Generalized BGV, BFV, and CKKS for Homomorphic Encryption over Matrix Rings

Bence Mali
bencemali835@gmail.com

University of Budapest, Hungary

May, 2025

Abstract

Some of the most valuable applications of homomorphic encryption, such as encrypted machine learning inference, require efficient large-scale plaintext-ciphertext and ciphertext-ciphertext matrix multiplications. Current state-of-the-art techniques for matrix multiplications all build on the ability to pack many ciphertexts into a ciphertext and compute on them in a Single Instruction, Multiple Data (SIMD) manner. However, to fit the operation of matrix multiplication into this computational model, a large number of additional costly operations need to be performed, such as the rotation of elements between the plaintext slots.

In this work, we propose an orthogonal approach to performing encrypted matrix operations with BGV-like encryption schemes, where the plaintext and ciphertext spaces are generalized to a matrix ring of arbitrary dimension. To deal with the inherent problem of noncommutativity in the case of matrix rings, we present a new superoperator technique to better represent linear and quadratic expressions in the secret key, which allows for the relinearization of ciphertexts after multiplication. The security of the modified encryption schemes is based on Module-LWE with module rank equal to the dimension of the matrices. With this construction, we demonstrate that Ring-LWE, Module-LWE, and LWE are potentially equally efficient for homomorphic encryption, both in terms of useful information density and noise growth, only for different sizes of matrices.

Contents

1 Introduction

1.1 Background

Ever since the foundational work of Gentry [1], which kick-started the work on today's fully homomorphic encryption (FHE) schemes, the defining challenge has been improving efficiency. Over the years, several so-called "generations" of schemes have been developed [2]. The first truly practical schemes came with the second such generation, and they are today represented by the almost identical [3] Brakerski–Gentry–Vaikuntanathan (BGV) [4, 5] and Brakerski/Fan–Vercauteren (BFV) [6, 7] encryption schemes. These schemes have multiplication and bootstrapping operations that are relatively expensive; however, they can pack many plaintext elements into one ciphertext in a Single Instruction, Multiple Data (SIMD) fashion [8][9, section C.2]. For this reason, they can be characterized as high-latency/high-throughput options.

The third generation of FHE constructions [10, 11, 12] are characterized by very fast bootstrapping and are today represented by the TFHE (or equivalently CGGI) [12] scheme. This generation of schemes will not be covered in this work.

The fourth generation refers to the CKKS encryption scheme [13], which diverges from the previous solutions and implements approximate arithmetic of complex numbers, as opposed to the exact discrete operations available previously. Despite the dissimilar plaintext space, the CKKS construction is actually a descendant of the BGV/BFV schemes, and they can all be described by an abstract scheme. We will refer to the BGV, BFV, and CKKS schemes as the BGV-like schemes.

All of the above mentioned schemes base their security on variants of the Learning With Errors (LWE) problem [14, 15, 16], which is the de facto hard problem and corresponding hardness assumption in modern lattice-based cryptography. For performance reasons, most implementations use a structured variant of LWE called Ring-LWE [17].

Previous matrix multiplication techniques. Because of their plaintext packing capabilities, the BGV-like schemes are more commonly used for encrypted matrix operations. For a brief introduction to the capabilities of these schemes, see [18, section 2]. Halevi and Shoup provide a table listing the available operations and their cost in terms of running time and noise growth [19, section 2, table 1]. In a nutshell, each ciphertext encrypts a vector of plaintext elements, where ciphertext addition and multiplication correspond to element-wise addition and multiplication of the underlying plaintext vectors. Addition of and multiplication by a constant are also supported. For more functionality, one can rotate the plaintext elements between the slots in a ciphertext, and also perform automorphisms on the encrypted plaintexts. As for the cost of these operations, plaintext-ciphertext and ciphertext-ciphertext addition are very cheap, plaintext slot rotations, computing automorphisms on plaintext elements, and plaintext-ciphertext multiplication are moderately expensive, while ciphertext-ciphertext multiplication is very expensive.

Here we give a short overview of previous matrix multiplication techniques, but we would like to emphasize that the techniques introduced in this work are in some sense orthogonal to previous techniques and can even be used in conjunction. Current techniques view the encryption schemes as black-box schemes with a fixed computational model (SIMD vector of plaintexts), then they try to fit the application (matrix multiplication) as efficiently as possible into this model. In essence, they try to solve a problem pertaining to algebraic structure with parallelism, which is not sufficient. Unfortunately, this requires using many operations from the moderately expensive category (see previous paragraph) to align and move around plaintext elements within one ciphertext. We will sometimes refer to these kinds of techniques as the outside-in approach, as opposed to modifying

the encryption scheme to accommodate matrix multiplication, which is the inside-out approach.

The naive method for performing homomorphic multiplication of $d \times d$ matrices is to encrypt the matrices entry-by-entry. As flexible as this approach is, it becomes vastly inefficient as the dimension of the matrices grows. To counter this, (outside-in) techniques have been developed to exploit the batching capabilities of the encryption schemes, and thus decrease the number of ciphertexts and operations performed.

1. The first notable work was due to Halevi and Shoup [19], who proposed methods for supporting linear algebra in the HElib homomorphic encryption library [20]. They focus on matrix-vector multiplication, where the vector is always encrypted and the matrix can either be known to the evaluator a priori or be encrypted. In domain specific jargon, these would be abbreviated PC-Mv (Plaintext-Ciphertext Matrix-vector) and CC-Mv (Ciphertext-Ciphertext Matrix-vector) multiplication, respectively. If the matrix is encrypted, either its columns or rows are encrypted in individual ciphertexts. If the matrix is known in plaintext, they implement a so-called systolic multiplication algorithm.

2. In 2018, Jian *et al.* [21] introduced a method to encode a matrix in one ciphertext vector, then showed how to perform encrypted matrix multiplication, CC-MM (Ciphertext-Ciphertext Matrix-Multiplication), along with matrix transposition and the ability to batch multiple matrices into one ciphertext. They use their construction to evaluate neural networks on encrypted inputs using the CKKS encryption scheme.

3. Also in 2018, Cheon *et al.* [22] presented a variant of CKKS where they can pack complex numbers into multidimensional tensors as multivariate polynomials. They base the security of this modified scheme on multivariate PLWE. This construction differs from the others in that it is a proposed modification of the CKKS scheme (thus partially an inside-out approach) and not just a packing method on top. However, it still requires complicated and expensive multiplications and data movements between plaintext slots.

As mentioned previously, all of the above approaches suffer from the need to perform many supplementary operations to achieve matrix multiplication in a computation model that is not inherently built for that algebraic structure. Oftentimes the number of these operations scales linearly or even quadratically with the dimension of the matrices. Here we do not discuss the computational complexity of the above algorithms in terms of multiplications, rotations, and automorphisms required, as these measures will not be applicable to our proposed technique. We refer the interested reader to [23, section 1].

Encryption schemes with matrix plaintext spaces have been proposed before. Some built on other frameworks (GSW [10]), while others had only limited homomorphic capabilities, or none at all. We list some of these works and their relevance to this work.

1. The first work to introduce a variant of LWE, where the plaintext space is extended to matrices over the underlying ring, similar to this work, is one by Peikert and Waters [24, section 6.2], then later also shown by Micciancio [25, definition 2.8, lemma 2.9] to be equivalent (in hardness) to the standard variant of LWE. This extension of LWE is fairly standard and natural. In the standard formulation of decision-LWE, we are asked to distinguish between a uniformly sampled tuple and a pseudorandom tuple, both of the form $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. In the pseudorandom case, $\mathbf{A}$ is uniformly sampled from $\mathbb{Z}_q^{m \times n}$, and $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$, where s is a secret vector uniformly from $\mathbb{Z}_q^n$, and $\mathbf{e}$ is a low-norm noise term from $\mathbb{Z}_q^m$. In the above works, they replace both $\mathbf{s}$

and $\mathbf{e}$, with matrices to get a matrix equation $\mathbf{B} = \mathbf{AS} + \mathbf{E}$. Peikert and Waters already identified that this construction is linearly homomorphic, meaning one can take linear combinations of ciphertexts to get encryptions of the linear combinations of the underlying plaintexts. Our work will build on this generalization, but for a more general variant of LWE, namely Module-LWE.

