

Article

Optimizing Message Range and Ciphertext Storage in Gentry–Sahai–Waters Encryption Using Chinese Remainder Theorem and PVW-like Compression Scheme

Kung-Wei Hu * , Huan-Chih Wang , Wun-Ting Lin and Ja-Ling Wu 

Department of Computer Science and Information Engineering, National Taiwan University,
Taipei City 10617, Taiwan; whcjimmy@cmlab.csie.ntu.edu.tw (H.-C.W.); d07922020@ntu.edu.tw (W.-T.L.);
wjl@cmlab.csie.ntu.edu.tw (J.-L.W.)

* Correspondence: r11922141@cmlab.csie.ntu.edu.tw

Abstract: This paper explores advancements in the Gentry-Sahai-Waters (GSW) fully homomorphic encryption scheme (FHE), addressing challenges related to message data range limitations and ciphertext size constraints. We leverage the well-known parallelizing technology—the Chinese Remainder Theorem (CRT)—to tackle the message decomposition, significantly expanding the allowable input message range to the entire plaintext space. This approach enables unrestricted message selection in the GSW scheme and supports parallel homomorphic operations without intermediate decryption. Additionally, we adapt existing ciphertext compression techniques, such as the PVW-like scheme, to reduce the memory overhead associated with ciphertexts. Our experimental results demonstrate the effectiveness of combining the proposed CRT-based decomposition with the PVW-like compression 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 in successive HE operations. 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; PVW-like compression; ciphertext compression



Academic Editor: Carlo Blundo

Received: 30 December 2024

Revised: 20 February 2025

Accepted: 21 February 2025

Published: 24 February 2025

Citation: Hu, K.-W.; Wang, H.-C.; Lin, W.-T.; Wu, J.-L. Optimizing Message Range and Ciphertext Storage in Gentry–Sahai–Waters Encryption Using Chinese Remainder Theorem and PVW-like Compression Scheme. *Cryptography* **2025**, *9*, 14. <https://doi.org/10.3390/cryptography9010014>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Cryptography has significantly evolved with the advent of FHE, a groundbreaking concept that enables computations on encrypted data without requiring decryption in advance. This paper delves into the intricacies of the Gentry-Sahai-Waters (GSW) fully homomorphic encryption scheme [1], focusing on the challenges and solutions related to the upper bound of message data ranges and constraints on ciphertext size.

The GSW scheme supports conducting addition and multiplication homomorphically in the encryption domain, which is essential for performing complex computations directly on encrypted data. However, each HE operation increases the error in the ciphertext, potentially growing exponentially with successive operations. This necessitates mechanisms to control the error growth to ensure obtaining correct results. This work explores methods to mitigate error accumulation, notably employing the CRT for message decomposition. This selection significantly extends the allowable message range of the GSW scheme to the entire plaintext space. However, when using the existing GSW variant decryption

method [2], successful message decryption necessitated meticulous design, rendering the process unintuitive and cumbersome.

Consequently, we opted for an approximation method [3]. While this approach may 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 need help with the shortage of extensive ciphertext storage requirements. Therefore, this study also examines compression techniques proposed in [1] to reduce ciphertext memory overhead. 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.

In summary, this write-up makes several key contributions to the field of FHE, explicitly focusing on improvements and extensions to the GSW scheme:

1. We propose a practical approach to message range extension in the GSW scheme by leveraging CRT for message decomposition. This method significantly expands the allowable input 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 [3]. This adoption 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 GSW FHE scheme's practicality, efficiency, and applicability, paving the way for its broader adoption in secure computation scenarios.

1.1. Related Work

1.1.1. 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, sensitive data must be encrypted before being uploaded to the cloud.

FHE was first introduced by Rivest, Adleman, and Dertouzos [4] in 1978. However, it wasn't until 2009 that Gentry proposed the first FHE scheme [5], which allowed computations on plaintext without decryption through bootstrapping a somewhat homomorphic encryption (SWHE) scheme by adding a noise refresh process. 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.

Computationally, Partially Homomorphic encryption (PHE) schemes are generally more efficient than SWHE and FHE schemes, mainly because they are homomorphic concerning only one type of operation: addition or multiplication. SWHE is more general than PHE because it supports only a limited number of homomorphic operations. For those interested in the early-stage studies of homomorphic encryption (HE), the following write-ups are highly recommended: Acar et al. [6] provide a comprehensive overview of the

development history of HE, summarizing the underlying theories and detailing various prominent HE algorithms. They also compare the implementation libraries of several HE algorithms developed prior to 2018. Alaya et al. [7] explore different application approaches for HE and present a range of scenarios, including medical treatment, image processing, and other relevant fields. Wood et al. [8] offer insights into applying fully homomorphic encryption (FHE) in medicine and bioinformatics, explaining how it can be effectively implemented by considering the distinct characteristics of each domain.

FHE supports directly conducting both homomorphic addition and multiplication operations on ciphertexts. The main merit of FHE is that it offers cryptographically strong guarantees to some privacy and security-sensitive applications in various fields, such as Healthcare, Finance, and Online Auctions [9]. Still, these guarantees come at the cost of massive computational overhead. Therefore, many researchers have shifted their investigation attention to FHE acceleration. In general, there are two different research directions: First, it focuses on algorithm optimization to accelerate the FHE schemes [10,11]. Second, it uses CPU, GPU, and FPGA hardware to accelerate computation [12–14]. Although the FHE acceleration schemes have made good progress, there is still a large room for reducing the gaps between theoretical development and practical usage.

In the survey paper [15], the authors categorize all current FHE schemes into four generations based on their proposal order and mention several simple applications. The second generation mentioned in [15] includes BV [16,17], BGV [18], and B/FV [19] schemes. These can operate in finite fields and allow adequate packing, meaning computations can be performed simultaneously on entire vectors of integers. However, the papers in this generation don't handle noise exceptionally well.

According to [15], the third generation includes schemes like GSW (the focus of our work) [1], FHEW [20], and TFHE [10]. These schemes aim to address the issue of excessive noise by converting the process of previously published FHE methods into bit-wise operations and improving the bootstrapping technique based on the skills proposed by Gentry in [5,21]. Generally, the schemes in this generation can reduce error generation, but they lack packing support.

Finally, the most famous scheme in the fourth generation is CKKS [22]. 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 parallelizing skill reduces the average computation time for individual operations. Moreover, it can be extended to operations on real numbers.

1.1.2. Ciphertext Compression

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

Using smaller compression keys and packing multiple ciphertexts (encrypted using the same secret key) are the two commonly used approaches to perform FHE ciphertext compression. Naehrig et al. [23] proposed switching to smaller parameters in the Learning with Errors (LWE)-based FHE schemes before sending the ciphertexts out. Ref. [24] presents an enhanced compression scheme that uses an additive homomorphic encryption (AHE) scheme with small ciphertexts to compress large homomorphic ciphertexts based on LWE, achieving up to 90% for single ciphertexts and 99% for many ciphertexts. Key switching is a standard intermediate procedure in the Ring Learning with Errors (RLWE)-based FHE

schemes. Such techniques result in smaller sizes of ciphertext containing the same message as the original one.

Initially, “ciphertext packing” was proposed in [18,25], which involves aggregating and encrypting multiple plaintexts into a single ciphertext. Later research [2] introduced a method that effectively reduces the dimensionality of the ciphertext while still encrypting multiple plaintexts simultaneously, yielding better results than previous techniques. Other works focused on compressing GSW ciphertexts [2,26], where the ciphertexts are used internally in following procedures such as bootstrapping [10].

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 present some ideas and associated mechanisms to deal with these problems. (d) Sections 6 and 7 present the final experimental results and conclusions.

Since FHE involves lots of definitions and mathematical derivations, we defined and reported the parameters used in various addressed schemes and the corresponding experiments as clearly as possible in the Appendices A and B. For better readability, we also put most of the detailed definitions of the discussed cryptographic schemes and their mathematical derivations in Appendix A. Moreover, for completeness, an illustrative detailed experimental numerical data are presented in Appendix B.

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 $\mathbf{s}\mathbf{A} + \mathbf{e} \equiv \mathbf{b} \pmod{q}$,

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

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 $\mathbf{s}\mathbf{A} = \mathbf{b}$ 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, ensuring 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 encryption scheme [1], 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 $\mathbf{G}(\mathbf{G}^{-1}(\mathbf{C})) = \mathbf{C}$, ensuring that the original ciphertext $\mathbf{C}$ is recoverable.

