Optimizing Message Range and Ciphertext Storage in GSW Encryption Using CRT and PVW-like Compression Scheme

Kung-Wei Hu[1], Huan-Chih Wang[2], and Ja-Ling Wu[3]

[1] National Taiwan University, Taiwan
r11922141@cmlab.csie.ntu.edu.tw
[2] National Taiwan University, Taiwan
whcjimmy@cmlab.csie.ntu.edu.tw
[3] National Taiwan University, Taiwan
wjl@cmlab.csie.ntu.edu.tw

Abstract. This paper explores advancements in the Gentry-Sahai-Waters (GSW) fully homomorphic encryption scheme, addressing challenges related to message data range limitations and ciphertext size constraints. We introduce a novel approach utilizing the Chinese Remainder Theorem (CRT) for message decomposition, significantly expanding the allowable message range to the entire plaintext space. This method enables unrestricted message selection and supports parallel homomorphic operations without intermediate decryption. Additionally, we adapt existing ciphertext compression techniques, such as the PVW-like scheme, to reduce memory overhead associated with ciphertexts. Our experimental results demonstrate the effectiveness of the CRT-based decomposition in increasing the upper bound of message values and improving the scheme's capacity for consecutive homomorphic operations. However, compression introduces a trade-off, necessitating a reduced message range due to error accumulation. This research contributes to enhancing the practicality and efficiency of the GSW encryption scheme for complex computational scenarios while managing the balance between expanded message range, computational complexity, and storage requirements.

Keywords: Fully Homomorphic Encryption · GSW · Chinese Remainder Theorem · Ciphertext Compression

1 Introduction

Cryptography has significantly evolved with the advent of fully homomorphic encryption, a groundbreaking concept that enables computations on encrypted data without requiring decryption. This paper delves into the intricacies of the Gentry-Sahai-Waters (GSW) fully homomorphic encryption scheme [11], focusing on the challenges and solutions related to the upper bound of message data ranges and constraints on ciphertext size.

The GSW scheme supports homomorphic addition and multiplication, which is essential for performing complex computations on encrypted data. However, each operation increases the error in the ciphertext, potentially growing exponentially with successive operations. This necessitates mechanisms to control error growth. The research presented here explores methods to mitigate error accumulation, notably employing the Chinese Remainder Theorem (CRT) for message decomposition. This approach significantly extends the allowed message range to the entire plaintext space. However, when using the previous GSW variant decryption method [10], successful message decryption necessitated meticulous design, rendering the process unintuitive and cumbersome. Consequently, we opted for an approximation method [12]. While this approach may theoretically generate increased noise as the message value grows, the aforementioned message decomposition technique mitigates this issue. Furthermore, the approximation method offers enhanced intuitiveness and practicality.

Furthermore, current encryption systems face challenges related to extensive ciphertext storage requirements. Therefore, this study also examines compression techniques proposed in [11], aimed at reducing memory overhead associated with ciphertexts. As the increase in dimension size leads to exponential growth in time and space complexity, we adopt a vector form of the private key, differing from the matrix form presented in the original paper. Consequently, our compression focus shifts from multiple messages to a single message. Due to this adjustment, only the PVW-like compression scheme remains applicable from the two compression methods initially proposed in the paper, namely PVW-like and nearly square gadget matrix.

This paper makes several key contributions to the field of fully homomorphic encryption, explicitly focusing on improvements and extensions to the Gentry-Sahai-Waters (GSW) scheme:

1. We propose a novel approach to message range extension in the GSW scheme by implementing CRT for message decomposition. This method significantly expands the allowed message range to encompass the entire plaintext space, enhancing the scheme's versatility and applicability.
2. We use an approximation method to address the challenges of the previous GSW variant decryption method [12]. This approach not only improves the intuitiveness but also the practicality of the decryption process
3. We present a comprehensive examination of ciphertext compression techniques, focusing on reducing the memory overhead associated with ciphertexts. Our analysis adapts the PVW-like compression scheme to work effectively with our modified encryption system, which utilizes a vector form of the private key.

These contributions collectively enhance the practicality, efficiency, and applicability of the GSW fully homomorphic encryption scheme, paving the way for its broader adoption in secure computation scenarios.

1.1 Related Work

Fully Homomorphic Encryption (FHE). In contemporary society, the rapid development of the Internet, cloud computing, and artificial intelligence has led to a vast amount of personal data being stored and processed remotely. However, this practice raises significant security concerns. To mitigate these risks, data must be encrypted before being uploaded to the cloud.

FHE was first introduced by Rivest, Adleman, and Dertouzos [16] in 1978. However, it wasn't until 2009 that Gentry proposed the first FHE scheme [8], which allowed computations on plaintext without decryption. Nevertheless, it was often limited by excessive noise and large ciphertext dimensions, leading to decryption failures after a few computations and prolonged computation times. Subsequently, based on Gentry's ideas, various novel schemes were proposed.

In the survey paper [13], they categorize all current FHE schemes into four generations based on their proposal order and mention several simple applications. The second generation mentioned in the paper includes BV [2, 3], BGV [1], and B/FV [7] schemes. These can operate in finite fields and allow effective packing, meaning computations can be performed simultaneously on entire vectors of integers. However, they don't handle noise exceptionally well.

The third generation includes schemes like GSW (the focus of our research) [11], FHEW [6], and TFHE [5]. These schemes aim to address the issue of excessive noise. Their approach involves converting previous processing methods into bit-wise operations and improving the bootstrapping technique, proposed by Gentry in [8] and [9]. These methods can reduce error generation, but they lack packing support.

Finally, the most famous scheme in the fourth generation is CKKS [4]. Although its bootstrapping technique is relatively inferior to the third generation, it focuses on optimizing packing or batching, allowing multiple data to be processed simultaneously. This reduces the average computation time for individual operations. Moreover, it can be extended to operations on real numbers.

Ciphertext Compression. Ciphertext compression is a significant research area within cryptographic systems. Many encryption schemes face the challenge of producing substantially larger ciphertexts than plaintexts. Compressing ciphertexts can enhance data transmission and increase storage efficiency while maintaining the security of the encrypted data.

Initially, "ciphertext packing" was proposed [14, 1], which involves aggregating and encrypting multiple plaintexts into a single ciphertext. However, later research [10] introduced a method that effectively reduces the dimensionality of the ciphertext while still encrypting multiple plaintexts simultaneously, yielding better results than previous techniques.

1.2 Organization

This writeup is structured as follows: a) Section 2 provides the background knowledge for understanding the article. b) Section 3 introduces the GSW encryption process, thoroughly examining each critical design element. c) Sections 4 and 5 address two significant issues in the current GSW scheme: the limited user message data range and the large ciphertext size. We propose improvements or ideas for these problems. d) Sections 6 and 7 present the final experimental results and conclusions.

2 Background

2.1 Learning With Error (LWE)

The LWE problem represents a significant challenge in the field of cryptographic systems and is used to enhance security by adding noise. This technique masks precise information by injecting minimal noise, hindering unauthorized parties from straightforwardly deducing sensitive data. Mathematically, the LWE problem requires identifying a vector $\mathbf{s}$ in $\mathbb{Z}_q^n$ such that $\mathbf{sA} + \mathbf{e} \equiv \mathbf{b} \pmod{q}$,

$$[\mathbf{s}_1 \ \mathbf{s}_2 \ \cdots \ \mathbf{s}_n] \begin{bmatrix} \mathbf{a}_{1,1} & \mathbf{a}_{1,2} & \cdots & \mathbf{a}_{1,m} \\ \mathbf{a}_{2,1} & \mathbf{a}_{2,2} & \cdots & \mathbf{a}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_{n,1} & \mathbf{a}_{n,2} & \cdots & \mathbf{a}_{n,m} \end{bmatrix} + [\mathbf{e}_1 \ \mathbf{e}_2 \ \cdots \ \mathbf{e}_m] = [\mathbf{b}_1 \ \mathbf{b}_2 \ \cdots \ \mathbf{b}_m] \pmod{q}.$$

where $\mathbf{A}$ is a matrix in $\mathbb{Z}_q^{n\times m}$ with $n < m$, and $\mathbf{e}$ denotes a small error vector. Matrix $\mathbf{A}$ and vector $\mathbf{b}$ constitute the public key, while vector $\mathbf{s}$ acts as the secret key in cryptographic applications.

Without error vector $\mathbf{e}$, the linear equation $\mathbf{sA} = \mathbf{b}$ solution can be determined directly using Gaussian elimination. However, introducing $\mathbf{e}$ makes traditional methods such as Gaussian elimination ineffective, thus making the problem more complicated. This complexity benefits encryption, as it ensures the system's security as long as the secret key $\mathbf{s}$ remains undisclosed. Even if an adversary gains access to the public key components $\mathbf{A}$ and $\mathbf{b}$, extracting the secret key solely from these elements is exceedingly challenging. Consequently, the LWE problem is integral to the defense of cryptographic systems, significantly reducing the likelihood of direct attacks.

2.2 Gadget Matrices

The gadget matrix $\mathbf{G}$ plays a crucial role in the GSW (Gentry-Sahai-Waters) encryption scheme [11], especially when paired with ciphertext flattening or the *BitDecomp* operation G^{-1}. This matrix is characterized by its unique ability to revert the transformations applied by its inverse through the operation $\mathbf{G}(G^{-1}(C)) = C$, ensuring that the original ciphertext C is recoverable.