2. In 2010, Gentry *et al.* constructed a pre-FHE encryption scheme [26] along the lines of the previous matrix encryption technique. They use GPV trapdoors [27], where along with the public matrix $\mathbf{A}$, a trapdoor matrix $\mathbf{T}$ is also generated at key generation time, for which $\mathbf{AT} = \mathbf{0}$. The ciphertext encrypting a matrix M is $\mathbf{C} = \mathbf{AS}+\mathbf{E}+\mathbf{M}$, where $\mathbf{S}$ is a random matrix and $\mathbf{E}$ is again a low-norm noise term. To decrypt, one can multiply $\mathbf{C}$ by $\mathbf{T}$ from the left to annihilate $\mathbf{A}$. It is easy to verify that this scheme is additively homomorphic. To get one multiplication along with many additions, the authors employ a transposition technique, where they multiply $\mathbf{C}_1$ and $\mathbf{C}_2$ as $\mathbf{C}_1 \cdot \mathbf{C}_2^T$ to get an encryption of the message $\mathbf{M}_1 \cdot \mathbf{M}_2^T$. This is needed because matrix multiplication is noncommutative and if we want to annihilate both occurrences of $\mathbf{A}$ in the ciphertext after multiplication, we need to have the $\mathbf{A}$'s on the outsides of the term $\mathbf{AS}_1 \cdot (\mathbf{AS}_2)^T = \mathbf{AS}_1\mathbf{S}_2^T\mathbf{A}^T$. This scheme highlights the core problem with matrices, that is matrix multiplication is not commutative. They achieve one matrix multiplication with this transposition technique.

3. A recent work by Park [28] borrows the previous transposition technique and implements it in the context of Ciphertext-Ciphertext Matrix-Multiplication (CC-MM) with batched CKKS Ring-LWE ciphertexts, which makes it an outside-in technique. To implement the transposition technique, they construct a Ciphertext Matrix-Transpose (C-MT) algorithm ([28, section 3.3]), then they use this to multiply matrices. To make the scheme fully homomorphic, they perform a key switching operation to transform a term of the form $\mathbf{SA}_1\mathbf{A}_2^T\mathbf{S}^T$ into a term of the form $\mathbf{A}_3\mathbf{S}^2$ ([28, section 4.2]), which can then be relinearized.

4. Another scheme that generalizes the plaintext space along the same lines as this work is due to Hiromasa *et al.* [29], where they construct a matrix variant of GSW. Similar to this work, the native plaintext space of the scheme is no longer the ring of scalars (as in a binary value for GSW, or polynomial in this work), but a matrix ring (binary matrices in Hiromasa *et al.*, and a matrix ring with polynomial entries in this work), and ciphertext addition and multiplication correspond to homomorphic addition and multiplication in the matrix ring.

Contributions. We propose a generalization of the BGV, BFV, and CKKS homomorphic encryption schemes to operate over a (noncommutative) matrix ring, instead of a polynomial ring. At the heart of this work is a new superoperator technique 3.4 to deal with the noncommutativity of the matrix ring, as compared to the standard commutative polynomial ring in the standard Ring-LWE instantiations. This superoperator technique allows us to represent linear and quadratic expressions in the secret key as linear operators, which then opens up the possibility for relinearization. The modified schemes base their security on Module-LWE with module rank equal to the dimension of the square matrices. In a way, this newly introduced matrix dimension parameter is exactly as natural to the encryption schemes in question as Ring-LWE being a special case of Module-LWE with module rank equal to 1. This work presents a new direction, which allows trading the parallelism offered by Ring-LWE-based schemes (the ring dimension) for the algebraic structure needed in matrix multiplications (module rank).

1.2 Technical Overview

For a reference on notation used, see section 2.1, for a high-level description of the standard instantiations of BGV, BFV, and CKKS, see section 2.3. For the sake of brevity, we omit details and simplify things that are not essential, such as the different methods of encoding a message into a plaintext.

Matrix encryption. We are looking to generalize the main equation of Ring-LWE, and thus the emergent ciphertexts, from elements of the polynomial ring R_q to matrices $M_d(R_q)$. Since dimension d square matrices form a ring, this generalization can be viewed as a noncommutative Ring-LWE over the matrix ring $M_d(R_q)$. The standard Ring-LWE instantiation can be viewed as a 1×1 matrix or scalar case of this general construction.

$$\text{(Ring-LWE encryption)} \quad (-\mathbf{a} \cdot \mathbf{s} + \mathbf{e} + \mathbf{m}, \mathbf{a}) \in R_q \times R_q \tag{1}$$

$$\text{(Module-LWE matrix encryption)} \quad (-\mathbf{A} \cdot \mathbf{S} + \mathbf{E} + \mathbf{M}, \mathbf{A}) \in M_d(R_q) \times M_d(R_q) \tag{2}$$

In the matrix case, both $\mathbf{A}$ and $\mathbf{S}$ become d number of R_q-vectors concatenated together [1]. The security of this matrix encryption can easily be based on Module-LWE with module rank d. Note that since we essentially reuse every vector of both $\mathbf{A}$ and $\mathbf{S}$ d times in calculating the entries of their product, this dimension increase from Ring-LWE to Module-LWE comes for free, as it does not just increase the ciphertext size, but also the plaintext space. Contrast this with standard LWE-based (or more generally Module-LWE-based) encryptions, where the ciphertext is in $\mathbb{Z}_q^{d+1}$, but the message is just encoded as one element from $\mathbb{Z}_q$. With this view, Ring-LWE, Module-LWE, and standard LWE are equally efficient, only for different sizes of matrices, and we can freely interpolate between Ring-LWE and LWE. This becomes even more interesting when we show that the per multiplication noise growth is identical if we keep the effective lattice dimension (and thus security level) constant.

Encoding. The encoding procedures specific to the actual BGV-like encryption scheme can be performed entry-by-entry, in which case the unencoded message matrix would come from $M_d(R_t)$, and would produce an encoded matrix in $M_d(R_q)$. This is due to the fact that the encoding/decoding procedures reduce to scalar multiplication of matrices (which commutes with matrix multiplication) and entry-by-entry rounding or modular reduction procedures.

Homomorphic properties. This construction is already linearly homomorphic, as pointed out by Peikert and Waters [24, section 6.2]. The problem comes with multiplication, as matrix multiplication is noncommutative. First we write the ciphertext as a polynomial in X.

$$C(\mathbf{X}) = \mathbf{C}_0 + \mathbf{C}_1 X = (-\mathbf{A} \cdot \mathbf{S} + \mathbf{E} + \mathbf{M}) + \mathbf{A}X \tag{3}$$

In the Ring-LWE case, we depend on the commutativity of the underlying ring to combine all linear terms into one, then more crucially, to get a quadratic term in the form $\mathbf{c}_2\mathbf{x}^2$, which can then be evaluated with one multiplication by $\mathbf{s}^2$ ([2]). These will not work in the noncommutative case.

$$\begin{aligned} C(X) \cdot D(X) &= (\mathbf{C}_0 + \mathbf{C}_1 X) \cdot (\mathbf{D}_0 + \mathbf{D}_1 X) \\ &= \mathbf{C}_0\mathbf{D}_0 + \mathbf{C}_0\mathbf{D}_1 X + \mathbf{C}_1 X\mathbf{D}_0 + \mathbf{C}_1 X\mathbf{D}_1 X \end{aligned} \tag{4}$$

[1] Row vectors in the case of $\mathbf{A}$, column vectors with $\mathbf{S}$.

[2] Technically a dot product, but it is morally one multiplication.

Now the product still gives a valid polynomial that decrypts to the correct result, however, we cannot collapse the linear terms into one, which would make the ciphertext grow over time, and the quadratic term needs two substitutions of $\mathbf{S}$, which actually prevents relinearization.

