longestrun	1	0.122325	0.122325	10/10	PASS
Rank	1	0.534146	0.534146	10/10	PASS
Fft	1	0.350485	0.350485	10/10	PASS
Non-overlapping template	148	0.000199	0.991468	8-10/10	PASS
Overlapping template	1	0.534146	0.534146	9/10	PASS
Universal	1	0.739918	0.739918	9/10	PASS
Approximate entropy	1	0.350485	0.350485	10/10	PASS
Random excursions	8	—	—	4/4	PASS
Random excursions variant	18	—	—	4/4	PASS
Serial	2	0.534146	0.911413	10/10	PASS
Linear complexity	1	0.534146	0.534146	10/10	PASS

TABLE 3: Comparison of performance metrics (encryption/decryption time) and storage efficiency (ciphertext size) between the old and the new optimized exponential encryption techniques across 10 independent rounds

Round*	λ/φ Ratio	OLD Enc. (ms)	NEW Enc. (ms)	OLD Dec. (µs)	NEW Dec. (µs)	Size Δ (%)
1	0.083	7.87	0.83	9.91	2.44	−33.4
2	0.125	8.01	0.84	7.81	2.19	−33.4
3	0.500	8.00	0.83	6.05	2.13	−33.5
4	0.002	8.06	0.84	7.96	1.97	−33.4
5	0.500	7.62	0.80	4.54	2.06	−33.4
6	0.050	7.79	0.81	6.28	2.09	−33.4
7	0.500	7.90	0.83	5.60	2.05	−33.4
8	0.500	7.56	0.80	6.90	1.98	−33.4
9	0.056	7.79	0.82	5.60	2.39	−33.5
10	0.500	7.64	0.81	4.81	2.19	−33.4
Mean	0.282	7.83	0.82	6.55	2.15	−33.4
Std. dev.	0.232	0.17	0.01	1.55	0.15	0.0

*Each round uses independently generated 512-bit primes *P* and *q*

original using $pk = n \times z$, $ek = w \times \varphi(n)$, with random z and no validation; and the proposed technique using $pk = p \times z$, $ek = w \times \lambda(n)$, with a secure, large random z and $C \neq M$ validation.

Both were tested using 512-bit p and q private keys, and the same bit length for the z-value. The benchmark consisted of 10 independent rounds, each with its own p and q keys, and 5000 encryption/decryption operations per round. Messages were generated randomly with lengths ranging from 8 to 32 bits, after a 20-iteration JVM warm-up. Encryption and decryption times, and ciphertext size,

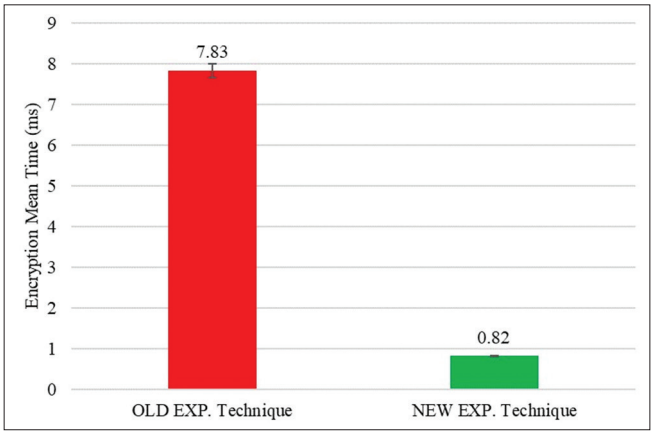

Fig. 2. Comparison of the average encryption times before and after implementing the new exponential technique.

