Leveled Homomorphic Encryption over Composite Groups

Mahdi Mahdavi[1], Ehsan Meamari[2], Emad Heydari Beni[3], and Maryam Sheikhi[4]

[1] Universitat Oberta de Catalunya (UOC)
m_mahdavi@uoc.edu
[2] University of Delaware
ehsan@udel.edu
[3] COSIC, KU Leuven & Nokia Bell Labs
emad.heydaribeni@esat.kuleuven.be
[4] University of Hamburg
maryam.sheikhi.garjan@uni-hamburg.de

Abstract. Homomorphic encryption is a powerful tool that enables computation on encrypted data without requiring decryption. While many Fully Homomorphic Encryption schemes, supporting arbitrary computations on encrypted data, have been developed using lattice-based and AGCD-based approaches, progress in composite groups has been limited to Partial Homomorphic Encryption schemes, which only support either addition or multiplication. This paper introduces the first ℓ-leveled homomorphic encryption schemes over composite groups, based on factoring problem, that achieve both multiplicative and additive homomorphic properties. Our novel design eliminates the need for relinearization operation, which is common in LWE-based HE schemes, and removes the requirement for the circular security assumption. For applications where the traffic must be indistinguishable as encrypted, a scrambled scheme is designed using a labeling technique. While the initial schemes offer an expanded message space, the introduction of the double-sized Message technique enables the encryption of messages up to twice the size without increasing the ciphertext size. Implementation results show that our schemes significantly outperform existing solutions, particularly in multiplication operations, achieving speeds up to 1000 times faster than well-known schemes such as BFV, BGV, CKKS, and TFHE.

Keywords: Homomrphic Encryption · Leveled Homomrphic Encryption · Composit Group · Factoring · Bézout's coefficients.

1 Introduction

Homomorphic Encryption (HE) is a robust cryptographic technique that enables computations on encrypted data. Consequently, sensitive information can be processed and analyzed without being exposed, ensuring a high level of confidentiality. This capability makes HE particularly valuable for numerous applications in privacy and security. In machine learning [CJP20] and image processing

[MNR24], HE allows for secure computation on encrypted data, enabling companies to collaborate on sensitive datasets without revealing the underlying information. In cloud computing [LATV12a], HE can potentially facilitate secure outsourcing of computations to untrusted servers, maintaining the confidentiality of the data being processed. Despite its valuable applications, designing a HE scheme capable of evaluating complicated functions on encrypted data is not straightforward. The concept of homomorphic encryption began with the first class, Partially Homomorphic Encryption (PHE), which supports either addition or multiplication operations on encrypted data, but not both. GM [GM82] and RSA [RSA78] are two PHE schemes that support addition or multiplication. The next class of HE, Somewhat Homomorphic Encryption (SHE), extends the capabilities of PHE by allowing both addition and multiplication operations on encrypted data, but with certain limitations. [BGN05] proposed a SHE scheme which supports one multiplication but unlimited addition operations.

Leveled Homomorphic Encryption (LHE) extends the capability to handle computations up to a certain complexity or depth, defined by the number of multiplicative as level ℓ. [Bra12,FV12] proposed an LHE scheme which supports maximum of ℓ multiplication, but unlimited addition operations. Gentry et al. [Gen09] made significant breakthroughs in the development of the first Fully Homomorphic Encryption (FHE) which provides an unlimited number of multiplications and additions. This breakthrough demonstrated that it is possible to create a cryptographic system capable of arbitrary computation on encrypted data with confidentiality preserving.

1.1 Mathematical Foundations in HE

There are various mathematical techniques used to design Homomorphic Encryption schemes, each offering distinct properties such as security, efficiency, and so on. The different approaches can be classified as follows:

- **Pairing and Multi-linear Maps:** This technique, first introduced in [BGN05], utilizes pairings and supports unlimited additions along with a single multiplication. Although [GGH13] later developed ℓ-Multilinear maps, which allow for ℓ multiplications, this approach did not gain wide adoption due to the non-standard nature of multilinear assumptions in cryptography.

- **Ideal-Lattice:** Gentry's HE scheme [Gen09], based on Ideal-Lattices, introduced bootstrapping techniques, enabling the first FHE. Although efforts were made by schemes such as [SS10] to enhance the efficiency of this category, the proposed methods remained extremely slow, preventing this approach from gaining much traction.

- **LWE-based:** This category, typically derives from the Regev encryption system [Reg09], utilizes Learning with Errors (LWE) and its variant, the Ring Learning with Errors (RLWE) problems [BGV12,Bra12,FV12,CGGI17,GIKV23,MHWW24].

– **NTRU-based:** NTRU is a lattice-based problem utilized in the design of HE schemes, such as LTV [LATV12b] and YASHE [BLLN13]. This area has demonstrated notable progress, with several recent publications [JMPP24] and [Klu22] contributing to its development.

– **AGCD:** Another direction for proposed HE schemes utilizes the Approximate Greatest Common Divisor (AGCD) problem, which operates over integers [vDGHV10,Per20,BBL17,Per21].

– **Composite Group:** This direction, which is based on the factoring problem, is initially explored by the RSA [RSA78], and then GM [GM82], and Paillier [Pai99] schemes.

Among the listed categories, some are well-researched, while others are not. The "LWE-based" category is regarded as the most prominent and thoroughly investigated class, while the "AGCD" category is also well-studied. In contrast, categories such as "Pairing and Multi-linear Maps" and "Composite Group" exhibit limited progress. Specifically, the schemes within the "Composite Group" category have made minimal advancements, resulting in only partial homomorphic encryption schemes. For example, RSA [RSA78] supports only multiplication, and GM [GM82] supports only addition. The most notable scheme in this category is Paillier [Pai99], which supports addition and scalar multiplication. To the best of our knowledge, despite numerous attempts, there is currently no known scheme that provides at least somewhat homomorphic encryption over composite groups. This paper aims to overcome prior obstacles and advance this area by introducing l-Leveled Homomorphic Encryption in composite groups.

One advantage of the Composite Group is its potential to support a larger message space compared to other approaches.[5] As a result, the ratio of ciphertext space to a single message is smaller than in other schemes, leading to lower communication overhead.[6].

1.2 Design Techniques in HE

Other than the mathematical foundation of an HE scheme, there are different design techniques that matter while designing an HE scheme.

Relinearization, which decreases the size of the resulting ciphertext after each multiplication back into a fresh ciphertext, is required in schemes such as BGV [BGV12] and BFV [Bra12,FV12]. However, the relinearization key comprises an encrypted version of the secret key. Publishing this key as a public parameter

[5] Although LWE-based schemes like TFHE [CGGI17] can support homomorphism over large messages, they require additional computations or modifications.

[6] While other schemes utilize Batching and Single Instruction Multiple Data (SIMD) techniques to pack more data items into a single ciphertext, these techniques are ineffective for applications that require encrypting a single data item. Our schemes offer an alternative for such use cases.

relies on circular security which is potentially a security concern.

The decomposition technique, utilized in LWE-based [GSW13,ASP14,CGGI17] and AGCD-based schemes [BBL17,Per20], transforms the exponential growth of noise during multiplication into an additive growth pattern. Additionally, noise reduction can be achieved through another technique that involves switching the modulus of the ciphertext space, as applied in [BV14,BGV12,Bra12,FV12,CNT12].

Algebraic operations, such as addition and multiplication in ciphertext space, could be computed simply in evaluation. However, maintenance operations such as bit decomposition, bit-wise operations, real division, and rounding are complicated in evaluation computation. Most of practical FHE schemes, such as BGV [BGV12] and BFV [Bra12,FV12] schemes, require some of the maintenance operations for evaluation of an inquired function.

While relinearization, decomposition, modulus switching, and maintenance operations are powerful techniques essential for most HE schemes, they significantly increase the complexity and inefficiency of implementing the FHE systems. In contrast, our designs do not require these techniques, resulting in a less complex design and more efficiency compared to other HE schemes.

Beyond designing new FHE schemes, several areas of improvement aim to enhance their practicality, performance, and adaptability. These improvements include performance optimization, application-specific optimizations, and parallelism and distributed computing. Existing FHE schemes typically generate ciphertexts that appear random-like at first glance. However, an eavesdropper analyzing multiple ciphertexts may detect patterns in the traffic. While this does not compromise the confidentiality of the data, it enables the identification of traffic flows containing encrypted messages. In applications such as steganography, ensuring that traffic patterns remain indistinguishable from randomness is vital to avoid detection.

1.3 Our Contributions

The contributions are detailed as follows:

- This paper introduces the Straw-Man (SM) LHE scheme using composite groups. Then, the Labeled Scrambled Ciphertext (LSC) scheme is introduced to make the SM's ciphertext more scrambled. Eventually, a modified Rothblum's transformation technique is introduced to convert the symmetric schemes into public key versions.
- To enhance the performance of the schemes, a double-sized Message technique is introduced, allowing the encryption of a message with up to 2τ-bit without increasing the value of η.
- Implementation results demonstrate that our designed schemes can perform evaluation algorithms significantly faster, particularly for multiplication—the most computationally expensive operation in LHE schemes—achieving

speeds up to 1000 times faster than well-known schemes like BFV, BGV, CKKS, and TFHE.

1.4 Paper Organization

The paper proceeds in Section 2 with the preliminaries. Section 3 introduces two leveled homomorphic schemes: Straw-Man and Labeled Scrambled Ciphertext, followed by their public key versions. A performance enhancement technique, called doubling, is proposed in Section 4 to improve the efficiency of the schemes. Section 5 provides the implementation results of the schemes and compares them with the most well-known homomorphic encryption schemes. Finally, the conclusion in Section 6 summarizes the results of the paper.

2 Preliminaries

This section introduces notation, presents supporting mathematical lemmas, and provides an overview of homomorphic encryption.

2.1 Notation

Operations within a modulus x are denoted by $[.]_x$. The signs of $\lfloor x \rfloor$, $\lceil x \rceil$, and $\lfloor x \rceil$ represent the floor, ceiling, and rounding of x to the nearest integer, respectively.

$y \leftarrow \mathcal{A}(x)$ indicates that y is the output of algorithm (or attacker) $\mathcal{A}$ when run with x as the input.

The security parameter is shown by λ; ρ is the bit length of the noise where $\rho > \lambda$ to protect the scheme against exhaustive search. Messages are selected from $\mathbb{Z}_t$; while τ is the bit length of the message space (i.e., $\tau = \lceil \log_2 t \rceil$). The ciphertext is in modulo N, which is 2η-bit. Specifically, $N = p \cdot q$, where p and q are prime numbers with η-bit of size[7].

2.2 Supporting Mathematical Lemmas

Lemma 1. *Let $N = p \cdot q$, where p and q are prime numbers. For an integer $r \in \mathbb{Z}_N$, the equations $[qr]_N = q[r]_p$ and $[pr]_N = p[r]_q$ hold.*

Proof. We only prove $[pr]_N = p[r]_q$, and the proof of $[qr]_N = q[r]_p$ follows similarly. The Euclidean division concludes:

$$pr = \alpha N + r_N \tag{1}$$

where $r_N < N$, and

$$r = \beta q + r_q \tag{2}$$

[7] The complexity of the factoring problem is at least 2^λ using the best-known practical factoring algorithm known as the number field sieve (NFS). Quantum algorithms for factoring, e.g., Shor algorithm, are not taken into account.

where $r_q < q$. Equation 1 represents the division of pr by N, while Equation 2 corresponds to the division of r by q. Equation 2 concludes $p \cdot r = \beta pq + pr_q$. Since $r_q < q$, it follows that $pr_q < pq$. The uniqueness property of Euclidean division concludes that $pr_q = r_N$, implying that $p[r]_q = [pr]_N$. □

Remark 1. The lemma 1 shows that if the value of pr overflows from N, the result remains in the form of a multiple of p, specifically $p[r]_q$. The proposed lemma suggests that computations modulo N are equivalent to computations modulo p or q.

Lemma 2. *Let p and q be two odd primes and $N = p \cdot q$. For any number $c \in \mathbb{Z}_N$,a unique pair $(a, b) \in \mathbb{Z}_p \times \mathbb{Z}_q$ exists such that $aq + bp = c$. In other words, the equation $qx + py = c$ has a unique answer for x and y.*

Proof. Suppose there are two pairs (a_1, b_1) and (a_2, b_2) such that $a_1q + b_2p = a_2q + b_2p = c$, which can be written as $(a_1 - a_2)q = (b_2 - b_1)p$. Since p and q are primes, it follows that $a_1 - a_2 = \alpha p$ and $b_2 - b_1 = \alpha q$. Therefore, it can be concluded that $a_1 \equiv a_2 \mod p$, and $b_1 \equiv b_2 \mod q$. This implies that if the equation $a \cdot q + b \cdot p = c$ has a solution, the solution is unique. Next, it will be shown that the equation indeed has a solution.
Given the uniqueness of values for a and b, and since there are p possible values for a and q possible values for b, there are pq possible pairs (a, b) that cover all representations of $\mathbb{Z}_N$. This implies that such an equation has a unique solution. □

Lemma 3. *Let p and q be two odd primes, and let $N = p \cdot q$. It follows that $[p^q]_N = p$ and $[q^p]_N = q$.*

Proof. According to Lemma 1, for $r = [q^{p-1}]_N$ it can be concluded that $[q \cdot q^{p-1}]_N = q[q^{p-1}]_p$. This can be rewritten as $[q^p]_N = q[1]_p$. Similarly, the other equation holds. □

Remark 2. Let p and q be two primes and $N = pq$. Consider $\hat{p} = [p^{q-1}]_N = \alpha p = \alpha' q + 1$ and $\hat{q} = [q^{p-1}]_N = \beta q = \beta' p + 1$. For any i, the relations $[\hat{p}^i]_N = \hat{p}$ and $[\hat{q}^i]_N = \hat{q}$ are hold, means that the order of $\hat{p}$ and $\hat{q}$ are equal to one.

2.3 Overview of Homomorphic Encryption

Instead of computing the function f on the message m, Homomorphic Encryption allows for the computation of a function f' on the encrypted data `ct` such that decrypting the output of f' yields the same result as applying f to m. Therefore, HE can ensure data confidentiality during computation. HE schemes typically comprise four fundamental algorithms: Key Generation, Encryption, Evaluation, and Decryption, denoted as `KeyGen`, `Enc`, `Eval`, and `Dec`, respectively. In this section, we present the symmetric-key HE algorithms, which are transformable to the public-key version using Rothblum's transformation [Rot11]. The functionality of each algorithm is as follows:

- $\texttt{KeyGen}(1^\lambda, t, \ell)$: This algorithm takes security parameter λ, space of messages t, and maximum depth ℓ as input, and outputs the secret key $\texttt{sk}$, public parameters $\texttt{pp}$, and the evaluation key $\texttt{evk}$.

- $\texttt{Enc}(m, \texttt{sk}, \texttt{pp})$: This algorithm utilizes the secret key $\texttt{sk}$ and public parameters $\texttt{pp}$ to generate a ciphertext $\texttt{ct}$ corresponding to the input message m.

- $\texttt{Eval}(\overrightarrow{\texttt{ct}}, \texttt{evk}, f)$: This algorithm takes a set of ciphertexts $\overrightarrow{\texttt{ct}}$, evaluation key $\texttt{evk}$, and an evaluation function f as input, and outputs $\texttt{ct}_f$, which is the ecnryption of $f(m_1, m_2, ..., m_k)$. The evaluation algorithm is valid for any function f that the number of multiplications is less than ℓ, and consists of two sub-algorithms as follows:
  - $\texttt{Add}$: represents addition evaluation function.
  - $\texttt{Mult}$: represents multiplication evaluation function.
- $\texttt{Dec}(\texttt{ct}, \texttt{sk})$: This algorithm takes a ciphertext $\texttt{ct}$ and secret key $\texttt{sk}$ as inputs and generates the plaintext message.

Definition 1 (Correct Homomorphic Encryption). *The scheme* $\mathcal{E} = \{\texttt{KeyGen}, \texttt{Enc}, \texttt{Eval}, \texttt{Dec}\}$ *is correct if, for any key pair* $(\texttt{sk}, \texttt{evk})$ *generated by* $\texttt{KeyGen}$*, and any* l *ciphertexts* $\vec{c} = \langle c_1, c_2, ..., c_l \rangle$ *encrypted with* l *messages* $m_1, ..., m_l$*, for a given* l*-input function* f*, the following relation holds:*

$$\texttt{Dec}(\texttt{sk}, \texttt{Eval}(\vec{c}, \texttt{evk}, f)) = f(m_1, ..., m_l)$$

3 Proposed HE Schemes

This section introduces three LHE schemes: the Straw-Man (SM), the Labeled Scrambled Ciphertext (LSC), and the public key version of LSC. The Straw-Man scheme designs an LHE using composite groups. However, the ciphertext is placed near to a multiple of the secret key. This pattern makes the traffic of the Straw-Man scheme identifiable as encrypted, such as other schemes like [vDGHV10], which is undesirable in applications like steganography. To address this issue, the Labeled Scrambled Ciphertext scheme makes the ciphertext more indistinguishable. Eventually, the public key version of the LSC scheme is designed using a revised Ron Rothblum's technique.

3.1 Straw-Man LHE

The Straw-Man scheme works as follows:

- $\texttt{SM.KeyGen}(1^\lambda, \ell, t)$: Based on the security parameters λ, select two η-bit prime numbers p and q, set the secret key as $\texttt{sk} := p$, and publish $\texttt{evk} := N = p \cdot q$ as evaluation key. For $\tau = \lceil \log_2 t \rceil$, set noise length $\rho > \lambda$[8] such that $p > 2^{\ell.(\rho+\tau)}$, where ℓ is the level (maximum number of multiplication).

[8] To protect against exhaustive search.

This algorithm also defines the public parameters as $\mathtt{pp} := \{\lambda, \rho, \tau, \ell, t\}$ and publishes them along with $\mathtt{evk}$, both of which serve as inputs for the subsequent algorithms.

- $\mathtt{SM.Enc}(m, \mathtt{sk})$: For $m \in \mathbb{Z}_t$ as a message, select a ρ-bit random integer e as an error (or noise) and compute the plaintext in form of $\mu = t \cdot e + m$[9]. Next, select a random number $r \in \mathbb{Z}_q$ and return

$$\mathtt{ct} = p \cdot r + \mu$$

as the ciphertext[10].

- $\mathtt{SM.Eval}(\mathtt{ct}_1, \mathtt{ct}_2, \cdots, \mathtt{ct}_k, f)$: Consist of the two addition and multiplication sub-algorithms:
  - $\mathtt{SM.Add}(\mathtt{ct}_1, \mathtt{ct}_2, N)$: Addition algorithm computes $\mathtt{ct}_+ = [\mathtt{ct}_1 + \mathtt{ct}_2]_N$.
  - $\mathtt{ST.Mult}(\mathtt{ct}_1, \mathtt{ct}_2, N)$: Multiplication algorithm computes $\mathtt{ct}_\times = [\mathtt{ct}_1.\mathtt{ct}_2]_N$.
- $\mathtt{SM.Dec}(\mathtt{ct}, \mathtt{sk})$: Return $m = [[\mathtt{ct}]_p]_t$.

Remark 3. Since the Eval algorithm does not use any maintenance operation, so that it can compute $\mathtt{ct}_f = [f(\mathtt{ct}_1, \mathtt{ct}_2, \cdots, \mathtt{ct}_k)]_N$ to compute the encryption of the $[f(m_1, m_2, \cdots, m_k)]_t$.

Remark 4. Scalar addition (adding a constant $c \in \mathbb{Z}_t$ with ciphertext $\mathtt{ct}$) and scalar multiplication (multiplying a constant $c \in \mathbb{Z}_t$ with ciphertext $\mathtt{ct}$) can be computed as $[\mathtt{ct} + c]_N$ and $[c.\mathtt{ct}]_N$, respectively.

Remark 5. The $\mathtt{Enc}$ algorithm encodes the message m into the LSB part of μ. Although the LSB mode of the scheme is discussed, because of simplicity and efficiency, the MSB mode for the schemes could be designed similarly.

Remark 6. Our proposed schemes support ℓ multiplications, with the value of ℓ being increased by utilizing larger public parameters. Specifically, the relationship $\ell = \lfloor \frac{\eta - 1}{\rho + \tau} \rfloor$ governs the interaction between these parameters. This relationship guarantees the requirement $p > 2^{\ell \cdot (\rho + \tau)}$. The number of additions is significantly greater than ℓ, and therefore, it is not discussed further in this document.

Remark 7. In the proposed schemes, the value of noise e is chosen from a uniform distribution, which is less complex than other schemes based on Lattice, such as vDGHV, BGV, BFV, etc., where the noise is chosen from a Gaussian distribution.

[9] This means that, μ is a $(\rho + \tau)$-bit random coding of message m, such that $m = [\mu]_t$.

[10] Since $r \leq q - 1$ and $\mu < p$, it follows that $pr + \mu < p(q - 1) + p$, which implies that $\mathtt{ct} < N$. Therefore, the ciphertext is given by $\mathtt{ct} = [pr + \mu]_N$, ensuring that the fresh ciphertext never exceeds the value of N.

Correctness The proof of correctness is as follows:

$$[[\mathtt{ct}]_p]_t = [[pr + \mu]_p]_t = [[p.r + \mu]_p]_t = [[\mu]_p]_t = [\mu]_t = m \tag{3}$$

The correctness condition in equation 3 holds as long as $\mu < p$. For fresh ciphertexts, μ remains below p since $\mu < 2^{\rho+\tau} < p$. While μ increases after evaluation, particularly during multiplication, the growth is manageable, and addition does not significantly increase μ. After ℓ multiplications, for the resulting μ_ℓ, it holds that $\mu_\ell < 2^{\ell(\rho+\tau)}$. Setting the parameter as $\ell = \lfloor \frac{\eta-1}{\rho+\tau} \rfloor$ ensures that $\mu_\ell < 2^{\ell(\rho+\tau)} \leq 2^{\eta-1} < p$. Therefore, as long as the evaluation function depth does not exceed ℓ, the correctness is preserved.

Homomorphism The homomorphic property of addition can be validated as follows.

$$\mathtt{ct}_+ = [pr_1 + \mu_1 + pr_2 + \mu_2]_N = [p(r_1 + r_2) + te_1 + m_1 + te_2 + m_2]_N$$

According to lemma 1, term $[p(r_1 + r_2)]_N$ can be written to $p[(r_1 + r_2)]_q$. If $(r_1 + r_2)$ exceeds from q, equivalently $p(r_1 + r_2)$ exceeds from N, the result also is in the same format, i.e., multiplication of p. So, the ciphertext can be simplified as

$$\mathtt{ct}_+ = [pr_+ + te_+ + m_+]_N$$

where $r_+ = r_1 + r_2$, $e_+ = e_1 + e_2 + cr$, and cr represents the carry value of $m_1 + m_2$, and $m_+ = [m_1 + m_2]_t$.

With similar explanation about the homomorphic property of addition, the homomorphic property of multiplication can be validated as follows.

$$\begin{aligned}
\mathtt{ct}_\times &= [(pr_1 + \mu_1).(pr_2 + \mu_2)]_N \\
&= [p.(\mu_1 r_2 + r_1\mu_2 + pr_1r_2) + \mu_1\mu_2]_N \\
&= [p.(\mu_1 r_2 + r_1\mu_2 + pr_1r_2) + m_1m_2 + t(m_1e_2 + te_1e_2 + e_1m_2)]_N \\
&= [pr_\times + m_\times + te_\times]_N
\end{aligned}$$

Note that $te_\times = t^2e_1e_2 + t(m_1e_2 + e_1m_2 + cr)$, where cr represents the carry of $m_1 + m_2$.

A similar discussion can represent the correctness of scalar addition and multiplication homomorphism.

3.2 Labeled Scrambled Ciphertext LHE

In the SM scheme, such as the vDGHV scheme, given a closer exploration of the ciphertext $\mathtt{ct}$ in the Straw-Man version, it is concluded that an eavesdropper can infer $\mathtt{ct}$ is nearly (in comparison to the size of N) equal to a multiple of the secret key p. If an eavesdropper intercepts network traffic, they may notice a consistent pattern in the data[11], leading them to conclude that the traffic

[11] This paper does not discuss detection techniques, but it can be observed that the ciphertexts generated by the SM scheme exhibit a consistent pattern.

contains encrypted messages. However, in applications such as steganography, the sender seeks to conceal this pattern. To protect the encrypted data from such threats, it is preferable to design a cryptographic scheme that produces scrambled ciphertexts. This paper introduces a labeling technique to enhance the scrambling of the SM scheme, which can also be applied to the DGHV scheme [vDGHV10].

This section describes the Labeled Scrambled Ciphertext (LSC) scheme in which the output of the encryption algorithm is more scrambled away from the multiplications of the secret key. In other words, the ciphertext generated by the SM scheme is predictable and follows a unique pattern, but the LSC scheme scrambled the distribution pattern in ciphertext space (i.e., $[0, N-1]$).

The LSC scheme requires that the labels of ciphertexts be attached and updated after each (multiplication) operation. Additionally, an equalizer algorithm is designed to *equalize* the labels of ciphertexts, which is necessary for the addition evaluation. The equalizer is a concept that can be used in any scheme which works with labels.

Definition 2 (Equalizer). *This algorithm takes as input two ciphertexts,* $\mathtt{ct}_1$ *and* $\mathtt{ct}_2$ *with labels* l_1 *and* l_2*, respectively. The algorithm updates the input ciphertexts to* $\mathtt{ct}_i^* = \mathtt{Eq}^{(l_m - l_i)} \cdot \mathtt{ct}_i$*, for* $i \in \{0,1\}$ *and* $l_m = max(l_1, l_2)$*, equalizing its label to* l_m *without affecting the included message* [12]*. Here,* $\mathtt{Eq}$ *is the equalizer parameter (i.e., encryption of* 1*,* $\mathtt{Enc}(1)$*)*[13].

Remark 8. In general, for a function $f(\mathbf{x}) = \sum_{i=1}^{v} a_i \prod_{j=1}^{w} x_j^{k_j}$ which is required to be evaluated, the algorithm computes the level for each term i as $l_i = \sum_j^w l_j \cdot k_j$, where l_j is the label of the ciphertext $\mathtt{ct}_j$. Then, the algorithm rewrites the function as $f'(\mathtt{Eq}; \mathbf{x}) = \sum_{i=1}^{v} a_i \prod_{j=1}^{w} \mathtt{Eq}^{(l_m - l_i)} x_j^{k_j}$, where l_m is the maximum of levels. Eventually, all terms of the rewritten function have the equalized label l_m.

Remark 9. Using an equalizer does not introduce new security concerns, as it is essentially the encryption of 1, and its security is tied to the scheme itself, rather than to other assumptions, such as circular security.

The LSC scheme's algorithms are as follows:

- $\mathtt{LSC.KeyGen}(1^\lambda, \ell, t)$: Based on the security parameters λ, select two η-bit prime numbers p and q. Then, choose a random integer $D \in \mathbb{Z}_p^*$, consider secret key as $\mathtt{sk} := (p, D)$, and publish $\mathtt{evk} := N = p \cdot q$ as evaluation key and equalizer $\mathtt{Eq} = \mathtt{LSC.Enc}(1, \mathtt{sk})$. For $\tau = \lceil \log_2 t \rceil$, set noise length $\rho > \lambda$ such that $p > 2^{\ell \cdot (\rho + \tau)}$. The algorithm then defines the public parameters as $\mathtt{pp} := \{\lambda, \rho, \tau, \ell, t\}$ and publishes them along with $\mathtt{evk}$, which both serve as inputs for the subsequent algorithms.

[12] Note that, the equalizer does not changes the ciphertext with maximum label. Without losing of generality, if $l_2 \geq l_1$, this algorithm only changes the ciphertext $\mathtt{ct}_1$.

[13] The values of $\mathtt{Eq}^{(i)}$ for $1 \leq i \leq \ell$ can be pre-computed and stored to reduce computation overhead.

– $\texttt{LSC.Enc}(m, \texttt{sk})$: For a message $m \in \mathbb{Z}_t$, select a ρ-bit random integer e as an error (or noise) and set the plaintext to $\mu = t.e + m$. Furthermore, select a random number $r \in \mathbb{Z}_q$ and return

$$\texttt{ct} = [pr + D\mu]_N$$

as the ciphertext. In addition, the label l, which shows the power of D and is 1 for fresh ciphertexts, should be attached to the ciphertext.

– $\texttt{LSC.Eval}(\texttt{ct}_1, l_1, \texttt{ct}_2, l_2, \cdots, \texttt{ct}_k, l_k, N, f)$: Consists of two sub-algorithms as follow:

- $\texttt{LSC.Add}(\texttt{ct}_1, l_1, \texttt{ct}_2, l_2)$: Update the two ciphertexts using the Equalizer algorithm to equalize the label of them. Then return $\texttt{ct}_+ = [\texttt{ct}_1 + \texttt{ct}_2]_N$ with the updated label.
- $\texttt{LSC.Mult}(\texttt{ct}_1, l_1, \texttt{ct}_2, l_2)$: Compute $\texttt{ct}_\times = [\texttt{ct}_1.\texttt{ct}_2]_N$, and update label using $l_\times = l_1 + l_2$.

– $\texttt{LSC.Dec}(\texttt{ct}, l, \texttt{sk})$: Compute $\mu = [D^{-l}.\texttt{ct}]_p$, and then return $m = [\mu]_t$ [14].

Remark 10. Since the Eval algorithm does not use any maintenance operation, so that it can call equalizer in general and receive the rewritten function f' and compute $\texttt{ct}_f = [f'(\texttt{ct}_1, \texttt{ct}_2, \cdots, \texttt{ct}_k)]_N$ to calculate the encryption of $[f(m_1, m_2, \cdots, m_k)]_t$. Note that the function f is computed modulo t on the plaintext, but f' is computed modulo N on the ciphertext.

Remark 11. For scalar multiplication and scalar addition, the scheme treats with the constant $c \in \mathbb{Z}_t$ like a ciphertext with label 0. Therefore, scalar multiplication for ciphertext $\texttt{ct}$ with label l can be computed straightforward as $[c.\texttt{ct}]_N$ without changing the label value. For scalar addition, the scheme calls equalizer, computes $[c \cdot \texttt{Eq}^l]_N$ and then returns $[\texttt{ct} + c \cdot \texttt{Eq}^l]_N$.

Correctness The correctness of decryption in LSC could be validated, as follows:

$$\begin{aligned}[D^{-1} \cdot \texttt{ct}]_p &= [D^{-1}[pr + D\mu]_N]_p = [D^{-1}(pr + D\mu)]_p \\ &= [D^{-1}D\mu]_p = [\mu]_p = \mu \end{aligned} \tag{4}$$

Clearly, $[\mu]_t = m$, with the same explanation about the correctness of the SM scheme.

[14] Since the label l is limited by ℓ, the values of $\{D^{-1}, D^{-2}, \cdots, D^{-\ell}\}$ and $\{\texttt{Eq}^{-1}, \texttt{Eq}^{-2}, \cdots, \texttt{Eq}^{-\ell}\}$ can be pre-computed and stored.

Homomorphism The addition homomorphism property can be proved for two ciphertexts with the equalized labels l as follows:

$$\begin{aligned}\mathtt{ct}_+ &= [\mathtt{ct}_1 + \mathtt{ct}_2]_N = [pr_1 + (D)^l\mu_1 + pr_2 + (D)^l\mu_2]_N = [p(r_1 + r_2) + (D)^l(\mu_1 + \mu_2)]_N \\ &= [p(r_1 + r_2) + (D)^l(te_1 + m_1 + te_2 + m_2)]_N \\ &= [pr_+ + (D)^l(te_+ + m_+)]_N\end{aligned}$$

The multiplication homomorphism property of the scheme could be validated as follows:

$$\begin{aligned}\mathtt{ct}_\times &= [\mathtt{ct}_1.\mathtt{ct}_2]_N = [(pr_1 + (D)^{l_1}\mu_1)(pr_2 + (D)^{l_2}\mu_2)]_N \\ &= [p^2(r_1r_2) + p(D)^{l_2}\mu_2r_1 + p(D)^{l_1}\mu_1r_2 + (D)^{l_1+l_2}(\mu_1\mu_2)]_N \\ &= [pr_\times + (D)^l(te_\times + m_\times)]_N\end{aligned}$$

The correctness of the equalizer algorithm can be simply approved similar to the correctness of multiplication.

The correctness of the scalar addition and multiplication homomorphism can be discussed similarly.

Remark 12. It can be observed that the ciphertexts produced by the LSC scheme are more scrambled than those generated by the SM scheme, which does not employ the labeling technique. This technique can also be similarly applied to the DGHV scheme [vDGHV10] to enhance its scrambling.

3.3 Public Key LHE

This section introduces the public key version of the LSC scheme. Usually, the Ron Rothblum's technique [Rot11] is used to transform a symmetric HE scheme to a public-key one, defined as follows:

$$Enc_{pk}(m) = \left[\sum_{i\in[k]} b_i \cdot \mathtt{p}_i + m\right]_N$$

where $b_i \in \{0, 1\}$, and $\mathtt{p}_i = \mathtt{Enc}(0)$, which is defined as the public key pk. However, this technique does not work for the LSC scheme, so that, a modified version of Rothblum's technique is defined as follows:

$$Enc_{pk}(m) = \left[\sum_{i\in[k]} b_i \cdot \mathtt{p}_i + m \cdot \mathtt{Eq}\right]_N \tag{5}$$

Note that, in equation 5, since the message m is multiplied by $\mathtt{Eq}$, the noise in $\mathtt{Eq}$ increases. This should be considered when defining the bounds of ℓ and p.

4 Optimization

This section investigates techniques to improve the performance of the proposed schemes.

4.1 Double sized Straw-Man

Recalling the SM scheme from section 3.1, the decryption algorithm can compute the value of the random integer r from the ciphertext $\mathtt{ct} = [p.r + \mu]_N$. However, the random integer r could be substituted with a plaintext. This capability is used to propose a technique to encrypt a message with up to 2τ-bit without increasing the value of η. Consider the message space in $\mathbb{Z}_t$ where $t = \dot{t} \cdot \ddot{t}$, and $\gcd(\dot{t}, \ddot{t}) = 1$. Assume $\dot{t}$ and $\ddot{t}$ are τ_1 and τ_2-bit long, respectively. For simplicity, let $\tau_1 = \tau_2 = \tau$. Given a message $m \in \mathbb{Z}_t$, we can split it into two components: $\dot{m} = [m]_{\dot{t}}$ and $\ddot{m} = [m]_{\ddot{t}}$. Knowing $\dot{m}$ and $\ddot{m}$, the Chinese Remainder Theorem (CRT) could be used to recover m as:

$$m = \dot{m} \cdot \ddot{t} \cdot [\ddot{t}^{-1}]_{\dot{t}} + \ddot{m} \cdot \dot{t} \cdot [\dot{t}^{-1}]_{\ddot{t}}$$

With doubling the space of m, two decryption keys are needed where each of them retrieves one of the messages.

By this introduction, this subsection proposes a Double-sized Straw-Man (DSM) scheme which provides the possibility of encryption of double-sized messages over composite message space, using remark 2. The Algorithms of the DSM scheme works as follows.

- $\mathtt{DSM.KeyGen}(1^\lambda, \ell, t)$: Select two η-bit prime numbers p and q and compute $N = p \cdot q$ as evaluation key $\mathtt{evk}$. The space of the message is $\mathbb{Z}_t$ (or $\mathbb{Z}_{\dot{t}} \times \mathbb{Z}_{\ddot{t}}$) where $t = \dot{t}.\ddot{t}$ and $\gcd(\dot{t}, \ddot{t}) = 1$. Then, compute $(\hat{p}, \hat{q}) = ([p^{q-1}]_N, [q^{p-1}]_N) \in \mathbb{Z}_N^2$, and set the secret key as $\mathtt{sk} := (p, q, \hat{p}, \hat{q})$. We should have $2^{\ell(\rho+\tau)} < p$ and $2^{\ell(\rho+\tau)} < q$. This algorithm also defines public parameters as $\mathtt{pp} := \{\lambda, \rho, \tau, \ell, t, \dot{t}, \ddot{t}\}$ and publish them with $\mathtt{evk}$ to be inputs to the next algorithms.

- $\mathtt{DSM.Enc}(m, \mathtt{sk})$: For $\dot{m} = [m]_{\dot{t}}$, $\ddot{m} = [m]_{\ddot{t}}$, select two ρ-bit random integers $\dot{e}, \ddot{e}$ and define two plaintexts $\dot{\mu} = \dot{t}.\dot{e} + \dot{m}$ and $\ddot{\mu} = \ddot{t}.\ddot{e} + \ddot{m}$, and then return

  $$\mathtt{ct} = [\hat{p}\dot{\mu} + \hat{q}\ddot{\mu}]_N$$

  as ciphertext[15].

- $\mathtt{DSM.Eval}(\mathtt{ct}_1, \mathtt{ct}_2, \cdots, \mathtt{ct}_k, f)$: two sub-algorithms are as follows:

  - $\mathtt{DSM.Add}(\mathtt{ct}_1, \mathtt{ct}_2, N)$: Compute $\mathtt{ct}_+ = [\mathtt{ct}_1 + \mathtt{ct}_2]_N$.
  - $\mathtt{DSM.Mult}(\mathtt{ct}_1, \mathtt{ct}_2, N)$: Compute $\mathtt{ct}_\times = [\mathtt{ct}_1.\mathtt{ct}_2]_N$.

- $\mathtt{DSM.Dec}(\mathtt{ct}, \mathtt{sk})$: Compute $\dot{m} = [[\mathtt{ct}]_q]_{\dot{t}}$ and $\ddot{m} = [[\mathtt{ct}]_p]_{\ddot{t}}$, and then return $m = \dot{m}\ddot{t}.[\ddot{t}^{-1}]_{\dot{t}} + \ddot{m}\dot{t}.[\dot{t}^{-1}]_{\ddot{t}}$.

[15] The ciphertext of the DSM scheme could be rewritten as $[p\ddot{r} + \ddot{\mu}] = [q\dot{r} + \dot{\mu}]$, which is similar to the ciphertext of the SM scheme. For more details, see the correctness.

Remark 13. Similar to the previous schemes, scalar multiplication is straightforward in this scheme too. For scalar addition, the DSM scheme returns $\mathtt{ct}+c\cdot E_1$, where E_1 is the encryption of one, $E_1 = \mathtt{DSM.Enc}(1, \mathtt{sk})$, which is like an equalizer in the LSC scheme.

Remark 14. In the DSM scheme, let assume $\gcd(\dot{t}, \ddot{t}) = t_g \neq 1$. For two double-sized messages $m_1 = (\dot{m}_1, \ddot{m}_1) = (0, 1)$ and $m_2 = (\dot{m}_2, \ddot{m}_2) = (1, 0)$, an adversary could query the encryption and receive ciphertexts $\mathtt{ct}_1$ and $\mathtt{ct}_2$. The adversary can then compute $[\mathtt{ct}_1]_{t_g}$, revealing $[gp]_{t_g}$, and $[\mathtt{ct}_2]_{t_g}$, revealing $[dq]_{t_g}$. This is why the condition $\gcd(\dot{t}, \ddot{t}) = 1$ is necessary.

Correctness The correctness of the decryption algorithm for a message $\dot{m}$ can be verified by considering remark 2 as follows.

$$\begin{aligned}[\mathtt{ct}]_q &= [[\hat{p}\dot{\mu} + \hat{q}\ddot{\mu}]_N]_q = [[(\alpha'.q + 1)\dot{\mu} + \beta.q\ddot{\mu}]_N]_q \\ &= [[\dot{\mu} + q(\alpha'.\dot{\mu} + \beta.\ddot{\mu})]_N]_q = [\dot{\mu} + q(\alpha'.\dot{\mu} + \beta.\ddot{\mu})]_q = [q\dot{r} + \dot{\mu}]_N = [\dot{\mu}]_q = \dot{\mu}\end{aligned}$$

Note that $\dot{m} = [\dot{\mu}]_{\dot{t}}$, with considering the same explanation about the correctness of the SM scheme. The correction of the decryption for a message $\ddot{m}$ is similar.

Homomorphism The addition homomorphism property can be proved as follows:

$$\begin{aligned}\mathtt{ct}_+ &= [\mathtt{ct}_1 + \mathtt{ct}_2]_N = [(\hat{p}.\dot{\mu}_1 + \hat{q}.\ddot{\mu}_1) + (\hat{p}.\dot{\mu}_2 + \hat{q}.\ddot{\mu}_2)]_N = [\hat{p}.(\dot{\mu}_1 + \dot{\mu}_2) + \hat{q}.(\ddot{\mu}_1 + \ddot{\mu}_2)]_N \\ &= [\hat{p}.(\dot{t}\dot{e}_1 + \dot{m}_1 + \dot{t}\dot{e}_2 + \dot{m}_2) + \hat{q}.(\ddot{t}\ddot{e}_1 + \ddot{m}_1 + \ddot{t}\ddot{e}_2 + \ddot{m}_2)]_N \\ &= [\hat{p}.(\dot{t}\dot{e}_+ + \dot{m}_+) + \hat{q}.(\ddot{t}\ddot{e}_+ + \ddot{m}_+)]_N\end{aligned}$$

The multiplication homomorphism property can be proved as follows:

$$\begin{aligned}\mathtt{ct}_\times &= [\mathtt{ct}_1.\mathtt{ct}_2]_N = [(\hat{p}.\dot{\mu}_1 + \hat{q}.\ddot{\mu}_1).(\hat{p}.\dot{\mu}_2 + \hat{q}.\ddot{\mu}_2)]_N \\ &= [(\hat{p}^2.(\dot{\mu}_1.\dot{\mu}_2) + \hat{p}\hat{q}(\dot{\mu}_1\ddot{\mu}_2 + \dot{\mu}_2.\ddot{\mu}_1) + \hat{q}^2.(\dot{\mu}_1.\ddot{\mu}_2))]_N \\ &= [\hat{p}^2.(\dot{\mu}_1.\dot{\mu}_2) + \hat{q}^2.(\ddot{\mu}_1.\ddot{\mu}_2))]_N \\ &= [\hat{p}.\dot{\mu}_\times + \hat{q}.\ddot{\mu}_\times]_N = [\hat{p}.(\dot{t}\dot{e}_\times + \dot{m}_\times) + \hat{q}.(\ddot{t}\ddot{e}_\times + \ddot{m}_\times)]_N\end{aligned}$$

The correctness of the scalar addition and multiplication homomorphism can be approved similarly.

4.2 Double-sized Labeled Scrambled Ciphertext

With a similar discussion about the idea of the DSM scheme, this subsection describes the Double-sized Labeled Scrambled Ciphertext (DLSC) scheme which works as follows.

- $\mathtt{DLSC.KeyGen}(1^\lambda)$: For the space of the message $\mathbb{Z}_t$ (or $\mathbb{Z}_{\dot{t}} \times \mathbb{Z}_{\ddot{t}}$), where $t = \dot{t}.\ddot{t}$ and $\gcd(\dot{t}, \ddot{t}) = 1$, select two η-bit prime numbers p and q and publish $\mathtt{evk} :=$

$N = p \cdot q$ as evaluation key. Then, select random pair $(d, g) \in \mathbb{Z}_p \times \mathbb{Z}_q$, and let $D = dq$ and $G = gp$. Eventually, set the secret key as $\mathtt{sk} := (p, q, D, G)$ and equalizer as $\mathtt{Eq} = \mathtt{DLSC.Enc}(1, \mathtt{sk})$. Two requirements must be satisfied: $2^{\ell(\rho+\tau)} < p$ and $2^{\ell(\rho+\tau)} < q$. Eventually, the algorithm define public parameters as $\mathtt{pp} := \{\lambda, \rho, \tau, \ell, t, \dot{t}, \ddot{t}\}$ and publish them with $\mathtt{evk}$ as inputs to the next algorithms.

- $\mathtt{DLSC.Enc}(m, \mathtt{sk})$: For $\dot{m} = [m]_{\dot{t}}$ and $\ddot{m} = [m]_{\ddot{t}}$, select two η-bit random integers $\dot{e}$ and $\ddot{e}$ and set the plaintexts as $\dot{\mu} = \dot{t}.\dot{e} + \dot{m}$ and $\ddot{\mu} = \ddot{t}.\ddot{e} + \ddot{m}$. Return

$$\mathtt{ct} = [G.\dot{\mu} + D.\ddot{\mu}]_N = [gp.\dot{\mu} + dq.\ddot{\mu}]_N$$

  as the ciphertext. In addition, the label value l, which is 1 for fresh ciphertexts, should be attached to the ciphertext.

- $\mathtt{DLSC.Eval}(\mathtt{ct}_1, l_1, \mathtt{ct}_2, l_2, \cdots, \mathtt{ct}_k, l_k, N, f)$: Consists of two sub-algorithms as follow:

  - $\mathtt{DLSC.Add}(\mathtt{ct}_1, l_1, \mathtt{ct}_2, l_2)$: Update the two ciphertexts using the Equalizer algorithm to equalize the label of them. Then return $\mathtt{ct}_+ = [\mathtt{ct}_1 + \mathtt{ct}_2]_N$ with the updated label.
  - $\mathtt{DLSC.Mult}(\mathtt{ct}_1, l_1, \mathtt{ct}_2, l_2)$: Compute $\mathtt{ct}_\times = [\mathtt{ct}_1.\mathtt{ct}_2]_N$, and update label using $l_\times = l_1 + l_2$.

- $\mathtt{DLSC.Dec}(\mathtt{ct}, \mathtt{sk})$: Compute $\dot{\mu} = [G^{-1} \cdot \mathtt{ct}]_q$ and $\ddot{\mu} = [D^{-1} \cdot \mathtt{ct}]_p$, then return $[\dot{\mu}]_{\dot{t}} = \dot{m}$ and $[\ddot{\mu}]_{\ddot{t}} = \ddot{m}$. The message m can be recovered by CRT as $m = m_1 \dot{t}[\ddot{t}^{-1}]_{\dot{t}} + m_2 \dot{t}[\dot{t}^{-1}]_{\ddot{t}}$.

Remark 15. The scalar multiplication can be computed straightforward as $[c.\mathtt{ct}]_N$ without changing the label value. For the scalar addition of the constant c with the ciphertext $\mathtt{ct}$ with label l, the scheme treats the constant like a ciphertext with label 0. So it computes $[c \cdot \mathtt{Eq}^l]_N$ using equalizer, and then returns $[\mathtt{ct} + c \cdot \mathtt{Eq}^l]_N$.

Remark 16. Alternatively, plaintexts can be computed through $\dot{\mu} = \frac{[\mathtt{ct}.g^{-1}p^{q-1}]_N}{p}$ and $\ddot{\mu} = \frac{[\mathtt{ct}.d^{-1}q^{p-1}]_N}{q}$ for decryption.

Remark 17. Clearly, if $D = \hat{q}$ and $G = \hat{p}$, the DLSC scheme is simplified to the DSM scheme. So the DSM scheme is a special case of the DLSC.

Remark 18. Values $\dot{m}$ and $\ddot{m}$ could be two separate messages instead of parts of a single message m. In this situation, the two separate messages are effectively batched together.

Correctness The correctness of the decryption methods can be verified similar to the correctness of the LSC scheme in section 3.2.

$$\begin{aligned}[G^{-1} \cdot \mathtt{ct}]_q &= [G^{-1}[gp\dot{\mu} + dq\ddot{\mu}]_N]_q = [G^{-1}(G\dot{\mu} + dq\ddot{\mu})]_q \\ &= [G^{-1}G\dot{\mu}]_q = [\dot{\mu}]_q = \dot{\mu}\end{aligned}$$

The correctness holds with considering the same explanation about the correctness of SM scheme. The decryption of the second part, $\ddot{\mu}$, also can be proved similarly.

The correctness of the alternative decryption algorithm can be validated as follows.

$$\begin{aligned}\frac{[\mathtt{ct}.f^{-1}p^{q-1}]_N}{p} &= \frac{[(fp.\dot{\mu} + dq.\ddot{\mu}).f^{-1}p^{q-1}]_N}{p} = \frac{[p^q.\dot{\mu} + dq.f^{-1}p^{q-1}\ddot{\mu}]_N}{p} \\ &= \frac{[p.\dot{\mu}]_N}{p} = \frac{p.[\dot{\mu}]_q}{p} = [\dot{\mu}]_q = \dot{\mu}\end{aligned}$$

The equations are valid because of lemma 3 and lemma 1, and the same explanation about the correctness of the SM scheme. The decryption of the second part, $\ddot{\mu}$, also can be proved similarly.

Homomorphism The addition homomorphism property can be proved as follows:

$$\begin{aligned}\mathtt{ct}_+ &= [\mathtt{ct}_1 + \mathtt{ct}_2]_N = [(G\dot{\mu}_1 + D\ddot{\mu}_1) + (G\dot{\mu}_2 + D\ddot{\mu}_2)]_N = [G(\dot{\mu}_1 + \dot{\mu}_2) + D(\ddot{\mu}_1 + \ddot{\mu}_2)]_N \\ &= [G(\dot{te}_1 + \dot{m}_1 + \dot{te}_2 + \dot{m}_2) + D(\ddot{te}_1 + \ddot{m}_1 + \ddot{te}_2 + \ddot{m}_2)]_N \\ &= [G(\dot{te}_+ + \dot{m}_+) + D(\ddot{te}_+ + \ddot{m}_+)]_N\end{aligned}$$

The multiplication homomorphism property can be proved as follows:

$$\begin{aligned}\mathtt{ct}_\times &= [\mathtt{ct}_1.\mathtt{ct}_2]_N = [(G.\dot{\mu}_1 + D.\ddot{\mu}_1).(G.\dot{\mu}_2 + D.\ddot{\mu}_2)]_N \\ &= [g^2p^2.(\dot{\mu}_1.\dot{\mu}_2) + gdpq(\dot{\mu}_1\ddot{\mu}_2 + \dot{\mu}_2.\ddot{\mu}_1) + d^2q^2.(\dot{\mu}_1.\ddot{\mu}_2)]_N \\ &= [g^2p^2.(\dot{\mu}_1.\dot{\mu}_2) + d^2q^2.(\ddot{\mu}_1.\ddot{\mu}_2)]_N \\ &= [G^2.(\dot{te}_\times + \dot{m}_\times) + D^2.(\ddot{te}_\times + \ddot{m}_\times)]_N\end{aligned}$$

Since G and D have the power of two, the resulting ciphertext is labeled 2.

A similar discussion can prove the correctness of scalar addition and multiplication homomorphism.

Remark 19. The proposed doubling technique can be extended to increase the message size by more than double, k. In the k-sized technique, the evaluation key should be updated to $N = p_1 \cdot p_2 \cdots p_k$, where each p_i is a large prime number. The ciphertext would be defined as $\mathtt{ct} = \mu_1 \frac{N}{p_1} + \mu_2 \frac{N}{p_2} + \cdots + \mu_k \frac{N}{p_k}$, and Bézout's coefficients are then used to decrypt the ciphertext into k parts.

4.3 Removing Label

Recall that the labels in the LSC and DLSC schemes are necessary when using the addition algorithm to equalize the labels of two ciphertexts. However, these labels introduce additional communication and computation costs, so that removing the labels can improve performance. In other words, the propuse of this section is providing a scrambled ciphertext without using labeling.

Initially, fresh ciphertexts have a label of one, which increases after each multiplication. Therefore, to eliminate the label attached to a ciphertext, the label must be reduced to one after each multiplication. So, a *reducer* is defined as an extra part of evaluation key as $\mathtt{Re} = [p \cdot r + D^{-1}]_N$ for LSC scheme, and $\mathtt{Re} = [g^{-1} \cdot p^{q-2} + d^{-1} \cdot q^{p-2}]_N$ for DLSC scheme. Therefore, multiplication algorithm should be changed to $\mathtt{ct}_\times = [\mathtt{ct}_1.\mathtt{ct}_2.\mathtt{Re}]_N$. With this changes, the addition algorithm does not need equalizer, and the labels of ciphertexts could be removed.

However, for a ciphertext $\mathtt{ct}$ of the LSC/DLSC scheme, the eavesdropper can compute $\mathtt{ct}' = [\mathtt{ct} \cdot \mathtt{Re}]_N$ to convert $\mathtt{ct}$ into a ciphertext of the SM/DSM scheme, $\mathtt{ct}'$, sacrificing the security advantages of the LSC and DLSC schemes. In addition, this approach relies on circular security which is not ideal from security perspectives. As a result, the proposed idea of removing labels is not realistic, but it is discussed to provide insights for future research.

5 Performance Evaluation and Comparison

This section first evaluates the performance of the LSC scheme, as the main contribution, and then compares it with the most well-known homomorphic schemes; BGV, BFV, TFHE, and CKKS. Implementations are executed on a MacBook Pro with an Apple M1 Pro chip, featuring 10 cores (8 performance and 2 efficiency) and 16 GB of memory, and the times, reported in the tables, represent the average of 5-10 runs.

5.1 Implementation Results

We have implemented the SM and LSC schemes in Rust[16]. The current implementation operates in single-threaded mode, but performance improvements are possible with a multi-threaded optimization. Table 1 presents the timing results for four algorithms: KeyGen, Enc, Eval (Add and Mult), and Dec, with varying values of λ set to 80, 92, 112, and 128, where η is the size of secret key (2η is the size of N). As expected, higher security levels associated with larger λ values result in an increase in the size of N, leading to longer execution times for all algorithms.

[16] The implementation is available at https://github.com/MahdaviCrypt/LHECG and https://github.com/Lhecg/LHECG-Labeled

Table 1. Timing results for four algorithm; ($t = 256$)

λ	η	KeyGen (s)	Enc (μs)	Add (μs)	Mult (μs)	Dec (μs)
80	512	0.21	4.0	0.22	2.54	4.1
92	768	0.43	4.9	0.34	3.5	6.8
112	1024	1.0	7.2	0.4	4.5	11.5
128	1536	3.39	8.2	0.45	8.7	14.5

Table 2 presents the timing results for four algorithms as the multiplicative depth (ℓ) varies. For simplicity, the values of t are chosen as powers of 2 ($t = 2^\tau$); however, the scheme also supports other values. Row 2 indicates that for $\tau = 1$, the scheme can support up to 11 multiplications with an N size of 3072 bits. If the evaluation function requires 50 or 100 multiplications for the same value of τ, the size of N would need to increase accordingly to 13102, and the timing values increase as reported in row 3. If the value of τ increases (row 8), while keeping N constant at 3072 bits, the value of ℓ decreases. Meanwhile, increasing the size of N allows for a greater number of multiplications. For computing ℓ, we used the relationship $\ell = \lfloor \frac{\eta - 1}{\rho + \tau} \rfloor$, while $\rho = \lambda + 2$.

Table 2. Timing results for four algorithm; ($\lambda = 128$)

τ	ℓ	η	KeyGen (s)	Enc (μs)	Add (μs)	Mult (μs)	Dec (μs)
1	11	1536	3.39	8.2	0.45	8.7	14.5
1	50	6551	537.8	28.2	1.3	185.6	132.2
1	100	13101	12824	40.4	2.27	405.8	221
8	11	1536	4.71	9.2	0.57	8.8	6.2
8	50	6901	978.28	40.1	1.8	229.7	159.2
8	100	13801	13729	62.2	2.3	627.4	422.3
16	10	1536	1.18	19.4	1.8	19.4	15.4
16	50	7301	1924	17.8	1.85	298.3	67.4
16	100	14601	17870	93.2	2.9	733.8	615.4
32	9	1536	2.38	20.4	3.04	21.54	19.4
32	50	8101	1712	25.9	1.04	152.7	111
32	100	16201	18207	102.9	5.3	1112.3	738.1

5.2 Comparison

This section compares the implemented LSC scheme with some of the most well-known schemes, including BGVrns, BFVrns, CKKSrns[17], and TFHE[18], in

[17] For BGV, BFV and CKKS, we used the implementation at https://github.com/openfheorg/openfhe-development

[18] For TFHE, we used the implementation at https://github.com/zama-ai/tfhe-rs

security of 128 bits, and for $\ell = 10$[19]. Although these schemes are fully homomorphic, while LSC is leveled, the comparison is limited to leveled evaluations.

Table 3 presents the implementation times. Since the equalizer is a multiplication operation, if addition requires equalization using pre-computed values of $\mathtt{Eq}^{(i)}$, the time required for addition will increase by approximately one additional multiplication. Consequently, such an addition will take nearly as long as a multiplication. Given this, the addition and decryption times for the LSC scheme are comparable to the best times among the other four schemes. However, the LSC scheme performs multiplication approximately 1,000 times faster than the established schemes. Additionally, the Encryption algorithm of the LSC scheme is 100 times faster than that of the other four schemes.[20] Although the KeyGen algorithm of the LSC scheme is longer than that of the other four schemes, this is acceptable since it only needs to be run once.

Table 3. Performance results for different schemes; ($\lambda = 128$)

Scheme	τ	ℓ	KeyGen (ms)	Enc (μs)	Add (μs)	Mult (μs)	Dec (μs)
BGVrns	16	10	2.434	1686	32	19771	226
BFVrns	16	10	1.906	2405	23	15783	324
CKKSrns	16	10	2.627	1697	43	23368	107
TFHE	16	NA	546.5	285.72	107111	382582	6.06
LSC	16	10	1180	19.4	1.8	19.4	15.4

While the compared schemes offer quantum security, our schemes provide significant performance advantages in non-quantum settings. In many practical applications where quantum resistance is not yet a critical requirement, our schemes deliver a highly efficient and favorable solution.

6 Conclusion

We have proposed the first leveled homomorphic encryption schemes over integers within composite groups. Specifically, we designed a Straw-Man scheme that supports ℓ multiplications, and then to make the traffic of ciphertexts indistinguishable as encrypted, the Labeled Scrambled Ciphertext scheme is designed using the labeling technique. Additionally, we introduced a modified version of the Rothblum transformation to create a public key version of the symmetric schemes. Furthermore, we presented double-sized message schemes to enhance

[19] TFHE can support each level since it runs programmable bootstrapping after each multiplication.

[20] It is important to note that our scheme is in its early stages of development, whereas the other schemes have undergone years of refinement. Therefore, we anticipate that further optimizations will make our scheme even more efficient over time.

efficiency. Implementation results demonstrate that our schemes significantly outperform well-known homomorphic schemes, particularly in multiplication. For future work, we aim to explore the use of SIMD to further improve performance and investigate the potential of bootstrapping to transform these leveled homomorphic encryption schemes into fully homomorphic encryption schemes.

Acknowledgments

The firs author was supported by the Spanish Ministry of Science and Innovation through the PID2021-125962OB-C31 "SECURING" project. Additional funding was provided by the ARTEMISA International Chair of Cybersecurity (C057/23) and the DANGER Strategic Project of Cybersecurity (C062/23), both funded by the Spanish National Institute of Cybersecurity through the European Union — NextGenerationEU and the Recovery, Transformation, and Resilience Plan. We also thank Jeongeun Park from NTNU and Hilder Vitor Lima Pereira from Unicamp for their valuable comments.

References

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 297–314. Springer, 2014.

[BBL17] Daniel Benarroch, Zvika Brakerski, and Tancrède Lepoint. Fhe over the integers: decomposed and batched in the post-quantum regime. In *IACR International Workshop on Public Key Cryptography*, pages 271–301. Springer, 2017.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[BLLN13] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings 14*, pages 45–64. Springer, 2013.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual International Cryptology Conference*, 2012.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, 43(2):831–871, 2014.

[CGGI17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In *International Conference on the Theory and Application of Cryptology and Information Security*, 2017.

[CJP20] Ilaria Chillotti, Marc Joye, and Pascal Paillier. New challenges for fully homomorphic encryption. In *Privacy-preserving machine learning (PPML-PriML 2020) NeurIPS 2020 workshop*, 2020.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 446–464. Springer, 2012.

[CP19] Jean-Sébastien Coron and Hilder VL Pereira. On kilian's randomization of multilinear map encodings. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 325–355. Springer, 2019.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*, pages 1–17. Springer, 2013.

[GIKV23] Robin Geelen, Ilia Iliashenko, Jiayi Kang, and Frederik Vercauteren. On polynomial functions modulo pe and faster bootstrapping for homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 257–286. Springer, 2023.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, page 365–377, New York, NY, USA, 1982. Association for Computing Machinery.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *IACR Cryptol. ePrint Arch.*, 2013:340, 2013.

[JMPP24] Robin Jadoul, Axel Mertens, Jeongeun Park, and Hilder VL Pereira. Ntru-based fhe for larger key and message space. In *Australasian Conference on Information Security and Privacy*, pages 141–160. Springer, 2024.

[Klu22] Kamil Kluczniak. Ntru-v-um: secure fully homomorphic encryption from ntru with small modulus. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, pages 1783–1797, 2022.

[LATV12a] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

[LATV12b] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic

encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

[MHWW24] Shihe Ma, Tairong Huang, Anyu Wang, and Xiaoyun Wang. Accelerating bgv bootstrapping for large p using null polynomials over z pe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 403–432. Springer, 2024.

[MNR24] Axel Mertens, Georgio Nicolas, and Sergi Rovira. Convolution-friendly image compression in fhe. Cryptology ePrint Archive, Paper 2024/559, 2024. https://eprint.iacr.org/2024/559.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[Per20] Hilder Vitor Lima Pereira. Efficient agcd-based homomorphic encryption for matrix and vector arithmetic. In *International Conference on Applied Cryptography and Network Security*, pages 110–129. Springer, 2020.

[Per21] Hilder Vitor Lima Pereira. Bootstrapping fully homomorphic encryption over the integers in less than one second. In *Public-Key Cryptography–PKC 2021: 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10–13, 2021, Proceedings, Part I 24*, pages 331–359. Springer, 2021.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In Yuval Ishai, editor, *Theory of Cryptography*, pages 219–234, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 377–394. Springer, 2010.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *International Conference on the Theory and Application of Cryptographic Techniques*, 2010.

Appendix 1: Security Assumptions

This appendix defines two security assumptions and discusses their hardness.

Definition 3 (Double Modulus Problem). *Let $N = pq$, where p and q are two η-bit prime numbers such that factoring of N is λ-bit secure. The computation of $[[r]_p]_t$ for any integer $r \in \mathbb{Z}_N$ and a known integer $t \ll p, q$ is referred to as the Double Modulus (DM) Problem.*

Theorem 1. *Let $N = pq$ where p and q are two η-bit prime numbers such that factoring of N is λ-bit secure. For any attacker $\mathcal{A}$ that can solve the Double Modulus Problem (Def. 3), there exists an attacker $\mathcal{B}$ that can factor N.*

Proof. To factor N, the attacker $\mathcal{B}$ can run algorithm 1 as follow.
Considering I and F defined in line 3, since $F - I = 2^{\eta}$, and $p < 2^{\eta}$, so there

Algorithm 1: Factoring with DM attacker

Input: $\mathcal{A}$, and $N \in [2^{2\eta-1}, 2^{2\eta})$
Output: $p \in [2^{\eta-1}, 2^{\eta})$

```
1  Select I' ∈ Z_N;
2  Query I' from A; (z' = [[I']_p]_t ← A(I'))
3  Set I = I' − z' , and F = I + 2^η;
4  if F > N then
5      go to step 1
6  k = 2^8;
7  while k < F − I do
8      r = [⌊(I+F)/2⌋];
9      z = [[r]_p]_t ← A(r)
10     if z = [r − I]_t then
11         I = r − z
12     else
13         F = r
14 p = 1
15 while p = 1 do
16     for i = I : F do
17         p = gcd(i, N)
18 return p;
```

exists an integer T such that $I < T < F$ and $[T]_p = 0$. If $I' = \alpha \cdot p + te + z'$, then $I = \alpha \cdot p + te$, and the target point that the algorithm aims to find can be defined as $T = (\alpha + 1)p$, and then extract $p = \gcd(T, N)$.
Even though the probability of line 4's condition being true is negligible, we consider it to maintain accuracy. This is because if $F > N$, the target point T becomes equivalent to N, which is not helpful for factoring. Additionally, in each step of the proposed algorithm, I is set such that $[[I]_p]_t = 0$.

While $I = \alpha p + te$, and r defines as line (8), if $p > r$, the value of r definitely equals $\alpha p + te' + z$, and then $[r - I]_t = [t(e' - e) + z]_t = z$. This value would be equal to the value of z received from attacker $\mathcal{A}$, line (9). Then I should be substituted with the new value $r - z$; otherwise, $p < r$ and the value of r definitely equals $(\alpha + 1)p + te' + z$, and then $[r - I]_t = [p + t(e' - e) + z]_t \neq z$, so the value of F should be substitute by r (line 13). During this process, the target point would be between I and F and always $[[I]_p]_t = 0$. By repeating the process, the attacker $\mathcal{B}$ reaches to a searchable space, such as $k = 2^8$, which includes the target point T. Eventually, the attacker $\mathcal{B}$ can search to find p through computing $gcd(T, N)$.

The complexity of algorithm 1 is $O(\log N)$, indicating its efficiency in factoring N. □

Remark 20. Algorithm 1 is effective when the attacker $\mathcal{A}$ can solve the DM problem with a probability of 1. However, if the attacker $\mathcal{A}$ solves the DM problem with a probability of $\frac{1}{t}+\epsilon$, where ϵ is non-negligible, attacker $\mathcal{B}$ can still factorize N with high probability. To do this, for each specific r, attacker $\mathcal{B}$ queries $(2j+1)$ times, where $r_j = r+i$ and $-j \leq i \leq j$, where j is much smaller than 2^λ. Then, the attacker $\mathcal{A}$ returns $z_i = [[r_i]_p]_t$ and attacker $\mathcal{B}$ computes the possible values of $z = [[r]_p]_t$ by calculating $z^{(i)} = z_i - i$. The final answer is determined by selecting the most frequent value from the set $\vec{Z} = \langle z^{(-j)}, \ldots, z^{(-1)}, z^{(0)}, z^{(1)}, \ldots, z^{(j)} \rangle$.

Corollary 1. *For $t = 2$, the DMDM problem is equivalent to computing $LSB(r)$. It follows from Algorithm 1 and Remark 20 that $LSB(r)$ is a hardcore bit of a random element $r \in \mathbb{Z}_N$. Hence, any adversary that predicts $LSB(r)$ with non-negligible advantage can be leveraged to construct an efficient algorithm for factoring N.*

Definition 4 (Decisional Modified Double Modulus Problem). *Let $N = pq$, where p and q are two η-bit prime numbers such that factoring of N is λ-bit secure. Given a random vector $\vec{x} = \langle x_0, ..., x_{k-1} \rangle \in \mathbb{Z}_N^k$ such that $[[x_i]_p]_t = 0$ for $0 \leq i \leq k-1$, and a challenge pair $(x, z) \in \mathbb{Z}_N \times \mathbb{Z}_t$, where $|x - x_i| \gg 2^\eta$, deciding about $z = [[x]_p]_t$ is defined as the Decisional Modified Double Modulus (DMDM) Problem* [21].

Unlike the previous discussion, which proved the hardness of the DM problem, the following explanation offers a discussion aimed at providing insights into the hardness of the DMDM problem.

The DMDM problem is similar to the AGCD (and noise-free AGCD) problem. Below, a comparison between these two problems is discussed. In the DMDM problem, the x_is are 2η-bit integers that can be expressed as $x_i = pr_i + te$, where p and r_i are η-bit integers, and noise e is a ρ-bit integer in the level 1 version. In contrast, for the AGCD problem, $x_i = pr_i + e$ is typically a λ^5-bit integer, where p is about λ^2-bit, r_i is about λ^3-bit, and noise e is about λ-bit[22]. For both DMDM and AGCD problems, it is evident that the larger the noise compared to the given number x_i, the more difficult it becomes to find p. To have a better comparison between the two problems, the ratio of the noise size to the given number size should be defined. Specifically, this ratio is equal to the ratio of the noise size to the size of x_0 in AGCD ($\frac{\rho}{|x_0|}$), and the ratio of the noise size to the size of N in DMDM problem ($\frac{\rho}{|N|}$), where $\frac{\rho}{|N|} > \frac{\rho}{|x_0|}$. For instance, for $\lambda = 128$, in the AGCD problem, $\frac{\rho}{|x_0|} = \frac{\rho}{\lambda^5} \approx \frac{1}{\lambda^4} = 2^{-28}$, whereas in DMSM

[21] The level 1 version of the DMDM problem is defined as $t \cdot 2^\rho \leq [x_i]_p < t^2 \cdot 2^{2\rho}$, where $\rho > \lambda$, and the level l version is defined as $t^l \cdot 2^{l\rho} \leq [x_i]_p < t^{l+1} \cdot 2^{(l+1)\rho}$.

[22] Note that in noise-free AGCD, x_0 is defined as $x_0 = pr_0$. For a better understanding of the definition of x_0, refer to [vDGHV10,CP19]

problem, $\frac{\rho}{|N|} = \frac{\rho}{2\eta} \approx \frac{128}{3072} = \frac{1}{24} \approx 2^{-5}$. It is evident that as this ratio increases, the security of the assumption strengthens, leading to the conclusion that the proposed attacks on the AGCD and noise-free AGCD problems are less likely to succeed against the DMDM problem.

Remark 21. The DMDM problem for (x, z) can be solved if there exist x_i such that $\delta = |x - x_i| < 2^{\lambda}$. Specifically, x and x_i can be rewritten as $\alpha' p + te + z'$ and $\alpha p + te_1$ respectively, and then $|x - x_i| = |(\alpha' - \alpha)p + t(e - e_1) + z'|$. Since $|x - x_i| < 2^{\lambda}$, with hight probability $\alpha' = \alpha$. So the value of z' can be computed as $z' = [x - x_i]_t$, then it can be compare with the given z. If $z = z'$, it can be concluded that $z = [[x]_p]_t$; otherwise, z is not a valid value for $[[x]_p]_t$.

Lemma 4. *Let $N = pq$, where p and q are prime numbers. Finding $x \in \mathbb{Z}_p$, or $y \in \mathbb{Z}_p$, for any integer $c \in \mathbb{Z}_N$ such that $qx + py = c$ is equivalent to factoring N.*

Proof. Let an algorithm $\mathcal{A}$ find x for any equation $qx + py = c$, where $c \in \mathbb{Z}_N$. Then, assume that $x = a$ is a solution for $qx+py = c$ and $x' = a'$ for $qx'+py' = c'$. This leads to the equations $[q + pya^{-1}]_N = [ca^{-1}]_N$ and $[q + py(a')^{-1}]_N = [c'(a')^{-1}]_N$. Therefore, $[q+pya^{-1} - (q+py(a')^{-1})]_N = [ca^{-1} - c'(a')^{-1}]_N$, which simplifies to $[pya^{-1} - py(a')^{-1}]_N = [ca^{-1} - c'(a')^{-1}]_N$. Consequently, it can be concluded that $\gcd(N, [ca^{-1} - c'(a')^{-1}]_N) = p$.[23] □

Corollary 2. *By an argument analogous to that employed in Algorithm 1 and Remark 20, it follows that either $LSB(x)$ or $LSB(y)$ is a hardcore bit of the value $c = qx + py$ introduced in 4. Consequently, any adversary capable of predicting either $LSB(x)$ or $LSB(y)$ with non-negligible advantage can be transformed into an algorithm that efficiently recovers x or y.*

Appendix 2: Security Proofs for the Schemes

The following discusses the security proofs for the proposed SM and LSC schemes.

Theorem 2. *If an attacker $\mathcal{A}$ can win the Ind-CPA game for the proposed SM scheme, there is an attacker $\mathcal{B}$ who can solve the DMDM problem.*

Proof. Let $\mathcal{A}$ be an attacker that can break the proposed SM scheme with a non-negligible advantage ϵ. So, an attacker $\mathcal{B}$ can be simulated that solves DMDM problem[24]. Suppose that the challenger generates the inputs of DMDM problem as $\{N, \vec{x}\}$ where $\vec{x} = \langle x_0, x_1, ..., x_{k-1} \rangle$, and $[[x_i]_p]_t = 0$. The challenger flips a fair coin ν and sets $z^{\star} = [[x^{\star}]_p]_t$ if $\nu = 1$; otherwise, $z^{\star} \in \mathbb{Z}_t$ such that $z^{\star} \neq [[x^{\star}]_p]_t$. The attacker $\mathcal{B}$ aims to distinguish if $z^{\star} = [[x^{\star}]_p]_t$ or not and return his guess ν', with non-negligible advantage. The security game works as follows.

[23] Note that the proof can be extended such that the algorithm $\mathcal{A}$ computes only $x = a$ for $qx + py = c$ without computing $x' = a'$ for $qx' + py' = c'$.

[24] To be more precise, we should consider DMDM with level 1 version to guarantee evaluation and computation over encrypted data.

– *Initialization.* The attacker $\mathcal{B}$ receives $\{N, \vec{x}\}$, and challenge pair $(x^\star, z^\star)$ as a DMDM problem inputs.

– *Setup.* The attacker $\mathcal{B}$ sets N as an evaluation key and sends it to the attacker $\mathcal{A}$.

– *Query 1.* The attacker $\mathcal{A}$ queries for encryption of the message m. For any received query m, the attacker $\mathcal{B}$ simulates the encryption algorithm through $\mathtt{ct} = [x_i + m]_N$, where x_i is a random element of the vector $\vec{x}$, and forwards the ciphertext $\mathtt{ct}$ to the attacker $\mathcal{A}$[25].

– *Challenge.* The attacker $\mathcal{A}$ selects two messages (m_0, m_1) and sends them to the attacker $\mathcal{B}$. Then attacker $\mathcal{B}$ selects a random bit $b \in \{0, 1\}$ and a ρ-bit random noise e, and computes $\mathtt{ct}^\star = x^\star - z^\star + m_b$ and sends it to the attacker $\mathcal{A}$.

– *Query 2.* The query 1 phase is repeated.

– *Guess.* The attacker $\mathcal{A}$ forwards its guess about b as b' to the attacker $\mathcal{B}$. If $b' = b$, the attacker $\mathcal{B}$ outputs $\nu' = 1$, indicating that $z^\star = [[x^\star]_p]_t$ in the given DMDM problem instance; otherwise, it outputs $\nu' = 0$ indicating that $z^\star$ is a random.

Note that if $z^\star = [[x^\star]_p]_t$, then $\mathtt{ct}^\star$ is the encryption of m_b. Because, by running decryption algorithm, m_b can be achieved (i.e. $[[\mathtt{ct}]_p]_t = m_b$). In this case, attacker $\mathcal{A}$ can distinguish with probability $\frac{1}{2}+\epsilon$. On the other hand, if $z^\star = [[x^\star]_p]_t$ does not hold, $\mathtt{ct} = x^\star - z^\star + m_b$ seems as a random number and the probability of success for the attacker $\mathcal{A}$ is equal to $\frac{1}{2}$. The overall advantage of the attacker $\mathcal{B}$ for DMDM problem is:

$$\begin{aligned} \mathbf{Adv}_{\mathcal{B}}^{DMDM} &= \frac{1}{2} Pr[\nu' = \nu | \nu = 0] + \frac{1}{2} Pr[\nu' = \nu | \nu = 1] - \frac{1}{2} \\ &= \frac{1}{2} \cdot (\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2} \end{aligned} \tag{6}$$

In equation 6, the value of $\frac{\epsilon}{2}$ is non-negligible, so the probability of solving the DMDM problem is non-negligibly greater than $\frac{1}{2}$. □

Remark 22. In the SM scheme, if the value of the evaluation key, N, is not published, the risk of computing the secret key using a (quantum) factoring algorithm will be eliminated. However, under such circumstances, the ciphertext size is not constant and tends to increase with each subsequent evaluation.

[25] The value of k is big enough, but if the number of encryption query is greater than k, attacker $\mathcal{B}$ can select a ρ-bit random noise e and a random set $S \subset \{0, 1, ..., k\}$, and then compute $\mathtt{ct} = \sum_{i \in S} x_i + m + te$.

Remark 23. According to Remark 21, if an attacker has two ciphertexts, $\mathtt{ct}$ and $\mathtt{ct}'$, such that $|\mathtt{ct} - \mathtt{ct}'| \ll 2^{\eta}$, and one of the associated messages, m or m', they may attempt to retrieve the other message. Suppose $\mathtt{ct} = pr + te + m$ and $\mathtt{ct}' = pr' + te' + m'$. Given that $|p(r - r') + t(e - e') + (m - m')| \ll 2^{\eta}$, it follows that $r = r'$ with high probability. The attacker can then compute $m - m'$ via $[t(e - e') + (m - m')]_t$. If the attacker knows one of m or m', they can compute the other message. This attack becomes practical if $[r]_q = [r']_q$ holds. However, the probability of this condition being satisfied is approximately $\frac{1}{q}$, which is considered negligible.

Theorem 3. *The security of the proposed LSC scheme is at least as strong as the security of SM scheme. Therefore, if an attacker $\mathcal{D}$ can break the LSC scheme, then there exists an attacker $\mathcal{A}$ who can break the SM scheme.*

Proof. Let $\mathcal{D}$ be an attacker that can break the LSC scheme. We simulate an attacker $\mathcal{A}$ that can break the SM scheme. The SM scheme oracle sends the public parameters and evaluation key to the attacker $\mathcal{A}$. The attacker $\mathcal{A}$ request the encryption of 1 from the SM oracle and receive $E_1 = SM.Enc(1) = pr + te + 1$. The attacker $\mathcal{A}$ selects an η-bit integer D as a part of secret key in the LSC scheme and computes $D \cdot E_1 = pr' + D(te + 1)$ and defines it as equalizer $\mathtt{Eq}$. The attacker $\mathcal{A}$ forwards the public parameters, evaluation key, and the equalizer $\mathtt{Eq}$ to the attacker $\mathcal{D}$. Then the attacker $\mathcal{D}$ queries the encryption of message m. The attacker $\mathcal{A}$ forwards the query to SM oracle and receives ciphertext $\mathtt{ct}_{\mathtt{SM}}$. The attacker $\mathcal{A}$ forwards $\mathtt{ct}_{\mathtt{LSC}} = D \cdot \mathtt{ct}_{\mathtt{SM}}$ to the attacker $\mathcal{D}$. Suppose $\mathtt{ct}^{\star}_{\mathtt{SM}} = pr^{\star} + te^{\star} + m^{\star}$ is a challenge ciphertext of the SM scheme. Attacker $\mathcal{A}$ computes $[D \cdot \mathtt{ct}^{\star}_{\mathtt{SM}}]_N$ with label 1 and sends it to the attacker $\mathcal{D}$. In the viewpoint of attacker $\mathcal{D}$, this value equals to $\mathtt{ct}^{\star}_{\mathtt{LSC}}$ as follows:

$$\begin{aligned}[D \cdot \mathtt{ct}^{\star}_{\mathtt{SM}}]_N &= [D(pr^{\star} + te^{\star} + m^{\star})]_N \\ &= [pr + D(te + m^{\star})]_N = \mathtt{ct}^{\star}_{\mathtt{LSC}}\end{aligned}$$

Since $\mathtt{ct}^{\star}_{\mathtt{LSC}}$ is the encryption of the challenge message $m^{\star}$ under the LSC scheme, the attacker $\mathcal{D}$ can retrieve $m^{\star}$ and sends it to attacker $\mathcal{A}$, which means that the attacker $\mathcal{A}$ breaks the SM scheme. □