Superoperator technique. The trick is to realize that the linear terms together can be described as a linear superoperator acting on X, whereas the quadratic term is a linear superoperator acting on $X \otimes X$.

$$\begin{aligned} &\mathbf{C}_0\mathbf{D}_0 + \mathbf{C}_0\mathbf{D}_1 X + \mathbf{C}_1 X \mathbf{D}_0 + \mathbf{C}_1 X \mathbf{D}_1 X \\ &= \mathbf{C}_0\mathbf{D}_0 + \Psi_1(X) + \Psi_2(X \otimes X) \end{aligned} \tag{5}$$

$$\begin{aligned} \Psi_1(\cdot): \quad & M_d(R_q) \to M_d(R_q) \\ & X \mapsto (\underbrace{I \otimes \mathbf{C}_0\mathbf{D}_1 + \mathbf{D}_0^T \otimes \mathbf{C}_1}_{\Psi_1 \in M_{d^2}(R_q)})X \end{aligned} \tag{6}$$

$$\begin{aligned} \Psi_2(\cdot): \quad & M_d(R_q) \otimes M_d(R_q) \to M_d(R_q) \\ & X \otimes X \mapsto \underbrace{\mathbf{M}_\mu(\mathbf{C}_1 \otimes \mathbf{D}_1)}_{\Psi_2 \in M_{d^2 \times d^4}(R_q)}(X \otimes X) \end{aligned} \tag{7}$$

The second equation involves the multiplication operator $\mathbf{M}_\mu \in M_{d^2 \times d^4}(R_q)$, which collapses a tensor product, $\mathbf{M}_\mu \cdot (A \otimes B) = AB$. The operators can easily be verified to give the correct result, and both operators can be described with matrices given the ciphertext coefficients. With this modification, a general ciphertext will be a tuple in $M_d(R_q) \times M_{d^2}(R_q)$. Note that after several rounds of multiplications, the linear term is the sum of many tensor products and should not be thought of as this sum, but an actual matrix in $M_{d^2}(R_q)$, but initial ciphertexts can always be written as a tuple in $M_d(R_q) \times M_d(R_q)$. We would like to emphasize that this is optimal for the encryption of a matrix in $M_d(R_t)$, and matches the useful information density of the Ring-LWE case.

Relinearization. Since we reduced the quadratic term to a matrix-vector multiplication, $\Psi_2 \cdot (\mathrm{vec}(X) \otimes \mathrm{vec}(X))$, we can now provide a masked version of $\mathrm{vec}(\mathbf{S}) \otimes \mathrm{vec}(\mathbf{S})$, analogously to the masked version of $\mathbf{s}^2$ used in the standard formulations. We give the blueprint for a trivial but inefficient relinearization key, the size of which scales with d^4. This is a first attempt at relinearization and we expect that the rich algebraic structure would allow for significant optimizations.

Identical noise growth. Any norm we can define on our polynomial ring R, we can extend to the matrix ring $M_d(R)$, by defining the norm of a matrix to be the maximum of its entries' norms. This is the right approach for the infinity norm, as a low infinity norm for a noise term matrix means that each of its entries has a low-norm, which in turn means that each entry of the message matrix can be decoded correctly.

We can also extend the ring expansion factor δ_{R_q} to the matrix ring, $\delta_{M_d(R_q)}$. The ring expansion factor, introduced in [30], tells us the maximum ratio by which the norm increases when we reduce a polynomial in a quotient polynomial ring.

$$\delta_R = \max\{||\mathbf{a} \cdot \mathbf{b}||_\infty / (||\mathbf{a}||_\infty \cdot ||\mathbf{b}||_\infty) : \mathbf{a}, \mathbf{b} \in R\} \tag{8}$$

For the ambient ring $R = \mathbb{Z}[x]/(x^N + 1)$ used in Ring-LWE, $\delta_R = N$ with respect to the infinity norm. Note that this expansion factor is a worst-case value, whereas we can bound the noise expansion by $2 \cdot \sqrt{N}$ in practical applications [31, section 6.1].

An important conclusion we reach is that for the matrix ring, $\delta_{M_d(R_q')} = d \cdot N'$, where d is the module rank (or equivalently the dimension of the matrices), and N' is the dimension of the underlying polynomial ring $R_q' = \mathbb{Z}_q[x]/(x^{N'}+1)$. What this means is that if we transition from a Ring-LWE-based instantiation to our Module-LWE-based matrix instantiation, and in the process we choose the parameters so that $d \cdot N' = N$ ([3]), which just means that we keep the effective lattice dimension constant [4], then we also preserve the ring expansion factor. We emphasize that in the Ring-LWE case, the expansion factor applies to a polynomial-polynomial multiplication in R_q, whereas in the Module-LWE matrix case, it is a matrix-matrix multiplication in $M_d(R_q')$.

2 Preliminaries

2.1 Basic notation.

We refer to the ring $R = \mathbb{Z}[x]/(x^N+1)$ as the ambient ring, where N is a power of two. With N a power of two, x^N+1 is a cyclotomic polynomial, and its roots over the field of complex numbers are the primitive $2N$-th roots of unity. $\mathbb{Z}_q$ refers to the zero-centered set of integers $(-q/2, q/2]$, and $R_q = R/qR = \mathbb{Z}_q[x]/(x^N+1)$. Note that reduction to the set $(-q/2, q/2]$ after operations in R_q is implicit.

Elements of R are denoted with lowercase bold letters; for example $\mathbf{a} \in R$. The infinity norm of an element is denoted by $||\cdot||_\infty$, and is defined to be the maximum absolute value among the coefficients of a polynomial. The expansion factor of R is $\delta_R = \max\{||\mathbf{a}\cdot\mathbf{b}||_\infty / (||\mathbf{a}||_\infty \cdot ||\mathbf{b}||_\infty) : \mathbf{a}, \mathbf{b} \in R\}$.

For $a \in \mathbb{Z}$, we use the notation $[a]_q$ to indicate the representative of a modulo q in $\mathbb{Z}_q$. For $\mathbf{a} \in R$, $[\mathbf{a}]_q$ means coefficient-wise reduction. The operations $\lfloor\cdot\rfloor$, $\lceil\cdot\rceil$, and $\lfloor\cdot\rceil$ round down, up, and to the nearest integer, respectively.

The set of $n \times m$ matrices with elements from a ring Q is denoted as $M_{n\times m}(Q)$, whereas $M_d(Q)$ is the $d \times d$ matrix ring with elements from Q. Uppercase bold letters are used for matrices; for example $\mathbf{A} \in M_d(R_q)$. All previously defined operations should be defined for matrices, performed entry-by-entry.

To sample uniformly an element x from a set S, the notation $x \leftarrow S$ is used. When a distribution like χ is explicitly described, the notation $x \leftarrow \chi$ means random sampling according to the distribution χ.

For an element $a \in \mathbb{Z}_q$, Base2Decomp(a) denotes the binary decomposition vector of a, and similarly for a polynomial $\mathbf{p}$, Base2Decomp($\mathbf{p}$) denotes the vector of polynomials, with coefficients from the binary decomposition of $\mathbf{p}$'s coefficients.

We distinguish between an unencoded message element, denoted $\mathbf{m}$ or $\mathbf{M}$, and the encoded plaintext element $\hat{\mathbf{m}}$ or $\hat{\mathbf{M}}$. In addition, we use the notation $\text{enc}_{(\mathbf{s})}(\mathbf{m})$ for an encryption of the message $\mathbf{m}$ under the secret key $\mathbf{s}$, whereas $\text{mask}_{(\mathbf{s})}(\mathbf{v})$ denotes a mask-like encryption of an unencoded value $\mathbf{v}$ (see equation 15).

The vectorization function $\text{vec}(\cdot) : M_d(R) \to R^{d^2}$ produces a column vector from a square matrix by stacking its column on top of each other, starting with the left-most column on the top. The inverse vectorization function $\text{vec}^{-1}(\cdot) : R^{d^2} \to M_d(R)$ does the same thing in reverse.