The previously mentioned ciphertext flattening is achieved through the ‘short preimage’ function $\mathbf{G}^{-1}$ discussed in [27,28], which is why [29] refers to it as the Flattening Gadget. This useful tool in lattice-based cryptosystems can transform a vector with high norms into a higher-dimensional vector with low norms while preserving linear algebraic properties. The transformation $\mathbf{G}^{-1}$ decomposes each coordinate into its binary representation:

$$\mathbf{G}^{-1} \left(\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} \right) = \begin{bmatrix} c_{0,0} \\ \vdots \\ c_{0, \lceil \log_2 q \rceil} \\ c_{1,0} \\ \vdots \\ c_{1, \lceil \log_2 q \rceil} \\ \vdots \\ c_{n,0} \\ \vdots \\ c_{n, \lceil \log_2 q \rceil} \end{bmatrix}$$

where each c_i in the input vector is decomposed into binary coefficients $c_{i,j} \in 0, 1$ such that:

$$c_i = \sum_{j=0}^{\lceil \log_2 q \rceil} c_{i,j} \cdot 2^j.$$

This effectively represents each coordinate as a binary vector of length $\lceil \log_2 q \rceil$.

GSW frequently applies this tool to matrices as well. This application significantly reduces the error burden during ciphertext computations in GSW, as matrices are converted to smaller values, typically in binary representation. Consequently, the computation results remain manageable, leading to substantially reduced error generation. Therefore, G^{-1} is an efficiently computable function.

Typically, the structure of the gadget matrix $\mathbf{G}$ is defined as $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}$, where $\mathbf{I}_n$ is the identity matrix of size n , and $\mathbf{g}$ is a vector given by $\mathbf{g} = (1, 2, \dots, 2^{\lceil \log_2 q \rceil})$. 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 $2^{\lceil \log_2 q \rceil}$. The specific form of $\mathbf{G}$ is illustrated as follows:

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & \dots & 2^{\lceil \log_2 q \rceil} & 0 & \dots & 0 \\ & & & & 1 & 2 & \dots & 2^{\lceil \log_2 q \rceil} & \dots & 0 \\ & & & & \vdots & & \ddots & & & \vdots \\ & & & 0 & 0 & \dots & 1 & 2 & \dots & 2^{\lceil \log_2 q \rceil} \end{bmatrix}.$$

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 $m_1, m_2, \dots, m_n$ are pairwise coprime, then for any integers $r_1, r_2, \dots, r_n$, the system of equations

$$C : \begin{cases} x \equiv r_1 \pmod{m_1} \\ x \equiv r_2 \pmod{m_2} \\ \vdots \\ x \equiv r_n \pmod{m_n} \end{cases} \quad (1)$$

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

1. Let $\mathbf{M} = m_1 \times m_2 \times \dots \times m_n = \prod_{i=1}^n m_i$ be the product of integers $m_1, m_2, \dots, m_n$, and let $\mathbf{M}_i = \mathbf{M}/m_i, \forall i \in \{1, 2, \dots, n\}$, 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 : $\mathbf{t}_i \mathbf{M}_i \equiv 1 \pmod{m_i}, \forall i \in \{1, 2, \dots, n\}$.
3. The general solution of the system of Equation (1) takes the form: $x = r_1 \mathbf{t}_1 \mathbf{M}_1 + r_2 \mathbf{t}_2 \mathbf{M}_2 + \dots + r_n \mathbf{t}_n \mathbf{M}_n + k\mathbf{M} = k\mathbf{M} + \sum_{i=1}^n r_i \mathbf{t}_i \mathbf{M}_i, k \in \mathbb{Z}$. 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.

2.4. Regev and PVW Encryption

In the Regev encryption scheme [30], 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 $\langle s, c \rangle = \lceil q/2 \rceil \cdot \sigma + e \pmod{q}$, 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. Notice that the plaintext space has been 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 $\langle s, c \rangle = \lceil q/p \rceil \cdot \sigma + e \pmod{q}$, 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 [25] 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 $\Sigma \in \mathbb{Z}_p^{k \times m}$. Here, the decryption of the two ciphertexts results in $S \cdot c = \lceil q/p \rceil \cdot \sigma + \mathbf{e} \pmod{q}$ and $SC = \lceil q/p \rceil \cdot \Sigma + \mathbf{E} \pmod{q}$, 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. The GSW-like Encryption Scheme

This section introduces the GSW homomorphic encryption scheme [1], illustrated in Figure 1, which offers significant data security and privacy guarantees, presents considerable challenges, particularly in managing errors and limiting message sizes.

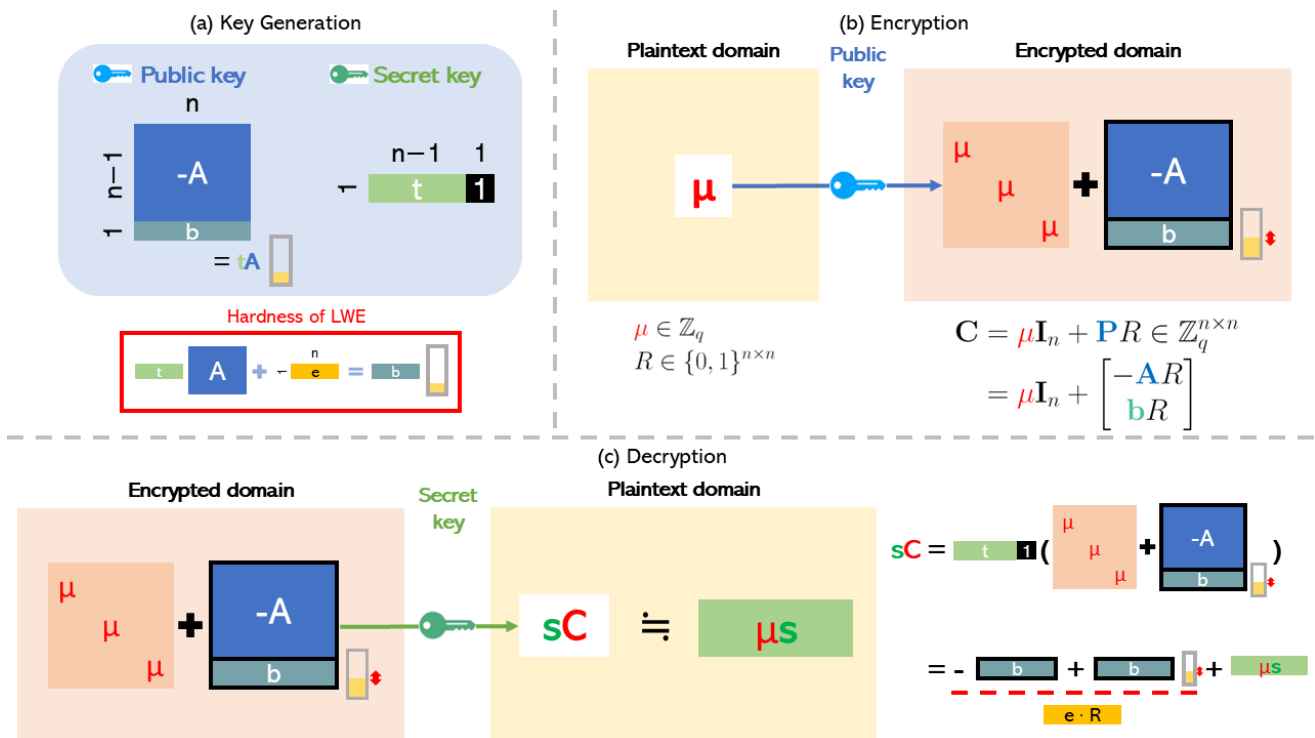


Figure 1. The initial encryption is accomplished through (a) key generation, (b) encryption, and (c) decryption processes. This diagram snapshots the detailed mathematical operations involved in each process.

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 [3] 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.

3.1. 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 b is computed as $b = tA + e$, following the LWE scheme. In Figure 1, a bar chart adjacent to vector b displays the current error magnitude as a yellow segment. Successful decryption requires the error to remain below the bar chart's upper bound. The public key P is then defined as $P = \begin{bmatrix} -A \\ b \end{bmatrix} \in \mathbb{Z}_q^{n \times m}$ and the secret key s as $s = [t \mid 1] \in \mathbb{Z}_q^n$, ensuring that the product $s \times P \equiv e \pmod{q}$ confirms the error model of LWE.

3.2. 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 C is computed using the formula

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

where G typically represents a gadget matrix facilitating the embedding of μ within the encryption process. During encryption, interactions between $\mathbf{b}$ and R affect the error magnitude, shown by the dynamic error level changes in the bar chart. These variations significantly impact decryption accuracy and scheme security.