Typically, the structure of the gadget matrix $\mathbf{G}$ is defined as $\mathbf{G} = I_n \otimes \mathbf{g}$, where I_n is the identity matrix of size n, and $\mathbf{g}$ is a vector given by $\mathbf{g} = (1, 2, \cdots, 2^{\lfloor \log_2 q \rfloor})$. This definition forms $\mathbf{G}$ as a block matrix where each block is a scaled identity matrix with the scaling factors being powers of 2, up to

$2^{\lfloor \log_2 q \rfloor}$. The specific form of $\mathbf{G}$ is illustrated as follows:

$$\begin{bmatrix} 1 & 2 & \cdots & 2^{\lfloor \log_2 q \rfloor} & 0 & & & & \cdots & & & 0 \\ & & 0 & & 1 & 2 & \cdots & 2^{\lfloor \log_2 q \rfloor} & \cdots & & & 0 \\ & & \vdots & & & \vdots & & & \ddots & & & \vdots \\ & & 0 & & & 0 & & & \cdots & 1 & 2 & \cdots & 2^{\lfloor \log_2 q \rfloor} \end{bmatrix}.$$

This structured approach leverages the mathematical properties of the identity matrix and the geometric progression in $\mathbf{g}$ to facilitate the operations required in the GSW encryption system.

2.3 The Chinese Remainder Theorem

"有物不知其數，三三數之剩二，五五數之剩三，七七數之剩二。問物幾何？"

The above is a famous ancient Chinese Quiz; it presented an excellent example of the CRT, a fundamental principle in number theory used to determine an unknown number based on its remainders when divided by several coprime integers. Assuming integers $m_1, m_2, \cdots, m_n$ are pairwise coprime, then for any integers $r_1, r_2, \cdots, r_n$, the system of equations

$$C : \begin{cases} \mathbf{x} \equiv r_1 \pmod{m_1} \\ \mathbf{x} \equiv r_2 \pmod{m_2} \\ \vdots \\ \mathbf{x} \equiv r_n \pmod{m_n} \end{cases}. \tag{1}$$

has a solution, and the general solution can be constructed as follows:

1. Let $\mathbf{M} = m_1 \times m_2 \times \cdots \times m_n = \prod_{i=1}^{n} m_i$ be the product of integers $m_1, m_2, \cdots, m_n$, and let $\mathbf{M}_i = \mathbf{M}/m_i$, $\forall i \in \{1, 2, \cdots, n\}$, i.e., $\mathbf{M}_i$ is the product of the $n-1$ integers excluding m_i.
2. Let $\mathbf{t}_i = \mathbf{M}_i^{-1}$ be the modular multiplicative inverse of $\mathbf{M}_i$ modulo m_i: $\mathbf{t}_i\mathbf{M}_i \equiv 1 \pmod{m_i}$, $\forall i \in \{1, 2, \cdots, n\}$
3. The general solution of the system of Equation 1 takes the form: $x = r_1\mathbf{t}_1\mathbf{M}_1 + r_2\mathbf{t}_2\mathbf{M}_2 + \cdots + r_n\mathbf{t}_n\mathbf{M}_n + k\mathbf{M} = k\mathbf{M} + \sum_{i=1}^{n} r_i\mathbf{t}_i\mathbf{M}_i, k \in \mathbb{Z}$. In the sense of modulo $\mathbf{M}$, the system of Equation 1 has only one solution: $x = \sum_{i=1}^{n} r_i\mathbf{t}_i\mathbf{M}_i$.

This solution not only satisfies the original conditions but also demonstrates the power of the CRT in parallel solving such systems, provided the modules are coprime to each other. Let us consider a simple illustrative example in Appendix B.

2.4 Regev and PVW Encryption

In the Regev encryption scheme [15], a bit $\sigma \in \{0, 1\}$ is encrypted into a ciphertext $c \in \mathbb{Z}_q^{n+1}$. When this ciphertext is decrypted using a secret key $s \in \mathbb{Z}_q^{n+1}$ with the last coordinate being 1, the result is given by $\langle s, c\rangle = \lceil q/2 \rfloor \cdot \sigma + e \pmod q$, where the error term must be $|e| < q/4$. This is because the decryption process involves dividing the result by $q/2$, and if the residual error is less than $1/2$, rounding off the result will yield the correct value. The plaintext space is expanded to p, with $p < q$ to broaden the applicability. In this case, the encrypted message $\sigma \in \mathbb{Z}_p$ during decryption leads to $\langle s, c\rangle = \lceil q/p \rfloor \cdot \sigma + e \pmod q$, where the error now needs to be less than $q/2p$ to ensure that rounding the output yields the correct message σ.

The subsequent PVW encryption system [14] is more widely applicable, allowing the selected message format to be transformed into a vector $\sigma \in \mathbb{Z}_p^k$, or even a matrix $\sum \in \mathbb{Z}_p^{k\times m}$. Here, the decryption of the two ciphertexts results in $S \cdot c = \lceil q/p \rfloor \cdot \sigma + \mathbf{e} \pmod q$ and $SC = \lceil q/p \rfloor \cdot \sum + \mathbf{E} \pmod q$, with secret key $S = (S' \mid I) \in \mathbb{Z}_q^{n\times(n+k)}$ and now the conditions that the error terms $\mathbf{e}$ and $\mathbf{E}$ must be less than $q/2p$.

Both Regev and PVW encryption systems support homomorphic addition and multiplication, but a prerequisite is that the error must remain sufficiently small. Therefore, the plaintext space p cannot be excessively large, as it would otherwise be challenging to manage.

3 GSW-Like Encryption Scheme

This section introduces the GSW homomorphic encryption scheme [11], which, despite offering significant data security and privacy advantages, presents considerable challenges, particularly in managing errors and limiting message sizes.

One of the core challenges in homomorphic encryption is the accumulation of errors during computations in the encrypted domain. Each operation on ciphertexts introduces an error, which accumulates without intermediate decryption steps and can grow exponentially with the number of operations performed. This accumulated error can severely compromise the integrity of decrypted results, necessitating stringent control mechanisms. Consequently, messages in the GSW scheme are initially restricted to binary selection.

Subsequently, we considered limiting messages to a selection between 0 and 1 was overly restrictive. Therefore, we attempted to increase the upper bound of messages to ϕ. This adjustment still maintains secure encryption and accurate decryption of message values. Moreover, by introducing an *approximation method* [12] at the output stage of decryption, an additional layer of approximation error is added. Consequently, the feasible upper bound ϕ must be significantly reduced, further constraining the operational range and flexibility of the encryption scheme.

Key Generation. The generation of the public and secret keys begins with the random selection of the matrix A from $\mathbb{Z}_q^{(n-1)\times m}$, vector t from $\mathbb{Z}_q^{n-1}$, and a small noise vector e from the distribution χ^{n-1} conforms to an integer-normal modulo q, where $m = n \cdot \ell$ to accommodate an expanded message encoding scheme. The vector $\mathbf{b}$ is computed as $\mathbf{b} = tA + e$, following the LWE scheme. The public key $\mathbf{P}$ is then defined as $\mathbf{P} = \begin{bmatrix} -A \\ \mathbf{b} \end{bmatrix} \in \mathbb{Z}_q^{n\times m}$ and the secret key $\mathbf{s}$ as $\mathbf{s} = [t \mid 1] \in \mathbb{Z}_q^n$, ensuring that the product $\mathbf{s} \times \mathbf{P} \equiv e \pmod{q}$ confirms the error model of LWE.

Encryption. The encryption process commences with the selection of the message μ from $\{0, 1\}$, accompanied by a randomly generated matrix R from $\{0, 1\}^{m\times m}$. The ciphertext $\mathbf{C}$ is computed using the formula

$$\mathbf{C} = \mu \cdot \mathbf{G} + \mathbf{P} \times R \pmod{q} \in \mathbb{Z}_q^{n\times m},$$

where $\mathbf{G}$ typically represents a gadget matrix facilitating the embedding of μ within the encryption process.

The range for selecting the message μ was initially limited to between 0 and 1 due to error considerations during the design of the GSW encryption system. However, we later deemed this restriction overly stringent. Experimental results indicated that the range could be expanded. Therefore, we attempt to extend the message range to a new upper bound (denoted as ϕ), aiming to more efficiently exploit the fault tolerance of the scheme without compromising security. With the revised approach, encryption now begins with the selection of a message μ within the range $[0, \phi)$.

Table 1. Memory requirements for ciphertext before decryption.

n	8	9	10	11	12	13	14	15	16	17	18
Ciphertext Size(KB)	3.625	5.188	7.156	9.578	12.5	15.97	20.03	24.73	30.13	36.25	43.16
n	19	20	21	22	23	24	25	26	27	28	
Ciphertext Size(KB)	50.89	59.5	69.03	79.53	91.05	103.6	117.3	132.2	148.2	165.5	

Table 1 succinctly illustrates the memory space required to store the selected message's encrypted ciphertext under different security parameters. Here, n represents the security parameter, with higher values indicating greater security and the ciphertext size is measured in kilobytes (KB). The table shows that the required memory space also grows as the security parameter increases. Notably, even at $n = 8$, a message originally only a few bits in size requires 3.625 KB to store the ciphertext. This issue is not unique to the GSW encryption scheme; other existing encryption schemes exhibit similar behavior.

Decryption. Decryption is executed using the secret key $\mathbf{s}$ and the received ciphertext $\mathbf{C}$ as inputs. The decryption formula, as pre-defined, is written as

$$\mathbf{sC} = \mu \cdot \mathbf{s} \times \mathbf{G} + \mathbf{s} \times \mathbf{P} \times R \pmod{q}.$$

Let e' denote the accumulated error after the decryption is completed. This above formula can be simplified to:

$$\mathbf{sC} \equiv \mu \cdot \mathbf{s} \times \mathbf{G} + e' \pmod{q} \in \mathbb{Z}_q^m. \tag{2}$$

After applying the decryption formula, an output operation is necessary to derive the correct message value μ. This process involves separating $\mu \cdot \mathbf{s} \times \mathbf{G}$ from e' within the decrypted output.