2.2 LWE, Module-LWE, and Ring-LWE.

In this section we recall the hard problems underlying the standard variants of the encryption schemes in question, BGV [5], BFV [6, 7], and CKKS [13]. We define the most

[3] Here N' is the ring dimension of the Module-LWE instance, and N is the Ring-LWE ring dimension.
[4] This roughly means that we keep the security level constant.

general case, namely Module-LWE [32], then show that both LWE [14] and Ring-LWE [17] are special cases of Module-LWE.

Definition 2.1 (Decision Module-LWE ($\mathsf{MLWE}_{N,d,q,\chi_s,\chi_e}$))**.** Fix the following integers: the ring dimension N, the module rank d, the modulus q. Also fix two distributions χ_s and χ_e, both over R. These are the secret key and noise distributions, respectively.

Let $\mathbf{s} \leftarrow \chi_s^d$. The $\mathsf{MLWE}_{N,d,q,\chi_s,\chi_e}$ problem asks to distinguish between two distributions:

1. Sample $\mathbf{a} \leftarrow R_q^d$, $\mathbf{e} \leftarrow \chi_e$, and return
$$(\mathbf{a}, \mathbf{b} := \langle \mathbf{a}, \mathbf{s} \rangle + \mathbf{e})$$
2. Sample $\mathbf{a} \leftarrow R_q^d$, $\mathbf{b} \leftarrow R_q$, and return
$$(\mathbf{a}, \mathbf{b})$$

The **Decision Module-LWE assumption** (with parameters $(N, d, q, \chi_s, \chi_e)$) states that without the knowledge of $\mathbf{s}$, no polynomial-time adversary can distinguish the two distributions significantly better than random guessing.

As we can see, decision Module-LWE asks us to distinguish between noisy dot products with a secret value, and uniformly random tuples.

Definition 2.2 (Decision Ring-LWE ($\mathsf{RLWE}_{N,q,\chi_s,\chi_e}$))**.** Decision Ring-LWE is the instantiation of Decision Module-LWE, with module rank $d = 1$.

$$\mathsf{RLWE}_{N,q,\chi_s,\chi_e} = \mathsf{MLWE}_{N,1,q,\chi_s,\chi_e}$$

The **Decision Ring-LWE assumption** is equivalent to the respective Decision Module-LWE assumption.

The decision Ring-LWE problem asks us to distinguish between noisy products with a secret polynomial $\mathbf{s}$, and uniformly random pairs of elements from R_q.

Definition 2.3 (Decision LWE ($\mathsf{LWE}_{d,q,\chi_s,\chi_e}$))**.** Decision Ring-LWE is the instantiation of Decision Module-LWE, with ring dimension $N = 1$.

$$\mathsf{LWE}_{d,q,\chi_s,\chi_e} = \mathsf{MLWE}_{1,d,q,\chi_s,\chi_e}$$

The **Decision LWE assumption** is equivalent to the respective Decision Module-LWE assumption.

The decision LWE problem asks us to distinguish between noisy dot products with a secret value (this time over $\mathbb{Z}_q$), and uniformly random tuples.

Effective lattice dimension. The LWE and Ring-LWE problems are each extremal cases of the general Module-LWE problem along different dimensions. One is the ring

dimension N, which is minimized in the LWE case. The other is the module rank d, which is minimized in the Ring-LWE case. Once the rest of the parameters are fixed, the hardness of Module-LWE scales with $d \cdot N$ [33], which we will refer to as the *effective lattice dimension*. Note that a larger rank d and lower dimension N gives rise to a less algebraically structured underlying lattice in the reductions to hard problems in lattices. To the best of the author's knowledge, techniques that exploit this structure are not known [34, section 2.1, Related problems], except for the most pathological of cases [35, 36], and the hardness of a Module-LWE problem with parameters N and d is taken to be the hardness of the LWE problem with $N' = d \cdot N$.

2.3 Standard formulations of BGV, BFV, CKKS.

Typical instantiations of the BGV [4, 5], BFV [6, 7], and CKKS [13] encryption schemes base their security on Decision Ring-LWE. The ciphertext structure and homomorphic operations of the three schemes can be generalized to an abstract scheme. In this section, we describe this abstract scheme. The three schemes differ in their methods of encoding a message into a plaintext, and also in their management of the noise term. For our generalization, these differences will be irrelevant. For simplicity, we also only describe the secret key formulation of the schemes.

Ciphertext format. Briefly, to encrypt a message $\mathbf{m} \in R_t$ under the secret key $\mathbf{s} \in R_q$, depending on the specific scheme, one encodes $\mathbf{m}$ to get $\hat{\mathbf{m}} \in R_q$, then samples $\mathbf{a} \leftarrow R_q$ and $\mathbf{e} \leftarrow \chi_e$, then constructs the following tuple.

$$(\mathbf{c}_0, \mathbf{c}_1) = (-\mathbf{a} \cdot \mathbf{s} + \mathbf{e} + \hat{\mathbf{m}}, \mathbf{a}) \in R_q \times R_q \tag{9}$$

The modulus q is referred to as the ciphertext modulus, while t is the plaintext modulus. To decrypt, one computes a linear combination with the secret key $\mathbf{s}$.

$$\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} = -\mathbf{a} \cdot \mathbf{s} + \mathbf{e} + \hat{\mathbf{m}} + \mathbf{a} \cdot \mathbf{s} = \mathbf{e} + \hat{\mathbf{m}} \approx \hat{\mathbf{m}} \tag{10}$$

Message encoding. Each scheme leverages a different encoding procedure to get rid of the small-norm noise term $\mathbf{e}$, and get the message $\mathbf{m}$.

In the BGV case, $\hat{\mathbf{m}} = \mathbf{m}$ and $\mathbf{e}$ is divisible by t, the plaintext modulus. This way, decoding is just reduction modulo t.

$$[t\mathbf{e} + \hat{\mathbf{m}}]_t = \mathbf{m} \tag{11}$$

In the BFV case, $\hat{\mathbf{m}} = \Delta\mathbf{m}$, where $\Delta = \lfloor q/t \rfloor$. Decoding is multiplication by t/q followed by a rounding.

$$[\lfloor t/q \cdot (\Delta\mathbf{m} + \mathbf{e}) \rceil]_t = \mathbf{m} \tag{12}$$

As we can see, BGV leaves the message unchanged and scales up the noise term, while BFV scales up the message. They can be thought of as storing the message in the least significant or most significant bits of the plaintext, but in both cases the message and noise need to be separated at all times.

In contrast, CKKS does not separate the message and the noise in the plaintext polynomial $\hat{\mathbf{m}}$. In fact, the message polynomial $\mathbf{m}$ is from R_q and it is already an approximate representation of the underlying vector of complex elements, then the encoding step $\hat{\mathbf{m}} = \mathbf{m} + \mathbf{e}$ just makes this approximation a little worse.