The range for selecting the message μ was initially limited to between 0 and 1 due to error considerations in 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 $[0, \phi)$.

The range for selecting the message μ was initially limited to between 0 and 1 due to error considerations in the design of the GSW encryption system. However, in [2], authors discovered that the message range can be expanded while complying with the learning with errors problem (c.f. Section 2.1). Therefore, we also 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 $[0, \phi)$. The experimental results in Section 6.1 demonstrate that the range can be expanded.

Table 1 succinctly illustrates the memory space required to store the selected message's encrypted ciphertext under different security parameters, as shown graphically in Section 6.3. 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 grows as the security parameter increases. Notably, even at $n = 8$, a message initially 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.

Table 1. Memory requirements of GSW ciphertext concerning different security parameters before decryption. Numerical results are depicted in Figure 2.

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

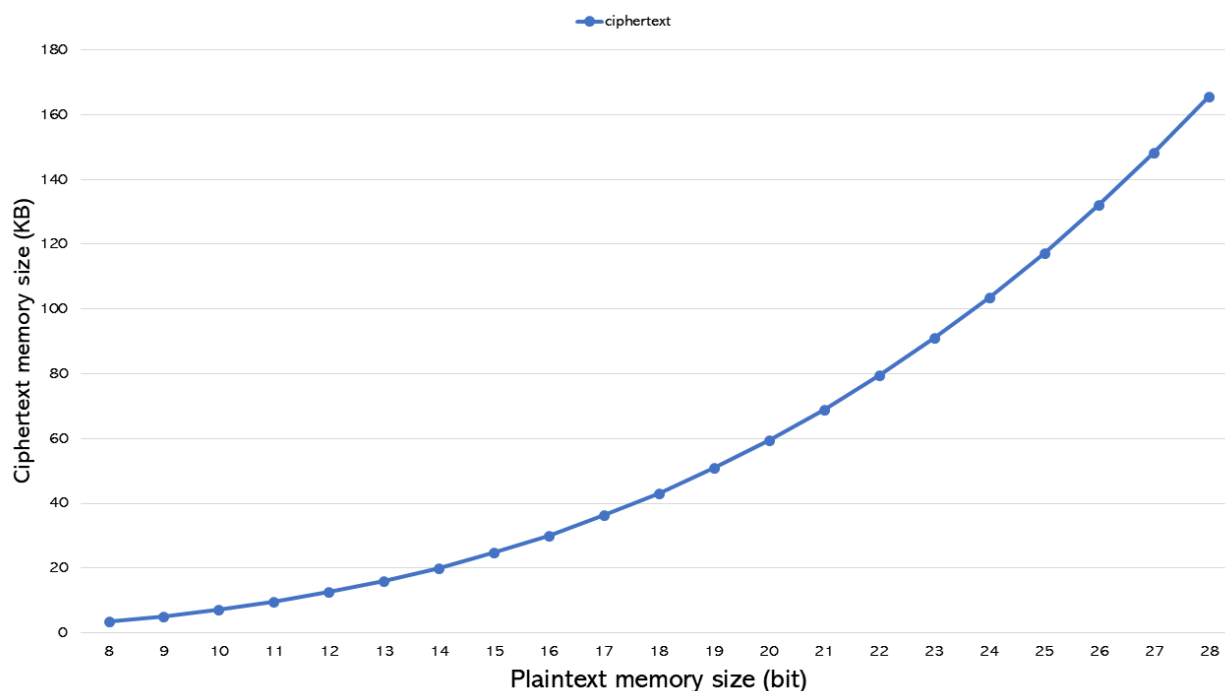


Figure 2. Comparative and quantitative analyses of GSW encryption scheme's plaintext and ciphertext on the memory size. The blue line indicates the memory size under different plaintext sizes.

3.3. 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

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

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

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

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.

3.4. 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 [2] and an approximate method [3]. Each aimed to address the complex challenge of error management.

1. A Variant of GSW [2]. 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:
 - $\mathbf{M} = \mathbf{G}^{-1}(0)$ such that $\mathbf{GM} = 0 \pmod{q}$, effectively nullifying the message-related component in the matrix multiplication due to congruence modulo q .
 - $\mathbf{M}$ 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 $\mathbf{Z} = \mathbf{s} \cdot \mathbf{C} \cdot \mathbf{M} = \mu \mathbf{sG} + e' \cdot \mathbf{M} = e' \cdot \mathbf{M} \pmod{q}$, which isolates the error due to $\mathbf{GM} = 0 \pmod{q}$.
- **Step 2:** Extract e' by applying the inverse of $\mathbf{M}$,

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

- **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 $\mathbb{Z}$ —creating 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 component 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 [3]. 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

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

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 the approaches mentioned above 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.

3.5. Homomorphic Operation

Two primary operations in the cryptographic framework that support homomorphic operations are homomorphic addition (C^+) and homomorphic multiplication (C^*). Let C_1 and C_2 be ciphertexts, respectively encrypting plaintexts μ_1 and μ_2 . The operations are mathematically described as follows:

- Homomorphic addition: $C^+ = C_1 + C_2 \pmod{q}$
- Homomorphic multiplication: $C^* = C_1 \times G^{-1}(C_2) \pmod{q}$

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

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

where each $c_{i,j} \in \mathbb{Z}_q$ is decomposed into a binary representation such that each binary bit $c_{x,y,z}$ lies in $\{0, 1\}$. The original element $c_{i,j}$ is then expressed as $c_{i,j} = c_{i,j,0} + 2^1 \cdot c_{i,j,1} + \cdots + 2^{\ell-1} \cdot c_{i,j,\ell-1}$, 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 to free their choices without restriction. As noted, errors can easily exceed the tolerance limit when selecting a large message. 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 following addressed base decomposition method (as shown in the middle of Figure 3) or the "CRT decomposition method", which uses CRT to expand the message range further (as shown at the bottom of Figure 3).

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.

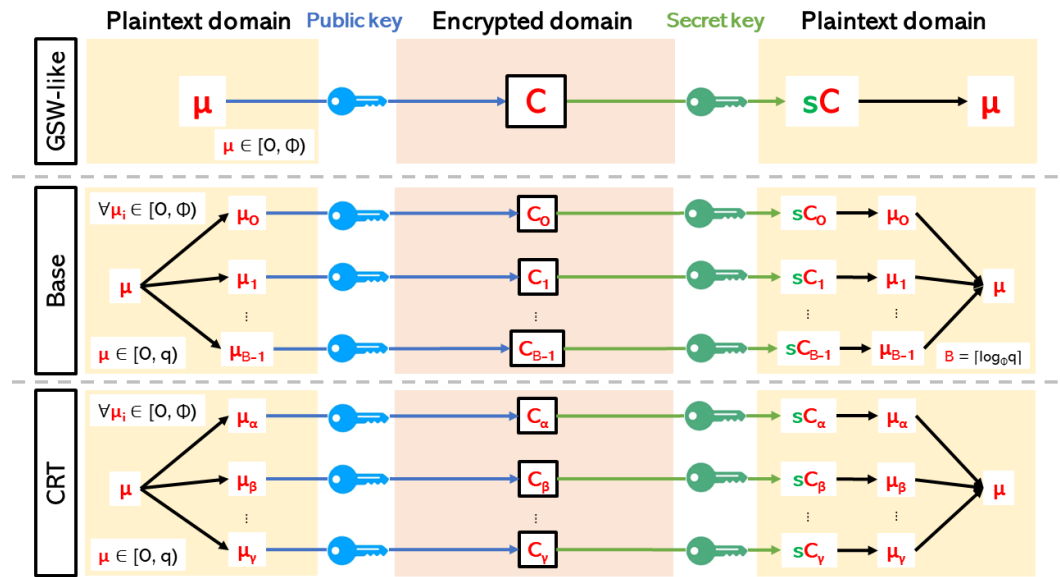


Figure 3. Simplified flowcharts for benchmarked schemes: GSW-like (**top**), Base Decomposition (**middle**), and CRT (**bottom**), respectively.

4.1. Base Decomposition (Figure 3, the Middle Block)

Key Generation, Decomposition and Encryption. The public key (P) and secret key (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 $(\mu_0, \mu_1, \dots, \mu_{\ell-1})$, expressed as $\mu = \mu_0 + 2 \cdot \mu_1 + \dots + 2^{\ell-1} \cdot \mu_{\ell-1} \pmod{q}$, 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 3.2, 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 $(\mu_0, \mu_1, \dots, \mu_{B-1})$, expressed as $\mu = \mu_0 + \phi \cdot \mu_1 + \dots + \phi^{B-1} \cdot \mu_{B-1} \pmod{q}$, 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 s with the ciphertext 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 $\mu = \mu_0 + \phi \cdot \mu_1 + \dots + \phi^{B-1} \cdot \mu_{B-1} \pmod{q}$. After computing the result, the original large message is successfully reconstructed.

4.2. CRT Decomposition (Figure 3, 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 $(m_\alpha, m_\beta, \dots, m_\gamma)$ is all the moduli utilized in the CRT. Each modulus is pairwise coprime and less than ϕ . Recombination also employs the CRT, formulated as:

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

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

The complete processes of the benchmarked schemes are briefly addressed 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 a GSW-based encryption scheme.

Table 2. The key generation (KeyGen), Decomposition (Decomp), Encryption (Enc), Decryption (Dec), Output, and Recombination (Recomb) processes of the benchmarked schemes.

	Bit	Base	CRT (Section 4.2)
KeyGen.	Public key : $\mathbf{P} = \begin{bmatrix} -A \\ \mathbf{b} \end{bmatrix} \in \mathbb{Z}_q^{n \times m}$, and Secret key: $\mathbf{s} = [t \mid 1] \in \mathbb{Z}_q^n$		
	Selecting the message $\mu \in [0, q)$ and $R \in \{0, 1\}^{m \times m}$		
Decomp.	$(\mu_0, \mu_1, \dots, \mu_{\ell-1})$	$(\mu_0, \mu_1, \dots, \mu_{B-1})$	$\begin{cases} \mu \equiv \mu_\alpha \pmod{m_\alpha} \\ \mu \equiv \mu_\beta \pmod{m_\beta} \\ \vdots \\ \mu \equiv \mu_\gamma \pmod{m_\gamma} \end{cases}$
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 $\forall i, \text{s.t. } \mathbf{sC}_i = \mu_i \cdot \mathbf{s} \times \mathbf{G} + e_i \pmod{q} \in \mathbb{Z}_q^n$		
Output.	Approximate method		
Recomb.	$\mu = \sum_i 2^i \mu_i$	$\mu = \sum_i \phi^i \mu_i$	$\mu = \sum_i \mu_i \mathbf{t}_i \mathbf{M}_i$

4.3. 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, the CRT-based method supports parallel computation, a capability not available in base decomposition-based methods. Furthermore, the base decomposition approach frequently encounters issues with mathematical carry operations, which are inherently absent in the CRT method. This advancement increases the potential and applicability of the proposed encryption scheme in daily use.

4.3.1. Base Decomposition

A straightforward conclusion is that the base decomposition approach cannot perform homomorphic addition after decomposition; the main reason comes from the occurrence of mathematical carry. A *mathematical carry* is possible when the decomposed bits or elements are added to the original plaintext. Unfortunately, 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 induced complexity prevents efficient decryption after such operations.

Moreover, even if a technique exists to manage mathematical 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 the plaintext decompositions are conducted using base decomposition.

4.3.2. CRT Decomposition.

On the other hand, the CRT decomposition avoids carrying issues, facilitating parallel computation, as shown in Figure 4. However, an additional limitation arises: it must be ensured that the aggregate of selected messages, $(\sum_j \mu^j)$, remains below the modulus value M to avoid operational failures. Despite this limitation, the CRT decomposition method allows selecting messages without constraints and effectively supports homomorphic addition, even over many consecutive operations without interruption. Therefore, this limitation is acceptable. (According to the definition of CRT, similar arguments hold for homomorphic multiplications.)

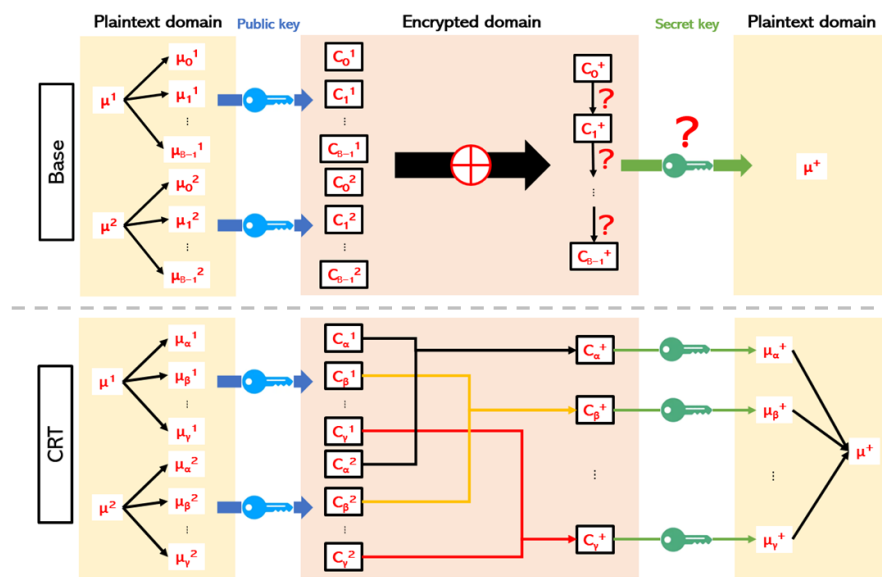


Figure 4. The processes of the base decomposition (the top row), the CRT decomposition (the bottom row), and the homomorphic addition (the middle column).

5. Ciphertext Compression

As depicted in Table 1, storing a brute-force GSW-encrypted ciphertext requires a significant amount of 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.

Reference [2] has proposed a compression method applicable to the GSW encryption scheme, introducing two compression techniques: “PVW-like” and “Nearly Square Gadget Matrix”. The kernel merit of [2] to achieve efficient ciphertext compression is founded on encrypting multiple messages simultaneously. Nevertheless, 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. For completeness, the details of the “Nearly Square Gadget Matrix” method will be included in Appendix A.4.2. It’s crucial to compare both methods to understand their strengths and limitations.

Regarding effectiveness, reference [2]’s approach, which compresses multiple messages simultaneously, will naturally provide more impressive results 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

Before introducing the PVW-like Scheme’s compression process, let us explain the core idea behind the method first. The top and bottom portions of Figure 5 show the uncompressed and compressed processes of the PVW-like scheme, respectively.

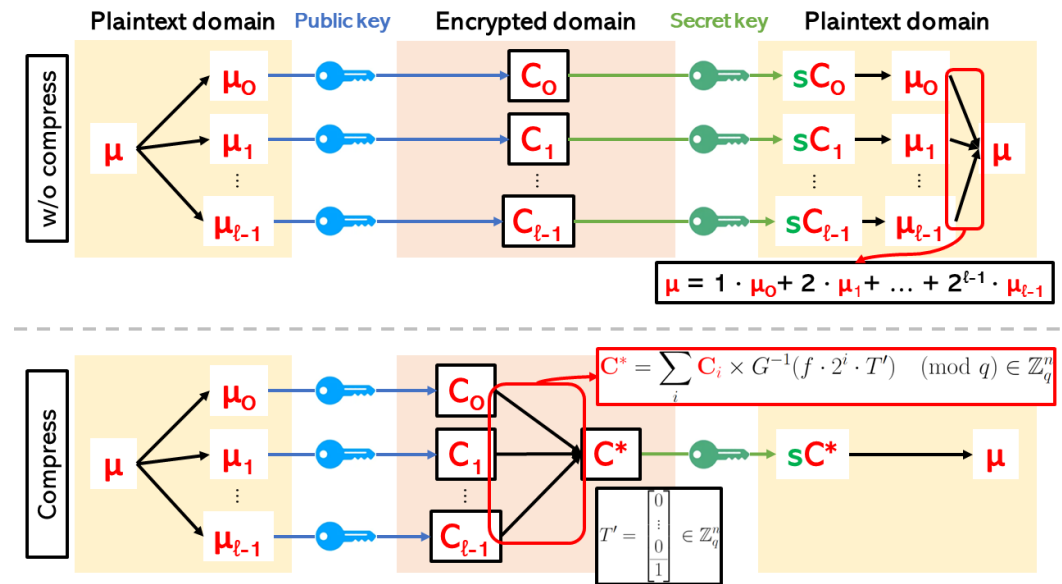


Figure 5. The architectures and the corresponding processes of PVW-like schemes: the upper part is for without compression case while the lower part is for compression one.

The top portion, which shows the uncompressed process, follows the same idea of the base decomposition scheme 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 this compression method is that it combines the computation steps earlier in the process. Specifically, instead of waiting until the final decryption and output stages are finished and combining the results, ref. [2] advances the combination process such that the combination happens immediately after each small message is encrypted.

5.1.1. 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 $\mathbf{s}$.