The Decryption Output. In advancing the effectiveness of homomorphic encryption, particularly in the output stages where accurate decryption is crucial, two approaches will be investigated: a *variation of the GSW encryption scheme* [10] and an *approximate method* [12]. Each aimed to address the complex challenge of error management.

1. **A Variant of GSW [10].** The proposed variant of the GSW encryption scheme seeks to directly extract and eliminate the error component e' from the decryption result. The method involves a matrix $\mathbf{M}$ that satisfies two crucial properties:

   Property 1. $\mathbf{M} = G^{-1}(0)$ such that $\mathbf{GM} = 0 \pmod{q}$, effectively nullifying the message-related component in the matrix multiplication due to congruence modulo q.

   Property 2. $\mathbf{M}$ must be invertible, allowing for the reversal of transformations applied during the output operation.

   Output Steps for the Variant GSW Approach:
   - **Step 1**: Multiply the decryption result by $\mathbf{M}$, isolating e' as $\mathbf{Z} = \mathbf{s} \cdot \mathbf{C} \cdot \mathbf{M} = \mu\mathbf{sGM} + e' \cdot \mathbf{M} = e' \cdot \mathbf{M} \pmod{q}$, which isolates the error due to $\mathbf{GM} = 0 \pmod{q}$.
   - **Step 2**: Extract e' by applying the inverse of $\mathbf{M}$,

   $$\mathbf{ZM}^{-1} = (e' \cdot \mathbf{M} \pmod{q}) \cdot \mathbf{M}^{-1} = e' \tag{3}$$

   - **Step 3**: Remove e' from the original decryption to isolate the message: $\mathbf{sC} - \mathbf{ZM}^{-1} = \mathbf{sC} - e' = \mu\mathbf{sG}$.
   - **Step 4**: Deduce the message μ is deduced from the known secret key $\mathbf{s}$ and $\mathbf{G}$.

   Challenges and Limitations: The critical limitation arises in Equation 3. However, since $\mathbf{M}$ and $\mathbf{M}^{-1}$ must exist under different *field* conditions — $\mathbf{M}$ under finite field conditions and $\mathbf{M}^{-1}$ ideally in the infinite field $\mathbb{Z}$ — this creates a fundamental operational flaw. In practice, $\mathbf{M}^{-1}$ must be

determined under mod q, and hence, obtaining such an inverse matrix where $\mathbf{M}$ is nonsingular modulo q becomes infeasible.
Therefore, thinking of possible solutions to the equation requires adjusting $e' \cdot \mathbf{M} \pmod q$, which requires the elements in $\mathbf{M}$ to be kept very small. This adjustment is crucial to ensure that when multiplied by e', each element does not exceed $q-1$. Although this strategy lacks rigor, it represents a feasible solution. Identifying such $\mathbf{M}$ is highly challenging, prompting us to switch to approximation methods as an alternative. This approach might be preferable if a more straightforward way can be discovered to determine $\mathbf{M}$.

2. **An Approximation Method [12].** This more straightforward approach does not seek to directly and accurately correct the error but instead minimizes it through comparative analysis :
   - **Step 1**: Directly divide the decryption result by the secret key and the gadget matrix $\mathbf{G}$, attempting to isolate μ as

$$\begin{aligned}\frac{\mathbf{sC}}{\mathbf{sG}} &= \mu \cdot \mathbf{J}_{1,m} + \frac{e'}{\mathbf{sG}}, \text{ where } \mathbf{J}_{1,m} = [1\ 1\ \cdots\ 1] \in 1^m \\ &= \left[\mu_0^e\ \mu_1^e\ \cdots\ \mu_{m-1}^e\right].\end{aligned}$$

     Ideally, this should leave us with the correct message and a small amount of error. Therefore, the smaller the error, the closer the value at that position is to the correct result.
   - **Step 2**: Each element resulting from the division is examined to determine which produces the smallest error, approximated as

$$\forall i,\ \mathbf{s} \cdot \mathbf{C} - \mu_i^e \mathbf{sG} = e_i'.$$

     The element with the smallest resultant error e_i' is considered the closest approximation to μ.

   Challenges and Limitations: However, since this method is essentially an approximation, it inherently introduces additional deviations, particularly as the magnitude of the message increases. Consequently, the final calculated error significantly exceeds the theoretical error, resulting in a substantially reduced upper bound ϕ. Despite these limitations, this approach provides a simpler and more convenient alternative than the previous one.

Both approaches offer potential pathways to handle errors in the output stage of decryption. However, although theoretically robust, the variant GSW approach faces practical implementation hurdles due to the matrix inversion issues. On the other hand, while more straightforward and practicable, the *approximate method* suffers from error magnification as message values increase.

Homomorphic Operation. Two primary operations in the cryptographic framework that support homomorphic operations are homomorphic addition ($\mathbf{C}^+$) and homomorphic multiplication ($\mathbf{C}^*$). Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be ciphertexts, respectively encrypting plaintexts μ_1 and μ_2. The operations are mathematically described as follows:

- Homomorphic addition: $\mathbf{C}^+ = \mathbf{C}_1 + \mathbf{C}_2 \pmod{q}$
- Homomorphic multiplication: $\mathbf{C}^* = \mathbf{C}_1 \times G^{-1}(\mathbf{C}_2) \pmod{q}$

where transforming each element of the ciphertext matrix into its binary representations, that is:

$$\mathbf{C} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix} \implies G^{-1}(\mathbf{C}) = \begin{bmatrix} \mathbf{c}_{1,1,0} & \mathbf{c}_{1,2,0} & \cdots & \mathbf{c}_{1,n,0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{1,1,\ell-1} & \mathbf{c}_{1,2,\ell-1} & \cdots & \mathbf{c}_{1,n,\ell-1} \\ \mathbf{c}_{2,1,0} & \mathbf{c}_{2,2,0} & \cdots & \mathbf{c}_{2,n,0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{2,1,\ell-1} & \mathbf{c}_{2,2,\ell-1} & \cdots & \mathbf{c}_{2,n,\ell-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{n,1,0} & \mathbf{c}_{n,2,0} & \cdots & \mathbf{c}_{n,n,0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{n,1,\ell-1} & \mathbf{c}_{n,2,\ell-1} & \cdots & \mathbf{c}_{n,n,\ell-1} \end{bmatrix}$$

where each $c_{i,j} \in \mathbb{Z}_q$ is decomposed into a binary representation such that each binary bit $\mathbf{c}_{x,y,z}$ lies in $\{0, 1\}$. The original element $c_{i,j}$ is then expressed as $c_{i,j} = \mathbf{c}_{i,j,0} + 2^1 \cdot \mathbf{c}_{i,j,1} + \cdots + 2^{\ell-1} \cdot \mathbf{c}_{i,j,\ell-1}$, with $\ell = \lceil \log_2 q \rceil$.

Performing operations in the encrypted domain without intermediate decryptions presents significant challenges, notably regarding error accumulation and message size constraints. As a result, the error can increase exponentially with the number of operations, profoundly affecting the integrity of the decryption outcome. Therefore, when multiple homomorphic operations are involved, ϕ must be adjusted downward to prevent the total error from exceeding the decryption algorithm's capacity for correction or tolerance. This adjustment restricts the magnitude of message values that can be securely encrypted and accurately decrypted after applying homomorphic operations.

4 The CRT-based Decomposition

This section explores expanding the message selection range to the entire plaintext space ($[0, q)$), allowing users unrestricted choice. As noted, errors can easily exceed the tolerance limit when the large message is selected. To address this, we considered decomposing the large message into multiple smaller values, each containing minor errors, which can be individually encrypted and decrypted before recombining them. This approach leads us to consider the previous base decomposition method (as shown in the middle of Figure 1) or one of our primary research topics, the "CRT decomposition method," which uses CRT to expand the message range further (as shown at the bottom of Figure 1).

Our CRT decomposition method solves the issue of the base decomposition method's inability to perform homomorphic addition. It allows for multiple consecutive operations without intermediate decryption, significantly outperforming the original non-decomposition method.

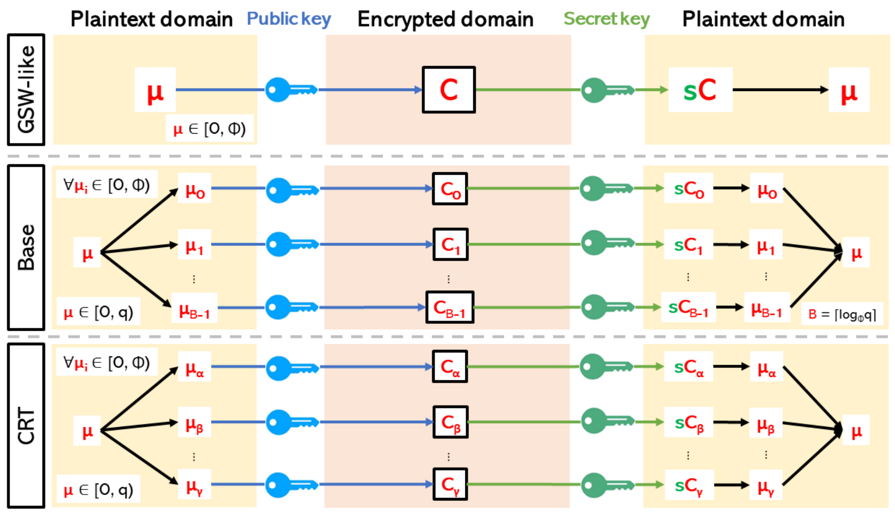

Fig. 1. Simplified flowcharts for benchmarking schemes are presented in Sections 3 (top), 4.1 (middle), and 4.2 (bottom), respectively.