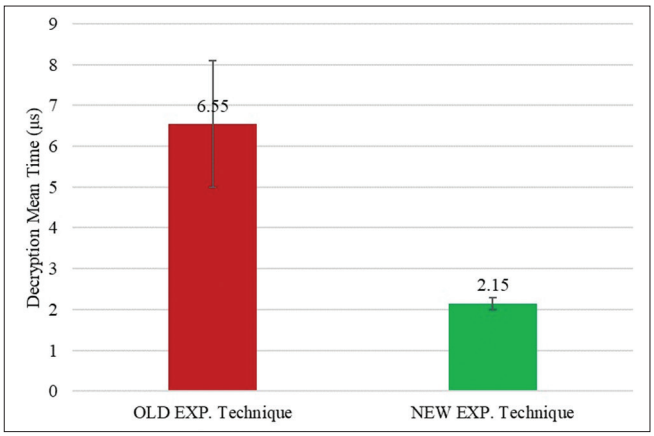

Fig. 3. A comparison between the old and new exponential techniques, with respect to average decryption time.

were measured as means with standard deviations across all rounds (Table 3).

Table 4 presents a performance comparison across 10 rounds of testing involving 100,000 operations. The new technique demonstrates improvements, with encryption time decreasing from 7.83 ± 0.17 ms to 0.82 ± 0.01 ms, representing an 89.5% speed increase as shown in Fig. 2. Decryption performance also improved by 67.2%, with time decreasing from 6.55 ± 1.55 μs to 2.15 ± 0.15 μs (Fig. 3). The ciphertext size decreased by 33.4%, from an average of 1532 bits to 1020 bits (Fig. 4). The average $\lambda(n)/\phi(n)$ ratio across all rounds was 0.2816, indicating a 71.8% reduction in exponent size when

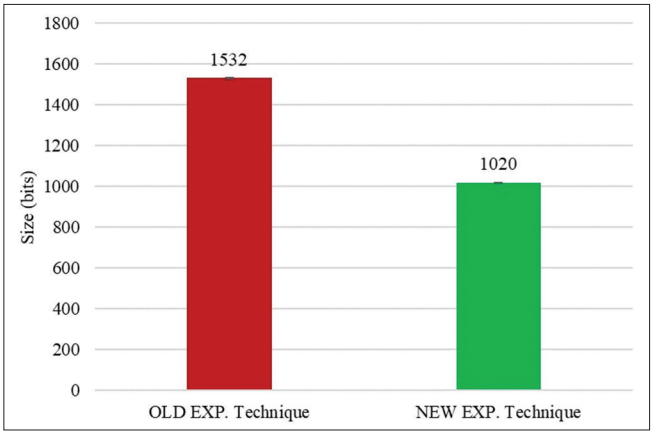

Fig. 4. A comparison of the average ciphertext size between the old and new exponential techniques.

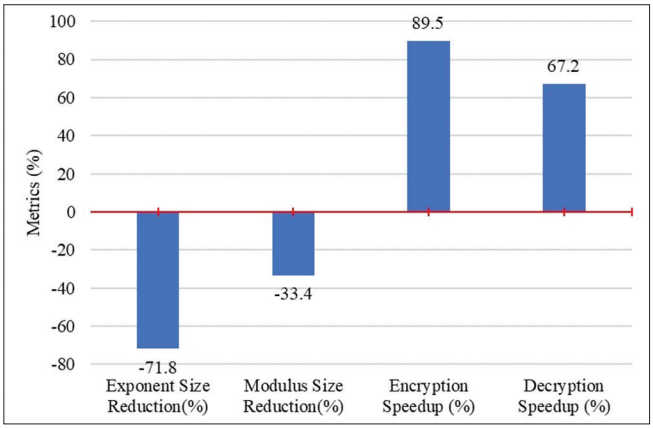

Fig. 5. Exponent and modulus optimization impact on performance.

using the Carmichael function compared to Euler's factor. Ratios ranging from 0.0024 to 0.5, depending on the prime factorization of $(p-1)$ and $(q-1)$. Fig. 5 shows how reduction contributes to performance improvement.