Relinearization. To make the addition and especially the multiplication operation more intuitive, we can also describe a ciphertext as a polynomial $c(x) = \mathbf{c}_0 + \mathbf{c}_1 x \in R_q[x]$, where decryption is the evaluation of this polynomial at the secret key $\mathbf{s}$.

$$c(s) = \mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} \approx \mathbf{m} \tag{13}$$

With this in mind, ciphertext addition and multiplication are just polynomial addition and multiplication. After both operations, evaluation at $\mathbf{s}$ gives the sum or product of the messages, with the noise term having grown in the process. However, with multiplication, the degree of our ciphertext went up from one to two, and now we have a quadratic ciphertext.

$$c(x) = \mathbf{c}_0 + \mathbf{c}_1 x + \mathbf{c}_2 x^2 \tag{14}$$

This is not a problem in itself, as the new quadratic ciphertext can still be evaluated at $\mathbf{s}$ to decrypt, but it does not match our original ciphertext format, and with additional multiplications the degree of the ciphertext would grow exponentially. For this reason a relinearization step is performed after each multiplication, which makes use of a relinearization key $\texttt{rlk}$, which allows us to evaluate the quadratic monomial with a masked version of $\mathbf{s}^2$. The key consists of a list of linear polynomials, each of which mask $\mathbf{s}^2$ scaled by powers of 2 (or optionally any other base) [7, section 4, Relinearisation: Version 1]. Note that these are strictly speaking not encryptions of $\mathbf{s}^2$ scaled, since these values have not been encoded, similarly to the plaintext space of CKKS. Let $l = \lceil \log_2(q) \rceil$.

$$\texttt{rlk} = [\underbrace{(-\mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i + 2^i \cdot \mathbf{s}^2, \mathbf{a}_i)}_{\text{mask}_{(\mathbf{s})}(2^i \cdot \mathbf{s}^2)} : 0 \leq i < l] = [\text{rlk}_i(x) : 0 \leq i < l] \tag{15}$$

$$\text{rlk}_i(\mathbf{s}) = \mathbf{e}_i + 2^i \cdot \mathbf{s}^2 \tag{16}$$

To transform our quadratic polynomial back into a linear one, we use the masked $\mathbf{s}^2$ value to evaluate the quadratic term.

$$c'(x) = \mathbf{c}_0 + \mathbf{c}_1 x + \langle \text{Base2Decomp}(\mathbf{c}_2), \texttt{rlk} \rangle \tag{17}$$

Notice that $\sum \text{Base2Decomp}(\mathbf{c}_2)[i] \cdot 2^i = \mathbf{c}_2$, and thus the dot product gives a noisy evaluation of the quadratic term. The base 2 decomposition is needed to manage the noise introduced from the product of the random R_q element $\mathbf{c}_2$ and the noise terms in the relinearization key $\texttt{rlk}$. Also notice that the dot product produces a linear polynomial, which summed with the linear part of $c'(x)$ again produces a linear polynomial.

3 Matrix–BGV/BFV/CKKS

3.1 Matrix encryption.

Now we present our baseline matrix encryption scheme along the lines of [24, section 6.2]. The encryption scheme is defined as follows.

1. $\mathsf{MatEnc.SecKeyGen}(1^\lambda)$: For $0 \leq i < d$ sample $\mathbf{s}_i \leftarrow \chi_s^d$, and output $\texttt{sk} = (\mathbf{s}_0 | \ldots | \mathbf{s}_{d-1})$.
2. $\mathsf{MatEnc.SecKeyEnc}(\texttt{sk}, \mathbf{M})$: To encrypt a message matrix $\mathbf{M} \in M_d(R_t)$, first encode the matrix into the plaintext matrix $\hat{\mathbf{M}} \in M_d(R_q)$ [5], then sample $\mathbf{A} \leftarrow M_d(R_q)$, $\mathbf{E} \leftarrow \chi_e^{d \times d}$. Let $\mathbf{S} = \texttt{sk}$. Output the following tuple.

$$\texttt{ct} = (-\mathbf{AS} + \mathbf{E} + \hat{\mathbf{M}}, \mathbf{A}) \in M_d(R_q) \times M_d(R_q) \tag{18}$$

[5] The encoding procedure is inherited from either BGV, BFV, or CKKS, performed entry-by-entry.

3. MatEnc.Dec(sk, ct): Let $\mathbf{S} = \mathtt{sk}$, and $(\mathbf{C}_0, \mathbf{C}_1) = ct$. First, take the following linear combination of the ciphertext tuple and the secret key.

$$\hat{\mathbf{M}}' = \mathbf{C}_0 + \mathbf{C}_1\mathbf{S} \tag{19}$$

Perform the decoding procedure on the matrix $\hat{\mathbf{M}}$' to get the message matrix $\mathbf{M}$'. Output $\mathbf{M}$'.

Note that this secret key encryption scheme can be turned into a public key encryption scheme along the lines of Regev's original construction [14], often referred to as primal Regev. To do that, we can augment our encryption scheme with two additional functions.

4. MatEnc.PubKeyGen(sk): Let $\mathbf{S} = \mathtt{sk}$. Sample $\mathbf{A} \leftarrow M_d(R_q)$, $\mathbf{E} \leftarrow \chi_e^{d\times d}$. Output the following tuple.

$$\mathtt{pk} = (-\mathbf{AS} + \mathbf{E}, \mathbf{A}) \in M_d(R_q) \times M_d(R_q) \tag{20}$$

5. MatEnc.PubKeyEnc(pk, $\mathbf{M}$): Let $(\mathbf{P}_0, \mathbf{P}_1) = \mathtt{pk}$. To encrypt a message matrix $\mathbf{M} \in M_d(R_t)$, first encode the matrix into the plaintext matrix $\hat{\mathbf{M}} \in M_d(R_q)$, then sample $\mathbf{U} \leftarrow \chi_s^{d\times d}$, $\mathbf{E}_1, \mathbf{E}_2 \leftarrow \chi_e^{d\times d}$. Output the following tuple.

$$\mathtt{ct} = (\mathbf{UP}_0 + \mathbf{E}_1 + \hat{\mathbf{M}}, \mathbf{UP}_1 + \mathbf{E}_2) \tag{21}$$

The hardness results of the BGV-like schemes can be adopted to the decision Module-LWE problem.

Theorem 3.1. *The IND-CPA security of the above scheme MatEnc can be based on the hardness of (decision)* $\mathsf{MLWE}_{N,d,q,\chi_s,\chi_e}$.

Proof. The indistinguishability argument is identical to that of [24, lemma 6.2] and is therefore omitted. See also [25, lemma 2.9]. □

3.2 Homomorphic properties.

Our matrix encryption scheme already defines a somewhat homomorphic encryption scheme [37, section 2.1.1] along the lines of the BGV-like schemes, with polynomial-like addition and multiplication of ciphertext evaluating the sum and product of the underlying plaintext elements. Having said that, until we define a relinearization procedure, it is only a non-compact [37, section 2.2.2] somewhat homomorphic encryption scheme.

Theorem 3.2. *The encryption scheme MatEnc 3.1, along with polynomial-like addition and multiplication of ciphertexts, defines a* ***non-compact somewhat homomorphic encryption scheme.***

Proof. To verify the homomorphic properties, we identify initial ciphertexts with linear polynomials in $M_d(R_q)[X]$.

$$C(X) = \mathbf{C}_0 + \mathbf{C}_1 X = (-\mathbf{A}_1\mathbf{S} + \mathbf{E}_1 + \hat{\mathbf{M}}_1) + \mathbf{A}_1 X \quad \text{(Ciphertext 1)}$$

$$D(X) = \mathbf{D}_0 + \mathbf{D}_1 X = (-\mathbf{A}_2\mathbf{S} + \mathbf{E}_2 + \hat{\mathbf{M}}_2) + \mathbf{A}_2 X \quad \text{(Ciphertext 2)}$$

The decryption procedure is equivalent to the evaluation map $\mathrm{eval}_{(\mathbf{S})}(\cdot) : M_d(R_q)[X] \rightarrow$

$M_d(R_q)$.

$$C(\mathbf{S}) = \mathbf{C}_0 + \mathbf{C}_1\mathbf{S} = -\mathbf{A}_1\mathbf{S} + \mathbf{E}_1 + \hat{\mathbf{M}}_1 + \mathbf{A}_1\mathbf{S} = \mathbf{E}_1 + \hat{\mathbf{M}}_1$$

$$D(\mathbf{S}) = \mathbf{D}_0 + \mathbf{D}_1\mathbf{S} = -\mathbf{A}_2\mathbf{S} + \mathbf{E}_2 + \hat{\mathbf{M}}_2 + \mathbf{A}_2\mathbf{S} = \mathbf{E}_2 + \hat{\mathbf{M}}_2$$

Since the evaluation map is an R_q-algebra homomorphism, the decryption function (evaluation map) and R_q-algebra operations commute. This implies the somewhat-homomorphic property. The scheme is non-compact, since the degree of the polynomials in $M_d(R_q)[X]$ can grow exponentially with the number of multiplication operations. □

However, there is a problem with generalizing the ciphertext-ciphertext multiplication to the matrix ring. Specifically, the Ring-LWE-based constructions rely on the commutativity of the polynomial ring in the relinearization step. As long as we cannot perform relinearization, we are stuck with a non-compact homomorphic encryption scheme.