A message is selected during the encryption, and the two decomposition methods mentioned earlier can be used. After decomposing the message into smaller values, each value is encrypted separately.

For the discussion of this subsection, 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 $(\mu_0, \mu_1, \dots, \mu_{\ell-1})$, and the encrypted ciphertexts are $(C_0, C_1, \dots, C_{\ell-1})$.

5.1.2. 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:

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

In the above equation, 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 5. 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 the following.

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

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

For base decomposition f is equal to $\lceil q/p \rceil$, 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, $f = \lceil q/\mathbf{M} \rceil$, where $\mathbf{M}$ is the product of all the CRT decomposition moduli and represents the range of the plaintext domain.

5.1.3. Decryption and Its Output

Similarly, we can derive the decryption scenario after compression as follows:

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

$$\mathbf{s} \times \mathbf{C}^* = f \cdot \mu + \mathbf{e}' \pmod{q}. \quad (5)$$

Equation (4) evidences that the previously distributed errors are 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 in this context to build our output mechanism. That is,

$$\text{Output: } \left\lceil \frac{\mathbf{s} \mathbf{C}^*}{f} \right\rceil = \left\lceil \frac{f \cdot \mu + \mathbf{e}'}{f} \right\rceil = \mu \pmod{p}, \text{ with } \|\mathbf{e}'\| < \frac{f}{2}. \quad (6)$$

However, applying this method to CRT decomposition increases the likelihood of failure. This is caused by the limitations encountered during the original CRT recombination. In the final step of CRT, the reconstructed sum is calculated under the modulus $\mathbf{M}$, yielding an unknown value. The result tends to be very large without performing the modulo $\mathbf{M}$ operation. In this case, after dividing by f , the output directly yields a value between 0 and $\mathbf{M} - 1$, rather than the expected result from CRT. This discrepancy results in a high

failure rate in our experiments. Experimental results revealed that if the result of $q/\mathbf{M}$ is an exact integer or the decimal produced after division is tiny, the method mentioned above is more likely to succeed. However, this requires careful selection and testing of the CRT's modulus set $(m_\alpha, m_\beta, \dots, m_\gamma)$ to ensure proper functionality.

Although choosing each m_i meticulously may find a workable $\mathbf{M}$, trying each modulus one by one inhibits its usage in practice. Therefore, we attempted to modify the output method mentioned above. The process in [2] treats the decrypted result as a scalar, so they subsequently divide by f , allowing the result to be mapped back to the original plaintext space. However, their implementation of “division by f ” seems somewhat unreasonable.

In Equation (6), $f \cdot \mu + \mathbf{e}'$ is, in fact, a result in the finite field $\mathbb{Z}_q$. In finite fields, division by f should be realized using the multiplicative inverse of f , denoted as f^{-1} , rather than directly dividing by f as done in a real-number field. Therefore, we intended to change the output method to $\mathbf{sC}^* \cdot f^{-1}$. We further divided the use cases into those with and without errors to verify this rewritten formula's correctness and the associated security. That is,

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

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 $\mathbf{M}$. In other words, performing an additional modulus $\mathbf{M}$ step does yield the correct result, indicating that this approach is feasible.

However, when testing for security by introducing noise, $\mathbf{e}'$ will be multiplied by a scalar f^{-1} , 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, although our proposed adjustment is mathematically sound, the inability to control the noise amount limits its applicability. This explains why [2] treats the decrypted result as a scalar and subsequently divides it by f to correctly allow the result to be mapped back to the original plaintext space.

6. Experiment

We present a comparative analysis and experimental results of the above technologies, especially focusing on comparing CRT decomposition and compression techniques. This investigation examines the impact of our proposed CRT decomposition method and compression technique on the upper bound ϕ , observes the consecutive computational iterations achievable by the CRT decomposition method without decryption, and finally compares the differences in the ciphertext size required in memory for the various decomposition methods and compression mentioned above. We aim to verify whether these modifications align with our hypotheses. The actual numerical data involved in the following Figures are provided in Appendix B.

6.1. Upper Bound ϕ

Figure 6 represents our experimental upper bound results of the CRT decomposition method with and without the compression technique. It contains four entries: the last three curves represent the growth of the upper bound ϕ related to the increase of the security parameter for operations “without (denoted as *Without*)”, “with (denoted as *CRT*)” applying the CRT decomposition method and the compression technique (denoted as *Compression*). In contrast, the first entry (the green bar) indicates the minimum number of required moduli. The horizontal axis denotes the applied security parameters, the left

vertical axis represents the corresponding upper bound ϕ (displayed logarithmically), and the right vertical axis indicates the number of used moduli.

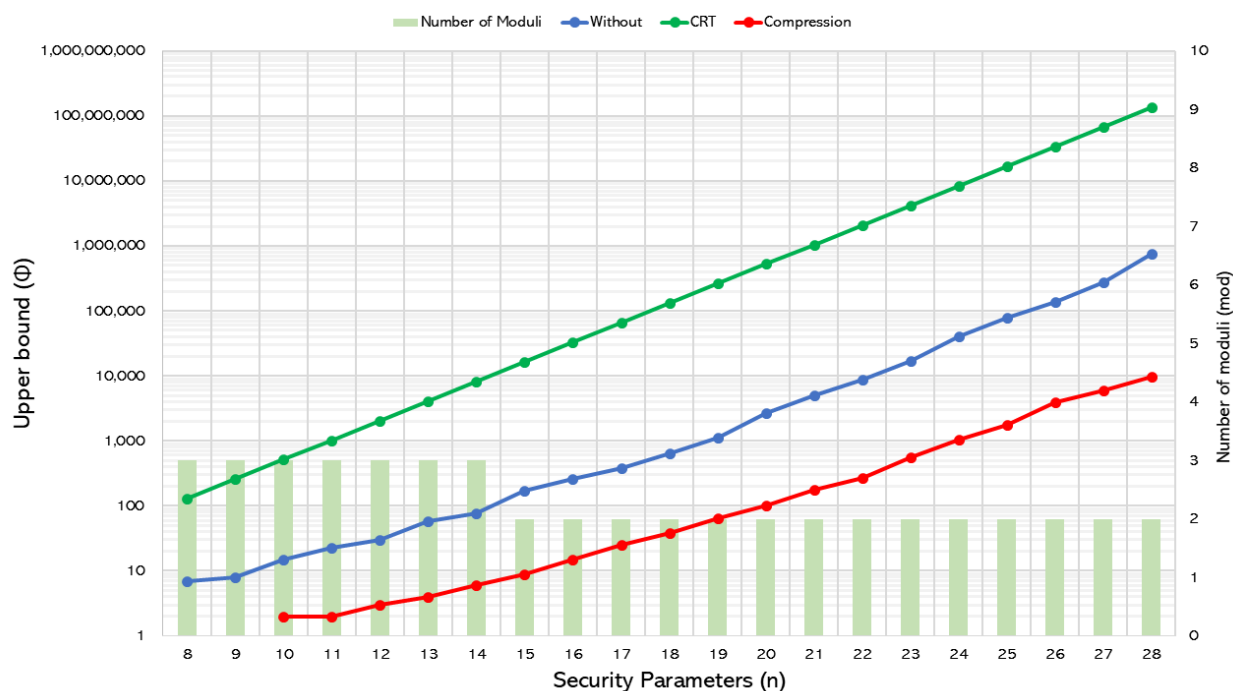


Figure 6. Comparative and quantitative Analyses of CRT Decomposition and Compression Methods on the Upper Bound ϕ of Message Range. The red line indicates the effect of compression, and the green bars indicate the number of used moduli.

From Figure 6, several remarks can now be drawn as follows:

1. It can be observed that our proposed CRT decomposition method significantly increases the upper bound. Regardless of the security parameter 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 [2] is well-suited for this scenario. Due to error considerations, the original PVW encryption system also restricts its 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 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 necessary to reach the goal.

6.2. Sequential Computation

Practical applications rarely involve just a single HE operation. Therefore, it is essential to examine how the encryption scheme, after our modifications, performs multiple consecutive additions without decryption. As illustrated in Figure 7, we analyze the performance under varying security parameters. The first two colored labels (orange and red) in the Figure represent the minimum and maximum consecutive operations successfully observed in 1000 repeated tests using our CRT method (described in Section 4.2). The remaining label (deep-blue) indicate the maximum successful consecutive operations for the original

GSW method without decomposition (cf. Section 3). The vertical axis displays the number of consecutive HE additions, the left axis pertains to our proposed process, and the right axis corresponds to the original approach.

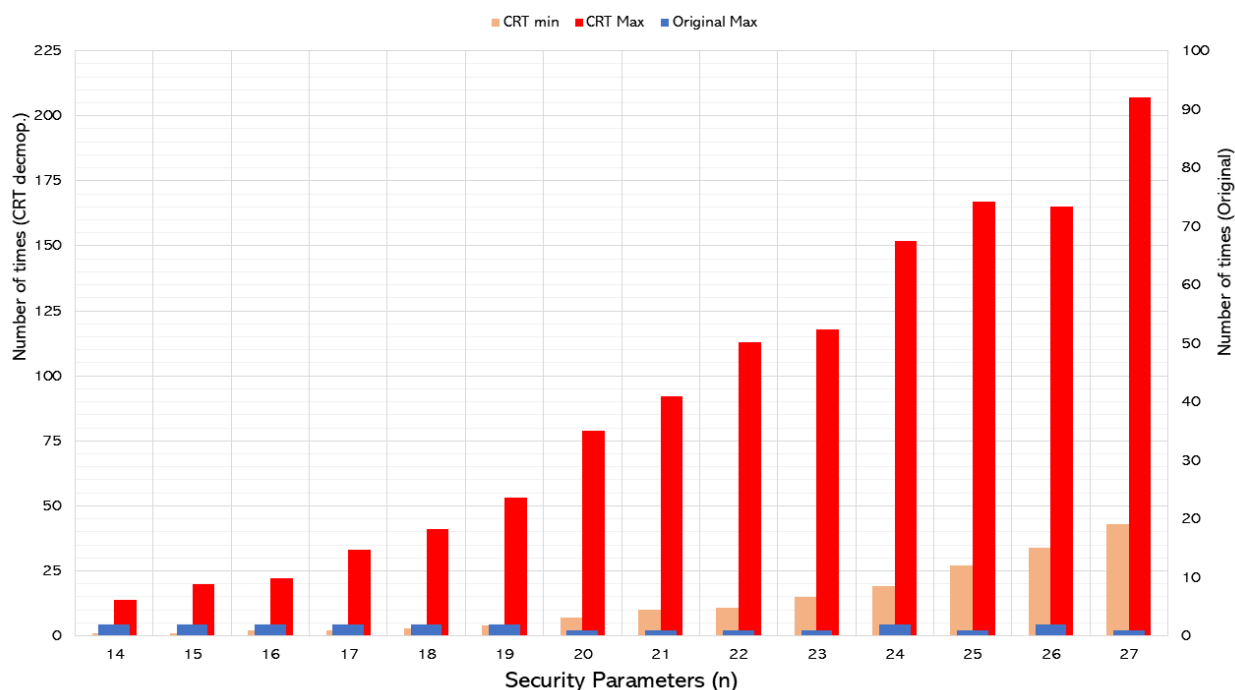


Figure 7. Comparing the number of consecutive operations that can be performed without decomposition (cf. Section 3 and indicated in deep-blue) and with CRT decomposition (cf. Section 4.2 and shown in orange and red) under different security parameters, all with the same message range in $[0, q)$.

From Figure 5, several remarks can now be drawn accordingly:

1. Our method outperforms the original GSW scheme across almost every security parameter. The original method, without decomposition, generally allows a maximum of one HE addition.
2. As the parameter n increases, the proposed method expands the numbers of both minimum and maximum successful consecutive operations. 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 the message range will make successfully executing a single HE addition more difficult. The CRT method demonstrates better noise tolerance due to its ability to perform more addition operations.

6.3. Memory Size Analysis

Figure 2 shows the relationship between plaintext and corresponding ciphertext of the GSW encryption scheme. The horizontal axis represents the number of bits of plaintext and also reflects different security parameter settings; the vertical axis represents the storage space required for the ciphertext generated after encryption in KB. It can be observed from the figure that even a small amount of plaintext bits will require considerable memory space to store its ciphertext after encryption. This highlights the storage efficiency challenges GSW encryption schemes in practical applications.

Figure 8 illustrates the comparison of ciphertext sizes generated by various decomposition techniques and compression methods previously discussed. The figure contains four different methods: bit decomposition (shown in gray), base decomposition (shown in blue),

CRT decomposition (shown in red), and the compression method (shown in green). All ciphertext sizes are measured in KB.

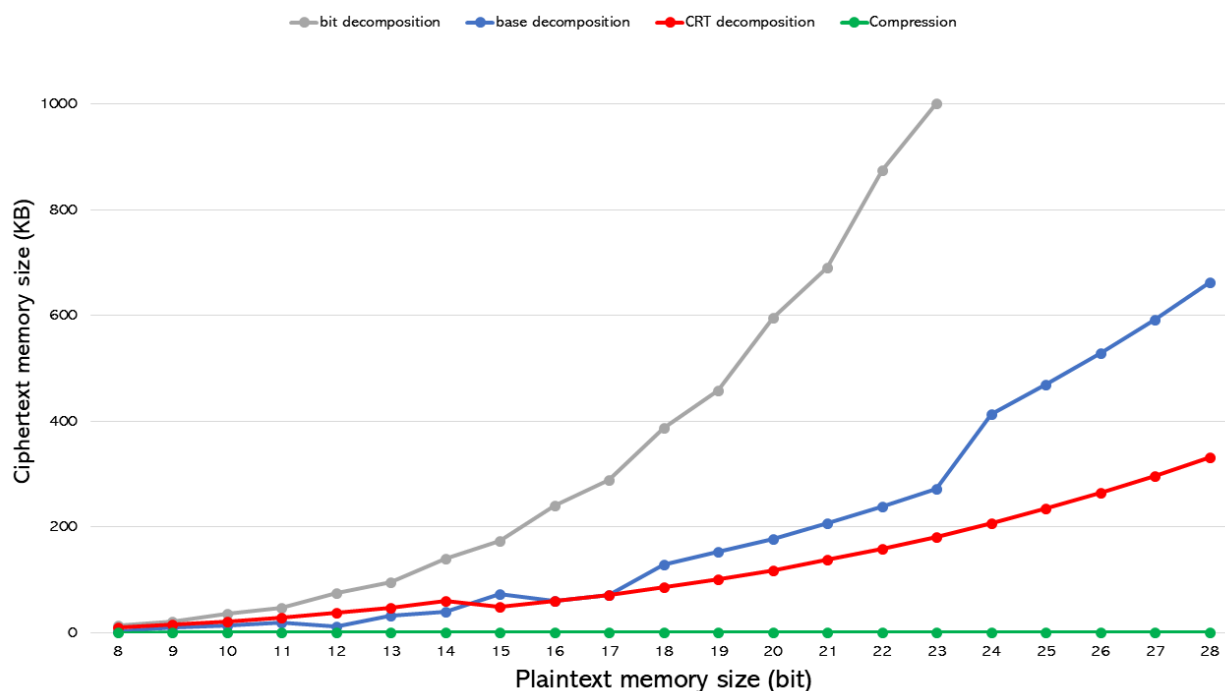


Figure 8. Comparing the ciphertext memory size across different techniques: three decomposition methods (cf. Sections 4.1 and 4.2 and indicated in grey, blue, and red) and compression method (cf. Section 5 and shown in green), measured against varying plaintext memory sizes from 8 to 28 bits.

From Figure 8, we can derive several key observations:

1. To allow users to freely select messages from the complete plaintext range, combined with bit decomposition's operational mechanism of splitting messages into individual bits, this method inevitably generates a considerable number of ciphertext components. This directly results in significantly larger storage space requirements compared to other methods.
2. As the plaintext size increases, our proposed CRT decomposition demonstrates excellent performance: when the plaintext length reaches 15 bits, only two values are needed to cover the entire plaintext range. This results in CRT decomposition requiring far less memory space than base decomposition for larger plaintext ranges.
3. The compression method achieves optimal space efficiency by combining the original ciphertext and substantially reducing dimensionality. The figure shows that its required storage space is significantly smaller than all decomposition techniques.

Our proposed CRT decomposition method facilitates unrestricted range message selection and supports parallelized, continuous, and uninterrupted HE computations. However, these performance gains come at the cost of increased memory space, as the original message has to be divided into many smaller values to fulfill the CRT representation, which may also result in slightly longer computation times.

6.4. Application Scenarios

We demonstrate the complete process from CRT decomposition to compression through a federated learning application in a single-key scenario. The application scenario can be envisioned as different branches of the same hospital performing computations while considering data privacy. All branches share the same public and secret keys, and their

hospital's server is cloud-based, requiring encrypted data for cloud computations. Figure 9 illustrates this process, showing actual computation results.