6. DISCUSSION

Experiment 1 confirms several features and reveals a new homomorphic capability. The unified decryption formula successfully recovered signed-integer plaintext encrypted using either linear or exponential approaches. Decryption succeeded using p alone, whereas attempts using q or $n = p \times q$ failed, confirming the single-prime optimization of Section 4.2.2 and demonstrating that relying on p alone provides greater security. Handling negative numbers by expanding the plaintext space to $[-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor]$ enables true homomorphic subtraction, where $C\_sub = C_1 - C_2 - ... - C_n$ correctly decrypts to negative or positive values. This extends the homomorphic properties of the original techniques to include subtraction, provided the constraint $(M_1 - M_2 - ... - M_n) \in [-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor)$ is satisfied.

Experiment 2 confirms that failure rates decrease significantly as z increases: increasing z from 10^6 to 10^{10} reduces failures from 0.483% to 0.010% (Table 4). Both cases decrease proportionally – Case 1 from 209 to 3, and Case 2 from 274 to 7. Importantly, Case 1 dominates in practice at cryptographically relevant parameter sizes, whereas Case 2 becomes statistically negligible at large z values. At larger z values ($\in [10^{17}, 10^{18}]$), Case 1 still holds, as illustrated by the failure case of $M = 11$. This validates the design choice of probabilistic retry when the $C \neq M$ check fails rather than deterministic constraint enforcement.

Regarding NIST tests, Table 3 shows that all 15 statistical tests passed, and all probability values exceeded the significance level of $\alpha = 0.01$, ranging from 0.122325 to 0.991468. Furthermore, all tests achieved the minimum pass rate for the ten tested binary sequences (8/10 for standard tests, 3/4 for Random Excursions). Regarding the non-overlapping template analysis: out of 148 template variants, only two cases showed marginal probability values of 0.000199; however, all binary sequences achieved a pass rate of 10/10. The

TABLE 4: Encryption failure rates versus *z*-range

z-Range	Samples	Success	Failure	Failure rate (%)	Case 1	Case 2
10^6–10^8	100000	99517	483	0.483	209	274
10^8–10^{10}	100000	99781	219	0.219	97	122
10^{10}–10^{12}	100000	99990	10	0.010	3	7

remaining 146 variables passed the test with probability values ranging from 0.017912 to 0.991468. Passing all NIST tests confirms the method satisfies randomness requirements and produces cryptographically strong ciphertexts.

Experiment 4 shows significant performance improvements and reduced storage requirements. Using the Carmichael function $\lambda(n)$ instead of Euler's $\varphi(n)$ reduces exponent size by an average of 71.8%, leading to fewer multiplications while ensuring correct encryption and decryption. These improvements remain consistent across different $\lambda(n)/\varphi(n)$ ratios (0.002–0.500). The smaller coefficient ($p \times z$ vs. $n \times z$) improves encryption speed and reduces ciphertext size, directly impacting storage and transmission costs while restricting decryption to a single key. For decryption, using the optimized equation $M = C - p \times \lfloor C/p \rfloor$, the 67.2% faster decryption is due to the ciphertext size difference. The 33.4% reduction in storage space represents significant cost savings for applications that process large amounts of encrypted data.

7. CONCLUSIONS

This research addresses the fundamental limitations of homomorphic cryptography through mathematical analysis and experimentation. We demonstrate that the proposed technique addresses critical challenges, including handling negative values, cryptographic failure under specific mathematical conditions, and computational inefficiencies.

The key contributions are: Identifying the precise conditions under which homomorphic exponential cryptography fails; defining the relationship between modulus random value characteristics and encryption failure rate; enabling true homomorphic subtraction using a unified decryption algorithm with rounding rather than flooring, expanding the plaintext space to include signed integers and marginal cases ($0, \pm 1$); and reducing complexity while enhancing security through Carmichael function $\lambda(n)$ selection and single-key decryption restriction. All statistical randomization tests were passed, and performance criteria showed significant improvements, confirming practical feasibility in resource-constrained environments. These results have broad implications for cloud computing, where calculations on sensitive encrypted data preserve user privacy.