We can verify that the polynomial multiplication of ciphertexts results in two linear terms and a quadratic term, where the quadratic term is of the general form $\mathbf{A}X\mathbf{B}X$.

$$\begin{aligned} C(X) \cdot D(X) &= (\mathbf{C}_0 + \mathbf{C}_1 X) \cdot (\mathbf{D}_0 + \mathbf{D}_1 X) \\ &= \mathbf{C}_0\mathbf{D}_0 + \mathbf{C}_0\mathbf{D}_1 X + \mathbf{C}_1 X\mathbf{D}_0 + \mathbf{C}_1 X\mathbf{D}_1 X \end{aligned} \tag{22}$$

Both the linear terms and quadratic term cause problems. The first problem is that we cannot collapse all linear terms into one, and with later multiplications their number would only increase. This is not desirable if we wish to satisfy a compactness property. The problem with the quadratic term is even worse, as its form inhibits relinearization. If the quadratic term involves two linear occurences of X, then due to noncommutativity, we would have to substitute in the secret key $\mathbf{S}$ twice. Since relinearization, which is as evaluation with a quasi-encryption of the secret value, involves a multiplication between a constant (usually $\mathbf{c}_1$) and a linear polynomial, the result in the usual case is a linear polynomial. With two substitutions however, our relinearization procedure would again involve a ciphertext-ciphertext multiplication, which defeats the purpose.

3.3 The superoperator technique.

Our main result is showing that we can represent the linear terms and the quadratic term with operator-valued linear maps (superoperators) Ψ_1 and Ψ_2 acting on the indeterminate X.

$$\begin{aligned} &\mathbf{C}_0\mathbf{D}_0 + \mathbf{C}_0\mathbf{D}_1 X + \mathbf{C}_1 X\mathbf{D}_0 + \mathbf{C}_1 X\mathbf{D}_1 X \\ &= \mathbf{C}_0\mathbf{D}_0 + \Psi_1(X) + \Psi_2(X \otimes X) \end{aligned} \tag{23}$$

$$\begin{aligned} \Psi_1(\cdot): \quad & M_d(R_q) \to M_d(R_q) \\ & X \mapsto \mathrm{vec}^{-1}((\underbrace{I \otimes \mathbf{C}_0\mathbf{D}_1 + \mathbf{D}_0^T \otimes \mathbf{C}_1}_{\Psi_1 \in M_{d^2}(R_q)})\mathrm{vec}(X)) \end{aligned} \tag{24}$$

$$\begin{aligned} \Psi_2(\cdot): \quad & M_d(R_q) \otimes M_d(R_q) \to M_d(R_q) \\ & X \otimes X \mapsto \mathrm{vec}^{-1}(\underbrace{\mathbf{M}_\mu(\mathbf{C}_1 \otimes \mathbf{D}_1)}_{\Psi_2 \in M_{d^2 \times d^4}(R_q)}(\mathrm{vec}(X) \otimes \mathrm{vec}(X))) \end{aligned} \tag{25}$$

We use the vectorization notation $\mathrm{vec}(\cdot)$ to highlight that the superoperators act on the space of operators with matrix-vector multiplication. The two resulting operators can be expressed with tensor products between the ciphertext coefficients.

$$
\begin{aligned}
&\text{vec}^{-1}((I \otimes \mathbf{C}_0\mathbf{D}_1 + \mathbf{D}_0^T \otimes \mathbf{C}_1)\text{vec}(X)) \\
&= \text{vec}^{-1}((I \otimes \mathbf{C}_0\mathbf{D}_1)\text{vec}(X) + (\mathbf{D}_0^T \otimes \mathbf{C}_1)\text{vec}(X)) \\
&= \text{vec}^{-1}(\text{vec}(\mathbf{C}_0\mathbf{D}_1X) + \text{vec}(\mathbf{C}_1X\mathbf{D}_0)) \\
&= \text{vec}^{-1}(\text{vec}(\mathbf{C}_0\mathbf{D}_1X)) + \text{vec}^{-1}(\text{vec}(\mathbf{C}_1X\mathbf{D}_0)) \\
&= \mathbf{C}_0\mathbf{D}_1X + \mathbf{C}_1X\mathbf{D}_0
\end{aligned}
\tag{26}
$$

$$
\begin{aligned}
&\text{vec}^{-1}(\mathbf{M}_\mu(\mathbf{C}_1 \otimes \mathbf{D}_1)(\text{vec}(X) \otimes \text{vec}(X))) \\
&= \text{vec}^{-1}(\mathbf{M}_\mu(\mathbf{C}_1\text{vec}(X) \otimes \mathbf{D}_1\text{vec}(X))) \\
&= \text{vec}^{-1}(\mathbf{M}_\mu(\text{vec}(\mathbf{C}_1X) \otimes \text{vec}(\mathbf{D}_1X)) \\
&= \text{vec}^{-1}(\text{vec}(\mathbf{C}_1X\mathbf{D}_1X)) \\
&= \mathbf{C}_1X\mathbf{D}_1X
\end{aligned}
\tag{27}
$$

If the vectorization of a matrix (e.g. $\mathbf{C}_0\mathbf{D}_1X$) is given in the appropriate encrypted form [6], we can perform the inverse vectorization operation $\text{vec}^{-1}(\cdot)$. We can also perform the matrix-vector product between a superoperator and an encrypted vectorized matrix (e.g., $\text{vec}(\mathbf{S}) \otimes \text{vec}(\mathbf{S})$), and then the inverse vectorization operation, given the encrypted vectorized matrix is well formed. In the next section we show initial attempts for performing this plaintext-ciphertext matrix-vector multiplication between a superoperator and an encrypted [7] version of $\text{vec}(\mathbf{S}) \otimes \text{vec}(\mathbf{S})$.

Generalized ciphertext format. We can observe that the initial ciphertexts can be represented as tuples from $M_d(R_q) \times M_d(R_q)$, but after performing multiplications (with relinearization, see next section), our ciphertexts become a tuple in $M_d(R_q) \times M_{d^2}(R_q)$.

$$
\mathtt{ct} = (\mathbf{C}_0, \mathbf{C}_1) \in M_d(R_q) \times M_{d^2}(R_q) \quad \text{(General ciphertext format)} \tag{28}
$$

Note that this general ciphertext definition includes the initial ciphertexts with a slight modification.

$$
\mathtt{ct}_{\text{initial}} = (\mathbf{C}_0, \mathbf{C}_1) = (-\mathbf{AS} + \mathbf{E} + \hat{\mathbf{M}}, \mathbf{I} \otimes \mathbf{A}) \quad \text{(Modified initial format)} \tag{29}
$$

Of course, this tensoring with the identity can be dropped for efficiency reasons. This ciphertext space still forms and R_q-module, thus the execution of linear operations is unchanged. The polynomial-like multiplication can also be adapted with the following small adjustments.

$$
\begin{aligned}
&(\mathbf{C}_0 + \mathbf{C}_1X) \cdot (\mathbf{D}_0 + \mathbf{D}_1X) = \\
&\quad \mathbf{C}_0\mathbf{D}_0 + ((\mathbf{I} \otimes \mathbf{C}_0)\mathbf{D}_1 + (\mathbf{D}_0^T \otimes \mathbf{I})\mathbf{D}_1)X + \mathbf{M}_\mu(\mathbf{C}_1 \otimes \mathbf{D}_1)(\text{vec}(\mathbf{X}) \otimes \text{vec}(\mathbf{X}))
\end{aligned}
\tag{30}
$$

3.4 Naive relinearization.

In this section, we describe a simple but inefficient relinearization technique for the homomorphic matrix encryption scheme.

As we have seen, the previous technique reduces relinearization to a plaintext-ciphertext matrix-vector multiplication between the unencrypted rectangular matrix $\mathbf{C}_2 \in M_{d^2 \times d^4}(R_q)$,

[6] Usually it will involve many ciphertexts encrypting the vector in pieces.