4.1 Base Decomposition (Figure 1, the middle block)

Key Generation, Decomposition and Encryption. The public key ($\mathbf{P}$) and secret key ($\mathbf{s}$) generation during the initial key generation stage remain unchanged. In the encryption phase, to align with our objectives, we now permit users to select messages without restrictions and denote the allowable range as $\mu \in [0, q)$. Previously, the method involved converting μ into its binary form $(\mu_0, \mu_1, \cdots, \mu_{\ell-1})$, expressed as $\mu = \mu_0 + 2 \cdot \mu_1 + \cdots + 2^{\ell-1} \cdot \mu_{\ell-1} \pmod q$, where $\forall i$, $\mu_i \in \{0, 1\}$ and $\ell = \lceil \log_2 q \rceil$. Each bit is independently encrypted and separately decrypted without affecting each other.

Additionally, as identified in Section 3, the upper bound of a message value can be as large as ϕ. Thus, the base value can also be increased to ϕ, significantly reducing the number of decomposed values. Now, μ can be represented as $(\mu_0, \mu_1, \cdots, \mu_{B-1})$, expressed as $\mu = \mu_0 + \phi \cdot \mu_1 + \cdots + \phi^{B-1} \cdot \mu_{B-1} \pmod q$, where $\forall i$, $\mu_i \in [0, \phi)$ and $B = \lceil \log_\phi q \rceil$. This alteration substantially lowers both the memory requirements and the processing time.

Decryption, Output, and Recombination: The decryption and output stages proceed as previously. Decryption involves multiplying the secret key $\mathbf{s}$ with the ciphertext $\mathbf{C}$, and the output employs the approximate method. Initially, the large message is split into smaller messages. During the recombination stage, the final output is recombined according to $\mu = \mu_0 + \phi \cdot \mu_1 + \cdots + \phi^{B-1} \cdot \mu_{B-1} \pmod q$. After computing the result, the original large message is successfully reconstructed.

Table 2. All benchmarked vector secret key schemes' decomposition processes.

	Bit	Base	**CRT** (Sec. 4.2)
KeyGen.	Public key : $\mathbf{P} = \left[\frac{-A}{\mathbf{b}}\right] \in \mathbb{Z}_q^{n\times m}$, and Secret key : $\mathbf{s} = [t \mid 1] \in \mathbb{Z}_q^n$		
Decomp.	Selecting the message $\mu \in [0, q)$ and $R \in \{0,1\}^{m\times m}$		
	$(\mu_0, \mu_1, \cdots, \mu_{\ell-1})$	$(\mu_0, \mu_1, \cdots, \mu_{B-1})$	$\begin{cases} \mu \equiv & \mu_\alpha \pmod{m_\alpha} \\ \mu \equiv & \mu_\beta \pmod{m_\beta} \\ & \vdots \\ \mu \equiv & \mu_\gamma \pmod{m_\gamma} \end{cases}$
Enc.	Encrypting each element into ciphertext by $\forall i, \text{s.t. } \mathbf{C_i} = \mu_i \cdot \mathbf{G} + \mathbf{P} \times R \pmod{q} \in \mathbb{Z}_q^{n\times m}$		
Dec.	Decrypting all ciphertext by $\forall i, \text{s.t. } \mathbf{s}\mathbf{C}_i = \mu_i \cdot \mathbf{s} \times \mathbf{G} + e_i \pmod{q} \in \mathbb{Z}_q^n$		
Output.	*Approximate method*		
Recomb.	$\mu = \sum_i 2^i \mu_i$	$\mu = \sum_i \phi^i \mu_i$	$\mu = \sum_i \mu_i \mathbf{t}_i \mathbf{M}_i$

4.2 CRT Decomposition (Figure 1, the bottom block)

At this stage, we have decided to implement the CRT for decomposition because it allows for converting the more significant value into multiple smaller values. The key generation, encryption, decryption, and output phases adhere to the principles established in the base decomposition approach. A distinctive feature of this approach is the methodology used for decomposition and recombination. Specifically, the CRT is utilized to decompose μ as follows:

$$\begin{cases} \mu \equiv & \mu_\alpha \pmod{m_\alpha} \\ \mu \equiv & \mu_\beta \pmod{m_\beta} \\ & \vdots \\ \mu \equiv & \mu_\gamma \pmod{m_\gamma} \end{cases},$$

where the set $(m_\alpha, m_\beta, \cdots, m_\gamma)$ is all the moduli utilized in the CRT. Each modulus is pairwise coprime and less than ϕ. Recombination also employs the CRT, formulated as:

$$\mu = \mu_\alpha \mathbf{t}_\alpha \mathbf{M}_\alpha + \mu_\beta \mathbf{t}_\beta \mathbf{M}_\beta + \cdots + \mu_\gamma \mathbf{t}_\gamma \mathbf{M}_\gamma = \sum_i \mu_i \mathbf{t}_i \mathbf{M}_i \pmod{\mathbf{M}},$$

where $\mathbf{M} = m_\alpha \times m_\beta \times \cdots \times m_\gamma = \prod_i m_i$.

These decomposition methods are detailed in Table 2, highlighting the differences in the decomposition and recombination stages. These decomposition methods efficiently handle large messages by splitting them, thereby enhancing the efficiency of our encryption scheme.

4.3 Homomorphic Addition after Decomposition

Now that both the base and CRT decomposition methods have proven effective in achieving our desired outcomes, it is pertinent to explore our CRT-based approach's distinct advantages and contributions. Notably, our method supports parallel computation, a capability not available in base decomposition methods. Furthermore, the base decomposition approach frequently encounters issues with mathematical carry operations, which are inherently absent in our CRT method. This advancement, therefore, increases the potential and applicability of this encryption scheme in daily use.

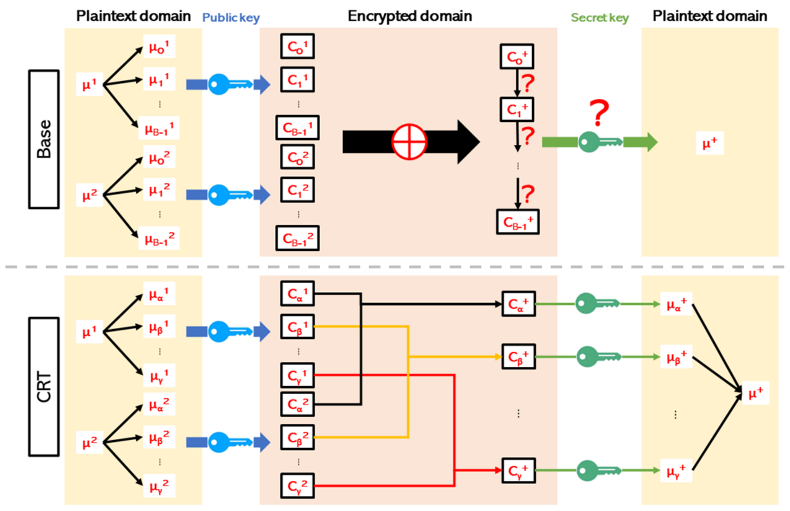

Fig. 2. The processes of the base decomposition (the top row), the CRT decomposition (the bottom row), and the homomorphic addition (the middle column).

Base Decomposition. The direct conclusion is that base decomposition cannot perform homomorphic addition after decomposition; the main reason is the concept of *mathematical carry*. A mathematical carry is possible when these decomposed bits or elements are added to the original plaintext domain. However, when these elements undergo encryption and are represented in ciphertext form, simulating or implementing carry functions within the encrypted domain — particularly in matrix formats — presents significant hurdles. This complexity prevents efficient decryption after such operations.

Moreover, even if a technique exists to manage to carry within a matrix of encrypted elements, executing parallel computations efficiently remains a formidable challenge due to the nature of carries. Consequently, performing homomorphic additions becomes inefficient, complex, and occasionally unfeasible if decompositions are conducted using base decomposition.

CRT Decomposition. However, our CRT decomposition avoids carrying issues, facilitating parallel computation, as shown in Figure 2. However, an additional limitation arises: it must be ensured that the aggregate of selected messages ($\sum_j \mu^j$) remains below the modulus value $\mathbf{M}$ to avoid operational failures. Despite this limitation, the CRT decomposition method allows for the selection of messages without constraints and effectively supports homomorphic addition, even over many consecutive operations without interruption. Therefore, this limitation is acceptable.

5 Ciphertext Compression

Table 1 shows that key encryption requires significant memory, even with a small security parameter. For example, when $n = 8$, a message that might initially be just a few bits can expand to require 3.625 KB. Moreover, such a small security parameter is insufficient for real-world applications, which might need $n = 128$ or more for adequate security. In this scenario, the storage space required becomes even more unimaginable. Therefore, any method that can compress these schemes would be incredibly beneficial.

[10] proposes a compression method applicable to the GSW encryption scheme, introducing two compression techniques: "*PVW-like*" and "*Nearly Square Gadget Matrix.*" The method in the paper compresses ciphertexts resulting from the encryption of multiple messages simultaneously. However, in our study, we focus on compressing ciphertexts from the encryption of a single message. Additionally, since the "*Nearly Square Gadget Matrix*" method is designed for multiple messages, our compression discussion will only cover the *PVW-like* method. The details of the "*Nearly Square Gadget Matrix*" method will be included in Appendix A.4. It's crucial to compare these methods to understand their strengths and limitations. In terms of effectiveness, the paper's approach, which compresses multiple messages simultaneously, will naturally be more impressive than compressing a single message. However, our proposed *PVW-like* method for single message compression holds promise and could offer significant benefits in specific scenarios.

5.1 The PVW-like Scheme

Ideas and Concepts. Before introducing their compression process, let us explain the concept behind their method first. The top and bottom portions of Figure 3 show the uncompressed and compressed processes of the *PVW-like* scheme, respectively.

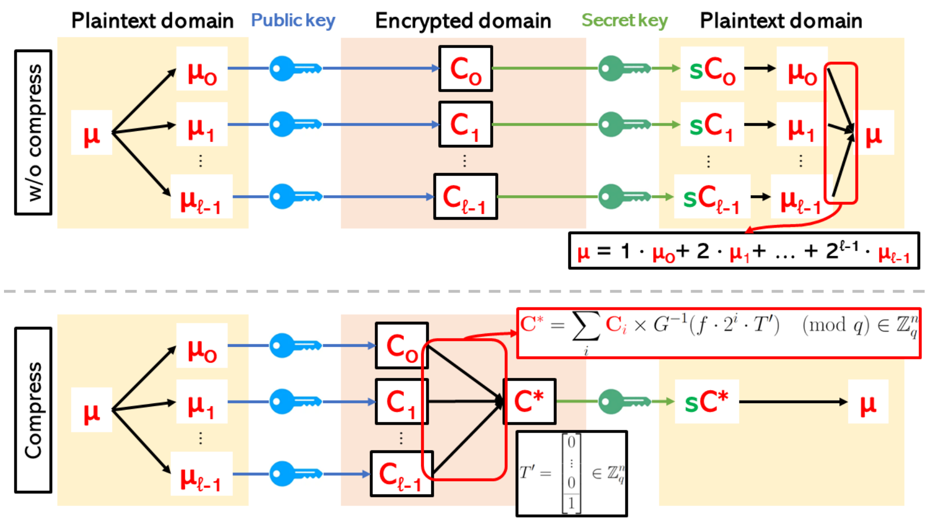

Fig. 3. The architectures are associated with PVW-like schemes without compression (the upper part) and compression (the lower part).

The top section, which shows the uncompressed process, follows the same idea of base decomposition introduced in section 4. Initially, the message is converted to binary form, and after decryption and output, it is converted back to decimal form.

Our understanding of their compression method is that it combines these steps earlier in the process. Specifically, instead of waiting until the final decryption and output stages to combine the results, the combination happens immediately after each small message is encrypted. This early combination is the core idea behind their compression method.

Key Generation and Encryption. Let us start by introducing the entire compression process. First, the key generation is the same as before, resulting in a pair of public key $\mathbf{P}$ and secret key $\mathbf{s}$.

During the encryption, a message is selected, and the two decomposition methods mentioned earlier can be used: base decomposition and CRT decomposition. After decomposing the message into smaller values, each value is encrypted separately.

For this explanation, we will assume the base decomposition method is used with a base of 2. Therefore, the larger message μ is decomposed into smaller messages, denoted as $(\mu_0, \mu_1, \cdots, \mu_{\ell-1})$, and the encrypted ciphertexts are $(\mathbf{C}_0, \mathbf{C}_1, \cdots, \mathbf{C}_{\ell-1})$.

Compression. The next critical step is compression. The *PVW-like* compression method was chosen for this purpose. Given that the ciphertext now takes the form of a matrix, the original combination formula is no longer applicable and has been transformed into

$$\mathbf{C}^* = \sum_i \mathbf{C}_i \times G^{-1}(f \cdot 2^i \cdot T') \pmod{q} \in \mathbb{Z}_q^n.$$

In the above equation, $\mathbf{C}_i$ represents the ciphertext generated by encrypting all previous small messages, and $T' = \begin{bmatrix} 0_{n-1} \\ 1 \end{bmatrix} \in \mathbb{Z}_q^n$ is a particular vector of length n, where only the last coordinate is one and all others are 0, as illustrated in the lower part of Figure 3. This vector T' plays a vital role in the compression process. As a result, what would have initially required ℓ ciphertexts of size $n \times m$ can now be effectively reduced to a single vector of length n, thereby achieving the desired compression.

This compression method combines all small messages immediately after encryption, effectively reducing the ciphertext size. Consequently, we have adapted the formulas for various decomposition methods, as illustrated in

$$\text{Base Decomposition: } \mathbf{C}^* = \sum_j \mathbf{C}_j \times G^{-1}(f \cdot \phi^j \cdot T') \pmod{q} \in \mathbb{Z}_q^n$$
$$\text{CRT Decomposition: } \mathbf{C}^* = \sum_k \mathbf{C}_k \times G^{-1}(f \cdot \mathbf{t}_k \cdot \mathbf{M}_k \cdot T') \pmod{q} \in \mathbb{Z}_q^n.$$

For base decomposition f is equal to $\lceil q/p \rfloor$, where q represents the modulus of the encryption domain and p represents the modulus of the plaintext domain, which is the message range that users can choose from. For the CRT decomposition, $f = \lceil q/\mathbf{M} \rfloor$, where $\mathbf{M}$ is the product of all the moduli in the CRT decomposition and also represents the plaintext domain.

Decryption and Its Output. Next, we similarly observe the decryption scenario after compression. That is,

$$\begin{aligned} \text{Decryption: } \mathbf{s} \times \mathbf{C}^* &= \sum_i \mathbf{s} \times \mathbf{C}_i \times G^{-1}(f \cdot 2^i \cdot T') \\ &= \sum_i (\mu_i \mathbf{s}\mathbf{G} + e_i') \times G^{-1}(f \cdot 2^i \cdot T') \\ &= \sum_i f \cdot 2^i \cdot \mu_i \cdot \mathbf{s} \times T' + \overbrace{\sum_i e_i' \times G^{-1}(f \cdot 2^i \cdot T')}^{\mathbf{e}'} \qquad (4) \\ &= f \cdot \mu \cdot \mathbf{s} \times T' + \mathbf{e}' = f \cdot \mu + \mathbf{e}' \pmod{q}. \qquad (5) \end{aligned}$$

In Equation 4, it is evident that the previously distributed error is indeed recombined. Consequently, it can be inferred that to ensure the smooth operation of the entire encryption scheme at this stage, the upper bound of the message range must be further reduced. Moreover, given that the dimensions after compression are significantly smaller than those of a single ciphertext, the upper bound ϕ is likely to decrease even more noticeably. Furthermore, the result after decryption in Equation 5 resembles the output of PVW decryption. Therefore, we follow the derivation of the PVW output method in this context.

$$\text{Output: } \left\lceil \frac{\mathbf{s}\mathbf{C}^*}{f} \right\rfloor = \left\lceil \frac{f \cdot \mu + \mathbf{e}'}{f} \right\rfloor = \mu \pmod{p}, \text{ with } \parallel \mathbf{e}' \parallel < \frac{f}{2}. \tag{6}$$

However, applying this method to CRT decomposition increases the likelihood of failure. This is due to the limitations encountered during the original CRT recombination. In the final step of CRT, the combined sum is calculated under the modulus $\mathbf{M}$, yielding an unknown value. The result tends to be very large without performing this modulo $\mathbf{M}$ operation. In this context, after dividing by f, the output method directly yields a value between 0 and $\mathbf{M} - 1$, rather than the expected significant result by CRT. This discrepancy results in a high failure rate in experiments. The experiments revealed that if the result of $q/\mathbf{M}$ is an exact integer or the decimal produced after division is tiny, the method is more likely to succeed. However, this requires careful selection and testing of the CRT's modulus set $(m_\alpha, m_\beta, \cdots, m_\gamma)$ to ensure proper functionality.

However, trying each modulus one by one is impractical. Therefore, we attempted to modify the output method. We understand that their method treats the decrypted result directly as a scalar, so they subsequently divide by f, allowing the result to be mapped back to the original plaintext space. However, "division by f" seemed somewhat unreasonable in implementation. In Equation 6, $f \cdot \mu + \mathbf{e}'$ is, in fact, a result in the finite field q. In finite fields, division should be performed using the multiplicative inverse of f, denoted as f^{-1}, rather than directly dividing by f. Therefore, we attempted to change the output method to $\mathbf{s}\mathbf{C}^* \cdot f^{-1}$. To verify the correctness of the formula and its security, we further divided the scenarios into those with and without error. That is,

$$\begin{aligned} &\text{No error:} && \mathbf{s}\mathbf{C}^* \cdot f^{-1} = f \cdot \mu \cdot f^{-1} = \mu \pmod{q} \\ &\text{With error:} && \mathbf{s}\mathbf{C}^* \cdot f^{-1} = (f \cdot \mu + \mathbf{e}') \cdot f^{-1} = \mu + \mathbf{e}' \cdot f^{-1} \pmod{q}. \end{aligned} \tag{7}$$

In Equation 7, both the base and CRT decomposition methods work without issues. The computed result will match the more significant value produced by the original CRT before performing modulus $\mathbf{M}$. Thus, performing an additional step of modulus $\mathbf{M}$ yields the correct result, indicating that this approach is feasible.

However, when testing for security by introducing noise, $\mathbf{e}'$ becomes multiplied by a scalar, significantly increasing the noise. Maintaining $|\mathbf{e}' \cdot f^{-1}|$ within 0.5 becomes almost impossible, leading to failure in execution for any decomposition method when an error is added. Consequently, our proposed adjustment, although mathematically sound, fails to balance the trade-off effectively—the inability to control the noise amount results in overall failure.

6 Experiment

We present a comparative analysis and experimental results of CRT decomposition and compression techniques. This study examines the impact of our proposed CRT decomposition method and compression technique on the upper bound ϕ, as well as observes the consecutive computational iterations achievable by the CRT decomposition method without decryption. We aim to verify whether these modifications align with our hypotheses. The actual numerical data used in the figures are provided in Appendix C.

6.1 Upper bound ϕ

Figure 4 represents the upper bound results of the CRT decomposition method and compression technique. It contains four entries: the last three represent the upper bound ϕ for operations without (denoted as *Without*), with the CRT decomposition method (denoted as *CRT*) and the compression technique (denoted as *Compression*). In contrast, the first entry indicates the minimum number of moduli. The horizontal axis denotes the security parameters, while the left vertical axis in both tables represents the upper bound ϕ, displayed logarithmically, and the right vertical axis represents the number of moduli.

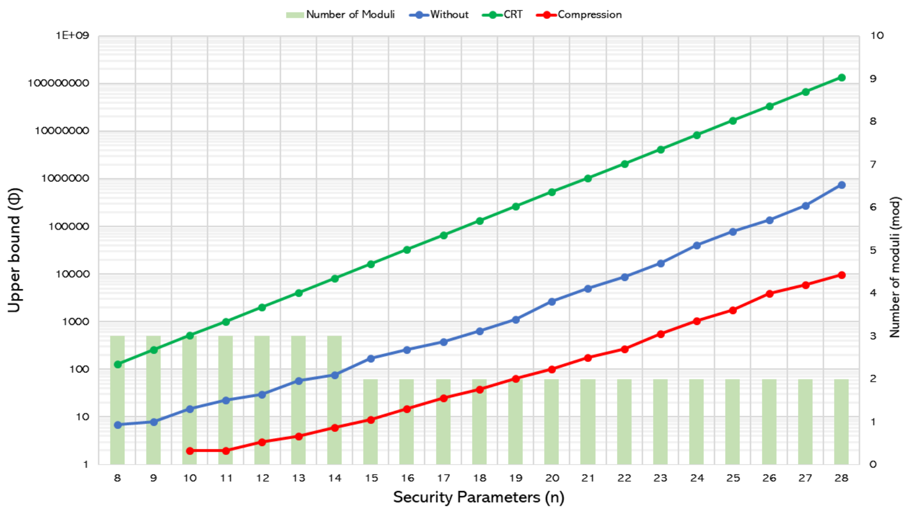

Fig. 4. Comparative Analysis of CRT Decomposition and Compression Methods on the Upper Bound ϕ of Message Range.

Several conclusions can now be drawn as follows:

1. It can be observed that, our proposed CRT decomposition method significantly increases the upper bound. Regardless of n, users can freely choose messages within the entire plaintext space without restriction.
2. As hypothesized, compression leads to a cumulative effect of errors, significantly reducing the upper bound of the message range, even below that of

the undecomposed original method. This considerably constrains the user's range of choices. The PVW-like compression method proposed in [10] is particularly well-suited for this scenario. Due to error considerations, the original PVW encryption system also restricts the plaintext space.

3. The original "Without decomposition's" upper bound ϕ indirectly affects the upper bound results and the number of moduli required for applying the CRT decomposition method. For example, when $n = 8$, the upper bound is 7, so the CRT moduli can only be chosen from the range 2 to 7. To achieve the theme of restriction-free, at least three values are needed to exceed 2^7. Therefore, if the upper bound could be increased, only two values would be needed to reach the goal.

6.2 Sequential computation

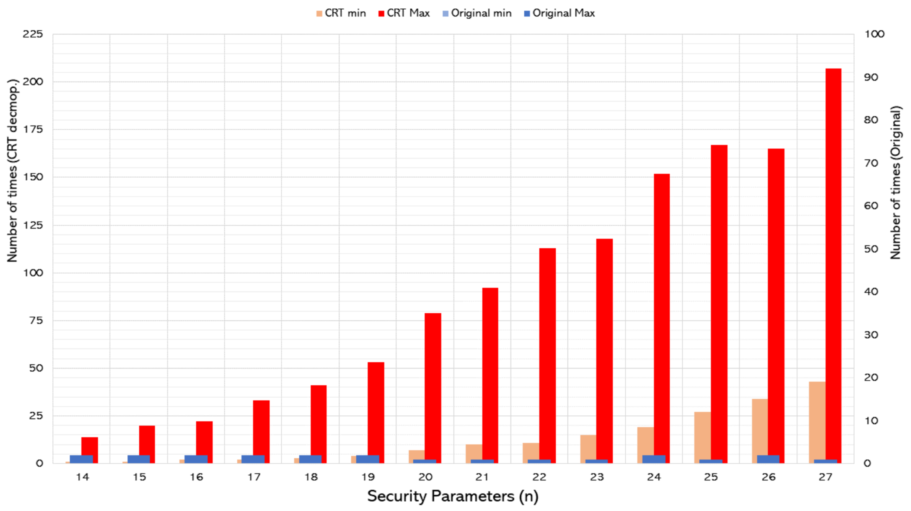

Fig. 5. Comparing the number of consecutive operations that can be performed without decomposition (Sec. 3) and with CRT decomposition (Sec. 4.2) under different parameter combinations, all with a message range of [0, q).

However, practical applications rarely involve just a single operation. Therefore, it is essential to observe how this encryption scheme, after our modifications, performs multiple consecutive additions without decryption. As illustrated in Figure 5, we analyze the performance under varying security parameters. The first two labels in the table represent the minimum and maximum consecutive operations observed in 1,000 repeated tests using our CRT method (described in Section 4.2). The remaining two labels indicate the minimum and maximum operations for the original GSW method without decomposition (Section 3). The vertical axes display the number of operations, while the left axis pertains to our method, and the right axis corresponds to the original approach.

Several conclusions can be drawn accordingly:

1. Our method outperforms the original GSW scheme across almost every security parameter. Unlike the original method, it consistently permits at least one operation, which allows a maximum of one operation.
2. As the parameter n increases, the range of both minimum and maximum operations expands for our method. In contrast, the original method remains limited to one or two operations, which aligns with our expected outcomes. Since the original method can only process messages within the range of ϕ, expanding this range will only make a single operation more difficult.

Our proposed CRT decomposition method facilitates unrestricted message selection and supports parallelized addition operations and continuous, uninterrupted computations. However, these performance gains come at the cost of increased memory space, as the message is divided into smaller values for representation, which may also result in slightly longer computation times.

7 Future Works and Conclusions

In conclusion, this research has effectively demonstrated the utility of CRT in expanding the range of encrypted messages within the GSW encryption scheme. By integrating CRT, the scheme now supports a larger message range and adapts existing ciphertext compression techniques for practical applications. These enhancements are crucial for deploying secure cryptographic systems, where flexibility and efficiency are paramount.

Given that [10] initially applied compression methods to the GSW matrix secret key scheme, and [17] subsequently referenced [10], categorizing it among various multikey FHE schemes that enhance ciphertext processing, our future work will focus on extending our current research to multikey scenarios. Moreover, as multikey schemes align more closely with contemporary real-world applications, successfully adapting our method to multikey scenarios could significantly enhance the potential and feasibility of implementing GSW encryption schemes in practical contexts.

We recognize that during compression, errors reaccumulate within the same ciphertext, significantly increasing the likelihood of decryption failure. Consequently, users must substantially reduce the upper bound of the message range when selecting messages. This trade-off necessitates a careful balance between compression ratio and error tolerance, particularly when handling sensitive or large-scale data. To address this challenge, we propose exploring decompression techniques that could redistribute the reaccumulated errors, potentially mitigating the issues introduced by compression. This work has the potential to significantly advance the practical applicability of FHE in real-world scenarios.

References

1. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
2. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. pp. 97–106 (2011). https://doi.org/10.1109/FOCS.2011.12
3. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Annual cryptology conference. pp. 505–524. Springer (2011)
4. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. pp. 409–437. Springer (2017)
5. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
6. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 617–640. Springer (2015)
7. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
8. Gentry, C.: A fully homomorphic encryption scheme. Stanford university (2009)
9. Gentry, C.: Computing arbitrary functions of encrypted data. Communications of the ACM **53**(3), 97–105 (2010)
10. Gentry, C., Halevi, S.: Compressible fhe with applications to pir. In: Theory of Cryptography Conference. pp. 438–464. Springer (2019)
11. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. pp. 75–92. Springer (2013)
12. Hedglin, N., Phillips, K., Reilley, A.: Building a fully homomorphic encryption scheme in python (2019), https://github.com/hedglinnolan/GSW-Homomorphic-Encryption-Python
13. Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F.H., Aaraj, N.: Survey on fully homomorphic encryption, theory, and applications. Proceedings of the IEEE **110**(10), 1572–1609 (2022)
14. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Annual international cryptology conference. pp. 554–571. Springer (2008)
15. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th annual ACM symposium on Theory of computing. pp. 84–93 (2005). https://doi.org/10.1145/1060590.1060603
16. Rivest, R.L., Adleman, L., Dertouzos, M.L., et al.: On data banks and privacy homomorphisms. Foundations of secure computation **4**(11), 169–180 (1978)
17. Yuan, M., Wang, D., Zhang, F., Wang, S., Ji, S., Ren, Y.: An examination of multi-key fully homomorphic encryption and its applications. Mathematics **10**(24), 4678 (2022)

A Matrix Secret Key

A.1 Learning With Error (LWE)

Find a $\mathbf{S} \in \mathbb{Z}_q^{n_0 \times k}$ matrix s.t. $\mathbf{SA}+\mathbf{E} = \mathbf{B} \pmod q$ under a small error $\mathbf{E}$, where $\mathbf{A} \in \mathbb{Z}_q^{k \times m}$ with $k < m$,

$$\begin{bmatrix} \mathbf{s}_{1,1} & \cdots & \mathbf{s}_{1,k} \\ \vdots & \ddots & \vdots \\ \mathbf{s}_{n_0,1} & \cdots & \mathbf{s}_{n_0,k} \end{bmatrix} \begin{bmatrix} \mathbf{a}_{1,1} & \cdots & \mathbf{a}_{1,m} \\ \vdots & \ddots & \vdots \\ \mathbf{a}_{k,1} & \cdots & \mathbf{a}_{k,m} \end{bmatrix} + \begin{bmatrix} \mathbf{e}_{1,1} & \cdots & \mathbf{e}_{1,m} \\ \vdots & \ddots & \vdots \\ \mathbf{e}_{n_0,1} & \cdots & \mathbf{e}_{n_0,m} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{1,1} & \cdots & \mathbf{b}_{1,m} \\ \vdots & \ddots & \vdots \\ \mathbf{b}_{n_0,1} & \cdots & \mathbf{b}_{n_0,m} \end{bmatrix}.$$

A.2 Single Key

KeyGen. Initially, the matrix $A \in \mathbb{Z}_q^{k \times m}$, $S \in \mathbb{Z}_q^{n_0 \times k}$, and small $E \in \chi^{n_0 \times m}$ are selected randomly, where for some small $\epsilon > 0$ s.t. $n_0 \approx 2k/\epsilon, n_1 = n_0 + k$ and $m = n_1 \cdot \ell$. Subsequently, the matrix $\mathbf{B} = SA + E \in \mathbb{Z}_q^{n_0 \times m}$ is calculated. Finally, the public key, represented by $\mathbf{P} = \left[\frac{-A}{\mathbf{B}}\right] \in \mathbb{Z}_q^{n_1 \times m}$, and the secret key, represented by $\mathbf{S} = [S \mid I_{n_0}] \in \mathbb{Z}_q^{n_0 \times n_1}$, are generated, ensuring that $\mathbf{S} \times \mathbf{P} = E \pmod q$.

Enc. Selecting the message $\mu \in [0, \phi)$ and $R \in \{0,1\}^{m \times m}$, and encrypting into ciphertext by $\mathbf{C} = \mu \cdot \mathbf{G} + \mathbf{P} \times R \pmod q \in \mathbb{Z}_q^{n_1 \times m}$.

Dec. Using the secret key ($\mathbf{S}$) and ciphertext ($\mathbf{C}$), and the result is computed using formula $\mathbf{SC} = \mu \cdot \mathbf{S} \times \mathbf{G} + \mathbf{S} \times \mathbf{P} \times R = \mu \cdot \mathbf{S} \times \mathbf{G} + E' \pmod q \in \mathbb{Z}_q^{n_0 \times m}$.

Output. The *approximate method* is adopted because it is more straightforward and convenient to understand.

A.3 All Decomposition Process

In this study, Table 3 illustrates the processes of all decomposition methods. The term 'Bit' indicates using base decomposition with a base of 2, corresponding to the approach in [10]. The entire process differs only in the decomposition and recombination stages.

Users can initially select message values from any range in the decomposition stage. The next step involves decomposition based on the respective methods, where all element in each decomposition method falls within the interval $[0, \phi)$. In other words, $\forall i$, s.t. $\mu_i \in [0, \phi)$.

Finally, in the recombination stage, all elements obtained from the output are recombined according to their respective combination methods to reconstruct the user's initially selected message.

A.4 Ciphertext Compression

Under a matrix secret key, the compression of ciphertext involves compressing multiple messages together, as proposed in [10]. The effectiveness of this compression method is expected to be significantly better than compressing a single message using a vector secret key.

Table 3. All benchmarked matrix secret key schemes' decomposition processes.

	Bit [10]	Base	**CRT**
KeyGen.	Public key : $\mathbf{P} = \left[\frac{-A}{\mathbf{B}}\right] \in \mathbb{Z}_q^{n_1 \times m}$, and Secret key : $\mathbf{S} = [S \mid I_{n_0}] \in \mathbb{Z}_q^{n_0 \times n_1}$		
Decomp.	Selecting the message $\mu \in [0, q)$ and $R \in \{0,1\}^{m \times m}$		
	$(\mu_0, \mu_1, \cdots, \mu_{\ell-1})$	$(\mu_0, \mu_1, \cdots, \mu_{B-1})$	$\begin{cases} \mu \equiv & \mu_\alpha \pmod{m_\alpha} \\ \mu \equiv & \mu_\beta \pmod{m_\beta} \\ & \vdots \\ \mu \equiv & \mu_\gamma \pmod{m_\gamma} \end{cases}$
Enc.	Encrypting each element into ciphertext by $\forall i, \text{s.t. } \mathbf{C_i} = \mu_i \cdot \mathbf{G} + \mathbf{P} \times R \pmod{q} \in \mathbb{Z}_q^{n_1 \times m}$		
Dec.	Decrypting all ciphertext by $\forall i, \text{s.t. } \mathbf{SC}_i = \mu_i \cdot \mathbf{S} \times \mathbf{G} + E_i \pmod{q} \in \mathbb{Z}_q^{n_0 \times m}$		
Output.	*Approximate method*		
Recomb.	$\mu = \sum_i 2^i \mu_i$	$\mu = \sum_i \phi^i \mu_i$	$\mu = \sum_i \mu_i \mathbf{t}_i \mathbf{M}_i \pmod{\mathbf{M}}$

Following the approach in [10], the goal is to compress n_0^2 messages simultaneously. Subsequently, using bit decomposition will result in $n_0^2 \cdot \ell$ elements. Therefore there will also be $n_0^2 \cdot \ell$ ciphertexts after encryption, denoted as $\mathbf{C}_{u,v,w} \in \mathbb{Z}_q^{n_1 \times m}$, where $u, v \in [n_0]$ and $w \in [\ell]$. Both the original compression methods, *PVW-like* and *nearly square gadget matrix*, can be utilized.

PVW-like.

Compression. At this point, the compression formulas corresponding to each decomposition method become

$$\text{Bit Decomposition} \quad \Longrightarrow \quad \mathbf{C}^* = \sum_{u,v,w} \mathbf{C}_{u,v,w} \times G^{-1}(f \cdot 2^w \cdot T'_{u,v}) \in \mathbb{Z}_q^{n_1 \times n_0}$$

$$\text{Base Decomposition} \quad \Longrightarrow \quad \mathbf{C}^* = \sum_{U,V,W} \mathbf{C}_{U,V,W} \times G^{-1}(f \cdot \phi^W \cdot T'_{U,V}) \in \mathbb{Z}_q^{n_1 \times n_0}$$

$$\text{CRT Decomposition} \quad \Longrightarrow \quad \mathbf{C}^* = \sum_{i,j,k} \mathbf{C}_{i,j,k} \times G^{-1}(f \cdot \mathbf{t}_k \cdot \mathbf{M}_k \cdot T'_{i,j}) \in \mathbb{Z}_q^{n_1 \times n_0},$$

where f remains the same as previously mentioned, and $T'_{u,v} = \begin{bmatrix} 0_{k \times n_0} \\ T_{u,v} \end{bmatrix} = \begin{bmatrix} 0_{k \times n_0} \\ \mathbf{e}_u \otimes \mathbf{e}_v \end{bmatrix} \in \mathbb{Z}_q^{n_1 \times n_0}$, $T_{u,v} = \mathbf{e}_u \otimes \mathbf{e}_v$ is an $n_0 \times n_0$ matrix with only the position (u, v) is 1, and all other positions are 0. Additionally, $\mathbf{e}_u$ represents a vector of length n_0, with the u-th position as one and all other positions being 0.

This transforms the original bit decomposition of $n_0^2 \times \ell$ matrices of size $n_1 \times m$ into a single $n_1 \times n_0$ matrix. The compression effect is far superior to the previous vector secret key method. However, because it now includes a significantly more significant amount of messages, the total error after compression will encompass the error from all these messages.

Decryption. The decryption of the bit decomposition compressed ciphertext is then performed as follows:

$$\begin{aligned}
\mathbf{S} \times \mathbf{C}^* &= \sum_{u,v,w} \mathbf{S} \times \mathbf{C}_{u,v,w} \times G^{-1}(f \cdot 2^w \cdot T'_{u,v}) \\
&= \sum_{u,v,w} (\mu_{u,v,w}\mathbf{SG} + E_{u,v,w}) \times G^{-1}(f \cdot 2^w \cdot T'_{u,v}) \\
&= \sum_{u,v,w} f \cdot 2^w \cdot \mu_{u,v,w}\mathbf{S}T'_{u,v} + \overbrace{\sum_{u,v,w} E_{u,v,w} \times G^{-1}(f \cdot 2^w \cdot T'_{u,v})}^{\mathbf{E}'} \\
&= f \cdot \sum_{u,v} \mu_{u,v}\mathbf{S}T'_{u,v} + \mathbf{E}' = f \cdot \overbrace{\sum_{u,v} \mu_{u,v} T_{u,v}}^{\mathbf{Z}} + \mathbf{E}',
\end{aligned}$$

where $\mathbf{Z} \in [p]^{n_0 \times n_0}$. After decryption and output, these $n_0 \times n_0$ messages are represented using a matrix, with each message placed according to its fixed position,

$$\mathbf{Z} = \begin{bmatrix} \mu_{0,0} & \mu_{0,1} & \cdots & \mu_{0,n_0-1} \\ \mu_{1,0} & \mu_{1,1} & \cdots & \mu_{1,n_0-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{n_0-1,0} & \mu_{n_0-1,1} & \cdots & \mu_{n_0-1,n_0-1} \end{bmatrix} \in \mathbb{Z}_p^{n_0 \times n_0}.$$

Nearly Square Gadget Matrix

Compression. At this point, we switch to another variant, the *nearly square gadget matrix.* The compression formula also changes. That is,

$$\begin{aligned}
&\text{Bit Decomposition} \implies \mathbf{C}^* = \sum_{u,v,w} \mathbf{C}_{u,v,w} \times G^{-1}(2^w \cdot T'_{u,v} \times H) \in \mathbb{Z}_q^{n_1 \times n_2} \\
&\text{Base Decomposition} \implies \mathbf{C}^* = \sum_{U,V,W} \mathbf{C}_{U,V,W} \times G^{-1}(\phi^W \cdot T'_{U,V} \times H) \in \mathbb{Z}_q^{n_1 \times n_2} \\
&\text{CRT Decomposition} \implies \mathbf{C}^* = \sum_{i,j,k} \mathbf{C}_{i,j,k} \times G^{-1}(\mathbf{t}_k \cdot \mathbf{M}_k \cdot T'_{i,j} \times H) \in \mathbb{Z}_q^{n_1 \times n_2},
\end{aligned}$$

which now includes an additional term, "H." According to the text, "H" is a nearly square matrix similar to the gadget matrix $\mathbf{G}$, with dimensions $n_0 \times n_2$, where $n_2 = n_0(1+\epsilon/2)$. This matrix "H" has special correlations and properties with a corresponding matrix called the "public trapdoor" matrix F. That is,

- **Property (a)**: $F = H^{-1}(0) \in \mathbb{Z}^{m \times m}$ such that $HF = 0 \pmod q$, effectively nullifying the message-related component in the matrix multiplication due to congruence modulo q.
- **Property (b)**: F has small entries ($\ll q$).
- **Property (c)**: F must be invertible over $\mathbb{R}$, allowing for the reversal of transformations applied during the output operation (but not over $\mathbb{Z}_q$).

Decryption and Output. This variant compression method also affects the decryption process for bit decomposition. That is,

$$\begin{aligned}
\mathbf{S} \times \mathbf{C}^* &= \sum_{u,v,w} \mathbf{S} \times \mathbf{C}_{u,v,w} \times G^{-1}(2^w \cdot T'_{u,v} \times H) \\
&= \sum_{u,v,w} (\mu_{u,v,w}\mathbf{S}\mathbf{G} + E_{u,v,w}) \times G^{-1}(2^w \cdot T'_{u,v} \times H) \\
&= \sum_{u,v,w} 2^w \cdot \mu_{u,v,w}\mathbf{S}T'_{u,v}H + \overbrace{\sum_{u,v,w} E_{u,v,w} \times G^{-1}(2^w \cdot T'_{u,v} \times H)}^{\mathbf{E}'} \\
&= \sum_{u,v} \mu_{u,v}\mathbf{S}T'_{u,v}H + \mathbf{E}' = (\overbrace{\sum_{u,v} \mu_{u,v}T_{u,v}}^{\mathbf{Z}}) \times H + \mathbf{E}',
\end{aligned}$$

where the term $\mathbf{Z}$ is similar to the one used in the *PVW-like* method. Next, we can proceed with the output. That is,

- **Step 1**: Multiply the decryption result by the public trapdoor F, isolating $\mathbf{E}'$ as $\mathbf{Z} = \mathbf{S} \times \mathbf{C}^* \times F = \mathbf{Z} \times H \times F + \mathbf{E}' \times F = \mathbf{E}' \times F \pmod q$, which isolates the error due to $HF = 0 \pmod q$ in **Property (a)**.
- **Step 2**: Extract $\mathbf{E}'$ by applying the inverse of F,

$$\mathbf{Z}F^{-1} = (\mathbf{E}' \times F \pmod q) \times F^{-1} = \mathbf{E}'. \tag{8}$$

- **Step 3**: Remove $\mathbf{E}'$ from the original decryption to isolate the message: $\mathbf{S}\mathbf{C}^* - \mathbf{Z}F^{-1} = \mathbf{S}\mathbf{C}^* - \mathbf{E}' = \mathbf{Z} \times H$.
- **Step 4**: Deduce the message $\mathbf{Z}$ is deduced from the known H.

However, we thought this approach has the same issues and methodology described in the previous Equation 3. Since in Equation 8, $\mathbf{E}' \times F \pmod q$ and F^{-1} exist under different field conditions, this is theoretically unreasonable. [10] provides an example of a public trapdoor F, which is

$$F = \begin{bmatrix} F' & 0 & \cdots & 0 \\ 0 & F' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F' \end{bmatrix} \text{ and } F' = \begin{bmatrix} p^{t-1} & p^{t-2} & \cdots & p & 1 \\ 1 & p^{t-1} & \cdots & p^2 & p \\ p & 1 & \cdots & p^3 & p^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p^{t-2} & p^{t-3} & \cdots & 1 & p^{t-1} \end{bmatrix}$$

where $q = p^t - 1$ for some integers p and t. However, F is nonsingular in the finite field q, rendering it unusable. To make it operational, the requirement in **Property (b)** is that the values of F are much smaller than q. This approach may succeed, aligning with the previously considered solution.

B Chinese Remainder Theorem Example

Find the smallest integer solution to the following congruence equations.

$$\begin{cases} \mathbf{x} \equiv 2 \pmod 3 \\ \mathbf{x} \equiv 3 \pmod 5 \\ \mathbf{x} \equiv 2 \pmod 7 \end{cases} .$$

Step-by-Step Solution by Using CRT

Step-1. Verify Coprimality and Calculate Product $\mathbf{M}$

- The moduli 3, 5, and 7 are pairwise coprime.
- Calculate the product of the moduli $3 \times 5 \times 7 = \mathbf{M}$ is 105.

Step-2. Calculate $\mathbf{M}_i$ *for each modulus*

- $\mathbf{M}_1 = \mathbf{M}/3 = 105/3 = 35$
- $\mathbf{M}_2 = \mathbf{M}/5 = 105/5 = 21$
- $\mathbf{M}_3 = \mathbf{M}/7 = 105/7 = 15$

Step-3. Find the Multiplicative Inverse $\mathbf{t}_i \triangleq (\mathbf{M}_i)^{-1} \mod m_i$

- For $\mathbf{M}_1 = 35$, we need $\mathbf{t}_1$ such that $\mathbf{t}_1 \times 35 \equiv 1 \pmod 3$. Testing small values, $\mathbf{t}_1 = 2$ works as $2 \times 35 = 70 \equiv 1 \pmod 3$.
- For $\mathbf{M}_2 = 21$, we need $\mathbf{t}_2$ such that $\mathbf{t}_2 \times 21 \equiv 1 \pmod 5$. Here, $\mathbf{t}_2 = 1$ works as $1 \times 21 = 21 \equiv 1 \pmod 5$.
- For $\mathbf{M}_3 = 15$, we need $\mathbf{t}_3$ such that $\mathbf{t}_3 \times 15 \equiv 1 \pmod 7$. Here, $\mathbf{t}_3 = 1$ works as $1 \times 15 = 15 \equiv 1 \pmod 7$.

Step-4. Compute $\mathbf{x}$ *Using the CRT Reconstruction Formula*

- Using the formula $\mathbf{x} = \sum_i r_i \mathbf{t}_1 \mathbf{M}_i \pmod M$ where r_i are the remainders. That is,

$$\mathbf{x} = 2 \cdot 2 \cdot 35 + 3 \cdot 1 \cdot 21 + 2 \cdot 1 \cdot 15 = 233 \equiv 23 \pmod{105}.$$

Thus, the solution to the above congruence system is $\mathbf{x} = 23$.

C Numerical Data for Figures

Table 4. Numerical results depicted in Figure 4.

n	Without CRT (ϕ)	With CRT (ϕ)	Compression (ϕ)	Number of Moduli
8	7	128		3
9	8	256		3
10	15	512	2	3
11	23	1024	2	3
12	30	2048	3	3
13	58	4096	4	3
14	78	8192	6	3
15	169	16384	9	2
16	259	32768	15	2
17	383	65536	25	2
18	649	131072	38	2
19	1106	262144	64	2
20	2724	524288	101	2
21	4961	1048576	174	2
22	8881	2097152	266	2
23	16851	4194304	567	2
24	40185	8388608	1052	2
25	78165	16777216	1787	2
26	138984	33554432	3970	2
27	277512	67108864	5911	2
28	750000	134217728	9853	2

Table 5. Numerical results depicted in Figure 5.

n	14	15	16	17	18	19	20	21	22	23	24	25	26	27
CRT min	1	1	2	2	3	4	7	10	11	15	19	27	34	43
CRT Max	14	20	22	33	41	53	79	92	113	118	152	167	165	207
Original min	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Original Max	2	2	2	2	2	2	1	1	1	1	2	1	2	1