This study has limitations: it supports a limited set of homomorphic arithmetic operations, requires careful management of plaintext range to prevent overshooting in complex multistep computations, and our evaluation focused on controlled experiments rather than production deployment. Future work should address: supporting additional operations such as division and comparison; exploring hybrid combinations with approximate cryptographic methods for floating-point handling; developing adaptive parameter-length selection mechanisms that balance security and performance according to application needs; and validating under various attack scenarios in production environments.

With growing global concerns about data privacy, effective and practical homomorphic encryption systems have become essential. This research contributes to that goal, and we hope it inspires continued innovation in secure data processing without compromising privacy.

REFERENCES

[1] K. Munjal and R. Bhatia. "A systematic review of homomorphic encryption and its contributions in healthcare industry". *Complex and Intelligent Systems*, vol. 9, no. 4, pp. 3759-3786, 2023.

[2] C. K. Yee and M. F. Zolkipli. "Review on confidentiality, integrity and availability in information security". *Journal of ICT in Education*, vol. 8, no. 2, pp. 34-42, 2021.

[3] S. Murthy and C. R. Kavitha. "Preserving Data Privacy in Cloud using Homomorphic Encryption". In: *2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, pp. 1131-1135, 2019.

[4] A. A. Abdullah, R. Khalaf and M. Riza. "A realizable quantum three-pass protocol authentication based on hill-cipher algorithm". *Mathematical Problems in Engineering*, vol. 2015, pp. 1-6, 2015.

[5] A. V. Kumar, M. S. Sujith, K. T. Sai, G. Rajesh and D. J. S. Yashwanth. "Secure Multiparty Computation Enabled E-Healthcare System with Homomorphic Encryption". In: *IOP Conference Series: Materials Science and Engineering*. IOP Publishing, p. 022079, 2020.

[6] W. Guo, J. Shao, R. Lu, Y. Liu and A. A. Ghorbani. "A privacy-preserving online medical prediagnosis scheme for cloud environment". *IEEE Access*, vol. 6, pp. 48946-48957, 2018.

[7] Q. Zhang. "An Overview and Analysis of Hybrid Encryption: The Combination of Symmetric Encryption and Asymmetric Encryption". In: *2021 2nd International Conference on Computing and Data Science (CDS)*. IEEE, pp. 616-622, 2021.

[8] A. Pattanshetti. "Review of encryption techniques to enhance data protection in cloud". *Vidhyayana-An International Multidisciplinary Peer-Reviewed E-Journal*. vol. 10, no. si4, pp. 84-98, 2025

[9] A. Haddad, M. H. Habaebi, E. A. Elsheikh, M. R. Islam, S. A. Zabidi and F. E. M. Suliman. "E2EE enhanced patient-centric blockchain-based system for EHR management". *Plos One*, vol. 19, no. 4, p. e0301371, 2024.

[10] B. M. Singh and J. Natarajan. "A novel secure authentication protocol for e-health records in cloud with a new key generation

method and minimized key exchange". *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 7, p. 101629, 2023.

[11] J. L. Fernández-Alemán, I. C. Señor, P. Á. O. Lozoya and A. Toval. "Security and privacy in electronic health records: A systematic literature review". *Journal of Biomedical Informatics*, vol. 46, no. 3, pp. 541-562, 2013.

[12] M. Ali, A. Abbas, M. U. S. Khan and S. U. Khan. "SeSPHR: A methodology for secure sharing of personal health records in the cloud". *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 347-359, 2021.

[13] H. Li, K. Yu, B. Liu, C. Feng, Z. Qin and G. Srivastava. "An efficient ciphertext-policy weighted attribute-based encryption for the internet of health things". *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 5, pp. 1949-1960, 2022.