[7] Technically, it is not an actual encryption of $\text{vec}(\mathbf{S} \otimes \mathbf{S})$, but a masked version analogously to 2.3.

and an encrypted[8] version of $\mathbf{v} = \text{vec}(\mathbf{S}) \otimes \text{vec}(\mathbf{S}) \in R_q^{d^4}$. Of course, we have to fit the vector $\mathbf{v}$ into plaintext matrices of size $d \times d$. We give a few equations for notation purposes.

$$C(X) = \mathbf{C}_0 + \mathbf{C}_1 X + \mathbf{C}_2(X \otimes X) \quad \text{(Quadratic ciphertext)} \tag{31}$$

$$\text{vec}^{-1}(\underbrace{\mathbf{C}_2 \cdot \mathbf{v}}_{\mathbf{q} \in R_q^{d^2}}) = \underbrace{\mathbf{C}_2(\mathbf{S} \otimes \mathbf{S})}_{\mathbf{Q} \in M_d(R_q)} \quad \text{(Unencrypted evaluation)} \tag{32}$$

$$\text{vec}^{-1}(\mathbf{C}_2 \cdot \text{mask}_{(\mathbf{S})}(\mathbf{v})) = \text{enc}_{(\mathbf{S})}(\mathbf{C}_2(\mathbf{S} \otimes \mathbf{S})) \quad \text{(Relinearization equation)} \tag{33}$$

The best way to think about this matrix-vector multiplication is to go through the steps in reverse. What we want to get at the end is a single ciphertext encrypting the evaluated quadratic monomial. To get this ciphertext, we will need to perform an inverse vectorization $\text{vec}^{-1}(\mathbf{q})$ on the matrix vector product, in the encrypted domain.

Inverse vectorization. The easiest situation we can hope for is that as a result of the plaintext-ciphertext matrix-vector multiplication, $\mathbf{q}$ is encrypted section-by-section into d number of ciphertexts, where the top section of length d from $\mathbf{q}$ is the first column of the first ciphertext (with the rest of the columns all zero), the second section from the top is the second column of the second ciphertext, and so on. In this situation, summing the ciphertexts results in an inverse vectorization. Let $\mathbf{Q}^{(i)}$ denote the i-th column of the matrix $\mathbf{Q}$ (equivalently the i-th section of $\mathbf{q}$), with indexing starting from 0. Let $\mathbf{0}$ denote a column vector of length d with all zero entries, and let $\mathbf{0}^n$ denote n number of these columns stacked together horizontally.

$$\underbrace{(\mathbf{Q}^{(0)}|\mathbf{0}^{d-1}) + (\mathbf{0}|\mathbf{Q}^{(1)}|\mathbf{0}^{d-2}) + \cdots + (\mathbf{0}^{d-1}|\mathbf{Q}^{(d-1)})}_{\text{inverse vectorization: } \text{vec}^{-1}(\mathbf{q})} = \mathbf{Q} \tag{34}$$

Our aim will be to perform this summation in the encrypted domain on ciphertexts.

Matrix-vector product block-by-block. To get d ciphertexts of the above form, we break the matrix $\mathbf{C}_2$ into blocks of size $d \times d$.

$$\mathbf{C}_2 = \begin{pmatrix} \mathbf{C}_{(0,0)} & \mathbf{C}_{(0,1)} & \cdots & \mathbf{C}_{(0,d^3-1)} \\ \mathbf{C}_{(1,0)} & \mathbf{C}_{(1,1)} & \cdots & \mathbf{C}_{(1,d^3-1)} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{C}_{(d-1,0)} & \mathbf{C}_{(d-1,1)} & \cdots & \mathbf{C}_{(d-1,d^3-1)} \end{pmatrix} \in M_{d \times d^3}(M_d(R_q))$$

The relinearization key `rlk` will consist of length d sections of $\mathbf{v}$ (denoted $\mathbf{v}^{(0)}$ to $\mathbf{v}^{(d^3-1)}$, from top to bottom), each encrypted separately as columns of plaintext matrices. We will have d^3 number of ciphertexts encrypting $(\mathbf{v}^{(i)}|\mathbf{0}^{d-1})$ for all $0 \le i < d^3$, then d ciphertexts encrypting $(\mathbf{0}|\mathbf{v}^{(i)}|\mathbf{0}^{d-2})$ for all i, and so on. In total, we have d^4 ciphertexts, where each length d section of $\mathbf{v}$ is encrypted in all d possible column positions.

$$\texttt{rlk}[i,j] := \text{mask}_{(\mathbf{S})}(\mathbf{0}^j|\mathbf{v}^{(i)}|\mathbf{0}^{d-1-j}) \quad (0 \le i < d^3 - 1, 0 \le j < d - 1)$$

Now we can perform the matrix-vector product block-by-block, where for each row of $\mathbf{C}_2$, we use the ciphertexts in `rlk` where the index of the row equals the index of the columns occupied in the plaintext matrix.

$$\text{enc}_{(\mathbf{S})}(\mathbf{0}^i|\mathbf{Q}^{(i)}|\mathbf{0}^{d-1-i}) = \langle(\mathbf{C}_{(i,0)}, \ldots, \mathbf{C}_{(i,d^3-1)}), (\texttt{rlk}[0,i], \ldots, \texttt{rlk}[d^3-1,i])\rangle \tag{35}$$

[8]For simplicity, we will refer to the masked version of the secret key as an encryption and a ciphertext.

Given the encrypted columns of $\mathbf{Q}$, we can sum them up to get the relinearized quadratic monomial.

$$\text{enc}_{(\mathbf{S})}(\mathbf{Q}) = \sum_{0 \leq i < d-1} \text{enc}_{(\mathbf{S})}(\mathbf{0}^i | \mathbf{Q}^{(i)} | \mathbf{0}^{d-1-i}) \tag{36}$$

3.5 Compact homomorphic encryption scheme.

Result 3.1. *The encryption scheme MatEnc 3.1 can be turned into a* ***compact somewhat homomorphic encryption scheme*** *with the above relinearization technique.*

Note that similarly to the Ring-LWE case 2.3, one needs to provide scaled versions of the relinearization key, then perform a decomposition on the quadratic coefficient $\mathbf{C}_2$. We would also like to emphasize that this is a first attempt at defining a relinearization technique and there is a lot of room for improvement. For instance, we expect that a ciphertext format that only encrypts column vectors is possible and would drastically reduce the size of the relinearization key.

A big problem with this relinearization key is that each length d segment of $\text{vec}(\mathbf{S}) \otimes \text{vec}(\mathbf{S})$ is encrypted as one column of an entire $d \times d$ matrix. Hence, one realistic approach to reducing the key size would be to encrypt only columns, not matrices. This can be achieved if one only uses one column from the secret key $\mathbf{S}$, then views that encryption under the one column as an encryption under $\mathbf{S}'$, where all other columns are replaced with zeros. If key switching keys are then provided from each column encryption to the original key $\mathbf{S}$, one can perform the matrix-vector product only on encrypted columns, then switch those to matrices and sum them up.

3.6 Ring expansion factor and noise growth.

In this section, we give the ring expansion factor for the matrix ring $M_d(R)$, with respect to the infinity norm on matrices.

Theorem 3.3. *The ring expansion factor $\delta_{M_d(Q)}$ for the matrix ring $M_d(Q)$, over any ring Q, is given by $d \cdot \delta_Q$.*

Proof. Let $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$ denote a general multiplication in $M_d(Q)$. To reach our conclusion for the matrix $\mathbf{C}$, we can inspect the product entry by entry. We can easily upper bound each entry with the upper bound on the Q-multiplications in the row-column dot product.

$$\begin{aligned}
&||\mathbf{C}_{(i,j)}||_\infty = ||\langle (\mathbf{A}_{(i,0)}, \ldots, \mathbf{A}_{(i,d-1)}), (\mathbf{B}_{(0,j)}, \ldots, \mathbf{B}_{(d-1,j)}) \rangle||_\infty = \\
&= ||\sum_{0 \leq k < d} \mathbf{A}_{(i,k)} \mathbf{B}_{(k,j)}||_\infty \leq \sum_{0 \leq k < d} ||\mathbf{A}_{(i,k)} \mathbf{B}_{(k,j)}||_\infty \leq \\
&\leq \delta_Q \cdot \sum_{0 \leq k < d} ||\mathbf{A}_{(i,k)}||_\infty ||\mathbf{B}_{(k,j)}||_\infty \leq d \cdot \delta_Q \cdot ||\mathbf{A}||_\infty \cdot ||\mathbf{B}||_\infty
\end{aligned}$$

Since we defined the infinity norm of a matrix entry by entry, this gives the equivalent upper bound for $\delta_{M_d(Q)}$. Furthermore, since δ_R is a lowest upper bound, with an appropriate choice of parameters all inequalities can be saturated, thus the inequality is sharp. □

Theorem 3.3 gives rise to a convenient connection between the hardness of the Module-LWE instance and the ring expansion factor it implies on the matrix encryption scheme 3.1.

Theorem 3.4. *For a Module-LWE-based encryption scheme as defined in section 3.1, the ring expansion factor for the underlying matrix ring $M_d(R)$ is equivalent to the effective lattice dimension of the Module-LWE instance.*

Proof. We defined the effective lattice dimension 2.2 as $d \cdot N$, which is equivalent to the expansion factor $\delta_{M_d(R)}$ of the ring $M_d(R)$, for $R = \mathbb{Z}/(x^N + 1)$. □

This means that if we start from a Ring-LWE-based instantiation of BGV/BFV or CKKS (which is just a $d = 1$ instantiation of our matrix encryption scheme 3.1) and we increase the matrix dimension to d, then we can decrease the ring dimension N to keep both the hardness of the Module-LWE instance and the ring expansion factor the same. This result points to a sense of naturality in this transition from the scalar to the matrix instantiation.

4 Conclusion and Future Work

We presented a generalization of the BGV, BFV, and CKKS homomorphic encryption schemes to matrix rings. This generalized construction is secure under a suitable decision Module-LWE hardness assumption, and inherits the linear homomorphic properties of the standard instantiations. A so called "superoperator technique" is then presented, which allows for the relinearization of ciphertexts after multiplication, turning the matrix encryption scheme into a compact somewhat homomorphic encryption scheme.

Open questions. Determining the practicality of the relinearization procedure and the development of further improvements constitute major open problems. Additionally, further work is needed to examine the feature parity of the generalized scheme compared with the standard instantiations, especially in the case of bootstrapping algorithms.

Acknowledgments. The author would like to thank Péter Kutas for the countless insightful discussions and encouragement.

References

[1] Gentry, Craig. "Fully homomorphic encryption using ideal lattices", *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 169–178.

[2] Marcolla, Chiara *et al.*, "Survey on fully homomorphic encryption, theory, and applications", *Proceedings of the IEEE*, Vol. 110 No. 10, (2022), pp. 1572–1609. `https://eprint.iacr.org/2022/1602`.

[3] Kim, Andrey, Polyakov, Yuriy, and Zucca, Vincent. "Revisiting homomorphic encryption schemes for finite fields", *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part III 27*. Springer. 2021, pp. 608–639. `https://eprint.iacr.org/2021/204`.

[4] Brakerski, Zvika and Vaikuntanathan, Vinod. "Efficient fully homomorphic encryption from (standard) LWE", *SIAM Journal on computing*, Vol. 43 No. 2, (2014), pp. 831–871. https://eprint.iacr.org/2011/344.

[5] Brakerski, Zvika, Gentry, Craig, and Vaikuntanathan, Vinod. "Fully homomorphic encryption without bootstrapping", *Proceedings of Innovations in Theoretical Computer Science (ITCS'12)*. 2012. http://eprint.iacr.org/2011/277.

[6] Brakerski, Zvika. "Fully homomorphic encryption without modulus switching from classical GapSVP", *Annual cryptology conference*. Springer. 2012, pp. 868–886. https://eprint.iacr.org/2012/078.

[7] Fan, Junfeng and Vercauteren, Frederik. "Somewhat practical fully homomorphic encryption", *Cryptology ePrint Archive*, (2012). https://eprint.iacr.org/2012/144.

[8] Smart, Nigel P and Vercauteren, Frederik. "Fully homomorphic SIMD operations", *Designs, codes and cryptography*, Vol. 71, (2014), pp. 57–81. https://eprint.iacr.org/2011/133.

[9] Gentry, Craig, Halevi, Shai, and Smart, Nigel P. "Fully homomorphic encryption with polylog overhead", *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, pp. 465–482. https://eprint.iacr.org/2011/566.

[10] Gentry, Craig, Sahai, Amit, and Waters, Brent. "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based", *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*. Springer. 2013, pp. 75–92. https://eprint.iacr.org/2013/340.

[11] Ducas, Léo and Micciancio, Daniele. "FHEW: bootstrapping homomorphic encryption in less than a second", *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2015, pp. 617–640. https://eprint.iacr.org/2014/816.

[12] Chillotti, Ilaria *et al.*, "TFHE: fast fully homomorphic encryption over the torus", *Journal of Cryptology*, Vol. 33 No. 1, (2020), pp. 34–91. https://eprint.iacr.org/2018/421.

[13] Cheon, Jung Hee *et al.*, "Homomorphic encryption for arithmetic of approximate numbers", *Advances in cryptology–ASIACRYPT 2017: 23rd international conference on the theory and applications of cryptology and information security, Hong kong, China, December 3-7, 2017, proceedings, part i 23*. Springer. 2017, pp. 409–437. https://eprint.iacr.org/2016/421.

[14] Regev, Oded. "On lattices, learning with errors, random linear codes, and cryptography", *Journal of the ACM (JACM)*, Vol. 56 No. 6, (2009), pp. 1–40. https://arxiv.org/abs/2401.03703.

[15] Regev, Oded. "The learning with errors problem", *Invited survey in CCC*, Vol. 7 No. 30, (2010), p. 11. https://cims.nyu.edu/~regev/papers/lwesurvey.pdf.

[16] Peikert, Chris and Pepin, Zachary. "Algebraically structured LWE, revisited", *Journal of Cryptology*, Vol. 37 No. 3, (2024), p. 28. https://eprint.iacr.org/2019/878.

[17] Lyubashevsky, Vadim, Peikert, Chris, and Regev, Oded. "On ideal lattices and learning with errors over rings", *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*. Springer. 2010, pp. 1–23. https://eprint.iacr.org/2012/230.

[18] Geelen, Robin and Vercauteren, Frederik. "Bootstrapping for BGV and BFV Revisited", *Journal of Cryptology*, Vol. 36 No. 2, (2023), p. 12. https://eprint.iacr.org/2022/1363.

[19] Halevi, Shai and Shoup, Victor. "Algorithms in helib", *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*. Springer. 2014, pp. 554–571. https://eprint.iacr.org/2014/106.

[20] Halevi, Shai and Shoup, Victor. "Design and implementation of HElib: a homomorphic encryption library", *Cryptology ePrint Archive*, (2020). https://eprint.iacr.org/2020/1481.

[21] Jiang, Xiaoqian *et al.*, "Secure outsourced matrix computation and application to neural networks", *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, pp. 1209–1222. https://eprint.iacr.org/2018/1041.

[22] Cheon, Jung Hee, Kim, Andrey, and Yhee, Donggeon. "Multi-dimensional packing for HEAAN for approximate matrix arithmetics", *Cryptology ePrint Archive*, (2018). https://eprint.iacr.org/2018/1245.

[23] Zheng, Xiaopeng, Li, Hongbo, and Wang, Dingkang. "A new framework for fast homomorphic matrix multiplication", *Designs, Codes and Cryptography*, (2025), pp. 1–23. https://eprint.iacr.org/2023/1649.

[24] Peikert, