An Optimized GHV-Type HE Scheme: Simpler, Faster, and More Versatile*

Liang Zhao[1(✉)], Ze Chen[1], Liqun Chen[2], and Xinyi Huang[3]

[1] School of Cyber Science and Engineering, Sichuan University, Chengdu, China
zhaoliangjapan@scu.edu.cn, xkwy8808@gmail.com
[2] Surrey Centre for Cyber Security, University of Surrey, Guildford, UK
liqun.chen@surrey.ac.uk
[3] Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China
xyhuang@fjnu.edu.cn

Abstract. In this paper we present an optimized variant of Gentry, Halevi and Vaikuntanathan (GHV)′s Homomorphic Encryption (HE) scheme. Our scheme is appreciably more efficient than the original GHV scheme without losing its merits of the (multi-key) homomorphic property and matrix encryption property. In this research, we first measure the density for the trapdoor pairs that are created by using Alwen and Peikert′s trapdoor generation algorithm and Micciancio and Peikert′s trapdoor generation algorithm, respectively, and use the measurement result to precisely discuss the time and space complexity of the corresponding GHV instantiations. We then propose a generic GHV-type construction with several optimizations that improve the time and space efficiency from the original GHV scheme. In particular, our new scheme can achieve asymptotically optimal time complexity and avoid generating and storing the inverse of the used trapdoor. Finally, we present an instantiation that, by using a new set of (lower) bound parameters, has the smaller sizes of the key and ciphertext than the original GHV scheme.

Keywords: Homomorphic encryption · LWE · Matrix operations

1 Introduction

Background and Related Work. Homomorphic Encryption (HE) allows running operations on ciphertexts so that decryptions match the results from the corresponding operations on plaintexts. HE has many interesting applications in the real-world, e.g., the electronic voting [12], the private matching [13], the computational private information retrieval [6], and the indistinguishability obfuscation [5]. Since introduced by Rivest, Adleman and Dertouzos [30] in

* A preliminary version of this paper appears in the proceedings of the 20th International Conference on Applied Cryptography and Network Security (ACNS 2022). This is the corresponding full version.

1978, the HE research has a long history in the modern cryptography. Early HE systems focused on evaluating asymmetric encryption and supports only one operation over encrypted data, either addition or multiplication. This type of HE is referred to as partially HE. Typical examples involve the additively HE schemes: Goldwasser-Micali and Paillier, and the multiplicatively HE schemes: RSA and ElGamal.

Breaking through the single operation homomorphism took a long time. The first step forward was given in 2005 by Boneh, Goh and Nissim (BGN for short) [6], who presented an additively HE scheme supporting one multiplication. An HE scheme that can evaluate two types of operations but only for a subset of operations is referred to as a Somewhat Homomorphic Encryption (SHE) scheme. The BGN scheme is the first SHE scheme. Breaking the security of this scheme is as hard as solving the subgroup-membership problem in composite-order groups that admit bilinear maps. Later Gentry, Halevi and Vaikuntanathan (GHV for short) in 2010 [15] proposed an additively HE scheme supporting one "direct" matrix multiplication. Notably, a "direct" matrix multiplication here means an ordinary matrix multiplication that does not require any extra computation. Security of their scheme is based on the standard Learning With Errors (LWE) assumption (see Sect. 2.1). The GHV scheme can be regarded as an improvement of the BGN scheme and has several inherent advantages (see Appendix A.2 on details of the GHV scheme). Specifically, one significant advantage is that there is a worst-case/average-case classical reduction from the standard LWE problem to the GHV security. Another important advantage is that the GHV scheme can encrypt messages from a large space (i.e., any matrix ring) and has no restriction for the output size. Moreover, the GHV scheme holds much of the flexibility of the LWE-based cryptosystem, e.g., it can be made identity-based and leakage-resilient. In a nutshell, the GHV cryptosystem is still an outstanding SHE scheme.

The first theoretically feasible construction capable of supporting arbitrary computations over ciphertexts, which is referred to as Fully HE (FHE), was introduced by Gentry in 2009 [14]. Since then, many FHE schemes have been proposed (e.g., [4, 10, 17, 19, 31, 33]). Generally speaking, the development of FHE until now involves three generations. Typical examples of the first generation are Gentry′s initial scheme based on ideal lattices [14] and van Dijk et al.′s proposal employing integer arithmetic [31]. The second generation includes Brakerski and Vaikuntanathan′s constructions [4, 10] that use new techniques to control the growth of noise. The third generation of FHE originates from the scheme of Gentry, Sahai and Waters (GSW for short) [17], which exhibits a somewhat distinct noise growth pattern. Although there is a great progress for the theoretical and practical improvements of FHE, for many applications, especially the applications requiring a single algebraic operation, this type of encryption is currently impractical because of the big key size, the large ciphertext expansion and the long evaluation time [7, 11, 28].

Besides the GHV scheme, there are two asymmetric HE schemes that can encrypt matrices and support homomorphic matrix addition and multiplication.

The first one was proposed in 2015 by Hiromasa, Abe, and Okamoto (HAO for short) [19], and their scheme is a matrix extension of the GSW-FHE scheme [17]. Security of the HAO scheme can be reduced from the standard LWE assumption, while an additional special circular security assumption is necessary. The homomorphic matrix multiplication does not correspond to the "direct" matrix multiplication and needs to employ a randomized function. In 2018, Wang, Wang, Xue, and Huang (WWXH for short) [33] presented another FHE scheme for encrypting matrices. Security of their scheme is based on hardness of the standard LWE problem, and the size of ciphertext matrices is smaller than that of the HAO scheme. However, for the WWXH scheme, the tensor product is largely employed to perform the homomorphic matrix multiplication, and the corresponding computational cost is $\Omega(m^4)$ for $m \times m$ input matrices. Thus, the complexity of using this scheme for some homomorphic computations (e.g., homomorphic computations over nondeterministic finite automata and linear algebra) is very large and not lower than that of the HAO scheme for the same computations. Some details of these two matrix-FHE schemes are listed in Table 1. Notably, total computational costs of both schemes are $\mathcal{O}(m^3)$.

Motivation and Our Target: Building a More Efficient GHV-Type HE Scheme. Based on the above descriptions, asymmetric matrix-FHE schemes [19, 33] currently do not match with very efficient cloud computing-related applications that only run a single (linear algebra) operation. A typical example is the private and verifiable delegation of linear algebra [23] that only allows a client to run $\mathcal{O}(m^{c'})$ computation for matrices of large size $m \times m$, where $c' \in [2, 3[$ is close to 2. SHE schemes are much more efficient and suitable for many applications. In particular, the GHV scheme has a sequence of desirable properties. Allowing encryption of a square matrix from any matrix ring in one operation and supporting the "direct" homomorphic matrix multiplication can make this scheme match with applications requiring the linear algebra computation over any ring, and be a powerful tool for the very efficient verifiable linear algebra computation. Although the construction of the GHV scheme is elegant, it seems that there are some optimizations left in its performance, and these optimizations can make it more versatile. This brings the main question that we want to answer in this work: *Can we create a more efficient GHV-Type HE scheme?* In more detail, this question involves the following three aspects:

- The new GHV-type HE scheme has lower time complexity, and in particular it is suitable for applications only permitting efficient privacy protection and verification (e.g., the private and verifiable delegation of linear algebra).
- The new GHV-type HE scheme has lower space complexity. To achieve this we need to first figure out whether some key employed by the original GHV scheme is not needed for the improved one.
- The new GHV-type HE scheme has smaller key and ciphertext sizes than those of the original GHV scheme.

Our Results. In this work, we propose an efficient GHV-type HE scheme together with optimized parameters. Security of our proposal is still based on the standard LWE assumption. Specifically, our contributions are four folds:

First Result (Sect. 3): Density of trapdoor matrix pairs. Trapdoor generation algorithms (e.g., [1, 3, 25]) play a big role in advanced lattice-based cryptographic primitives. They generate a pair of matrices ($\mathbf{A}^t$, $\mathbf{T}^t$), i.e., $\mathbf{A}^t$ is an (almost) uniformly random matrix and $\mathbf{T}^t$ is the corresponding trapdoor that is in the form of a nonsingular square matrix with short integer vectors. Some significant parameters related to the matrix pair, i.e., the lattice dimension and the quality of the trapdoor, generally have been explored when the corresponding trapdoor construction was given. In an asymmetric encryption scheme, ($\mathbf{A}^t$, $\mathbf{T}^t$) can be used as the public and secret keys. In particular, since the short basis $\mathbf{T}^t$ and its inversion $(\mathbf{T}^t)^{-1}$ used in the encryption schemes may multiply by matrices over a matrix ring, a natural question is how to evaluate the corresponding computational cost. To answer this question, we first introduce the concept of the density of a (trapdoor) matrix for matrix multiplication and give its definition. Actually, the density of a (trapdoor) matrix is measured by the number of nonzero elements of a matrix needed for a single matrix multiplication. Then, we take ($\mathbf{T}^t$, $(\mathbf{T}^t)^{-1}$) respectively generated by Alwen and Peikert′s trapdoor sampling algorithm (APTrapSamp for short) [3] and by Micciancio and Peikert′s trapdoor sampling algorithm (MPTrapSamp for short) [25] as targets and analyze their concrete density. Notably, the non-deterministically constructed components of these two trapdoor matrix pairs become the hard nut of the corresponding density analyses. Technically, we thus employ the matrix decomposition to simplify the complex components, which makes us simply focus on exploring components with the deterministic distribution. Using our concrete decompositions, for ($\mathbf{T}^t$, $(\mathbf{T}^t)^{-1}$) generated by APTrapSamp and MPTrapSamp, the analyses give accurate estimates on their density (see Lemma 4 to 7).

Second Result (Sect. 3 and 4): More accurate efficiency analyses. For the GHV-HE scheme, although the approximate result of its computational cost has been given in [15], the more accurate estimate on the computational cost is important, in particular for finding applications which the cryptosystem can be plugged directly into. Hence, we carefully analyze the encryption and decryption procedures of the GHV scheme using APTrapSamp and MPTrapSamp, and present accurate results on their computational cost and space cost (see Theorem 2 to 5). Technically, our analysis for the decryption procedure is based on the idea that multiplying matrices over a matrix ring with $\mathbf{T}^t$ (resp. $(\mathbf{T}^t)^{-1}$) is equivalent to multiplying matrices over a matrix ring with the decomposition form of $\mathbf{T}^t$ (resp. $(\mathbf{T}^t)^{-1}$). This implies that results on the density of $\mathbf{T}^t$ and $(\mathbf{T}^t)^{-1}$ are used for the efficiency analyses of the decryption procedure. We also employ the Hoeffding′s inequality to estimate a (near-)lower bound of the computational cost of the decryption procedure. Of course, the same idea is used to give the (time and space) efficiency analyses on our optimized GHV-type scheme (see Theorem 10).

Third Result (Sect. 4): Simpler construction and optimizations. Towards addressing the question of the above section in a systematic way, we first propose a generic GHV-type construction that removes the expensive matrix inversion computation for $(\mathbf{T}^t)^{-1}$ and the multiplication by $(\mathbf{T}^t)^{-1}$ on decryption. In our generic construction, a sparse matrix $\tilde{\mathbf{T}}$ that is easily built is employed to recover the plaintext message (see Sect. 4.2 on $\tilde{\mathbf{T}}$). Notice that, $\tilde{\mathbf{T}}$ is constructed deterministically, which means that it actually can be regarded as a "public" key for decryption. Moreover, our generic construction has an additional benefit for the multiplication by $\mathbf{T}^t$ on decryption. That is, a plaintext message can be recovered by multiplying with part of $\mathbf{T}^t$ instead of $\mathbf{T}^t$, which further reduces the computational cost and space cost of decryption. Then we present some simple optimizations on speeding up the matrix multiplication used in our generic construction (see Algorithm 2 and 3). For our optimizations, only element-wise additions are employed to achieve the multiplication by part of $\mathbf{T}^t$ and $\tilde{\mathbf{T}}$ on decryption, and a random, short component of $\mathbf{T}^t$ is used as the unique secret key (i.e., the component matrix $\mathbf{R}$ in Appendix A.1). This implies that our optimizations guarantee that any instantiation of our GHV-type scheme using $\mathsf{APTrapSamp}$-like trapdoor generation algorithm can have the asymptotically optimal time complexity and storage size of the secret key. Surprisingly, we achieve these efficiency improvements without having a negative effect on the security of the concrete GHV-type scheme.

Fourth Result (Sect. 4): Tighter parameters. To ensure that our GHV-type instantiation using $\mathsf{APTrapSamp}$ enjoys correctness and the same homomorphism as the original GHV instantiation using $\mathsf{APTrapSamp}$ holds, we show new bounds for the modulus q and the lattice dimension m (see Theorem 6 and 7). In particular, the parameter bounds that we establish are lower than those of the original GHV instantiation (see Appendix A.2). Since q has a direct impact on the key and ciphertext sizes, this means that sizes of elements of the public key and ciphertext can be smaller than those of the original GHV instantiation. Specifically, we first give a parameter setting for the case that our GHV-type instantiation only supports polynomially many additions (see Theorem 6). Then we present a parameter setting for the case that our GHV-type instantiation can permit polynomial number of additions and one multiplication (see Theorem 7).

Comparisons and Applications. A comparison of our optimized GHV (oGHV for short) scheme with the GHV scheme and other asymmetric matrix-(F)HE schemes is shown in Table 1, where we assume that all the schemes make use of the same security parameter n and plaintext matrix size $m \times m$. Notice that n and m are not the same; indeed, typically we have $m = \Theta(n \lg q)$, where $q = \mathsf{poly}(n)$.

Clearly, based on the above comparisons, we believe that our oGHV scheme can be plugged in as a "black box" to replace the original GHV scheme and deliver significant efficiency benefits in such applications discussed by Gentry et al. [15], e.g., electronic election protocols, private information retrieval protocols and identity-based encryption. Of course, the oGHV scheme may be used as a drop-in replacement in some other typical applications such as two-party compu-

Table 1. Comparisons of asymmetric LWE-based matrix-HE schemes with equal parameters n and m satisfying $n \ll m$.

Scheme	Classification	Homomorphism		Encryption Time	Decryption Time
		$\oplus^{\dagger}$	$\odot^{\dagger}$		
GHV(APTrapSamp) [15]	SHE	✓	✓	$\mathcal{O}(nm^2)$	$\mathcal{O}(m^3)$
GHV(MPTrapSamp)*	SHE	✓	✓	$\mathcal{O}(nm^2)$	$\mathcal{O}(m^3)$
HAO [19]	FHE	✓	✗	$\tilde{\mathcal{O}}(nm^2)$	$\mathcal{O}(m^3)$
WWXH [33]	FHE	✓	✗	$\tilde{\mathcal{O}}(nm^2)$	$\mathcal{O}(m^3)$
oGHV(APTrapSamp) [this paper]	SHE	✓	✓	$\mathcal{O}(nm^2)$	$\tilde{\mathcal{O}}(nm^2)$
oGHV(MPTrapSamp) [this paper]	SHE	✓	✓	$\mathcal{O}(nm^2)$	$\tilde{\mathcal{O}}(nm^2)$

* For GHV(MPTrapSamp), MPTrapSamp is used in the original GHV scheme.
† $\oplus$ and $\odot$: "Direct" matrix addition and multiplication between the input ciphertexts

tation protocols [22], graph encryption schemes supporting approximate shortest distance queries [24] and the protocol for private regular-expression searches on encrypted data [32]. Here we want to highlight that, compared with the GHV scheme, our oGHV scheme opens the door to more efficient real-world privacy-preserving applications. A such example is the private and verifiable delegation of linear algebra, which is always an important research subject in cryptography. Although Mohassel [23] has given the GHV scheme based delegation protocols for some linear algebra problems such as matrix multiplication and matrix inversion, as shown in Table 1, the GHV scheme actually should be excluded from consideration due to the "heavy" decryption performing roughly $\mathcal{O}(m^3)$ computations. Since our oGHV scheme achieves the desirable improvements in terms of the efficiency, it can be a natural match for private delegations of some linear algebra problems and even specific computations related to linear algebra.

2 Preliminaries

Notations. Throughout this paper, we use capital letters (e.g., X, Y) for random variables and probability distributions, standard letters (e.g., x, y) for scalars, and calligraphic letters (e.g., $\mathcal{X}$, $\mathcal{Y}$) for sets. We denote (column) vectors by standard bold letters (e.g., $\mathbf{x}$, $\mathbf{y}$) and matrices by capital bold letters (e.g, $\mathbf{X}$, $\mathbf{Y}$). For a matrix $\mathbf{X}$ over any ring, the ith column of $\mathbf{X}$ is denoted by $\mathbf{x}_i$, the ith element of a vector $\mathbf{x}$ is denoted by x_i, and the ith element of the jth column of $\mathbf{X}$ is denoted by $x_{i,j}$. $N_{\mathbf{X}}$ is a random variable (or probability distribution) on the number of nonzero elements of $\mathbf{X}$. We use $\mathbf{X}^t$ to denote the transpose of $\mathbf{X}$. The ith standard basis vector is denoted by $\mathbf{e}_i$. lg refers to the base 2 logarithm. We use $[x]$ to denote the set $\{1, 2, \cdots, x\}$. $x \xleftarrow{\$} \mathcal{X}$ is considered as sampling an element x from a finite set $\mathcal{X}$ uniformly at random, and $x \leftarrow X$ refers to sampling an element x according to a probability distribution X. For a finite set $\mathcal{X}$, we denote the uniform distribution over $\mathcal{X}$ by $\mathcal{U}(\mathcal{X})$. We denote the binomial distribution with parameters $\rho \in [0, 1]$ and $m \in \mathbb{N}_+$ by $\mathsf{Bin}_{\rho,m}$, where $\Pr[\mathsf{Bin}_{\rho,1} \neq 0] = \rho$

and $\Pr[\mathsf{Bin}_{\rho,1} = 0] = 1 - \rho$. We denote the discrete Gaussian (error) distribution over $\mathbb{Z}_q$ by $\overline{\Psi}_\beta(q)$ that may be generated by sampling $y \leftarrow \frac{1}{\beta}\exp(-\pi(\frac{x}{\beta})^2)$ and outputting $\lfloor q \cdot y \rceil \pmod q$, where $\beta > 0$ and $q \geq 2$. $X \sim D$ denotes that a random variable X follows a probability distribution D. For two distribution ensembles $X \overset{\mathsf{def}}{=} \{X_n\}$ and $Y \overset{\mathsf{def}}{=} \{Y_n\}$ indexed by $n \in \mathbb{N}_+$, $X \overset{s}{\approx} Y$ refers to the statistical indistinguishability between X and Y. $x \pmod q$ is considered as mapping x into the interval $]-\frac{q}{2}, \frac{q}{2}]$. Let $\mathbf{x} = (x'_1, x'_2, \ldots, x'_n) \in \{0,1\}^n$ be the binary representation of x. Then we call $\mathsf{bwt}(x) = \|\mathbf{x}\|_1 = \#\{i \in [n] | x'_i \neq 0\}$ the (Hamming) weight of x. Let t_a, t_m and t_g denote the running time of the (modulo) addition, (modulo) multiplication and discrete Gaussian sampling over the integers, respectively.

We also use the following simplified notations in this paper. Throughout, we denote the security parameter by $n \in \mathbb{N}_+$, and most parameters are functions of n, e.g., $m_1, m_2, m, q = \mathsf{poly}(n)$, $\beta = \frac{1}{\mathsf{poly}(n)}$ and $c = c(n) > 0$, where $\mathsf{poly}(n)$ denotes some polynomial function in n. Thus, we often omit n for the simplified notations. Moreover, overwhelming probability means that the probability is $1 - \psi$, where ψ is negligible in n.

2.1 Cryptographic Problem

We present below a famous hard learning problem, i.e., the Learning with Errors (LWE) problem, which has proven to be a rich and versatile source of many (post-quantum) cryptographic primitives.

Definition 1 (LWE [15, 29]) *Let n, m, q be positive integers, $\mathbf{s} \in \mathbb{Z}_q^n$ be a secret vector, and χ be a probability distribution over $\mathbb{Z}_q$. We denote the L-WE distribution by $L_{\mathbf{s},\chi,q}$ that is the probability distribution over $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$ given by choosing $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{m\times n}$, sampling a vector $\mathbf{x} \leftarrow \chi^m$ and outputting $(\mathbf{A}, \langle \mathbf{A}, \mathbf{s}\rangle + \mathbf{x}) = (\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$.*

The decision LWE problem $\mathsf{dLWE}(n, m, q, \chi)$ is the problem of distinguishing whether a sample $(\mathbf{A}, \mathbf{b})$ is drawn from $L_{\mathbf{s},\chi,q}$ or uniformly at random from $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$. The search LWE problem $\mathsf{sLWE}(n, m, q, \chi)$ is the problem of finding the secret $\mathbf{s}$ from a sample $(\mathbf{A}, \langle \mathbf{A}, \mathbf{s}\rangle + \mathbf{x})$ drawn according to $L_{\mathbf{s},\chi,q}$.

In particular, χ is generally the discrete Gaussian distribution $\overline{\Psi}_\beta(q)$ [21]. For the LWE version defined with $\overline{\Psi}_\beta(q)$, it is known as the "standard form". About the hardness of the standard LWE problem, there have been several results [8, 9, 27, 20, 29]. Specifically, Regev [29] first proved that solving $\mathsf{sLWE}(n, m, q, \beta)$ efficiently is as hard as finding a quantum solution for approximating certain worst-case lattice problems, i.e., the decision version of the Shortest Vector Problem (GAPSVP) and the Shortest Independent Vectors Problem (SIVP). Regev [29] also showed that $\mathsf{dLWE}(n, m, q, \beta)$ can be equivalent to (worst-case) $\mathsf{sLWE}(n, m, q, \beta)$ for a prime modulus $q \in [2, \mathsf{poly}(n)]$, with a loss of up to a $\mathsf{poly}(n) \cdot q$ factor in m. Then, Peikert [27] gave that solving $\mathsf{sLWE}(n, m, q, \beta)$ efficiently is (at least) as hard as approximating GAPSVP (and a GAPSVP variant) in the worst case via a classical (PPT) reduction with similar parameters. Moreover, based on the above Regev′s search-to-decision reduction, Peikert [27] provided a classical foundation for the hardness of $\mathsf{dLWE}(n, m, q, \beta)$. Notice that $\mathbf{s}$ can be sampled from the error distribution (i.e., $\overline{\Psi}_{\beta}(q)^n$) without any loss in security [2]. In what follows, since (post-quantum) cryptographic applications are typically based on $\mathsf{dLWE}(n, m, q, \beta)$, we summarize Regev and Peikert′s results for the decision variant.

Lemma 1 (Theorem 1.1 in [29], Theorem 3.3 in [27]) Let n be a positive integer, $\beta > 0$ and $q \in \mathbb{N}_+$ be a product of co-prime numbers, i.e., $q = \prod q_i$, where $\forall i \in \mathbb{N}_+$ $q_i = \mathsf{poly}(n)$. For $\beta q > 2\sqrt{n}$, if there is an efficient algorithm solving $\mathsf{dLWE}(n, m, q, \beta)$, there is an efficient quantum algorithm running in time $\mathsf{poly}(n)$ to approximate GAPSVP and SIVP on n-dimensional lattices in the worst case to within $\tilde{\mathcal{O}}(\frac{n}{\beta})$ factors, and an efficient classical algorithm running in time $\mathsf{poly}(n)$ to approximate a ζ-to-ζ' GAPSVP variant $\mathsf{GAPSVP}_{\zeta,\zeta'}$ on n-dimensional lattices in the worst case to within $\zeta = \tilde{\mathcal{O}}(q\sqrt{n})$ and $\zeta' = \tilde{\mathcal{O}}(\frac{n}{\beta})$ factors.

2.2 Trapdoor Sampling Algorithms

Here we recall two significant trapdoor generation algorithms for cryptographic lattices, which are inspired by Ajtai′s initial work [1]. The first proposal is the Alwen and Peikert trapdoor generator [3], denoted by $\mathsf{APTrapSamp}$. This randomized algorithm outputs a hard random lattice $\mathbf{A}^t \in \mathbb{Z}_q^{n\times m}$ together with some short orthogonal basis (i.e., trapdoor) $\mathbf{T}^t \in \mathbb{Z}^{m\times m}$ of the lattice $\Lambda_q^{\perp}(\mathbf{A}^t)$, where $m = \Theta(n \lg q)$. The block structures of $\mathbf{A}^t$ and $\mathbf{T}^t$ are shown in Fig. 1(a) (see Appendix A.1), where $\mathbf{A}_1 \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m_1}$ and $m_1 + m_2 = m$. In particular, $\mathsf{APTrapSamp}$ involves two concrete algorithms. Compared with Alwen and Peikert′s first algorithm, the second algorithm, denoted by $\mathsf{APSTrapSamp}$, can be regarded as an optimized algorithm with respect to the lattice dimension and the quality of the trapdoor. Then, $\mathsf{APSTrapSamp}$ is more suitable for efficient cryptographic applications. The second type of trapdoor generator is introduced by Micciancio and Peikert [25], which is the current state of the art in the trapdoor generation. This randomized algorithm, denoted by $\mathsf{MPTrapSamp}$, can output a hard random lattice $\mathbf{A}^t \in \mathbb{Z}_q^{n\times m}$ together with a sufficiently "short" integer matrix $\mathbf{R} \in \mathbb{Z}^{m_1\times m_2}$ as the gadget-based trapdoor (with tag (e.g., $\mathbf{I}$) over $\mathbb{Z}_q^{n\times n}$), where $m = \Theta(n \lg q)$ and $m_1 + m_2 = m$. $\mathsf{MPTrapSamp}$ includes the statistical instantiation, denoted by $\mathsf{MPSTrapSamp}$, and the computational instantiation. In particular, the statistically secure trapdoor construction from $\mathsf{MPSTrapSamp}$ is the better choice of cryptographic applications. Moreover, $\mathsf{MPSTrapSamp}$ may generate a good basis $\mathbf{T}^t$ for $\Lambda^{\perp}(\mathbf{A}^t)$ from knowledge of $\mathbf{R}$, which implies that $\mathsf{MPSTrapSamp}$ can also serve as a "traditional" trapdoor sampling algorithm. The corresponding block structures of $\mathbf{A}^t$ and $\mathbf{T}^t$ are given in Fig. 1(b) (see Appendix A.1), where $\mathbf{A}_1 \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m_1}$. Notice that, since the block structure of $\mathbf{T}^t$ generated by $\mathsf{MPSTrapSamp}$ is similar to that of the trapdoor from $\mathsf{APSTrapSamp}$, we refer to the "traditional" $\mathsf{MPSTrapSamp}$ as the $\mathsf{APTrapSamp}$-type trapdoor sampling algorithm. In what follows, we state some consequences related to $\mathsf{APSTrapSamp}$ and $\mathsf{MPSTrapSamp}$. In Appendix A.1, we present details of component matrices $\mathbf{G} \in \mathbb{Z}^{m_1 \times m_2}(\mathbf{G} \in \mathbb{Z}_q^{n \times m_2}), \mathbf{P} \in \mathbb{Z}^{m_2 \times m_1}, \mathbf{U} \in \mathbb{Z}^{m_2 \times m_2}$ and $\mathbf{R} \in \mathbb{Z}^{m_1 \times m_2}$ generated by $\mathsf{APSTrapSamp}$ and $\mathsf{MPSTrapSamp}$, respectively.

Lemma 2 (Theorem 3.2 in [3], Lemma 5.3 in [25])There are PPT randomized algorithms $\mathsf{APTrapSamp}$ and $\mathsf{MPSTrapSamp}$ that, on input 1^n, $q \geq 2$ and $m = \Theta(n \lg q)$, can generate matrices $\mathbf{A}^t \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T}^t \in \mathbb{Z}^{m \times m}$ such that

– $\mathbf{A}^t$ is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$.
– $\mathbf{T}^t$ is a "small" invertible matrix. In particular, the Euclidean norm of all columns of $\mathbf{T}^t$ from $\mathsf{APSTrapSamp}$ is bounded by $\mathcal{O}(n \lg q)$, where the constant hidden in the $\mathcal{O}(\cdot)$ is at most 20.
– $\mathbf{TA} = \mathbf{0} \pmod q$.

2.3 The Gentry-Halevi-Vaikuntanathan Encryption Scheme

The GHV scheme [15] is a public-key encryption scheme for encrypting matrices over any matrix ring $\mathbb{Z}_p^{m \times m}$, where $p \geq 2$. This scheme employs the idea of the trapdoor function given by Gentry, Peikert and Vaikuntanathan in 2008 [16], where a near-uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is the "public key" and an invertible "small" matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{TA} = \mathbf{0} \pmod q$ is the used trapdoor, to get the public and secret key pair for the encryption and decryption, and specifically runs the $\mathsf{APTrapSamp}$-type sampling algorithm (e.g., $\mathsf{APSTrapSamp}$ and $\mathsf{MPSTrapSamp}$) to output such a key pair $(\mathbf{A}, \mathbf{T})$. The basic construction of the GHV scheme, denoted by GHV, is due to the fact that the trapdoor $\mathbf{T}$ can solve the standard LWE instance relative to $\mathbf{A}$, which implies that security of GHV relies on the hardness of the standard LWE problem $\mathsf{dLWE}(n, m, q, \beta)$ (see Lemma 1). For more details, please refer to Appendix A.2.

2.4 Other Preliminaries

Definition 2 (Density of a Matrix for Matrix Multiplication) *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be matrices over any rings. In a single matrix multiplication* $\mathbf{XY}$ *over a matrix ring, density of* $\mathbf{X}$ (*resp.* $\mathbf{Y}$) *is equal to the number of necessary nonzero elements of* $\mathbf{X}$ (*resp.* $\mathbf{Y}$) *over the ring. These nonzero elements are used in* $\mathbf{XY}$.

Lemma 3 (Fact 1 in [15]) *Let positive integers* $n, q \geq 2$, $\beta > 0$ *and* $g = \omega(\sqrt{\lg n})$. *For* $\mathbf{x} \leftarrow \overline{\Psi}_\beta(q)^n$ *and an arbitrary vector* $\mathbf{y} \in \mathbb{Z}^n$, $|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \beta q g \|\mathbf{y}\|$ *with probability* $1 - \psi$, *where* $\|\mathbf{y}\|$ *is the Euclidean norm of* $\mathbf{y}$, *and* ψ *is negligible in* n.

Theorem 1 (Hoeffding Bound [18]) *Let* $X_1, X_2, \ldots, X_\kappa$, *where* $\kappa \in \mathbb{N}_+$, *be a sequence of independent random variables such that* $\forall i \in [1, \kappa]$ $\Pr[X_i \in [a_i, b_i]] = 1$. *Let* $X = \sum_{i=1}^{\kappa} X_i$. *Then, for any* $\tau > 0$

$$\Pr[|X - \mathbb{E}[X]| \geq \tau] \leq 2 \exp^{-\frac{2\tau^2}{\sum_{i=1}^{\kappa}(b_i - a_i)^2}} .$$

3 Efficiency Analyses of GHV

In this section, we precisely discuss (time and space) efficiency of the original scheme GHV and show why GHV using APSTrapSamp or even MPSTrapSamp is "relatively" inefficient and should be ruled out for some cryptographic applications that only run in time $\mathcal{O}(m^{c'})$, where $c' \in [2, 3[$. In particular, the density of the "special" trapdoor matrix pair ($\mathbf{T}$, $\mathbf{T}^{-1}$) has a direct influence on efficiency of GHV, which means that it should be first explored. For APSTrapSamp and MPSTrapSamp, although some significant parameters related to the output lattice associated with $\mathbf{A}^t$ (and $\mathbf{A}$) and the resulting basis $\mathbf{T}^t$ (and $\mathbf{T}$), e.g., the lattice dimension and the basis quality, have been explored in [3, 25], to the best of our knowledge, our work give the first measure of density of ($\mathbf{T}$, $\mathbf{T}^{-1}$) for matrix multiplication.

3.1 On the Density of Trapdoor Matrix Pair ($\mathbf{T}$, $\mathbf{T}^{-1}$)

We first give the density analysis of the matrix $\mathbf{T}$ generated by APSTrapSamp and MPSTrapSamp (i.e., $N_{\mathbf{T}}$), respectively. Then, we focus on exploring density of the corresponding inverse matrix $\mathbf{T}^{-1}$ over $\mathbb{Z}_p$ for $p \geq 2$ (i.e., $N_{\mathbf{T}^{-1}}$). Interestingly, we obtain $N_{\mathbf{T}}$ and $N_{\mathbf{T}^{-1}}$ based on simple and special decomposition forms of $\mathbf{T}$ and $\mathbf{T}^{-1}$. Notice that, for APSTrapSamp, when the modulus q is a prime, $\mathbf{H}$ can be of the form $[\, q\mathbf{e}_1 \cdots q\mathbf{e}_n \ \hat{\mathbf{H}} \,]$, where $\hat{\mathbf{H}} = \left[\begin{smallmatrix} \tilde{\mathbf{H}} \\ \mathbf{I} \end{smallmatrix}\right] \in \mathbb{Z}_q^{m_1 \times (m_1 - n)}$ is the column reduction form of the kernel of $\mathbf{A}_1$. Since $\mathbf{A}_1 \xleftarrow{\$} \mathbb{Z}_q^{n \times m_1}$ in APSTrapSamp, we present a mild assumption on $\mathbf{H}$ as follows: if q is a prime, $\forall i \in [n]$ and $\forall j \in [n+1, m_1]$ $h_{i,j} \xleftarrow{\$} \mathbb{Z}_q$, which means $\tilde{\mathbf{H}} \xleftarrow{\$} \mathbb{Z}_q^{n \times (m_1 - n)}$.

Lemma 4 *For the trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ generated by APSTrapSamp, it has the decomposition form $\mathbf{T} = \left(\left[\begin{smallmatrix} \mathbf{GU} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right] + \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]\left[\begin{smallmatrix} \mathbf{U} & \mathbf{P} \end{smallmatrix}\right]\right)^t$. Then, under the assumption that $\forall i \in [n]$ and $\forall j \in [n+1, m_1]$ $h_{i,j} \xleftarrow{\$} \mathbb{Z}_q$, where q is a prime, we have $N_{\mathbf{T}} = N_{\mathbf{R}} + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} + m + m_2 + n(w + \mathsf{bwt}(q-1))$, where $N_{\mathbf{R}} \sim \mathsf{Bin}_{\frac{1}{2}, dm_2}$ and $N_{\mathcal{P}_{\tilde{\mathbf{H}}}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2}, n(m_1-n)w}$, where $\mathcal{P}_{\tilde{\mathbf{H}}}$ is the binary representation of $\{h'_{i,j} | i \in [n], j \in [n+1, m_1]\}$.*

Proof. See Appendix A.3 for the proof.

Lemma 5 *Let the modulus q be a large enough prime. Consider that $\mathbf{R}$ is sampled from the distribution over $\{0, \pm 1\}^{m_1 \times m_2}$ that outputs 0 with probability $\frac{1}{2}$ and ± 1 each with probability $\frac{1}{4}$[4]. For the trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ generated by MPSTrapSamp, based on its decomposition form $\mathbf{T} = \left(\left[\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]\left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{P} & \mathbf{U} \end{smallmatrix}\right]\right)^t$, we have $N_{\mathbf{T}} = N_{\mathbf{R}} + N_{\mathbf{P}} + 2m + n(w - 2 + \mathsf{bwt}(q))$, where $N_{\mathbf{R}} \sim \mathsf{Bin}_{\frac{1}{2}, m_1 m_2}$ and $N_{\mathbf{P}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2}, nm_1 w}$.*

[4] We believe that a matrix sampled from the distribution over $\{0, \pm 1\}^{m_1 \times m_2}$ is generally sparser than a matrix from the discrete Gaussian distribution for some $\beta' \geq \eta_\upsilon(\mathbb{Z})$.

Proof. See Appendix A.4 for the proof.

Lemma 6 *For the inverse matrix $\mathbf{T}^{-1} \in \mathbb{Z}_p^{m\times m}$ corresponding to $\mathbf{T}$ generated by* $\mathsf{APSTrapSamp}$*, it is of the form $\left(\begin{bmatrix} \mathbf{U}^{-1}\mathbf{P}\mathbf{H}^{-1} & \mathbf{U}^{-1}(\mathbf{I}-\mathbf{P}\mathbf{H}^{-1}(\mathbf{G}+\mathbf{R})) \\ -\mathbf{H}^{-1} & \mathbf{H}^{-1}(\mathbf{G}+\mathbf{R}) \end{bmatrix}\right)^t$, and can be expressed as $\left(\begin{bmatrix} \mathbf{U}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\left(\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{P} \\ -\mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{H}^{-1} & -\mathbf{H}^{-1}(\mathbf{G}+\mathbf{R}) \end{bmatrix}\right)\right)^t$, where $\mathbf{U}^{-1} = diag(\mathbf{V}_{w_1}^{-1}, \cdots, \mathbf{V}_{w_{m_1}}^{-1}, \mathbf{I})$ and in particular $\forall i \in [w_k]$ the ith column of the $w_k \times w_k$ matrix $\mathbf{V}_{w_k}^{-1}$ (i.e., $\mathbf{v}_i^{-1}$) is $\sum_{j=1}^{i} 2^{i-j}\mathbf{e}_j$, where $k \in [m_1]$. Then, under the assumption that $\forall i \in [n]$ and $\forall j \in [n+1, m_1]$ $h_{i,j} \xleftarrow{\$} \mathbb{Z}_q$, where q is a prime, we have that $N_{\mathbf{T}^{-1}}$ is (at least) $2m+m_1+n\mathsf{bwt}(q-1)+N_{\mathcal{P}_{\tilde{\mathbf{H}}}}+Y_1+Y_2$ with $Y_1 \overset{s}{\approx} \mathsf{Bin}_{\frac{p-1}{p},n(m-n)}$ and $Y_2 \sim \mathsf{Bin}_{\frac{1}{2},(d-n)m_2}$.*

Proof. See Appendix A.5 for the proof.

Lemma 7 *Let the modulus q be a large enough prime. Consider that $\mathbf{R}$ is sampled from the distribution over $\{0, \pm 1\}^{m_1\times m_2}$ that outputs 0 with probability $\frac{1}{2}$ and ± 1 each with probability $\frac{1}{4}$. For the inverse matrix $\mathbf{T}^{-1} \in \mathbb{Z}_p^{m\times m}$ corresponding to $\mathbf{T}$ generated by* $\mathsf{MPSTrapSamp}$*, it has the decomposition form $\mathbf{T}^{-1} = \left(\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}^{-1} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\right)^t$, where $\mathbf{U}^{-1} = diag(\mathbf{V}_w^{-1}, \cdots, \mathbf{V}_w^{-1}, \mathbf{I})$. Then, we have $N_{\mathbf{T}^{-1}} = N_{\mathbf{R}} + N_{\mathbf{P}} + 3m + Y_3$, where $N_{\mathbf{R}} \sim \mathsf{Bin}_{\frac{1}{2},m_1m_2}$, $N_{\mathbf{P}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2},nm_1w}$ and $n(w-1) < Y_3 \leq \frac{n(w^2+3w-2)}{2}$.*

Proof. See Appendix A.6 for the proof.

3.2 Theoretical Efficiency of GHV

Now we analyze the computational cost and space cost of GHV when encrypting matrices over $\mathbb{Z}_p^{m\times m}$. In particular, the cases of employing APSTrapSamp and MPSTrapSamp are discussed, respectively. Using results on the density of trapdoor matrix pair $(\mathbf{T}, \mathbf{T}^{-1})$ in Lemma 4 to 7, we can show accurate estimates of these two costs. Notice that, we present (near-)lower bounds on these two costs of the decryption procedure of GHV.

Theorem 2 *For a plaintext matrix $\mathbf{B} \in \mathbb{Z}_p^{m\times m}$ $(p \geq 2)$ that is encrypted by* GHV *using* APSTrapSamp*,* Enc$(\mathbf{B})$ *takes at most $m^2((n+1)t_m + (n+2)t_a + t_g)$ time to generate a ciphertext matrix $\mathbf{C}$, and* Dec$(\mathbf{C})$ *needs to take at least $2m(\frac{p-1}{p}n(m-n) + (d - \frac{n}{2} - 3\sqrt{\frac{n}{2}})m_2 + (m_1 - n)nw + 4m)(t_m + t_a)$ time (with overwhelming probability) to recover $\mathbf{B}$ from $\mathbf{C}$.*

Proof. See Appendix A.7 for the proof.

Notice that, letting $m_1 = \frac{101}{100}n\lg q$ and $m_2 = \frac{402}{100}n\lg q$, which means $m = \frac{503}{100}n\lg q < \lfloor 8n\lg q \rfloor$, from the consequence on $\mathsf{Dec}(\mathbf{C})$ in Theorem 2, we see that the computational cost of the decryption procedure of GHV employing APSTrapSamp is at least $\frac{2}{5}m^3(t_m + t_a)$ $(\approx \mathcal{O}(m^3))$.

Theorem 3 *For a plaintext matrix $\mathbf{B} \in \mathbb{Z}_p^{m\times m}$ $(p \geq 2)$ that is encrypted by* GHV *using* $\mathsf{MPSTrapSamp}$, $\mathsf{Enc}(\mathbf{B})$ *takes at most $m^2((n+1)t_m + (n+2)t_a + t_g)$ time to generate a ciphertext matrix $\mathbf{C}$, and* $\mathsf{Dec}(\mathbf{C})$ *needs to take at least $2m(m_1(nw+m_2) - \sqrt{2n(m_1(nw+m_2)+1)} + 5m + (2w-3)n)(t_m+t_a)$ time (with overwhelming probability) to recover $\mathbf{B}$ from $\mathbf{C}$.*

Proof. See Appendix A.8 for the proof.

Let us consider $m_1 \approx n\lg q$ and $m_2 = n\lg\lceil q\rceil$, which are used in the "traditional" $\mathsf{MPSTrapSamp}$ construction. According to the consequence on $\mathsf{Dec}(\mathbf{C})$ in Theorem 3, we see that the computational cost of the decryption procedure of GHV employing $\mathsf{MPSTrapSamp}$ is about $m^3(t_m + t_a)$ $(\approx \mathcal{O}(m^3))$.

Theorem 4 *For a plaintext matrix $\mathbf{B} \in \mathbb{Z}_p^{m\times m}$ $(p \geq 2)$ that is encrypted by* GHV *using* $\mathsf{APSTrapSamp}$, $\mathsf{Enc}(\mathbf{B})$ *takes $2nm\lceil\lg q\rceil + m^2\lceil\lg p\rceil$ bits to generate a ciphertext matrix $\mathbf{C}$, and* $\mathsf{Dec}(\mathbf{C})$ *needs to take at least $2m^2\lceil\lg q\rceil + n(m-n)\lceil\lg p\rceil + 2dm_2 + n(m_1-n)w$ bits to recover $\mathbf{B}$ from $\mathbf{C}$.*

Proof. See Appendix A.9 for the proof.

Theorem 5 *For a plaintext matrix $\mathbf{B} \in \mathbb{Z}_p^{m\times m}$ $(p \geq 2)$ that is encrypted by* GHV *using* $\mathsf{MPSTrapSamp}$, $\mathsf{Enc}(\mathbf{B})$ *takes $2nm\lceil\lg q\rceil + m^2\lceil\lg p\rceil$ bits to generate a ciphertext matrix $\mathbf{C}$, and* $\mathsf{Dec}(\mathbf{C})$ *needs to take at least $2m^2\lceil\lg q\rceil + m_1(2m_2+nw)$ bits to recover $\mathbf{B}$ from $\mathbf{C}$.*

Proof. See Appendix A.10 for the proof.

4 Our Optimized GHV-Type Encryption Scheme

The above efficiency analysis confirms that GHV is not suitable for applications (e.g., the private and verifiable delegation of computation) that must use data protection techniques with roughly $\mathcal{O}(m^{c'})$ computational complexity, where $c' \in [2, 3[$ is close to 2. Hence, in this section, we modify the original scheme and are ready to present our optimized variant, denoted by oGHV, for keeping inherent merits of the scheme and making the corresponding running process more efficient, e.g., achieving $\tilde{\mathcal{O}}(nm^2)$ computational overhead. In particular, to make comparisons with the GHV instantiation that employs $\mathsf{APSTrapSamp}$ (see [15]), $\mathsf{APSTrapSamp}$ is still used in our oGHV instantiation. Of course, $\mathsf{MPTrapSamp}$ is also a candidate for oGHV. Notice that the trapdoor $\mathbf{T}^t := \left[\mathbf{T}_1^t\ \mathbf{T}_2^t\right]$, where $\mathbf{T}_1^t := \left[\begin{smallmatrix}(\mathbf{G}+\mathbf{R})\mathbf{U}\\ \mathbf{U}\end{smallmatrix}\right]$ and $\mathbf{T}_2^t := \left[\begin{smallmatrix}\mathbf{RP}-\mathbf{I}\\ \mathbf{P}\end{smallmatrix}\right]$, as adopted throughout the whole section.

4.1 Using a Sparse Matrix to Replace $\mathbf{T}^{-1}$

From Theorem 2, we know that GHV takes roughly $\mathcal{O}(nm^2)$ running time to encrypt an $m \times m$ matrix and uses $\mathcal{O}(m^3)$ time to recover this matrix. In particular, the computational cost of the decryption procedure is evidently larger than that of the encryption procedure. This means that we can focus on optimizing the decryption algorithm and reducing the corresponding cost to make the whole cryptosystem more efficient. Notice that there exist two steps in the decryption algorithm, i.e., $\mathbf{C}' = \mathbf{T}\mathbf{C}\mathbf{T}^t \pmod q$ and $\mathbf{B} = \mathbf{T}^{-1}\mathbf{C}'(\mathbf{T}^t)^{-1} \pmod p$. Specifically, based on the fact that $\mathbf{C}$ is of the form $\mathbf{A}\mathbf{S} + p\mathbf{X} + \mathbf{B} \pmod q$, computing $\mathbf{T}\mathbf{C}\mathbf{T}^t \pmod q$ is an indispensable step, which is used to cancel out $\mathbf{A}\mathbf{S}$. $\mathbf{T}^{-1}\mathbf{C}'(\mathbf{T}^t)^{-1} \pmod p$ can be seen as a "supplement" of $\mathbf{T}\mathbf{C}\mathbf{T}^t \pmod q$. The main purpose of this step is to cancel out $(\mathbf{T}, \mathbf{T}^t)$ and recover $\mathbf{B}$. Although the second step is similar to an additional operation, the corresponding computation is expensive in the decryption procedure and has great influence on the computational cost of the whole cryptosystem.

Thus, let us consider how to reduce the running time of the step $\mathbf{T}^{-1}\mathbf{C}'(\mathbf{T}^t)^{-1} \pmod p$ and improve efficiency of the whole decryption algorithm including the first step. Ideally, we would like to find a sufficiently sparse matrix to replace $\mathbf{T}^{-1}$ and "indirectly" recover $\mathbf{B}$ from $\mathbf{C}'$ by employing some other simple computation. Unfortunately, this is not a computationally feasible operation. However, from the definition of $\mathbf{T}$ in Sect. 2.2 (see Fig. 1(a)), we notice that the $m_2 \times m_2$ invertible component matrix $\mathbf{U}$ is the main part of $\mathbf{T}$ and satisfies $N_{\mathbf{U}} \ll m_2^2 - N_{\mathbf{U}}$. According to Lemma 6, we also know that $\mathbf{U}^{-1} = \mathit{diag}(\mathbf{V}_w^{-1}, \cdots, \mathbf{V}_w^{-1}, \mathbf{I})$ satisfies $N_{\mathbf{U}^{-1}} \ll m_2^2 - N_{\mathbf{U}^{-1}}$. These observations inspire us that we can construct an extremely sparse matrix involving $\mathbf{U}^{-1}$, denoted by $\tilde{\mathbf{T}}$, to decrypt $\mathbf{B}$ from $\mathbf{C}$. Specifically, the original plaintext matrix $\mathbf{B}$ should be first enlarged to $\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{B}\end{smallmatrix}\right]$ by padding zero elements in the encryption algorithm. Notice that, the number of the padded zero elements is far less than that of elements of $\mathbf{B}$. Then, in the decryption algorithm, $\mathbf{C}' = \mathbf{T}\mathbf{C}\mathbf{T}^t \pmod q$ is executed and, after that, $\tilde{\mathbf{T}} = \left[\begin{smallmatrix}\mathbf{U}^{-1}\\ \mathbf{0}\end{smallmatrix}\right]$ is used to recover $\mathbf{B}$ by running $\tilde{\mathbf{T}}^t\mathbf{C}'\tilde{\mathbf{T}} \pmod p$. As described above, our optimization for the construction of GHV is very simple but can surprisingly achieve the desired efficiency improvement. In Sect. 4.3 and 4.5, we give the detailed correctness analysis for the optimized scheme oGHV and also present the efficiency exploration of oGHV, which supports our optimization. Moreover, here we highlight another merit of using $\tilde{\mathbf{T}}$ instead of $\mathbf{T}^{-1}$. That is, it is unnecessary to store $\tilde{\mathbf{T}}$ for multiple encryptions. From Lemma 6, we have that $\forall i \in [m_1]$ $\mathbf{V}_{w_i}^{-1}$ can be seen as a deterministically-constructed matrix, which means that $\tilde{\mathbf{T}}$ is also a deterministically-constructed matrix that is easily reconstructed for multiple encryptions, while some components of $\mathbf{T}^{-1}$ must be stored for each GHV encryption. In Sect. 4.4, we introduce a concrete algorithm (i.e., Algorithm 3) to show how to efficiently run the multiplication between $\tilde{\mathbf{T}}$ (resp. $\tilde{\mathbf{T}}^t$) and $\mathbf{C}'$ without using $\tilde{\mathbf{T}}$ (resp. $\tilde{\mathbf{T}}^t$).

4.2 Generic Construction of oGHV

Now we give details on the generic construction of the optimized GHV-type HE scheme oGHV with parameters n, m_1, m_2, m, q, β for plaintext matrices over $\mathbb{Z}_p$ with any integer $p \geq 2$, where q is an odd prime, and β is a Gussian error parameter. In particular, oGHV including a triple of PPT algorithms ($\mathsf{oKeyGen}$, oEnc, oDec) is described below.

- $\mathsf{oKeyGen}(1^n) \to (\mathbf{A}, (\mathbf{T}, \tilde{\mathbf{T}}))$: Run $\mathsf{APTrapSamp}$-type trapdoor sampling algorithm to get a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ and its trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{m\times m}$ such that $\mathbf{TA} = \mathbf{0} \pmod q$. Generate a matrix $\tilde{\mathbf{T}} = \left[\begin{smallmatrix}\mathbf{U}^{-1}\\ \mathbf{0}\end{smallmatrix}\right] \in \mathbb{Z}^{m\times m_2}$, where $\mathbf{U}^{-1} \in \mathbb{Z}^{m_2\times m_2}$ is defined in Lemma 6. Output $(\mathbf{A}, (\mathbf{T}, \tilde{\mathbf{T}}))$ as the public and secret key pair.
- $\mathsf{oEnc}_{\mathbf{A}}(\mathbf{B}) \to \mathbf{C}$: Given a plaintext matrix $\mathbf{B} \in \mathbb{Z}_p^{m_2\times m_2}$, build an $m\times m$ matrix $\mathbf{B}' = \left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{B}\end{smallmatrix}\right]$ using three small zero matrices of respective size $m_1 \times m_1$, $m_1 \times m_2$ and $m_2 \times m_1$, where $m_1 + m_2 = m$. Choose $\mathbf{S} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$ and $\mathbf{X} \leftarrow \overline{\Psi}_\beta(q)^{m\times m}$ [5], and generate a ciphertext matrix $\mathbf{C} \in \mathbb{Z}_q^{m\times m}$ as
$$\mathbf{C} = \mathbf{AS} + p\mathbf{X} + \mathbf{B}' \pmod q.$$
- $\mathsf{oDec}_{(\mathbf{T},\tilde{\mathbf{T}})}(\mathbf{C}) \to \mathbf{B}$: Given the ciphertext matrix $\mathbf{C}$, run $\mathbf{C}' = \mathbf{TCT}^t \pmod q$ $= \mathbf{T}(p\mathbf{X} + \mathbf{B}')\mathbf{T}^t \pmod q$ and output $\mathbf{B} = \tilde{\mathbf{T}}^t\mathbf{C}'\tilde{\mathbf{T}} \pmod p$.

In the above construction, if $\mathsf{APSTrapSamp}$ is employed by $\mathsf{oKeyGen}$, from Sect. 2.2 (see Appendix A.1), we know that m_1 can be equal to $(1+\delta)n\lg q$ and $m_2 \geq (4+2\delta)n\lg q$, this implies that m_1 and m_2 can satisfy $m_2 \gg m_1$. Then, most of elements of $\mathbf{B}'$ come from $\mathbf{B}$, and we have, in some sense, "$\mathsf{oEnc}_{\mathbf{A}}(\mathbf{B}) \approx \mathsf{Enc}_{\mathbf{A}}(\mathbf{B})$", where Enc is the encryption algorithm of GHV. About concrete instantiations of the parameters m_1, m_2, m, q and β, which are used to guarantee that oGHV holds correctness, security and homomorphism, please refer to Sect. 4.3. In particular, according to properties of the proposed generic construction, the prime q related to the key and ciphertext sizes can be set to be smaller than that used for GHV. Moreover, some detailed optimizations based on the generic construction are presented in Sect. 4.4, which further reduce the computational cost and memory cost of oGHV and guarantee that the smallest key pair is employed. Notice that, similar to that in GHV, the post-multiplication by $\mathbf{T}^t$ and $\tilde{\mathbf{T}}$ on decryption in oGHV is unnecessary. This means that oDec simply runs $\tilde{\mathbf{T}}^t(\mathbf{TC} \pmod q) \pmod p$ for obtaining $\mathbf{B}$. The post-multiplication can be employed to decrypt product ciphertexts (see Sect. 4.3).

4.3 Homomorphic Operations and Concrete Parameters

Our optimized scheme oGHV enjoys the same homomorphic properties as GHV holds. Specifically, oGHV also supports addition and multiplication homomorphism. In particular, for two ciphertext matrices $\mathbf{C}_1 = \mathbf{AS}_1 + p\mathbf{X}_1 + \mathbf{B}'_1 \pmod q$ and $\mathbf{C}_2 = \mathbf{AS}_2 + p\mathbf{X}_2 + \mathbf{B}'_2 \pmod q$ corresponding to two plaintext matrices $\mathbf{B}_1$ and $\mathbf{B}_2$, considering the sum ciphertext $\mathbf{C} = \mathbf{C}_1 + \mathbf{C}_2 \pmod q$, we have
$$\mathbf{C} = \mathbf{C}_1 + \mathbf{C}_2 \pmod q = \mathbf{A}\underbrace{(\mathbf{S}_1 + \mathbf{S}_2)}_{\mathbf{S}} + p\underbrace{(\mathbf{X}_1 + \mathbf{X}_2)}_{\mathbf{X}} + \underbrace{\mathbf{B}'_1 + \mathbf{B}'_2}_{\mathbf{B}'} \pmod q.$$

[5] Clearly, the state-of-the-art discrete Gaussian sampling algorithms over the integers (e.g., [26]) can be considered as candidates used in oGHV to replace the sampling method proposed by Gentry et al. [15]. What is important is that the corresponding parameter setting needs to ensure that oGHV still holds the desired correctness, security and homomorphism.

It is easy to see that $\mathbf{C}_1 + \mathbf{C}_2 \pmod q$ can be decrypted to $\mathbf{B}_1 + \mathbf{B}_2 \pmod p$ if values of all the elements of $\mathbf{T}(p\mathbf{X} + \mathbf{B}')\mathbf{T}^t$ are smaller than $\frac{q}{2}$, where $\mathbf{B}' = \left[\begin{smallmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_1+\mathbf{B}_2 \end{smallmatrix}\right]$. Moreover, considering the product ciphertext $\mathbf{C} = \mathbf{C}_1\mathbf{C}_2^t \pmod q$, we have

$$\begin{aligned} \mathbf{C} &= \mathbf{C}_1\mathbf{C}_2^t \pmod q \\ &= \mathbf{A}\underbrace{\left(\mathbf{S}_1\mathbf{C}_2^t\right)}_{\mathbf{S}} + p\underbrace{\left(\mathbf{X}_1\left(p\mathbf{X}_2 + \mathbf{B}_2'\right) + \mathbf{B}_1'\mathbf{X}_2^t\right)}_{\mathbf{X}} + \underbrace{\mathbf{B}_1'(\mathbf{B}_2')^t}_{\mathbf{B}'} + \underbrace{\left(p\mathbf{X}_1 + \mathbf{B}_1'\right)\mathbf{S}_2^t}_{\tilde{\mathbf{S}}}\mathbf{A}^t \pmod q. \end{aligned}$$

This naturally implies that $\mathbf{C}_1\mathbf{C}_2^t \pmod q$ can be decrypted to $\mathbf{B}_1\mathbf{B}_2^t \pmod p$ when values of all the elements of $\mathbf{T}(p\mathbf{X} + \mathbf{B}')\mathbf{T}^t$ are smaller than $\frac{q}{2}$, where $\mathbf{B}' = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_1\mathbf{B}_2^t \end{bmatrix}$, as discussed above. In what follows, we present our answer on the parameter setting (for q, m_1, m_2, m and β), which guarantees the feasibility of the homomorphic operations.

Notably, according to the above analysis on the additive homomorphism of oGHV, we know that, similar to the case on decryption of the normal ciphertext, the post-multiplication by $\mathbf{T}^t$ is not required for decrypting a sum ciphertext $\mathbf{C} = \sum_{i=1}^{n^c}(\mathbf{A}\mathbf{S}_i + p\mathbf{X}_i + \mathbf{B}_i') \pmod q$, where $c > 0$. Moreover, compared with GHV, of which the correctness of decryption must rely on a condition that each element of $\mathbf{T}(p\mathbf{X}+\mathbf{B})\mathbf{T}^t$ is bounded by $\frac{q}{2}$, we want to show that the correctness of decryption of oGHV is able to depend on a more relaxed condition, resulting in the smaller parameters q and m that we can set. Specifically, consider that $\mathbf{C}' = \begin{bmatrix} \mathbf{C}_1' & \mathbf{C}_2' \\ \mathbf{C}_3' & \mathbf{C}_4' \end{bmatrix}$, where block matrices $\mathbf{C}_1' = \mathbf{T}_1\mathbf{C}\mathbf{T}_1^t \pmod q$, $\mathbf{C}_2' = \mathbf{T}_1\mathbf{C}\mathbf{T}_2^t \pmod q$, $\mathbf{C}_3' = \mathbf{T}_2\mathbf{C}\mathbf{T}_1^t \pmod q$ and $\mathbf{C}_4' = \mathbf{T}_2\mathbf{C}\mathbf{T}_2^t \pmod q$ are respective sizes $m_2 \times m_2$, $m_2 \times m_1$, $m_1 \times m_2$ and $m_1 \times m_1$, we have $\tilde{\mathbf{T}}^t\mathbf{C}'\tilde{\mathbf{T}} \pmod p = (\mathbf{U}^{-1})^t\mathbf{C}_1'\mathbf{U}^{-1} \pmod p$. This means that the final result can be recovered if the absolute value of each element in $\mathbf{T}_1(p\mathbf{X} + \mathbf{B}')\mathbf{T}_1^t$ (instead of $\mathbf{T}(p\mathbf{X} + \mathbf{B}')\mathbf{T}^t$) is bounded by $\frac{q}{2}$. Then, from the relaxed condition, we first set the parameters that simply ensure that oGHV is able to support n^c additions. After that, we establish the concrete parameters that guarantee that oGHV also holds the one-multiplication homomorphism.

Theorem 6 *Consider that APSTrapSamp is employed by oGHV. For the fixed parameters n and $c > 0$, let q, m_1, m_2, m, β be set as*

$$\begin{aligned} &q > 40n^{c+1}p\lg n, \\ &m = m_1 + m_2 \geq \lceil \frac{101}{100}n\lg q\rceil + \frac{201}{50}n\lg q, \textit{where } m_1 = \lceil\frac{101}{100}n\lg q\rceil \textit{ and } m_2 \geq \frac{201}{50}n\lg q, \\ &\beta = \frac{1}{5n^c p\sqrt{m_1\lg n}}. \end{aligned}$$

Then, oGHV with parameters n, m_1, m_2, m, q, β supports n^c homomorphic addition operations over the matrix ring $\mathbb{Z}_p^{m\times m}$ (and $\mathbb{Z}_p^{m_2\times m_2}$).

Proof. See Appendix A.11 for the proof.

Theorem 7 *Consider that APSTrapSamp is employed by oGHV. For the fixed parameters n and $c > 0$, let q, m_1, m_2, m, β be set as*

$$q > 2^{13} n^{3+3c} p^2 \lg^3 n,$$

$$m = m_1 + m_2 \geq \lceil \frac{101}{100} n \lg q \rceil + \frac{201}{50} n \lg q, \textit{where } m_1 = \lceil \frac{101}{100} n \lg q \rceil \textit{ and } m_2 \geq \frac{201}{50} n \lg q,$$

$$\beta = \frac{1}{2n^{\frac{3c}{2}} p \sqrt{m m_1 q \lg n}}.$$

Then, oGHV with parameters n, m_1, m_2, m, q, β supports n^c homomorphic addition operations and one homomorphic multiplication operation over the matrix ring $\mathbb{Z}_p^{m \times m}$ (and $\mathbb{Z}_p^{m_2 \times m_2}$).

Proof. See Appendix A.12 for the proof.

4.4 Computational Optimizations

Generally speaking, the matrix multiplication is a costly operation for cryptographic primitives related to the matrix. Here, according to concrete constructions of $\mathbf{T}^t$ and $\tilde{\mathbf{T}}$ from APSTrapSamp and the generic construction of the cryptosystem presented in Appendix A.1 and Sect. 4.2, some practical optimizations on speeding up the matrix multiplication used in oGHV and further improving efficiency of oGHV are given[6].

Algorithm 1: Ternary-Integer Matrix Product

```
Input: $\mathbf{X} \in \{0, \pm 1\}^{m_3 \times m_4}$, $\mathbf{Y} \in \mathbb{Z}_q^{m_4 \times m_5}$, $m_3$, $m_4$, and $m_5$
Output: $\mathbf{Z} = \mathbf{XY} \in \mathbb{Z}_q^{m_3 \times m_5}$
$\mathbf{Z} = \{0\}^{m_3 \times m_5}$;
for $i \in [m_3]$ do
    for $j \in [m_4]$ do
        for $k \in [m_5]$ do
            if $x_{i,j} == 0$ then
                $z_{i,k} += 0$;
            else if $x_{i,j} == 1$ then
                $z_{i,k} += y_{j,k}$;
            else if $x_{i,j} == -1$ then
                $z_{i,k} -= y_{j,k}$;
            end
        end
    end
end
return $\mathbf{Z}$;
```

Accelerating the Multiplication by a Ternary Matrix. Our idea is that, if a ternary matrix is involved in the matrix multiplication, the corresponding element multiplications are eliminated by running selections and additions. More concretely, a product can be obtained based on Algorithm 1.

[6] The proposed optimizations are not only focus on $\mathbf{T}^t$ and $\tilde{\mathbf{T}}$ from APSTrapSamp. Actually, some extremely similar optimizations can be developed for any APTrapSamp-type trapdoor sampling algorithm (e.g., MPTrapSamp).

Decomposing the Multiplication by $\mathbf{T}^t$ and $\mathbf{T}$. According to Algorithm 1, we can get a method to replace the multiplications in $\mathbf{TCT}^t$ by selections and additions. In particular, our technique is based on the decomposition form of $\mathbf{T}^t$ (resp. $\mathbf{T}$) in Lemma 4. Specifically, for $\left[\begin{smallmatrix}\mathbf{GU} & -\mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right]$ (resp. $\left[\begin{smallmatrix}\mathbf{GU} & -\mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right]^t$), there is (at most) a 1 in each column of $\mathbf{GU}$, and others are zero elements. Then, (at most) a 1 or -1 is in each column of $\left[\begin{smallmatrix}\mathbf{GU} & -\mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right]$, which means that the product of $\left[\begin{smallmatrix}\mathbf{GU} & -\mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right]$ (resp. $\left[\begin{smallmatrix}\mathbf{GU} & -\mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right]^t$) and some matrix can be achieved by simply employing selections shown in Algorithm 1. For $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]$ (resp. $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]^t$), since values of all elements are from $\{0, \pm 1\}$, Algorithm 1 can be directly used to obtain the product of $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]$ (resp. $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]^t$) and some matrix. For $[\,\mathbf{U}\ \mathbf{P}\,]$ (resp. $[\,\mathbf{U}\ \mathbf{P}\,]^t$), values of elements are from $\{-2, 0, 1\}$. Then, a modified Algorithm 1, where multiplying by -2 is replaced by two additions, is suitable for generating the product of $[\,\mathbf{U}\ \mathbf{P}\,]$ (resp. $[\,\mathbf{U}\ \mathbf{P}\,]^t$) and some matrix. In Algorithm 2, how to run the multiplication by $\mathbf{T}$ and $\mathbf{T}^t$ is shown. Notice that, as discussed in Sect. 4.3, for $\mathbf{C}' = \left[\begin{smallmatrix}\mathbf{C}'_1 & \mathbf{C}'_2\\ \mathbf{C}'_3 & \mathbf{C}'_4\end{smallmatrix}\right] = \mathbf{TCT}^t$, only $\mathbf{C}'_1 = \mathbf{T}_1\mathbf{C}\mathbf{T}_1^t$ is the required matrix corresponding to the final result. According to the decomposition form in Lemma 4, this means that Algorithm 2 simply needs to involve $\mathbf{T}_1^t = \left[\begin{smallmatrix}\mathbf{GU}\\ \mathbf{0}\end{smallmatrix}\right] + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{U}$, where $\mathbf{R}$ can be regarded as the only "secret" matrix.

Simplifying the Multiplication by $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{T}}^t$. For $\tilde{\mathbf{T}} = \left[\begin{smallmatrix}\mathbf{U}^{-1}\\ \mathbf{0}\end{smallmatrix}\right]$, where $\mathbf{U}^{-1} = diag(\mathbf{V}_w^{-1}, \cdots, \mathbf{V}_w^{-1}, \mathbf{I})$, we have $\forall i \in [w]$ $\mathbf{v}_i^{-1} = \sum_{j=1}^{i} 2^{i-j}\mathbf{e}_j$, which implies that $\mathbf{v}_{i'+1}^{-1} = \mathbf{e}_{i'+1} + 2\mathbf{v}_{i'}^{-1}$, where $i' \in [w-1]$. Based on this fact, for the case of multiplying some matrix (e.g., $\mathbf{C}'$) with $\tilde{\mathbf{T}}$ (resp. $\tilde{\mathbf{T}}^t$), elements of the $(wj+i+1)$th column (resp. row) of the corresponding product can be generated from elements of the $(wj+i)$th column (resp. row) of the product by running $2m$ additions, where $i \in [w-1]$ and $j \in [0, n-1]$. Consider that $\mathbf{C}' = \left[\begin{smallmatrix}\mathbf{C}'_1 & \mathbf{C}'_2\\ \mathbf{C}'_3 & \mathbf{C}'_4\end{smallmatrix}\right]$. Since the concrete multiplications can focus on $\mathbf{C}'_1$, the "whole" product of $(\mathbf{U}^{-1})^t\mathbf{C}'_1\mathbf{U}^{-1} = \tilde{\mathbf{T}}^t\mathbf{C}'\tilde{\mathbf{T}}$ is computed as shown in Algorithm 3. Notice that Algorithm 3 does not need any additional memory except the memory for storing $\mathbf{C}'_1$.

4.5 Property Analysis

In this section, we present the analyses on correctness, security and efficiency of the optimized encryption scheme oGHV, respectively.

Theorem 8 *For a plaintext matrix $\mathbf{B} \in \mathbb{Z}_p^{m_2\times m_2}$, oGHV with parameters n, m_1, m_2, m, q, β that we can establish has correct encryption and decryption.*

Proof. See Appendix A.13 for the proof.

Theorem 9 *If there is a distinguishing algorithm with advantage ϵ against the IND-CPA security of oGHV with parameters n, m_1, m_2, m, q, and β, then there must be a distinguisher against dLWE(n, m, q, β) with roughly the same running time and advantage (at most) $\frac{\epsilon}{2m}$.*

Algorithm 2: Multiplication by $\mathbf{T}^t$ and $\mathbf{T}$

```
**Input**: $\mathbf{C} \in \mathbb{Z}_q^{m \times m}$, $\mathbf{R} \in \{0, \pm 1\}^{m_1 \times m}$, $n$, $w$, $m_1$, $m_2$, and $m$
**Output**: $\mathbf{C}_1' \in \mathbb{Z}_q^{m_2 \times m_2}$
$\hat{\mathbf{C}} = \mathbf{C}$;
for $i \in [m]$ do                                   /* Running the multiplication by $\mathbf{T}_1^t$ */
    for $j \in [m_2]$ do
        $\hat{c}_{i,j} = \hat{c}_{i,(j+m_1)}$;
        for $k \in [m_1]$ do                         /* Invoking Algorithm 1 */
            $\hat{c}_{i,j} = \hat{c}_{i,j} + r_{k,j}\hat{c}_{i,k}$;
        end
    end
    for $j \in [0, n-1]$ do
        for $k \in [w-1]$ do
            $\hat{c}_{i,(wj+k+1)} = \hat{c}_{i,(wj+k+1)} - (\hat{c}_{i,(wj+k)} + \hat{c}_{i,(wj+k)})$;
        end
        $\hat{c}_{i,(wj+1)} = c_{i,(j+1)} + \hat{c}_{i,(wj+1)}$;
    end
end
for $i \in [m_2]$ do                                 /* Running the multiplication by $\mathbf{T}_1$ */
    for $j \in [n]$ do
        $\tilde{c}_{j,i} = \hat{c}_{j,i}$;
    end
    for $j \in [m_2]$ do
        $\hat{c}_{j,i} = \hat{c}_{(j+m_1),i}$;
        for $k \in [m_1]$ do                         /* Invoking Algorithm 1 */
            $\hat{c}_{j,i} = \hat{c}_{j,i} + r_{k,j}\hat{c}_{k,i}$;
        end
    end
    for $j \in [0, n-1]$ do
        for $k \in [w-1]$ do
            $\hat{c}_{(wj+k+1),i} = \hat{c}_{(wj+k+1),i} - (\hat{c}_{(wj+k),i} + \hat{c}_{(wj+k),i})$;
        end
        $\hat{c}_{(wj+1),i} = \tilde{c}_{(j+1),i} + \hat{c}_{(wj+1),i}$;
    end
end
$\mathbf{C}_1'$ = the top-left $m_2 \times m_2$ block of $\hat{\mathbf{C}}$;
return $\mathbf{C}_1'$;
```

Algorithm 3: Multiplication by $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{T}}^t$

```
**Input**: $\mathbf{C}_1' \in \mathbb{Z}_q^{m_2 \times m_2}$, $n$, $w$, and $m_2$
**Output**: $\mathbf{B} = (\mathbf{U}^{-1})^t \mathbf{C}_1' \mathbf{U}^{-1} \in \mathbb{Z}_p^{m_2 \times m_2}$
for $i \in [w-1]$ do
    for $j \in [0, n-1]$ do
        for $k \in [m_2]$ do
            $c'_{k,(wj+i+1)} = c'_{k,(wj+i+1)} + c'_{k,(wj+i)} + c'_{k,(wj+i)}$;
            $c'_{(wj+i+1),k} = c'_{(wj+i+1),k} + c'_{(wj+i),k} + c'_{(wj+i),k}$;
        end
    end
end
return $\mathbf{B} = \mathbf{C}_1'$;
```

Proof. The security proof follows directly from the proof of IND-CPA security for GHV (see Theorem 2 in [15]).

Theorem 10 *For a plaintext matrix $\mathbf{B} \in \mathbb{Z}_p^{m_2 \times m_2}$ that is encrypted by the optimized scheme oGHV using APSTrapSamp, oEnc$(\mathbf{B})$ takes (at most) $m^2(n+1)(t_a + t_m) + m^2 t_g + m_2^2 t_a$ time and $2nm\lceil \lg q \rceil + m_2^2 \lceil \lg p \rceil$ bits to generate a ciphertext matrix $\mathbf{C}$, and oDec$(\mathbf{C})$ needs to take (at most) $((\frac{1}{2}d + \sqrt{\frac{n}{2}} + 2)(m + m_2) + 4n(w - 1))m_2 t_a$ time and $(m^2 + m_2^2)\lceil \lg q \rceil + 2dm_2$ bits to recover $\mathbf{B}$.*

Proof. See Appendix A.14 for the proof.

5 Conclusions

In this paper, we have proposed an optimized GHV-type asymmetric HE scheme, which is more efficient than the original GHV scheme. In particular, it provides a much faster decryption algorithm, and the computational complexity of the decryption is decreased from $\mathcal{O}(m^3)$ to $\tilde{\mathcal{O}}(nm^2)$. As the same as the GHV scheme, security of our new GHV-type scheme is based on the standard LWE problem, and our scheme also supports matrix encryption. We have compared the performance of our scheme with two LWE-based FHE schemes, which support matrix operations, and the comparison result indicates that our scheme is more efficient. We also have discussed the options of using APSTrapSamp or MPSTrapSamp in the GHV scheme, and our optimizations can benefit both of these two options.

Although we have given the optimized GHV-type HE scheme, from the perspective of implementation, how to make this proposal more practical is an interesting open problem.

Acknowledgements. The authors would like to thank the anonymous reviewers for providing their valuable comments. This work was supported in part by the National Natural Science Foundation of China (No. 61302161, No. 61972269, No. 62032005), in part by the Doctoral Fund, Ministry of Education, China (No. 20130181120076), and in part by the European Union′ s Horizon 2020 research and innovation program under grant agreement No. 952697 (ASSURED) and grant agreement No. 101019645 (SECANT).

References

1. Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.), Automata, Languages and Programming–ICALP 1999, pp. 1–9. Springer, Heidelberg (1999).
2. Applebaum, B., Cash, D., Peikert, C., and Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.), Advances in Cryptology–CRYPTO 2009, pp. 595–618. Springer, Heidelberg (2009).
3. Alwen, J., and Peikert, C.: Generating shorter bases for hard random lattices. Theory of Computing Systems, 48(3), pp. 535–553 (2011).

4. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.), Advances in Cryptology–CRYPTO 2012, pp. 868–886. Springer, Heidelberg (2012).
5. Brakerski, Z., Döttling, N., Garg, S., and Malavolta, G.: Candidate iO from homomorphic encryption schemes. In: Canteaut, A., Ishai, Y. (eds), Advances in Cryptology–EUROCRYPT 2020, pp. 79–109. Springer, Heidelberg (2020).
6. Boneh, D., Goh, E.J., and Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed), Theory of Cryptography–TCC 2005, pp. 325–341. Springer, Heidelberg (2005).
7. Boyle, E., Ishai, Y., and Polychroniadou, A.: Limits of practical sublinear secure computation. In: Shacham, H., Boldyreva, A. (eds), Advances in Cryptology–CRYPTO 2018, pp. 302–332. Springer, Cham (2018).
8. Brakerski, Z., Kirshanova, E., Stehlé, D., and Wen, W.: Learning with errors and extrapolated dihedral cosets. In: Abdalla, M., Dahab, R. (eds), Public-Key Cryptography–PKC 2018, pp. 702–727. Springer, Cham (2018).
9. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., and Stehlé, D.: Classical hardness of learning with errors. In: STOC, pp. 575–584. ACM Press (2013).
10. Brakerski, Z., and Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM Journal on Computing, 43(2) pp. 831–871 (2014).
11. Clear, M., and McGoldrick, C.: Additively homomorphic IBE from higher residuosity. In: Lin, D., Sako, K. (eds), Public-Key Cryptography–PKC 2019, pp. 496–515. Springer, Cham (2019).
12. Damgård, I., and Jurik, M.: A generalisation, a simplification and some applications of Paillier′s probabilistic public-key system. In: Kim, K. (ed), Public Key Cryptography–PKC 2001, pp. 119–136. Springer, Heidelberg (2001).
13. Freedman, M.J., Nissim, K., and Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds), Advances in Cryptology–EUROCRYPT 2004, pp. 1–19. Springer, Heidelberg (2004).
14. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178, ACM Press (2009).
15. Gentry, C., Halevi, S., and Vaikuntanathan, V.: A simple BGN-type cryptosystem from LWE. In: Gilbert, H. (ed.), Advances in Cryptology–EUROCRYPT 2010, pp. 506–522. Springer, Heidelberg (2010).
16. Gentry, C., Peikert, C., and Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206, ACM Press (2008).
17. Gentry, C., Sahai, A., and Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.), Advances in Cryptology–CRYPTO 2013, pp. 75–92. Springer, Heidelberg (2013).
18. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58(301), pp. 13–30 (1963).
19. Hiromasa, R., Abe, M., and Okamoto, T.: Packing messages and optimizing bootstrapping in GSW-FHE. In: Katz, J. (ed), Public-Key Cryptography–PKC 2015, pp. 699–715. Springer, Heidelberg (2015).
20. Lyubashevsky, V., and Micciancio, D.: On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In: Halevi, S. (ed.), Advances in Cryptology–CRYPTO 2009, pp. 577–594. Springer, Heidelberg (2009).
21. Lindner, R., and Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.), Topics in Cryptology–CT-RSA 2011, pp. 319–339. Springer, Heidelberg (2011).

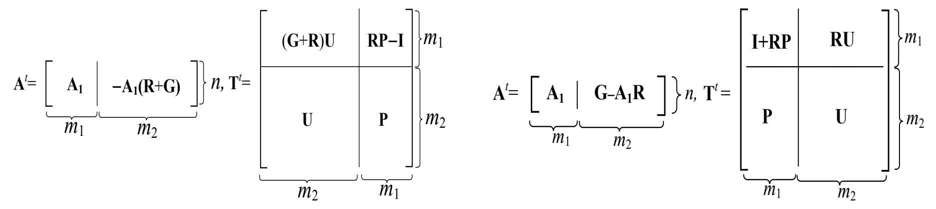
22. López-Alt, A., Tromer, E., and Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC, pp. 1219–1234, ACM Press (2012).
23. Mohassel, P.: Efficient and secure delegation of linear algebra. Cryptology ePrint Archive, Report 2011/605 (2011). https://eprint.iacr.org/2011/605
24. Meng, X., Kamara, S., Nissim, K., and Kollios, G.: GRECS: Graph encryption for approximate shortest distance queries. In: CCS, pp. 504–517, ACM Press (2015).
25. Micciancio, D., and Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. Cryptology ePrint Archive, Report 2011/501 (2011). https://eprint.iacr.org/2011/501
26. Micciancio, D., and Walter, M.: Gaussian sampling over the integers: efficient, generic, constant-time. In: Katz, J., Shacham, H. (eds), Advances in Cryptology–CRYPTO 2017, pp. 455–485. Springer, Cham (2017).
27. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC, pp. 333–342, ACM Press (2009).
28. Pereira, H.V.L. Efficient AGCD-based homomorphic encryption for matrix and vector arithmetic. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds), Applied Cryptography and Network Security–ACNS 2020, pp. 110–129. Springer, Cham (2020).
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 56(6) pp. 1–40 (2009).
30. Rivest, R.L., Adleman, L., and Dertouzos, M.L.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, pp. 169–180. Academic Press, London (1978).
31. van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.), Advances in Cryptology–EUROCRYPT 2010, pp. 24–43. Springer, Heidelberg (2010).
32. Wei, L., and Reiter, M.K.: Toward practical encrypted email that supports private, regular-expression searches. International Journal of Information Security, 14, pp. 397–416 (2015).
33. Wang, B., Wang, X., Xue, R., and Huang, X.: Matrix FHE and its application in optimizing bootstrapping. The Computer Journal, 61(12), pp. 1845–1861 (2018).

A Appendices

A.1 Details of Component Matrices

APSTrapSamp. Given a fixed constant $\delta > 0$, positive integers n, q, let $m_1 \geq d = (1+\delta)n \lg q$ and $m_2 \geq (4+2\delta)n \lg q$. Let $\mathbf{H} \in \mathbb{Z}^{m_1 \times m_1}$ denote the Hermite normal form of a lattice $\Lambda^{\perp}(\mathbf{A}_1)$. The component matrices $\mathbf{G}, \mathbf{P}, \mathbf{U}$ and $\mathbf{R}$ from the PPT algorithm are defined as follows:

– $\mathbf{G}$: Write $\mathbf{G} = [\,\mathbf{G}^{(1)}\ \mathbf{G}^{(2)}\ \cdots\ \mathbf{G}^{(m_1)}\ \mathbf{M}\ \mathbf{0}\,]$, where $\forall i \in [m_1]$ the block matrix $\mathbf{G}^{(i)}$ has $w_i = \lceil \lg h_{i,i} \rceil < 1 + \lg h_{i,i}$ column vectors, where the diagonal element $h_{i,i}$ of $\mathbf{H}$ is at least 1, and $\forall j \in [w_i]$ the jth column $\mathbf{g}_j^{(i)}$ of $\mathbf{G}^{(i)}$ satisfies $\mathbf{g}_j^{(i)} = 2^{j-1}\mathbf{e}_i$. Notice that, if q is a prime, all the values of $h_{i,i}$ that are greater than 1 are q, and the number of this type of $h_{i,i}$ is at most n. Then, $\sum_{i\in[m_1]} w_i \leq n\lceil \lg q \rceil$. Moreover, the special block matrix $\mathbf{M}$ has w column vectors, where w is the



(a) APSTrapSamp (b) MPSTrapSamp

Fig. 1. Block structures of $\mathbf{A}^t$ and $\mathbf{T}^t$.

largest power of 2 in the range $[d, m_2 - 2n \lg q] \supseteq [d, 2d]$. The top d rows of $\mathbf{M}$ are different rows of a square Hadamard matrix of size $w \times w$, and other elements of $\mathbf{M}$ are zero.

– $\mathbf{P}$: Write $\mathbf{P}^t = \left[\, (\mathbf{P}^{(1)})^t \; (\mathbf{P}^{(2)})^t \cdots (\mathbf{P}^{(m_1)})^t \; \mathbf{0} \; \mathbf{0} \,\right]$, where $\forall i \in [m_1]$ the block matrix $\mathbf{P}^{(i)} \in \mathbb{Z}^{w_i \times m_1}$. Let $\mathbf{H}' = \mathbf{H} - \mathbf{I}$. Then, $\forall i, j \in [m_1]$, the jth column $\mathbf{p}_j^{(i)}$ of $\mathbf{P}^{(i)}$ contains the binary representation of $h'_{i,j}$, where $h'_{i,j} \in [0, h_{i,i}[$. This implies that the element $p_{k,j}^{(i)} \in \{0, 1\}$ of $\mathbf{P}^{(i)}$ satisfies $\sum_{k \in [w_i]} 2^{k-1} p_{k,j}^{(i)} = h'_{i,j}$. Moreover, based on the definition of $\mathbf{G}^{(i)}$, $\mathbf{G}^{(i)} \mathbf{p}_j^{(i)} = \mathbf{e}_i \sum_{k \in [m_1]} 2^{k-1} p_{k,j}^{(i)} = h'_{i,j} \mathbf{e}_i$, which means $\mathbf{GP} = \mathbf{H}'$.

– $\mathbf{U}$: Write the block-diagonal matrix $\mathbf{U} = diag(\mathbf{V}_{w_1}, \cdots, \mathbf{V}_{w_{m_1}}, \mathbf{I})$, where $\forall i \in [m_1]$ the unimodular upper-triangular matrix $\mathbf{V}_{w_i} \in \mathbb{Z}^{w_i \times w_i}$. Specifically, the value of the diagonal element $v_{j,j}$ of $\mathbf{V}_{w_i}$ is 1 (i.e., $v_{j,j} = 1$ for $j \in [w_i]$), and the value of the upper diagonal element $v_{j,j+1}$ of $\mathbf{V}_{w_i}$ is -2 (i.e., $v_{j,j+1} = -2$ for $j \in [w_i - 1]$). All the other elements of $\mathbf{V}_{w_i}$ are zero. Notice that, from the definition of $\mathbf{G}$, $\forall i \in [n]$ $w_i = \lceil \lg q \rceil$ when q is a prime. Then, we can write $\mathbf{U} = diag(\mathbf{V}_w, \cdots, \mathbf{V}_w, \mathbf{I})$, where $w = \lceil \lg q \rceil$ and the total number of $\mathbf{V}_w$ is n.

– $\mathbf{R}$: $\mathbf{R}$ is a "short" matrix. In particular, each element in the top $d = (1+\delta) n \lg q$ rows of $\mathbf{R}$ is independently chosen from $\{0, \pm 1\}$ according to the following probability distribution: an element is 0 with probability $\frac{1}{2}$, -1 with probability $\frac{1}{4}$, and 1 with probability $\frac{1}{4}$. Moreover, elements of the remaining rows are zero.

MPSTrapSamp. Given positive integers n, q, let $m_1 > n \lg q$ (e.g., $m_1 = n \lg q + \omega(\log n)$) and $m_2 \geq n \lceil \lg q \rceil$. The component matrices $\mathbf{G}, \mathbf{P}, \mathbf{U}$ and $\mathbf{R}$ from the PPT algorithm can be defined as follows:

– $\mathbf{G}$: $\mathbf{G}$ is a sparse primitive matrix. In particular, write $\mathbf{G} = \left[\, \mathbf{G}^{(1)} \; \mathbf{G}^{(2)} \cdots \mathbf{G}^{(n)} \; \mathbf{0} \,\right]$, where $\forall i \in [n]$ the block matrix $\mathbf{G}^{(i)}$ has $w = \lceil \lg q \rceil$ column vectors, and $\forall j \in [w]$ the jth column $\mathbf{g}_j^{(i)}$ of $\mathbf{G}^{(i)}$ satisfies $\mathbf{g}_j^{(i)} = 2^{j-1} \mathbf{e}_i$. Notice that, the size of $\mathbf{e}_i$ is n.

– $\mathbf{P}$: $\mathbf{P} \in \mathbb{Z}^{m_2 \times m_1}$ is an arbitrary solution to $\mathbf{GP} = -\mathbf{A}_1 \pmod q$. Based on the definition of $\mathbf{G}$, we obtain an obvious solution $\mathbf{P}^t = \left[\, (\mathbf{P}^{(1)})^t \; (\mathbf{P}^{(2)})^t \cdots (\mathbf{P}^{(n)})^t \; \mathbf{0} \,\right]$, where $\forall i \in [n]$ the block matrix $\mathbf{P}^{(i)} \in \mathbb{Z}^{w \times m_1}$, and $\forall j \in [m_1]$, the jth column $\mathbf{p}_j^{(i)}$ of $\mathbf{P}^{(i)}$ contains the binary representation of $-(a_1)_{i,j}$.

- $\mathbf{U}$: $\mathbf{U}$ is the basis for $\Lambda^{\perp}(\mathbf{G})$, which implies that $\mathbf{GU} = \mathbf{0} \pmod q$. Write the block-diagonal matrix $\mathbf{U} = diag(\mathbf{V}_w, \cdots, \mathbf{V}_w, \mathbf{I}) \in \mathbb{Z}^{m_2 \times m_2}$, where the lower-triangular matrix $\mathbf{V}_w \in \mathbb{Z}^{w \times w}$, and the total number of $\mathbf{V}_w$ is n. Specifically, if $q = 2^w$ the value of the diagonal element $v_{j,j}$ of $\mathbf{V}_w$ is 2 (i.e., $v_{j,j} = 2$ for $j \in [w]$), and the value of the lower diagonal element $v_{j+1,j}$ of $\mathbf{V}_w$ is -1 (i.e., $v_{j+1,j} = -1$ for $j \in [w-1]$). If $q \neq 2^w$, for $j \in [w-1]$ the construction of the jth column $(\mathbf{v}_w)_j$ of $\mathbf{V}_w$ is the same as that of the case $q = 2^w$, while the wth column $(\mathbf{v}_w)_w$ of $\mathbf{V}_w$ contains the binary representation of q. Let $\mathbf{q} = (q'_1, q'_2, \ldots, q'_w) \in \{0,1\}^w$ be the binary representation of $q = \sum_{i=1}^{w} 2^{i-1} q'_i$. Then, $(\mathbf{v}_w)_w = \mathbf{q}$.
- $\mathbf{R}$: $\mathbf{R}$ is a random "short" matrix sampled from a probability distribution D over $\mathbb{Z}^{m_1 \times m_2}$. In particular, D is subgaussian with some (Gaussian) parameter $\beta' > 0$. This means that D can be a discrete Gaussian distribution for some $\beta' \geq \eta_v(\mathbb{Z})$, where η_v is the smoothing parameter and v is negligible in n, or the distribution over $\{0, \pm 1\}^{m_1 \times m_2}$ that outputs 0 with probability $\frac{1}{2}$ and ± 1 with probability $\frac{1}{4}$, respectively.

A.2 Details of GHV

The concrete GHV cryptosystem with parameters n, m, q, β involves a triple of PPT algorithms $\mathsf{GHV} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ below:

- $\mathsf{KeyGen}(1^n) \to (\mathbf{A}, \mathbf{T})$: On input 1^n, run the $\mathsf{APTrapSamp}$-type trapdoor sampling algorithm (e.g., $\mathsf{APSTrapSamp}$) to compute its public key $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and its trapdoor $\mathbf{T} \in \mathbb{Z}^{m \times m}$ as the secret key. Output the key pair $(\mathbf{A}, \mathbf{T})$.
- $\mathsf{Enc}_{\mathbf{A}}(\mathbf{B}) \to \mathbf{C}$: On input a plaintext message $\mathbf{B} \in \mathbb{Z}_p^{m \times m}$, sample a matrix $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and an "error matrix" $\mathbf{X} \leftarrow \overline{\Psi}_{\beta}(q)^{m \times m}$. Compute a ciphertext $\mathbf{C} \in \mathbb{Z}_q^{m \times m}$ as follows: $\mathbf{C} = \mathbf{AS} + p\mathbf{X} + \mathbf{B} \pmod q$.
- $\mathsf{Dec}_{\mathbf{T}}(\mathbf{C}) \to \mathbf{B}$: On input the ciphertext message $\mathbf{C}$, first run $\mathbf{C}' = \mathbf{TCT}^t \pmod q$ and then recover $\mathbf{B} = \mathbf{T}^{-1}\mathbf{C}'(\mathbf{T}^t)^{-1} \pmod p$.

The attractive thing is that GHV supports one multiplication and polynomially number (i.e., n^c for $c > 0$) of additions. This means that $\mathbf{B}_1\mathbf{B}_2^t = \mathsf{Dec}_{\mathbf{T}}(\mathbf{C}_1\mathbf{C}_2^t)$ and $\mathbf{B}_1 + \mathbf{B}_2 = \mathsf{Dec}_{\mathbf{T}}(\mathbf{C}_1 + \mathbf{C}_2)$, where $\mathsf{Enc}_{\mathbf{A}}(\mathbf{B}_1) = \mathbf{C}_1$ and $\mathsf{Enc}_{\mathbf{A}}(\mathbf{B}_2) = \mathbf{C}_2$. Consider the case that $\mathsf{APSTrapSamp}$ is used in KeyGen. To guarantee the corresponding homomorphic operations, if $p = 2$, parameters q, m and β can be set as $q > 2^{20}(c+4)^3 n^{3c+4} \lg^5 n$ (with q an odd prime), $m = \lfloor 8n \lg q \rfloor$ and $\beta = \frac{1}{27 n^{1+\frac{3c}{2}} \lg n \lg q \sqrt{qm}}$, and if $p > 2$, $q = \omega(p^2(c+4)^3 n^{3c+4} \lg^5 n)$ and the other parameters are the same as above. Moreover, GHV holds a "very special" property that the product of two encryptions under two distinct public keys can be correctly decrypted by pulling together the corresponding two secret keys. Specifically, given two key pairs $(\mathbf{A}_1, \mathbf{T}_1)$ and $(\mathbf{A}_2, \mathbf{T}_2)$ with both defined modulo the same prime q. Assume that $\mathbf{C}_1 = \mathsf{Enc}_{\mathbf{A}_1}(\mathbf{B}_1) = \mathbf{A}_1\mathbf{S}_1 + p\mathbf{X}_1 + \mathbf{B}_1 \pmod q$ and $\mathbf{C}_2 = \mathsf{Enc}_{\mathbf{A}_2}(\mathbf{B}_2) = \mathbf{A}_2\mathbf{S}_2 + p\mathbf{X}_2 + \mathbf{B}_2 \pmod q$. Then, $\mathbf{B}_1\mathbf{B}_2^t = \mathsf{Dec}_{(\mathbf{T}_1,\mathbf{T}_2)}(\mathbf{C}_1\mathbf{C}_2^t) = \mathbf{T}_1^{-1}(\mathbf{T}_1\mathbf{C}_1\mathbf{C}_2^t\mathbf{T}_2^t \pmod q)(\mathbf{T}_2^t)^{-1} \pmod p$. To the best of our knowledge, this property is known as the multi-key homomorphism which is first defined by López-Alt, Tromer and Vaikuntanathan in 2012 [22]. From this point of view, GHV is actually a multi-key homomorphic encryption scheme that may be used for multi-party computation.

A.3 Proof of Lemma 4

Proof. From Sect. 2.2 (see Fig. 1(a)), we obtain that $\mathbf{T} = \left(\left[\begin{smallmatrix} (\mathbf{G}+\mathbf{R})\mathbf{U} & \mathbf{RP}-\mathbf{I} \\ \mathbf{U} & \mathbf{P} \end{smallmatrix}\right]\right)^t = \left(\left[\begin{smallmatrix} \mathbf{GU} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right] + \left[\begin{smallmatrix} \mathbf{RU} & \mathbf{RP} \\ \mathbf{U} & \mathbf{P} \end{smallmatrix}\right]\right)^t = \left(\left[\begin{smallmatrix} \mathbf{GU} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right] + \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]\left[\begin{smallmatrix} \mathbf{U} & \mathbf{P} \end{smallmatrix}\right]\right)^t$. Then, $N_{\mathbf{T}}$ actually involves numbers of nonzero elements in $\left(\left[\begin{smallmatrix} \mathbf{GU} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]\right)^t$ and $(\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]\left[\begin{smallmatrix} \mathbf{U} & \mathbf{P} \end{smallmatrix}\right])^t$. Specifically, from the constructions of $\mathbf{G}$ and $\mathbf{U}$ in Sect. 2.2 (see Appendix A.1), since the modulus q that we need to set is prime, $\mathbf{GU}$ can be equal to $\left[\begin{smallmatrix} \mathbf{G}^{(1)}\mathbf{V}_w & \mathbf{G}^{(2)}\mathbf{V}_w & \cdots & \mathbf{G}^{(n)}\mathbf{V}_w & \mathbf{0} \end{smallmatrix}\right]$. In particular, $\forall i \in [n]$, the first column of $\mathbf{G}^{(i)}\mathbf{V}_w$ is $\mathbf{e}_i$, and other columns of $\mathbf{G}^{(i)}\mathbf{V}_w$ are zero vectors. Hence, there are $n+m_1$ nonzero elements in $\left(\left[\begin{smallmatrix} \mathbf{GU} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]\right)^t$. For $(\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]\left[\begin{smallmatrix} \mathbf{U} & \mathbf{P} \end{smallmatrix}\right])^t$, we first discuss $N_{\mathbf{R}}$, $N_{\mathbf{U}}$ and $N_{\mathbf{P}}$, respectively. Based on the fact that each element in the first d rows of $\mathbf{R}$ is nonzero with probability $\frac{1}{2}$, we have $N_{\mathbf{R}} \sim \mathsf{Bin}_{\frac{1}{2},dm_2}$. According to Sect. 2.2 (see Appendix A.1), since only the w diagonal elements and the $w-1$ upper diagonal elements of $\mathbf{V}_w$ are nonzero (i.e., 1 and -2), $\mathbf{U}$ consisting of n matrices $\mathbf{V}_w$ and an $(m_2-nw)\times(m_2-nw)$ identity matrix $\mathbf{I}$ satisfies that $N_{\mathbf{U}} = m_2 + n(w-1)$. For $\mathbf{P}$, it can be regarded as the binary representation of $\mathbf{H}' = \mathbf{H} - \mathbf{I}$, where $\mathbf{H} = \left[\begin{smallmatrix} q\mathbf{e}_1 & \cdots & q\mathbf{e}_n & \hat{\mathbf{H}} \end{smallmatrix}\right]$ when q is prime. Then, under the assumption that $h_{i,j} \overset{\$}{\leftarrow} \mathbb{Z}_q$ for $i \in [n]$ and $j \in [n+1, m_1]$, we can obtain that $N_{\mathbf{P}} = n\mathsf{bwt}(q-1) + N_{\mathcal{P}_{\tilde{\mathbf{H}}}}$, where we have $N_{\mathcal{P}_{\tilde{\mathbf{H}}}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2},n(m_1-n)w}$. Next, based on the above analysis, there are $2m_2 + n(w + \mathsf{bwt}(q-1) - 1) + N_{\mathbf{R}} + N_{\mathcal{P}_{\tilde{\mathbf{H}}}}$ nonzero elements in $\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} \mathbf{U} & \mathbf{P} \end{smallmatrix}\right]$. According to Definition 2, this means that $2m_2 + n(w + \mathsf{bwt}(q-1) - 1) + N_{\mathbf{R}} + N_{\mathcal{P}_{\tilde{\mathbf{H}}}}$ nonzero elements related to $(\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]\left[\begin{smallmatrix} \mathbf{U} & \mathbf{P} \end{smallmatrix}\right])^t$ are needed for matrix multiplication. Hence, $N_{\mathbf{T}} = n + m_1 + 2m_2 + n(w + \mathsf{bwt}(q-1) - 1) + N_{\mathbf{R}} + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} = m + m_2 + n(w + \mathsf{bwt}(q-1)) + N_{\mathbf{R}} + N_{\mathcal{P}_{\tilde{\mathbf{H}}}}$, which confirms our lemma.

A.4 Proof of Lemma 5

Proof. From the decomposition form of $\mathbf{T}$, we know that $N_{\mathbf{T}}$ actually involves numbers of nonzero elements in $(\left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{P} & \mathbf{U} \end{smallmatrix}\right])^t$ and $(\left[\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right])^t$. Specifically, from the definition of $\mathbf{R}$, since each element in $\mathbf{R}$ is nonzero with probability $\frac{1}{2}$, we have $N_{\mathbf{R}} \sim \mathsf{Bin}_{\frac{1}{2},m_1m_2}$. For $\mathbf{P}$, since each column contains the binary representation of an element of $-\mathbf{A}_1$, where $\mathbf{A}_1 \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m_1}$, we have $N_{\mathbf{P}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2},nm_1w}$. According to the definition of $\mathbf{U}$, we have $N_{\mathbf{U}} = nN_{\mathbf{V}_w} + (m_2 - nw)$, where $N_{\mathbf{V}_w} = 2w - 2 + \mathsf{bwt}(q)$. Then, $N_{\mathbf{U}} = n(w - 2 + \mathsf{bwt}(q)) + m_2$. Based on the above results, we obtain that $N_{\mathbf{T}} = N_{\mathbf{R}} + N_{\mathbf{P}} + n(w - 2 + \mathsf{bwt}(q)) + m_2 + (m_1 + m) = N_{\mathbf{R}} + N_{\mathbf{P}} + 2m + n(w - 2 + \mathsf{bwt}(q))$, which confirms our lemma.

A.5 Proof of Lemma 6

Proof. We employ the column transformation to generate the inversion of the trapdoor matrix $\mathbf{T}$, and the detailed procedure on computing $(\mathbf{T}^{-1})^t$ is shown in Fig. 2. Notice that, the second transformation in Fig. 2 is based on the fact that $\mathbf{GP} = \mathbf{H}'$. Then, $\mathbf{T}^{-1} = \left(\begin{bmatrix} \mathbf{U}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \left(\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{P} \\ -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{H}^{-1} & -\mathbf{H}^{-1}(\mathbf{G}+\mathbf{R}) \end{bmatrix}\right)\right)^t$. For $\mathbf{U}^{-1}$, from definitions of $\mathbf{U}$ and $\mathbf{V}$ in Appendix A.1, it is easy to find that $\mathbf{U}^{-1} = diag(\mathbf{V}_{w_1}^{-1}, \cdots, \mathbf{V}_{w_{m_1}}^{-1}, \mathbf{I})$, where $\forall i \in [w_k]$ with $k \in [m_1]$ the ith column of the $w_k \times w_k$ matrix $\mathbf{V}_{w_k}^{-1}$ is $\sum_{j=1}^{i} 2^{i-j}\mathbf{e}_j$. Here we first focus on exploring $\begin{bmatrix} \mathbf{H}^{-1} & -\mathbf{H}^{-1}(\mathbf{G}+\mathbf{R}) \end{bmatrix} \in \mathbb{Z}_p^{m_1 \times m}$ that involves three matrices $\mathbf{H}^{-1}$, $\mathbf{G}$ and $\mathbf{R}$. Specifically, $\mathbf{H}^{-1}$ is able to be directly generated from $\mathbf{H}$, which means that $\forall i \in [n], j \in [n+1, m_1]$ $h_{i,i}^{-1} = q^{-1}$, $h_{j,j}^{-1} = 1$, $h_{i,j}^{-1} = -q^{-1}h_{i,j}$, and other elements are zero. We write $\mathbf{H}^{-1} = \begin{bmatrix} (q^{-1})\mathbf{I} & \tilde{\mathbf{H}}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$ as a block matrix, where $\tilde{\mathbf{H}}^{-1} \in \mathbb{Z}_p^{n \times (m_1 - n)}$ corresponds to $\{h_{i,j}^{-1} | \forall i \in [n], j \in [n+1, m_1]\}$. Based on the assumption that $h_{i,j} \xleftarrow{\$} \mathbb{Z}_q$ and the random choice of $\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix}$, where $\mathbf{R}_1 \in \{0, \pm 1\}^{n \times m_2}$ and $\mathbf{R}_2 \in \{0, \pm 1\}^{(m_1 - n) \times m_2}$, we have that $\begin{bmatrix} \tilde{\mathbf{H}}^{-1} & \tilde{\mathbf{H}}^{-1}\mathbf{R}_2 \end{bmatrix}$ is statistically close to uniformly random by the matrix-vector leftover hash lemma [10]. Now letting $\mathbf{F} = \begin{bmatrix} (q^{-1})\mathbf{I} & \tilde{\mathbf{H}}^{-1} \end{bmatrix} \mathbf{R} = \begin{bmatrix} (q^{-1})\mathbf{I} & \tilde{\mathbf{H}}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix}$, we have that $\begin{bmatrix} \tilde{\mathbf{H}}^{-1} & \mathbf{F} \end{bmatrix}$ is an (almost) uniformly random matrix over $\mathbb{Z}_p^{n \times (m_2 + m_1 - n)}$, and so is the block matrix $\begin{bmatrix} \tilde{\mathbf{H}}^{-1} & -\begin{bmatrix} (q^{-1})\mathbf{I} & \tilde{\mathbf{H}}^{-1} \end{bmatrix}(\mathbf{G}+\mathbf{R}) \end{bmatrix}$. Hence, the number of nonzero elements of $\begin{bmatrix} \tilde{\mathbf{H}}^{-1} & -\begin{bmatrix} (q^{-1})\mathbf{I} & \tilde{\mathbf{H}}^{-1} \end{bmatrix}(\mathbf{G}+\mathbf{R}) \end{bmatrix}$ can be regarded as a random variable following a distribution that is statistically close to $\mathsf{Bin}_{\frac{p-1}{p}, n(m_2+m_1-n)}$. Moreover, since $\forall i \in [n+1, m_1], j \in [m_2]$ $g_{i,j} = 0$ in $\mathbf{G}$ when q is a prime, we obtain that $-\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}(\mathbf{G} + \mathbf{R}) = -\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix} = -\mathbf{R}_2$. This means that the number of nonzero elements of $-\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}(\mathbf{G} + \mathbf{R})$ can also be regarded as a random variable that follows $\mathsf{Bin}_{\frac{1}{2}, (d-n)m_2}$. According to the above analysis, we have that the number of nonzero elements of $\begin{bmatrix} \mathbf{H}^{-1} & -\mathbf{H}^{-1}(\mathbf{G}+\mathbf{R}) \end{bmatrix}$ is equal to $m_1 + Y_1 + Y_2$, where $Y_1 \overset{s}{\approx} \mathsf{Bin}_{\frac{p-1}{p}, n(m_2+m_1-n)}$ and $Y_2 \sim \mathsf{Bin}_{\frac{1}{2}, (d-n)m_2}$. For $\begin{bmatrix} \mathbf{U}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$, from the above discussion on $\mathbf{U}^{-1}$, we know that $N_{\mathbf{U}^{-1}} \in [m_2, m_2 + \frac{nw(w-1)}{2}]$ and $N_{\mathbf{U}^{-1}} = m_2$ if $p = 2$, which indicates that the number of nonzero elements in $\begin{bmatrix} \mathbf{U}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$ is (at least) m. For $\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, the number of corresponding nonzero elements is m_2. For $\begin{bmatrix} \mathbf{P} \\ -\mathbf{I} \end{bmatrix}$, based on the exploration of $N_{\mathbf{P}}$ in Lemma 4, the number of corresponding nonzero elements is $m_1 + n\mathsf{bwt}(q-1) + N_{\mathcal{P}_{\tilde{\mathbf{H}}}}$, where $N_{\mathcal{P}_{\tilde{\mathbf{H}}}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2}, n(m_1-n)w}$. Next, taking all numbers together, $N_{\mathbf{T}^{-1}}$ is (at least) $2m + m_1 + n\mathsf{bwt}(q-1) + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} + Y_1 + Y_2$, which confirms our lemma.

A.6 Proof of Lemma 7

Proof. From Lemma 5, we know that $\mathbf{T}^{-1} = \left(\left(\begin{bmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{P} & \mathbf{U} \end{bmatrix}\right)^{-1}\right)^t$, where $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{P} & \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{P} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U} \end{bmatrix}$. Then, we immediately obtain that $\mathbf{T}^{-1} = \left(\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}^{-1} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\right)^t$. In particular, $\mathbf{U}^{-1} = diag(\mathbf{V}_w^{-1}, \cdots, \mathbf{V}_w^{-1}, \mathbf{I})$, where there exists a decomposition $\mathbf{V}_w^{-1} = (\widehat{\mathbf{V}}_w\widetilde{\mathbf{V}}_w)^{-1} = \widetilde{\mathbf{V}}_w^{-1}\widehat{\mathbf{V}}_w^{-1}$, and $\mathbf{I}$ is of size $(m_2 - nw) \times (m_2 - nw)$. Specifically, for $\widehat{\mathbf{V}}_w$, the value of the diagonal element $\hat{v}_{i,i}$ is -1 (i.e., $\hat{v}_{i,i} = -1$ for $i \in [w]$), and the value of the upper diagonal element $\hat{v}_{i,i+1}$ is 2 (i.e., $\hat{v}_{i,i+1} = 2$

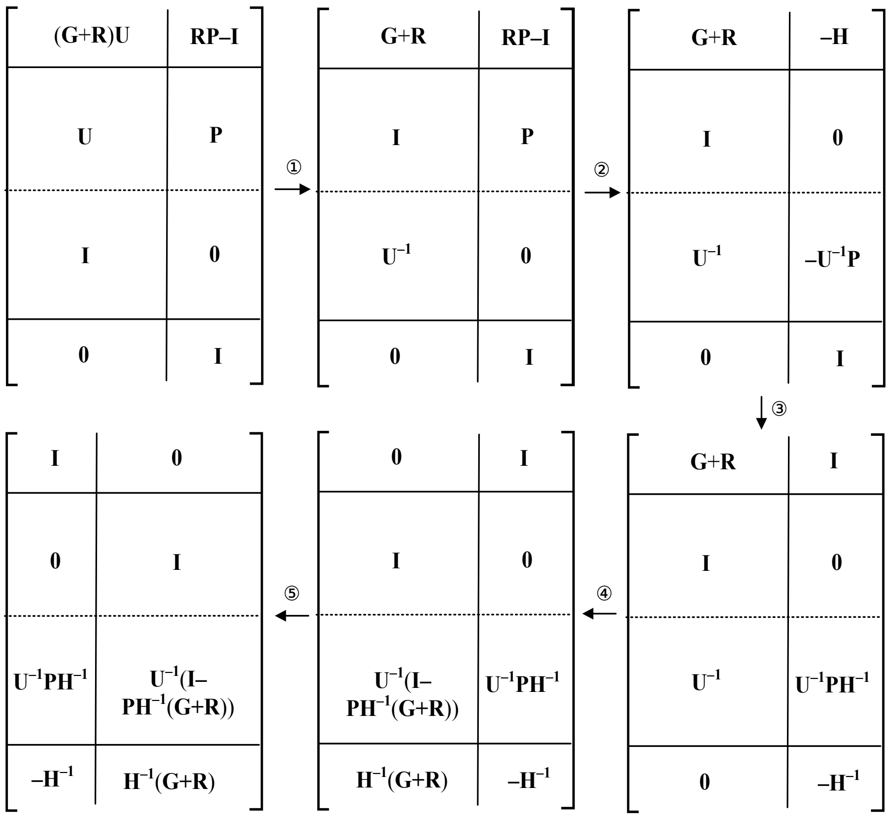


Fig. 2. The detailed procedure for computing $(\mathbf{T}^{-1})^t$.

for $i \in [w-1]$). All the other elements of $\widehat{\mathbf{V}}_w$ are zero. For $\widetilde{\mathbf{V}}_w$, the value of the lower diagonal element $\tilde{v}_{i+1,i}$ is 1 (i.e., $\tilde{v}_{i+1,i} = 1$ for $i \in [w-1]$), and $\forall j \in [w]$ $\tilde{v}_{j,w} = -\sum_{k=j}^{w} 2^{k-j} q'_k$, where $q'_k \in \{0,1\}$. All the other elements of $\widetilde{\mathbf{V}}_w$ are also zero. Then, the nonzero element of $\widehat{\mathbf{V}}_w^{-1}$ focuses on $\hat{v}_{i,j}^{-1}$, where $i,j \in [w]$ and $i \leq j$. This means that $N_{\widehat{\mathbf{V}}_w^{-1}} \geq w$ and $N_{\widehat{\mathbf{V}}_w^{-1}}$ is at most $\frac{(w+1)w}{2}$ when $p > 2$ is a prime. The nonzero element of $\widetilde{\mathbf{V}}_w^{-1}$ focuses on $\tilde{v}_{i,w}^{-1}$ and $\tilde{v}_{i,i+1}^{-1}$, where $i \in [w]$. Since q is a prime, $N_{\widetilde{\mathbf{V}}_w^{-1}} > w-1$, and $N_{\widetilde{\mathbf{V}}_w^{-1}}$ is at most $2w-1$. From the above analysis, we have $m_2 + n(w-1) < N_{\mathbf{U}^{-1}} \leq m_2 + \frac{n(w^2+3w-2)}{2}$. Moreover, according to Lemma 5, $N_{-\mathbf{R}} = N_{\mathbf{R}} \sim \mathsf{Bin}_{\frac{1}{2},m_1 m_2}$, and $N_{-\mathbf{P}} = N_{\mathbf{P}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2},n m_1 w}$. Hence, $N_{\mathbf{T}^{-1}} = N_{\mathbf{U}^{-1}} + N_{-\mathbf{R}} + N_{-\mathbf{P}} + 2m + m_1 = N_{\mathbf{R}} + N_{\mathbf{P}} + 3m + Y_3$, where $n(w-1) < Y_3 \leq \frac{n(w^2+3w-2)}{2}$. This confirms our lemma.

A.7 Proof of Theorem 2

Proof. Let us first show the computational cost of $\mathsf{Enc}(\mathbf{B}) = \mathbf{AS} + p\mathbf{X} + \mathbf{B} \pmod q$. Specifically, for the matrix multiplication $\mathbf{AS}$, since $\mathbf{A}$ is an (almost) uniformly random matrix over $\mathbb{Z}_q^{m\times n}$ and $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, it at most involves nm^2 element-wise multiplications and additions. For $p\mathbf{X}$, there are m^2 element-wise multiplications and discrete Gaussian samplings. Moreover, two necessary matrix additions in $\mathbf{AS}+p\mathbf{X}+\mathbf{B} \pmod q$ need $2m^2$ element-wise additions. Therefore, the computation of $\mathsf{Enc}(\mathbf{B})$ takes at most $m^2((n+1)t_m + (n+2)t_a + t_g)$ time.

Then, we explore the computational cost of $\mathsf{Dec}(\mathbf{C})$ that includes two steps. The first step $\mathbf{C}' = \mathbf{TCT}^t \pmod q$ involves two independent matrix multiplications (using the decomposition form of $\mathbf{T}$). From Lemma 4, we know that $N_{\mathbf{T}} = N_{\mathbf{R}} + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} + m + m_2 + n(w + \mathsf{bwt}(q-1))$, where $N_{\mathbf{R}} \sim \mathsf{Bin}_{\frac{1}{2},dm_2}$ and $N_{\mathcal{P}_{\tilde{\mathbf{H}}}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2},n(m_1-n)w}$. In particular, applying Theorem 1, $\Pr[N_{\mathbf{R}} + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} < \frac{1}{2}(dm_2 + n(m_1-n)w) - \sqrt{\frac{n}{2}}m_2] < \exp^{-n}$, which means that $N_{\mathbf{R}} + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} + n(w+\mathsf{bwt}(q-1))$ is at least $\frac{1}{2}(dm_2 + n(m_1-n)w) - \sqrt{\frac{n}{2}}m_2$ (with overwhelming probability). Then the computational overhead of $\mathbf{TCT}^t \pmod q$, which is equal to $2mN_{\mathbf{T}}(t_m + t_a)$, is at least $2m((\frac{1}{2}d + 1 - \sqrt{\frac{n}{2}})m_2 + m + \frac{1}{2}n(m_1-n)w)(t_m+t_a)$. The second step $\mathbf{B} = \mathbf{T}^{-1}\mathbf{C}'(\mathbf{T}^t)^{-1} \pmod p$ also involves two independent matrix multiplications (using the decomposition form of $\mathbf{T}^{-1}$). According to Lemma 6, $N_{\mathbf{T}^{-1}}$ is at least $2m + m_1 + n\mathsf{bwt}(q-1) + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} + Y_1 + Y_2$, where $Y_1 \overset{s}{\approx} \mathsf{Bin}_{\frac{p-1}{p},n(m-n)}$ and $Y_2 \sim \mathsf{Bin}_{\frac{1}{2},(d-n)m_2}$. In particular, applying Theorem 1,

$$\begin{cases} \Pr[Y_1 < \frac{p-1}{p}n(m-n) - \sqrt{\frac{n}{2}}m_2] < \exp^{-n}, \\ \Pr[Y_2 + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} < \frac{1}{2}((d-n)m_2 + (m_1-n)nw) - \sqrt{\frac{n}{2}}m_2] < \exp^{-n}, \end{cases}$$

which implies that (with overwhelming probability) $Y_1 + Y_2 + N_{\mathcal{P}_{\tilde{\mathbf{H}}}} + n\mathsf{bwt}(q-1)$ is at least $\frac{p-1}{p}n(m-n) - \sqrt{\frac{n}{2}}m_2 + \frac{1}{2}((d-n)m_2 + (m_1-n)nw) - \sqrt{\frac{n}{2}}m_2$. Hence, the computational overhead of $\mathbf{T}^{-1}\mathbf{C}'(\mathbf{T}^t)^{-1} \pmod p$, which is equal to $2mN_{\mathbf{T}^{-1}}(t_m+t_a)$, is at least $2m(\frac{p-1}{p}n(m-n) + \frac{1}{2}((d-n)m_2 + (m_1-n)nw) - \sqrt{2n}m_2 + 2m + m_1)(t_m+t_a)$. Taking both costs together, the computation of $\mathsf{Dec}(\mathbf{C})$ takes at least $2m(\frac{p-1}{p}n(m-n) + (d - \frac{n}{2} - 3\sqrt{\frac{n}{2}})m_2 + (m_1-n)nw + 4m)(t_m+t_a)$ time.

A.8 Proof of Theorem 3

Proof. For GHV using $\mathsf{MPSTrapSamp}$, the analysis of the computational cost of $\mathsf{Enc}(\mathbf{B}) = \mathbf{AS} + p\mathbf{X} + \mathbf{B} \pmod q$ is the same as that of the case when $\mathsf{APSTrapSamp}$ is employed by GHV. Then, the encryption also takes at most $m^2((n+1)t_m + (n+2)t_a + t_g)$ time.

Next, we focus on showing the computational cost of $\mathsf{Dec}(\mathbf{C})$ that includes two steps. For the first step $\mathbf{C}' = \mathbf{TCT}^t \pmod q$ involving two independent matrix multiplications (using the decomposition form of $\mathbf{T}$), according to Lemma 5, we have $N_{\mathbf{T}} = N_{\mathbf{R}} + N_{\mathbf{P}} + 2m + n(w - 2 + \mathsf{bwt}(q))$, where $N_{\mathbf{R}} \sim \mathsf{Bin}_{\frac{1}{2},m_1m_2}$ and

$N_{\mathbf{P}} \overset{s}{\approx} \mathsf{Bin}_{\frac{1}{2},nm_1w}$. In particular, applying Theorem 1, $\Pr[N_{\mathbf{R}} + N_{\mathbf{P}} < \frac{1}{2}m_1(nw + m_2) - \sqrt{\frac{n(m_1(nw+m_2)+1)}{2}}] < \exp^{-n}$, which implies that (with overwhelming probability) $N_{\mathbf{R}}+N_{\mathbf{P}}+n(w-2+\mathsf{bwt}(q))$ is at least $\frac{1}{2}m_1(nw+m_2)-\sqrt{\frac{n(m_1(nw+m_2)+1)}{2}}+n(w-2)$. Hence, the computational cost of $\mathbf{C}' = \mathbf{TCT}^t \pmod q$, which is equal to $2mN_{\mathbf{T}}(t_m + t_a)$, is at least $2m(\frac{1}{2}m_1(nw + m_2) - \sqrt{\frac{n(m_1(nw+m_2)+1)}{2}} + 2m + n(w-2))(t_m + t_a)$. Moreover, the second step $\mathbf{B} = \mathbf{T}^{-1}\mathbf{C}'(\mathbf{T}^t)^{-1} \pmod p$ also involves two independent matrix multiplications (using the decomposition form of $\mathbf{T}^{-1}$). According to Lemma 7, $N_{\mathbf{T}^{-1}} = N_{\mathbf{R}} + N_{\mathbf{P}} + 3m + Y_3$, where $n(w-1) < Y_3 \le \frac{n(w^2+3w-2)}{2}$. Thus, $N_{\mathbf{T}^{-1}}$ is at least $\frac{1}{2}m_1(nw + m_2) - \sqrt{\frac{n(m_1(nw+m_2)+1)}{2}} + 3m + n(w-1)$ (with overwhelming probability). This means that the computational overhead of the second step, which is equal to $2mN_{\mathbf{T}^{-1}}(t_m + t_a)$, is at least $2m(\frac{1}{2}m_1(nw + m_2) - \sqrt{\frac{n(m_1(nw+m_2)+1)}{2}} + 3m + n(w-1))(t_m + t_a)$. Based on the above analysis, the computation of $\mathsf{Dec}(\mathbf{C})$ takes at least $2m(m_1(nw + m_2) - \sqrt{2n(m_1(nw + m_2) + 1)} + 5m + (2w-3)n)(t_m + t_a)$ time.

A.9 Proof of Theorem 4

Proof. The whole computation procedure of $\mathsf{Enc}(\mathbf{B})$ needs four matrices, i.e., $\mathbf{A}$, $\mathbf{S}$, $\mathbf{X}$ and $\mathbf{B}$. Specifically, since $\mathbf{A}$ and $\mathbf{S}$ are matrices over $\mathbb{Z}_q^{m\times n}$ and $\mathbb{Z}_q^{n\times m}$, respectively, the size of $\mathbf{A}$ and $\mathbf{S}$ are bounded by $2nm\lceil\lg q\rceil$. Each element of $\mathbf{X}$ is generated by a discrete Gaussian sampling algorithm over the integers. Then, after generating an element of $p\mathbf{X}$, we can directly add it to an element of $\mathbf{AS}$. This implies that there is no need to store $\mathbf{X}$. For $\mathbf{B} \in \mathbb{Z}_p^{m\times m}$, the corresponding size is bounded by $m^2\lceil\lg p\rceil$. Therefore, the memory cost of $\mathsf{Enc}(\mathbf{B})$ is $2nm\lceil\lg q\rceil + m^2\lceil\lg p\rceil$ bits.

Next, we analyze the case of $\mathsf{Dec}(\mathbf{C})$. Notably, this computation involves four matrices, i.e., $\mathbf{C}$, $\mathbf{C}'$, $\mathbf{T}$ and $\mathbf{T}^{-1}$. In particular, the size of $\mathbf{C} \in \mathbb{Z}_q^{m\times m}$ and $\mathbf{C}' \in \mathbb{Z}_q^{m\times m}$ are bounded by $2m^2\lceil\lg q\rceil$. For $\mathbf{T}$ and $\mathbf{T}^{-1}$, we can employ the corresponding decomposition forms and simply store non-deterministic components for building these two matrices[7]. Consider $\mathbf{T} = \left(\left[\begin{smallmatrix}\mathbf{GU} & -\mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right] + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{U} & \mathbf{P}\end{smallmatrix}\right]\right)^t$. From the constructions of $\mathbf{G}$ and $\mathbf{U}$ in Appendix A.1, we know that $\forall i \in [0, n-1]$ the $(iw + 1)$th column of $\mathbf{GU}$ is $\mathbf{e}_i$, and other columns of $\mathbf{GU}$ are zero vectors. This implies that $\mathbf{GU}$ is easily reconstructed for each decryption. Then, there is no need to store the deterministically-constructed matrices $\mathbf{U}$ and $\mathbf{GU}$, and the memory cost is simply used for $\mathbf{R} \in \{0, \pm1\}^{m_1\times m_2}$ and $\mathbf{P} \in \{0, 1\}^{m_2\times m_1}$. Moreover, $\mathbf{P}$ is the binary representation of $\mathbf{H}' = \mathbf{H}-\mathbf{I}$, where $\mathbf{H} = \left[\begin{smallmatrix}q\mathbf{e}_1 & \cdots & q\mathbf{e}_n & \hat{\mathbf{H}}\end{smallmatrix}\right]$ and $\hat{\mathbf{H}} = \left[\begin{smallmatrix}\tilde{\mathbf{H}}\\ \mathbf{I}\end{smallmatrix}\right]$. The only non-deterministic component in $\mathbf{P}$ is the binary representation of $\tilde{\mathbf{H}}$. Hence the real memory cost for $\mathbf{T}$ is $2dm_2 + n(m_1 - n)w$. Now, let us consider $\mathbf{T}^{-1} = \left(\left[\begin{smallmatrix}\mathbf{U}^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\left(\left[\begin{smallmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right] + \left[\begin{smallmatrix}\mathbf{P}\\ -\mathbf{I}\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{H}^{-1} & -\mathbf{H}^{-1}(\mathbf{G}+\mathbf{R})\end{smallmatrix}\right]\right)\right)^t \in$

[7] In Sect. 4.4, some concrete optimizations on speeding up matrix multiplication are given, which demonstrate that why the memory cost is only used for the non-deterministic components.

$\mathbb{Z}_p^{m\times m}$. From Lemma 6, we know that $\mathbf{U}^{-1}$ is constructed deterministically. For $\left[\begin{smallmatrix}\mathbf{H}^{-1} & -\mathbf{H}^{-1}(\mathbf{G}+\mathbf{R})\end{smallmatrix}\right]$, where $\mathbf{H}^{-1} = \left[\begin{smallmatrix}(q^{-1})\mathbf{I} & \tilde{\mathbf{H}}^{-1}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]$, the non-deterministic component is $\left[\begin{smallmatrix}\tilde{\mathbf{H}}^{-1} & -\left[\begin{smallmatrix}(q^{-1})\mathbf{I} & \tilde{\mathbf{H}}^{-1}\end{smallmatrix}\right](\mathbf{G}+\mathbf{R})\end{smallmatrix}\right] \in \mathbb{Z}_p^{n\times(m-n)}$. This means that the real memory cost for $\mathbf{T}^{-1}$ can only be $n(m-n)\lceil\lg p\rceil$. Based on the above analysis, we believe that the memory cost of $\mathsf{Dec}(\mathbf{C})$ is (at least) $2m^2\lceil\lg q\rceil + n(m-n)\lceil\lg p\rceil + 2dm_2 + n(m_1-n)w$ bits.

A.10 Proof of Theorem 5

Proof. For GHV using $\mathsf{MPSTrapSamp}$, the analysis of the memory cost of $\mathsf{Enc}(\mathbf{B}) = \mathbf{AS} + p\mathbf{X} + \mathbf{B} \pmod q$ is the same as that of the case when $\mathsf{APSTrapSamp}$ is employed by GHV. Then, the encryption also takes $2nm\lceil\lg q\rceil + m^2\lceil\lg p\rceil$ bits.

For the case of $\mathsf{Dec}(\mathbf{C})$. The corresponding computation involves four matrices, i.e., $\mathbf{C}$, $\mathbf{C}'$, $\mathbf{T}$ and $\mathbf{T}^{-1}$. The size of $\mathbf{C} \in \mathbb{Z}_q^{m\times m}$ and $\mathbf{C}' \in \mathbb{Z}_q^{m\times m}$ are bounded by $2m^2\lceil\lg q\rceil$. For $\mathbf{T}$ and $\mathbf{T}^{-1}$, we still use the decomposition forms and simply store non-deterministic components for building these two matrices. From Lemma 5 and 7, we know $\mathbf{T} = \left(\left[\begin{smallmatrix}\mathbf{I} & \mathbf{R}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{P} & \mathbf{U}\end{smallmatrix}\right]\right)^t$ and $\mathbf{T}^{-1} = \left(\left[\begin{smallmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{U}^{-1}\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{P} & \mathbf{I}\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{I} & -\mathbf{R}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\right)^t$. According to the construction of $\mathbf{U}$ in Appendix A.1 and our analysis on the construction of $\mathbf{U}^{-1}$ in Lemma 7, we also know that $\mathbf{U}$ and $\mathbf{U}^{-1}$ can be constructed deterministically. Hence, it is simply necessary to store $\mathbf{R} \in \{0,\pm1\}^{m_1\times m_2}$ and $\mathbf{P} \in \{0,1\}^{m_2\times m_1}$, where elements of the last $m_2 - nw$ rows of $\mathbf{P}$ are zero. This means that the real memory cost for $\mathbf{T}$ and $\mathbf{T}^{-1}$ is $m_1(2m_2+nw)$, and $\mathsf{Dec}(\mathbf{C})$ takes (at least) $2m^2\lceil\lg q\rceil + m_1(2m_2+nw)$ bits to perform the decryption.

A.11 Proof of Theorem 6

Proof. Let $\mathbf{C} = \sum_{i=1}^{n^c}(\mathbf{AS}_i + p\mathbf{X}_i + \mathbf{B}'_i)$. Since the post-multiplication by $\mathbf{T}^t$ for decrypting $\mathbf{C}$ is redundant, we analyze the size of elements in $\sum_{i=1}^{n^c}\mathbf{T}(p\mathbf{X}_i+\mathbf{B}'_i)$. Notice that, as shown above, here we only need to focus on $\sum_{i=1}^{n^c}\mathbf{T}_1(p\mathbf{X}_i+\mathbf{B}'_i)$ that is directly related to the computation of $\mathbf{C}'_1$. Specifically, consider that $\mathbf{T}_1 = \left[\begin{smallmatrix}((\mathbf{G}+\mathbf{R})\mathbf{U})^t & \mathbf{U}^t\end{smallmatrix}\right]$. According to constructions of $\mathbf{G}$, $\mathbf{R}$ and $\mathbf{U}$ in Appendix A.1, for $i \in [nw]$, if $i \pmod w = 1$ the ith column of $(\mathbf{G}+\mathbf{R})\mathbf{U}$ is $\mathbf{e}_i + \mathbf{r}_i$, and if $i \pmod w \neq 1$ the ith column of $(\mathbf{G}+\mathbf{R})\mathbf{U}$ is $\mathbf{r}_i - 2\mathbf{r}_{i-1}$. Moreover, for $i \in [nw+1, m_2]$, the ith column of $(\mathbf{G}+\mathbf{R})\mathbf{U}$ is equal to $\mathbf{r}_i$. To determine the upper bound of the Euclidean norm of all the rows of $\mathbf{T}_1 = \left[\begin{smallmatrix}((\mathbf{G}+\mathbf{R})\mathbf{U})^t & \mathbf{U}^t\end{smallmatrix}\right]$, we explore the Euclidean norm of $\mathbf{r}_i - 2\mathbf{r}_{i-1}$, of which the upper bound is larger than that of $\mathbf{e}_i + \mathbf{r}_i$ and $\mathbf{r}_i$, where $i \in [m_2]$. Assume that $\forall j \in [m_1]$ $R_j := (r_{i,j} - 2r_{i-1,j})^2$. From the definition of $\mathbf{R}$ (see Appendix A.1), $\Pr[r_{i,j} = 0] = \frac{1}{2}$, $\Pr[r_{i,j} = 1] = \frac{1}{4}$ and $\Pr[r_{i,j} = -1] = \frac{1}{4}$. Then, we have the following probability distribution related to R_j: $\forall j \in [m_1]$, if $(r_{i,j}, r_{i-1,j}) = (0,0)$, $R_j = 0$ and $\Pr[R_j = 0] = \frac{1}{4}$; if $(r_{i,j}, r_{i-1,j}) \in \{(1,1),(-1,-1),(1,0),(-1,0)\}$, $R_j = 1$ and $\Pr[R_j = 1] = \frac{3}{8}$; if $(r_{i,j}, r_{i-1,j}) \in \{(0,1),(0,-1)\}$, $R_j = 4$ and $\Pr[R_j = 4] = \frac{1}{4}$; if $(r_{i,j}, r_{i-1,j}) \in \{(1,-1),(-1,1)\}$, $R_j = 9$ and $\Pr[R_j = 9] = \frac{1}{8}$. This implies that the mean of R_j is $\frac{5}{2}$. Applying Theorem 1, we obtain $\Pr[\sum_{j=1}^{m_1} R_j - \frac{5}{2}m_1 > \frac{3}{2}m_1] < \exp^{-\frac{1}{36}m_1}$. Now consider the general case that $m_1 = \lceil \frac{101}{100} n \lg q \rceil \gg 36$. The Euclidean norm of all the rows of $\mathbf{T}_1$ is less than $2\sqrt{m_1}$ with overwhelming probability. Based on Lemma 3 (with $g = \sqrt{\lg n} - \frac{3}{2n^c \sqrt{m_1} \beta q}$), each element of $\mathbf{T}_1 \mathbf{X}_i$ is bounded by $2\beta q g \sqrt{m_1}$ with overwhelming probability. Moreover, since every element of $\mathbf{B}_i$ is at most $p-1$, each element of $\sum_{i=1}^{n^c} \mathbf{U}^t \mathbf{B}_i$ is bounded by $3n^c p$, which implies that each element in $\sum_{i=1}^{n^c} \mathbf{T}_1 \mathbf{B}'_i = \sum_{i=1}^{n^c} \left[\mathbf{U}^t (\mathbf{G}+\mathbf{R})^t \ \mathbf{U}^t \right] \mathbf{B}'_i = \left[\mathbf{U}^t (\mathbf{G}+\mathbf{R})^t \ \mathbf{U}^t \right] \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sum_{i=1}^{n^c} \mathbf{B}_i \end{bmatrix} = \left[\mathbf{0} \ \sum_{i=1}^{n^c} \mathbf{U}^t \mathbf{B}_i \right]$ is also bounded by $3n^c p$. Hence, the absolute value of each element of $\sum_{i=1}^{n^c} \mathbf{T}_1 (p\mathbf{X}_i + \mathbf{B}'_i)$ is bounded by $2n^c \sqrt{m_1} \beta p q g + 3n^c p = 2n^c \beta p q \sqrt{m_1 \lg n} = \frac{2n^c p q \sqrt{m_1 \lg n}}{5pn^c \sqrt{m_1 \lg n}} = \frac{2}{5} q < \frac{q}{2}$. According to $\mathbf{TA} = \mathbf{0} \pmod q$, $\mathbf{C}'_1 = \mathbf{T}_1 \mathbf{C} \pmod q = \sum_{i=1}^{n^c} \mathbf{T}_1 (p\mathbf{X}_i + \mathbf{B}'_i) \pmod q = \sum_{i=1}^{n^c} \mathbf{T}_1 (p\mathbf{X}_i + \mathbf{B}'_i)$, which confirms our parameter setting. About the detailed proof of the correctness of the homomorphic addition operation, please see a similar proof for Theorem 8.

A.12 Proof of Theorem 7

Proof. We first discuss the case of supporting n^c homomorphic addition operations. Let $\mathbf{C} = \sum_{i=1}^{n^c} (\mathbf{AS}_i + p\mathbf{X}_i + \mathbf{B}'_i)$. The analysis for the size of elements in $\sum_{i=1}^{n^c} \mathbf{T}_1 (p\mathbf{X}_i + \mathbf{B}'_i)$ is nearly the same as that in the proof of Theorem 6. In particular, the absolute value of each element in $\sum_{i=1}^{n^c} \mathbf{T}_1 (p\mathbf{X}_i + \mathbf{B}'_i)$ is bounded by $2n^c \beta p q \sqrt{m_1 \lg n}$ that satisfies $2n^c \beta p q \sqrt{m_1 \lg n} = \frac{2n^c p q \sqrt{m_1 \lg n}}{2pn^{\frac{3c}{2}} \sqrt{m m_1 q \lg n}} = \frac{1}{n^{\frac{c}{2}}} \sqrt{\frac{q}{m}} \ll \sqrt{\frac{q}{m}}$, where inequality is based on the general condition that $n^{\frac{c}{2}} \gg 1$. Then, each element in $\sum_{i=1}^{n^c} \mathbf{T}_1 (p\mathbf{X}_i + \mathbf{B}'_i) \mathbf{T}_1^t$ is bounded by $2m\sqrt{m_1}\sqrt{\frac{q}{m}} = 2m\sqrt{\lceil \frac{101}{100} n \lg q \rceil \frac{q}{m}} \ll \frac{q}{2}$. Since $\mathbf{TA} = \mathbf{0} \pmod q$, we obtain $\mathbf{C}'_1 = \mathbf{T}_1 \mathbf{C} \mathbf{T}_1^t \pmod q = \sum_{i=1}^{n^c} \mathbf{T}_1 (p\mathbf{X}_i + \mathbf{B}'_i) \mathbf{T}_1^t \pmod q = \sum_{i=1}^{n^c} \mathbf{T}_1 (p\mathbf{X}_i + \mathbf{B}'_i) \mathbf{T}_1^t$, which confirms our parameter setting. About the detailed proof of the correctness of the homomorphic addition operation, please see a similar proof for Theorem 8.

Then, we explore the case of supporting one homomorphic multiplication operation. Specifically, assume that there is a circuit with one ℓ_1-fan-in addition layer, which is followed by a multiplication layer of fan-in two, and another ℓ_2-fan-in addition layer, where $\ell_1 + \ell_2 \le n^c$ [8]. As shown above, the ciphertext $\mathbf{C}_i$ at the output of the multiplication layer is of the form $\mathbf{AS}_i + p\mathbf{X}_i + \mathbf{B}'_i + \tilde{\mathbf{S}}_i \mathbf{A}^t$, and all the products from the output of the multiplication layer are of the same form. Notably, the product ciphertext $\mathbf{C}_i$ can be described as $\mathbf{AS}_i + (p\mathbf{X}_{i_1} + \mathbf{B}'_{i_1})(p\mathbf{X}_{i_2} + \mathbf{B}'_{i_2})^t + \tilde{\mathbf{S}}_i \mathbf{A}^t$, which implies that $\mathbf{T}\mathbf{C}_i\mathbf{T}^t \pmod q = \mathbf{T}(p\mathbf{X}_{i_1} + \mathbf{B}'_{i_1})(p\mathbf{X}_{i_2} + \mathbf{B}'_{i_2})^t \mathbf{T}^t \pmod q$, where both $p\mathbf{X}_{i_1} + \mathbf{B}'_{i_1}$ and $p\mathbf{X}_{i_2} + \mathbf{B}'_{i_2}$ are generated by adding up-to ℓ_1 encryptions. According to the relaxed condition, we simply analyze the size of elements in $\mathbf{T}_1(p\mathbf{X}_{i_1} + \mathbf{B}'_{i_1})(p\mathbf{X}_{i_2} + \mathbf{B}'_{i_2})^t \mathbf{T}_1^t$. Moreover, from the above analysis, we know that the absolute value of each element in $\mathbf{T}_1(p\mathbf{X}_{i_1} + \mathbf{B}'_{i_1})$ and $(\mathbf{T}_1(p\mathbf{X}_{i_2} + \mathbf{B}'_{i_2}))^t$ is bounded by $\frac{\ell_1}{n^{\frac{3c}{2}}} \sqrt{\frac{q}{m}}$. This means that every ele-

[8] The circuit used in our proof is the same as that in the proof of Theorem 1 in [15].

ment of $\mathbf{T}_1(p\mathbf{X}_{i_1} + \mathbf{B}'_{i_1})(p\mathbf{X}_{i_2} + \mathbf{B}'_{i_2})^t\mathbf{T}_1^t$ is bounded by $m\left(\frac{\ell_1}{n^{\frac{3c}{2}}}\sqrt{\frac{q}{m}}\right)^2 = \frac{\ell_1^2}{n^{3c}}q$. Next, for the final ciphertext $\mathbf{C} = \sum_{i=1}^{\ell_2}(\mathbf{AS}_i + p\mathbf{X}_i + \mathbf{B}'_i + \tilde{\mathbf{S}}_i\mathbf{A}^t)$ at the output of the addition layer, we still focus on considering the size of elements of $\sum_{i=1}^{\ell_2}\mathbf{T}_1(p\mathbf{X}_i + \mathbf{B}'_i)\mathbf{T}_1^t$ that is directly related to the computation of $\mathbf{C}'_1$, where $\ell_2 \le n^c - \ell_1$. Then, based on the above analysis, each element of $\sum_{i=1}^{\ell_2}\mathbf{T}_1(p\mathbf{X}_i + \mathbf{B}'_i)\mathbf{T}_1^t$ is bounded by $\frac{\ell_1^2\ell_2}{n^{3c}}q \le \frac{\ell_1^2(n^c-\ell_1)}{n^{3c}}q \le \frac{2}{9}q < \frac{q}{2}$, where the second inequality is based on the fact that $\ell_1^2(n^c - \ell_1)$ can achieve the maximum $\frac{2}{9}n^{3c}$ when $\ell_1 = \frac{2}{3}n^c$. Once again, we have $\mathbf{C}'_1 = \mathbf{T}_1\mathbf{C}\mathbf{T}_1^t \pmod q = \sum_{i=1}^{\ell_2}\mathbf{T}_1(p\mathbf{X}_i + \mathbf{B}'_i)\mathbf{T}_1^t \pmod q = \sum_{i=1}^{\ell_2}\mathbf{T}_1(p\mathbf{X}_i + \mathbf{B}'_i)\mathbf{T}_1^t$, which also confirms our parameter setting. About the detailed proof of the correctness of the homomorphic multiplication operation, please see a similar proof for Theorem 8.

A.13 Proof of Theorem 8

Proof. For a plaintext matrix $\mathbf{B} \in \mathbb{Z}_p^{m_2\times m_2}$, run oEnc to construct an enlarged matrix $\mathbf{B}' = \left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{B}\end{smallmatrix}\right] \in \mathbb{Z}_p^{m\times m}$, and generate its corresponding ciphertext matrix $\mathbf{C} = (\mathbf{AS} + p\mathbf{X} + \mathbf{B}') \pmod q \in \mathbb{Z}_q^{m\times m}$. Then, from $\mathbf{TA} \pmod q = \mathbf{0}$, we have $\mathbf{C}' = \mathbf{TCT}^t \pmod q = \mathbf{T}(\mathbf{AS} + p\mathbf{X} + \mathbf{B}')\mathbf{T}^t \pmod q = \mathbf{T}(p\mathbf{X} + \mathbf{B}')\mathbf{T}^t \pmod q$. Consider that APSTrapSamp is employed by oKeyGen. According to the parameter setting in Theorem 7 (and Theorem 6), $\mathbf{T}(p\mathbf{X} + \mathbf{B}')\mathbf{T}^t \pmod q = \mathbf{T}(p\mathbf{X} + \mathbf{B}')\mathbf{T}^t$, which means that $\mathbf{C}' = \mathbf{T}(p\mathbf{X} + \mathbf{B}')\mathbf{T}^t$ and $\tilde{\mathbf{T}}^t\mathbf{C}'\tilde{\mathbf{T}} \pmod p = \tilde{\mathbf{T}}^t\mathbf{TB}'\mathbf{T}^t\tilde{\mathbf{T}} \pmod p$. In particular, we know that $\mathbf{T}^t = \left[\,\mathbf{T}_1^t\ \mathbf{T}_2^t\,\right]$, where $\mathbf{T}_1^t = \left[\begin{smallmatrix}(\mathbf{G}+\mathbf{R})\mathbf{U}\\ \mathbf{U}\end{smallmatrix}\right] \in \mathbb{Z}^{m\times m_2}$ and $\mathbf{T}_2^t = \left[\begin{smallmatrix}\mathbf{RP}-\mathbf{I}\\ \mathbf{P}\end{smallmatrix}\right] \in \mathbb{Z}^{m\times m_1}$. This implies that we can get $\mathbf{TB}'\mathbf{T}^t = \left[\begin{smallmatrix}\mathbf{T}_1\\ \mathbf{T}_2\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{B}\end{smallmatrix}\right]\left[\,\mathbf{T}_1^t\ \mathbf{T}_2^t\,\right] = \left[\begin{smallmatrix}\mathbf{U}^t\\ \mathbf{P}^t\end{smallmatrix}\right]\mathbf{B}\left[\,\mathbf{U}\ \mathbf{P}\,\right]$. Hence, $\tilde{\mathbf{T}}^t\mathbf{TB}'\mathbf{T}^t\tilde{\mathbf{T}} \pmod p = (\left[\,\mathbf{U}\ \mathbf{P}\,\right]\tilde{\mathbf{T}})^t\mathbf{B}\left[\,\mathbf{U}\ \mathbf{P}\,\right]\tilde{\mathbf{T}} \pmod p$. Moreover, since $\tilde{\mathbf{T}} = \left[\begin{smallmatrix}\mathbf{U}^{-1}\\ \mathbf{0}\end{smallmatrix}\right]$, we have $\left[\,\mathbf{U}\ \mathbf{P}\,\right]\left[\begin{smallmatrix}\mathbf{U}^{-1}\\ \mathbf{0}\end{smallmatrix}\right] = \mathbf{I}$. Then the output of oDec is $\tilde{\mathbf{T}}^t(\mathbf{TCT}^t \pmod q)\tilde{\mathbf{T}} \pmod p = \mathbf{IBI} \pmod p = \mathbf{B}$, which confirms our result.

A.14 Proof of Theorem 10

Proof. We first analyze the overhead of encrypting an $m_2 \times m_2$ matrix $\mathbf{B}$. Specifically, the matrix multiplication involved in oEnc is $\mathbf{AS}$, where $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ and $\mathbf{S} \in \mathbb{Z}_q^{n\times m}$. The computation of $\mathbf{AS}$ needs at most $2m^2n$ element-wise arithmetic operations. The computation of $p\mathbf{X}$ includes m^2 element-wise multiplications and discrete Gaussian samplings. Moreover, two necessary matrix additions of $\mathbf{AS} + p\mathbf{X} + \mathbf{B} \pmod q$ involve at most $m^2 + m_2^2$ element-wise additions. Thus, the overall computational cost of oEnc($\mathbf{B}$) is (at most) $m^2(n+1)(t_a + t_m) + m^2t_g + m_2^2t_a$. The whole procedure of oEnc($\mathbf{B}$) involves four matrices, i.e., $\mathbf{A}$, $\mathbf{S}$, $\mathbf{X}$ and $\mathbf{B}$. In particular, $\mathbf{A}$ and $\mathbf{S}$ have $2nm\lceil\lg q\rceil$ bits. $\mathbf{B} \in \mathbb{Z}_p^{m_2\times m_2}$ has $m_2^2\lceil\lg p\rceil$ bits. Notice that, as discussed in Theorem 4, there is also no need to store $p\mathbf{X}$. Then, the memory cost of oEnc($\mathbf{B}$) is $2nm\lceil\lg q\rceil + m_2^2\lceil\lg p\rceil$ bits.

Next, we analyze the cost of oDec. Specifically, the first step $\mathbf{C}' = \left[\begin{smallmatrix}\mathbf{C}'_1 & \mathbf{C}'_2\\ \mathbf{C}'_3 & \mathbf{C}'_4\end{smallmatrix}\right] = \mathbf{TCT}^t \pmod q$ includes two independent matrix multiplications. Since $\mathbf{B}$ is recovered from the $m_2 \times m_2$ matrix $\mathbf{C}'_1$, the corresponding efficiency analysis actually can only focus on the generating process of $\mathbf{C}'_1$. From Lemma 4, we know that the first m_2 columns of $\mathbf{T}^t$ (i.e., $\mathbf{T}_1^t$) can be expressed as $\left[\begin{smallmatrix}\mathbf{GU}\\ \mathbf{0}\end{smallmatrix}\right] + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{U}$. Thus, based on the consequence of $N_{\mathbf{T}}$ in Lemma 4, $N_{\mathbf{T}_1} = N_{\mathbf{R}} + nw + 2m_2$. Moreover, from the second optimization in Sect. 4.4 (see Algorithm 2), we obtain that the computation of $\mathbf{C}'_1 = (\left[\begin{smallmatrix}\mathbf{GU}\\ \mathbf{0}\end{smallmatrix}\right] + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{U})^t\mathbf{C}(\left[\begin{smallmatrix}\mathbf{GU}\\ \mathbf{0}\end{smallmatrix}\right] + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{U})$ simply takes $(m_2 + m)N_{\mathbf{T}_1}t_a = (m_2 + m)(N_{\mathbf{R}} + nw + 2m_2)t_a$ time. In particular, according to Theorem 1, $\Pr[N_{\mathbf{R}} > \frac{1}{2}dm_2 + \sqrt{\frac{n}{2}}m_2 - nw] < \exp^{-n}$, which means that $N_{\mathbf{R}} + nw$ is at most $\frac{1}{2}dm_2 + \sqrt{\frac{n}{2}}m_2$ (with overwhelming probability). The second step $\mathbf{B} = \tilde{\mathbf{T}}^t\mathbf{C}'\tilde{\mathbf{T}} \pmod p$ involves two matrix multiplications related to $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{T}}^t$. By employing the third optimization in Sect. 4.4 (see Algorithm 3), the computation of $\tilde{\mathbf{T}}^t\mathbf{C}'\tilde{\mathbf{T}} \pmod p = (\mathbf{U}^{-1})^t\mathbf{C}'_1\mathbf{U}^{-1} \pmod p$ takes $4nm_2(w-1)t_a$ time. Therefore, the overall computational cost of $\mathsf{oDec}(\mathbf{C})$ is (at most) $((\frac{1}{2}d + \sqrt{\frac{n}{2}} + 2)(m + m_2) + 4n(w-1))m_2t_a$. As discussed in Sect. 4.4, since there is no need to store $\mathbf{T}$ (including $\mathbf{GU}$, $\mathbf{U}$ and $\mathbf{P}$) and $\tilde{\mathbf{T}}$, the whole procedure of $\mathsf{oDec}(\mathbf{C})$ involves three stored matrices, i.e., $\mathbf{R}$, $\mathbf{C}$ and $\mathbf{C}'$. In particular, the memory cost of $\mathbf{R} \in \{0, \pm 1\}^{m_1 \times m_2}$ needs $2dm_2$ bits, and $\mathbf{C}$ and $\mathbf{C}'$ (i.e., $\mathbf{C}'_1$) have $(m^2 + m_2^2)\lceil \lg q \rceil$ bits, which means that the overall memory cost of $\mathsf{oDec}(\mathbf{C})$ can be $(m^2 + m_2^2)\lceil \lg q \rceil + 2dm_2$ bits.