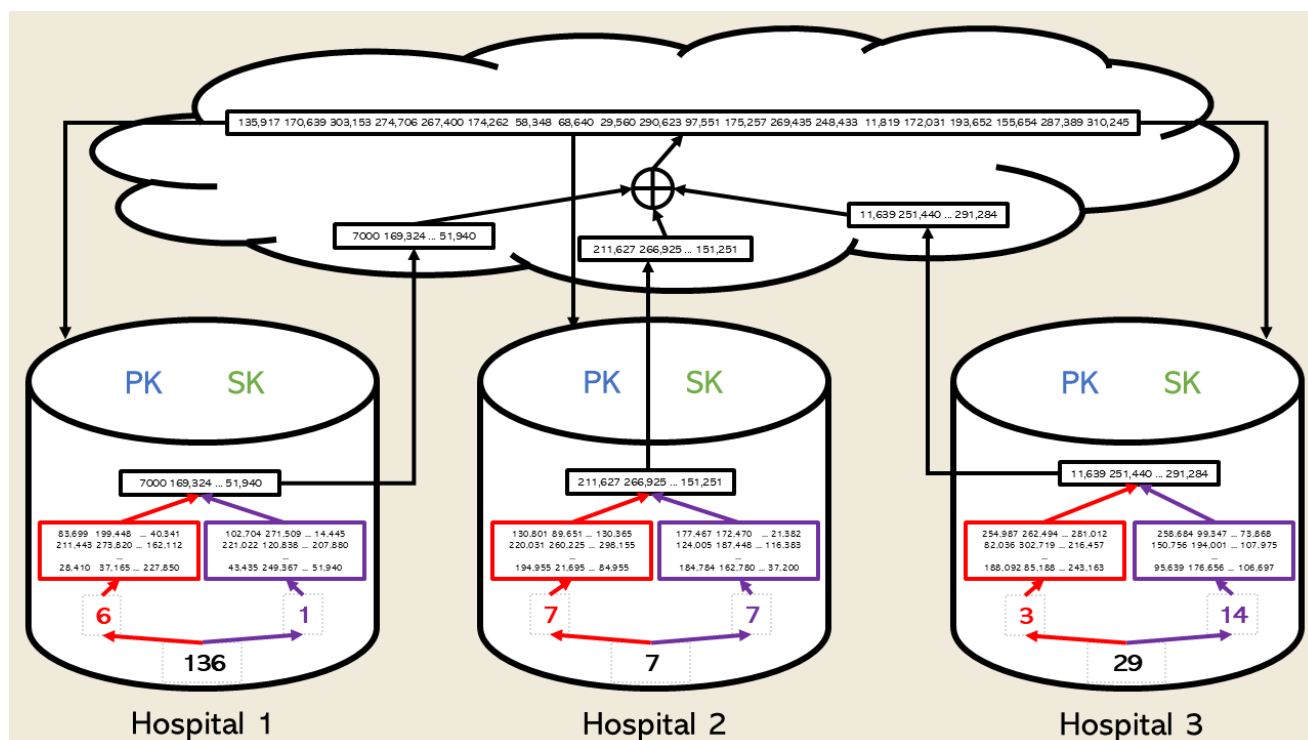


Figure 9. We simulate a simple single homomorphic addition operation under a practical Federated Learning scenario.

The process can be broken down into these steps:

1. **Initial Setup.** All three branches share the same public key (PK) and secret key (SK). The security parameter is set to $n = 20$.
2. **Message Selection and CRT Decomposition.** Each branch selects a message: 136, 7, and 29, respectively, with an expected final result of 172. CRT decomposition is then performed using moduli 13 and 15 to expand the operational message range, splitting each message into two smaller values.
3. **Encryption and Compression.** The decomposed values undergo GSW encryption followed by compression. Each branch only needs to transmit 272 bytes of encrypted compressed information to the cloud instead of the original 119 KB of encrypted data.
4. **Upload Data.** The computed ciphertexts are uploaded to the cloud.
5. **Homomorphic Addition.** The cloud performs homomorphic addition on the three compressed ciphertexts without decryption, maintaining the result size at 272 bytes.
6. **Return Result and Decryption.** The cloud returns identical computed results to all branches. Using their shared secret key, each branch can decrypt and obtain the expected outcome of 172.

While this simulation runs under relatively small security parameters due to computational resource and ciphertext size limitations, making real-world implementation currently challenging, it demonstrates the scheme's potential. The original GSW method made selecting messages arbitrarily within the entire plaintext space challenging and often failed to perform even a single operation. Our proposed CRT decomposition and compression significantly address these issues, greatly enhancing the practicality of the GSW encryption scheme.

7. Future Works and Conclusions

In conclusion, this write-up 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 more extensive message range and leverages existing ciphertext compression techniques to increase its value for practical applications. These enhancements are crucial for deploying secure cryptographic systems, where flexibility and efficiency are paramount.

Given that [2] initially applied compression methods to the GSW matrix secret key scheme and [31] subsequently referenced [2], categorizing it among various multikey FHE schemes that enhance ciphertext processing, our future work will focus on extending our current approach 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 can significantly advance FHE's practical applicability in real-world scenarios.

Author Contributions: Formal analysis, K.-W.H.; investigation, K.-W.H., H.-C.W., W.-T.L. and J.-L.W.; methodology, K.-W.H.; project administration, J.-L.W.; resources, J.-L.W.; algorithm development, K.-W.H.; supervision, H.-C.W., W.-T.L. and J.-L.W.; writing—original draft preparation, K.-W.H.; writing—review and editing, H.-C.W., W.-T.L. and J.-L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded and as a joint development program with the Taiwan Semiconductor Manufacturing Company, entitled Homomorphically Encrypted MLaaS POC. Also, this research was partially supported by the Minister of Science and Technology, Taiwan (grant number: MOST 111-2221-E-002-134-MY3) and National Taiwan University (grant number: NTU- 112L900902).

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Matrix Secret Key

Appendix 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 $k < m$,

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

Appendix A.2. Single Key

KeyGen. Initially, the matrix $\mathbf{A} \in \mathbb{Z}_q^{k \times m}$, $\mathbf{S} \in \mathbb{Z}_q^{n_0 \times k}$, and small $\mathbf{E} \in \chi^{n_0 \times m}$ 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 $\mathbf{B} = \mathbf{SA} + \mathbf{E} \in \mathbb{Z}_q^{n_0 \times m}$ is calculated. Finally, the public key,

represented by $\mathbf{P} = \begin{bmatrix} -\mathbf{A} \\ \mathbf{B} \end{bmatrix} \in \mathbb{Z}_q^{n_1 \times m}$, and the secret key, represented by $\mathbf{S} = [\mathbf{S} \mid I_{n_0}] \in \mathbb{Z}_q^{n_0 \times n_1}$, are generated, ensuring that $\mathbf{S} \times \mathbf{P} = \mathbf{E} \pmod{q}$.

Enc. Selecting the message $\mu \in [0, \phi)$ and $R \in \{0, 1\}^{m \times m}$, and encrypting it 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 ($\mathbf{C}$), the result is computed using formula $\mathbf{SC} = \mu \cdot \mathbf{S} \times \mathbf{G} + \mathbf{S} \times \mathbf{P} \times R = \mu \cdot \mathbf{S} \times \mathbf{G} + \mathbf{E}' \pmod{q} \in \mathbb{Z}_q^{n_0 \times m}$.

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

Appendix A.3. All Decomposition Process

In this study, Table A1 illustrates the processes of all decomposition methods. The term ‘Bit’ indicates using base decomposition with a base of 2, corresponding to the approach in [2]. As shown in Table A1, the entire process of the three investigated schemes differs only in the decomposition and the 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, assuming all elements in each decomposition fall 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 rebuild the user’s initially selected message.

Table A1. All benchmarked matrix secret key schemes’ decomposition processes.

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

Appendix A.4. Ciphertext Compression

Under a matrix secret key, the compression of ciphertext involves compressing multiple messages together, as suggested in [2]. The effectiveness of this compression method is expected to be significantly better than compressing a single message using a vector secret key.

Following the approach in [2], 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

$w \in [\ell]$. The original compression methods, *PVW-like* and *nearly square gadget matrix*, can be utilized.

Appendix A.4.1. PVW-like

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

$$\begin{aligned} \text{Bit Decomposition} &\implies \mathbf{C}^* = \sum_{u,v,w} \mathbf{C}_{u,v,w} \times G^{-1}(f \cdot 2^w \cdot T'_{u,v}) \in \mathbb{Z}_q^{n_1 \times n_0} \\ \text{Base Decomposition} &\implies \mathbf{C}^* = \sum_{U,V,W} \mathbf{C}_{U,V,W} \times G^{-1}(f \cdot \phi^W \cdot T'_{U,V}) \in \mathbb{Z}_q^{n_1 \times n_0} \\ \text{CRT Decomposition} &\implies \mathbf{C}^* = \sum_{i,j,k} \mathbf{C}_{i,j,k} \times G^{-1}(f \cdot \mathbf{t}_k \cdot \mathbf{M}_k \cdot T'_{i,j}) \in \mathbb{Z}_q^{n_1 \times n_0}, \end{aligned}$$

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 larger number 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:

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

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,

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

Appendix A.4.2. 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,

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

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 ,

$$\mathbf{Z} F^{-1} = (\mathbf{E}' \times F \pmod{q}) \times F^{-1} = \mathbf{E}'. \quad (\text{A1})$$

- **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 .

Note that this approach has the same issues as the methodology described in Equation (3). Since in Equation (A1), $\mathbf{E}' \times F \pmod{q}$ and F^{-1} exist under different field conditions, this is theoretically unreasonable. Ref. [2] 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 $\mathbb{Z}_q$, 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.

Appendix B. Numerical Data for Figures 6 and 7

Table A2. Numerical results depicted in Figure 6.

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	16,384	9	2
16	259	32,768	15	2
17	383	65,536	25	2
18	649	131,072	38	2
19	1106	262,144	64	2
20	2724	524,288	101	2
21	4961	1,048,576	174	2
22	8881	2,097,152	266	2
23	16,851	4,194,304	567	2
24	40,185	8,388,608	1052	2
25	78,165	16,777,216	1787	2
26	138,984	33,554,432	3970	2
27	277,512	67,108,864	5911	2
28	750,000	134,217,728	9853	2

Table A3. Numerical results depicted in Figure 7.

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

Table A4. Numerical results depicted in Figure 8.

Plaintext (bit)	Bit (KB)	Base (KB)	CRT (KB)	Compression (KB)
8	14.5	3.625	10.875	0.171875
9	20.75	10.375	15.5625	0.1796875
10	35.78125	14.3125	21.46875	0.1875
11	47.890625	19.15625	28.734375	0.1953125
12	75.0	12.5	37.5	0.203125
13	95.8125	31.9375	47.90625	0.2109375
14	140.21875	40.0625	60.09375	0.21875

Table A4. Cont.

Plaintext (bit)	Bit (KB)	Base (KB)	CRT (KB)	Compression (KB)
15	173.140625	74.203125	49.46875	0.2265625
16	241.0	60.25	60.25	0.234375
17	290.0	72.5	72.5	0.2421875
18	388.40625	129.46875	86.3125	0.25
19	458.015625	152.671875	101.78125	0.2578125
20	595.0	178.5	119.0	0.265625
21	690.3125	207.09375	138.0625	0.2734375
22	874.84375	238.59375	159.0625	0.28125
23	1001.515625	273.140625	182.09375	0.2890625
24	1243.5	414.5	207.25	0.296875
25	1407.75	469.25	234.625	0.3046875
26	1718.03125	528.625	264.3125	0.3125
27	1926.640625	592.8125	296.40625	0.3203125
28	2317.0	662.0	331.0	0.328125

References

1. Gentry, C.; Sahai, A.; Waters, B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Proceedings of the Advances in Cryptology—CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2013*; Proceedings, Part I; Springer: Berlin/Heidelberg, Germany, 2013; pp. 75–92.
2. Gentry, C.; Halevi, S. Compressible FHE with applications to PIR. In *Theory of Cryptography*; Springer: Cham, Switzerland, 2019; pp. 438–464.
3. Hedglin, N.; Phillips, K.; Reilley, A. Building a Fully Homomorphic Encryption Scheme in Python. 2019. Available online: <https://courses.csail.mit.edu/6.857/2019/project/15-Hedglin-Phillips-Reilley.pdf> (accessed on 21 July 2023).
4. Rivest, R.L.; Adleman, L.; Dertouzos, M.L. On data banks and privacy homomorphisms. *Found. Secur. Comput.* **1978**, *4*, 169–180.
5. Gentry, C. *A Fully Homomorphic Encryption Scheme*; Stanford University: Stanford, CA, USA, 2009.
6. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv. (Csur)* **2018**, *51*, 1–35. [\[CrossRef\]](#)
7. Alaya, B.; Laouamer, L.; Msilini, N. Homomorphic encryption systems statement: Trends and challenges. *Comput. Sci. Rev.* **2020**, *36*, 100235. [\[CrossRef\]](#)
8. Wood, A.; Najarian, K.; Kahrobaei, D. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–35. [\[CrossRef\]](#)
9. Chen, P.C.; Kuo, T.H.; Wu, J.L. A study of the applicability of ideal lattice-based fully homomorphic encryption scheme to ethereum blockchain. *IEEE Syst. J.* **2021**, *15*, 1528–1539. [\[CrossRef\]](#)
10. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast fully homomorphic encryption over the torus. *J. Cryptol.* **2020**, *33*, 34–91. [\[CrossRef\]](#)
11. Kuo, T.H.; Wu, J.L. A High Throughput BFV-Encryption-Based Secure Comparison Protocol. *Mathematics* **2023**, *11*, 1227. [\[CrossRef\]](#)
12. Moore, C.; O'Neill, M.; O'Sullivan, E.; Doröz, Y.; Sunar, B. Practical homomorphic encryption: A survey. In *Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Melbourne, VIC, Australia, 1–5 June 2014; pp. 2792–2795.
13. Lei, X.; Guo, R.; Zhang, F.; Wang, L.; Xu, R.; Qu, G. Accelerating homomorphic full adder based on fhw using multicore cpu and gpus. In *Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Zhangjiajie, China, 10–12 August 2019; pp. 2508–2513.
14. Özerk, Ö.; Elgezen, C.; Mert, A.C.; Öztürk, E.; Savaş, E. Efficient number theoretic transform implementation on GPU for homomorphic encryption. *J. Supercomput.* **2022**, *78*, 2840–2872. [\[CrossRef\]](#)
15. Marcolla, C.; Sucasas, V.; Manzano, M.; Bassoli, R.; Fitzek, F.H.; Aaraj, N. Survey on fully homomorphic encryption, theory, and applications. *Proc. IEEE* **2022**, *110*, 1572–1609. [\[CrossRef\]](#)
16. Brakerski, Z.; Vaikuntanathan, V. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, Palm Springs, CA, USA, 22–25 October 2011; pp. 97–106. [\[CrossRef\]](#)
17. Brakerski, Z.; Vaikuntanathan, V. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Advances in Cryptology—CRYPTO 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 505–524.

18. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory (TOCT)* **2014**, *6*, 1–36. [[CrossRef](#)]
19. Fan, J.; Vercauteren, F. Somewhat practical fully homomorphic encryption. *Cryptol. ePrint Arch.* **2012**, *2012*, 144.
20. Ducas, L.; Micciancio, D. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology—EUROCRYPT 2015*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 617–640.
21. Gentry, C. Computing arbitrary functions of encrypted data. *Commun. ACM* **2010**, *53*, 97–105. [[CrossRef](#)]
22. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *Proceedings of the Advances in Cryptology—ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017*; Proceedings, Part I 23; Springer: Berlin/Heidelberg, Germany, 2017; pp. 409–437.
23. Naehrig, M.; Lauter, K.; Vaikuntanathan, V. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, Chicago, IL, USA, 21 October 2011*; pp. 113–124.
24. Mahdavi, R.A.; Diao, A.; Kerschbaum, F. HE is all you need: Compressing FHE Ciphertexts using Additive HE. *arXiv* **2023**, arXiv:2303.09043.
25. Peikert, C.; Vaikuntanathan, V.; Waters, B. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology—CRYPTO 2008*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 554–571.
26. Brakerski, Z.; Döttling, N.; Garg, S.; Malavolta, G. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *Theory of Cryptography*; Springer: Cham, Switzerland, 2019; pp. 407–437.
27. Mukherjee, P.; Wichs, D. Two round multiparty computation via multi-key FHE. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, 8–12 May 2016*; Proceedings, Part II 35; Springer: Berlin/Heidelberg, Germany, 2016; pp. 735–763.
28. Micciancio, D.; Peikert, C. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology—EUROCRYPT 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 700–718.
29. Halevi, S. Homomorphic encryption. In *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 219–276.
30. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005*; pp. 84–93. [[CrossRef](#)]
31. Yuan, M.; Wang, D.; Zhang, F.; Wang, S.; Ji, S.; Ren, Y. An examination of multi-key fully homomorphic encryption and its applications. *Mathematics* **2022**, *10*, 4678. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.