[14] B. Rodrigues, I. Amorim, I. Silva and A. Mendes. "Patient-centric health data sovereignty: An approach using proxy re-encryption". In: S. Katsikas, F. Cuppens, N. Cuppens-Boulahia, C. Lambrinoudakis, J. Garcia-Alfaro, G. Navarro-Arribas, P. Nespoli, C. Kalloniatis, J. Mylopoulos, A. Antón and S. Gritzalis, Eds., *Computer Security. ESORICS 2023 International Workshops, Lecture Notes in Computer Science*. vol. 14398. Springer Nature, Cham, Switzerland, pp. 199-215, 2024.

[15] H. Jin, Y. Luo, P. Li and J. Mathew. "A review of secure and privacy-preserving medical data sharing". *IEEE Access*, vol. 7, pp. 61656-61669, 2019.

[16] J. H. Cheon, A. Kim, M. Kim and Y. Song. "Homomorphic encryption for arithmetic of approximate numbers". In: T. Takagi and T. Peyrin, Eds., *Advances in Cryptology - ASIACRYPT, Lecture Notes in Computer Science*. vol. 10624. Springer International Publishing, Cham, pp. 409-437, 2017.

[17] D. Huynh. "*CKKS Explained, Part 3: Encryption and Decryption - OpenMined*". Available from: https://openmined.org/blog/ckks-explained-part-3-encryption-and-decryption [Last accessed on 2025 Nov 03].

[18] D. Huynh. "*CKKS Explained: Part 1, Vanilla Encoding and Decoding*". OpenMined. Available from: https://openmined.org/blog/ckks-explained-part-1-simple-encoding-and-decoding [Last accessed on 2025 Nov 03].

[19] D. Huynh. "*CKKS Explained, Part 5: Rescaling*". OpenMined. Available from: https://openmined.org/blog/ckks-explained-part-5-rescaling [Last accessed on 2025 Nov 03].

[20] Z. T. Omar, F. S. Abed and S. K. Ahmed. "A new asymmetric fully homomorphic encryption scheme for cloud banking data". *Kurdistan Journal of Applied Research*, vol. 5, no. 2, pp. 152-165, 2020.

[21] A. T. Jalal and M. A. Mohammed. "Enhanced integer-based homomorphic encryption scheme with windowing mechanism for securing electronic health records". *UHD Journal of Science and Technology*, vol. 9, no. 2, pp. 77-91, 2025.

[22] "*Modulo*". Available from: https://en.wikipedia.org/w/index.php?title=modulo&oldid=130366402 [Last accessed on 2025 Nov 03].

[23] "*Multiplicative Order*". Available from: https://en.wikipedia.org/wiki/multiplicative_order#citerefnivenzuckermanmontgomery1991 [Last accessed on 2025 Nov 03].

[24] P. K. Siddharth, O. Pal and B. Alam. "A Homomorphic Encryption Scheme over Integers Based on Carmichael's Theorem". In: *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*. IEEE, Mysuru, India, pp. 17-20. 2016.

[25] T. Tosun, A. Moradi and E. Savas. "*Exploiting the Central Reduction in Lattice-Based Cryptography*". 2024. Available from: https://eprint.iacr.org/2024/066 [Last accessed on 2025 Nov 03].

[26] J. Chappelon. "On the multiplicative order a^n modulo n". *Journal of Integer Sequences*, vol. 13, Article 10.2.1, 2010.

[27] D. Boneh. "Twenty years of attacks on the RSA cryptosystem". *Notices of the American Mathematical Society*, vol. 46, pp. 203-213, 2002.

[28] 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.

[29] E. Barker. "*Recommendation for Key Management*: Part 1 – General." National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-57pt1r5, 2020.

[30] M. A. Will and R. K. Ko. "Computing Mod without Mod". Vol. 2014. IACR Cryptology ePrint Archive, California, p. 755, 2014.

[31] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, N. Heckert, J. Dray, S. Vo and L. E. Bassham. "*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*". National Institute of Standards and Technology, United States, 2010.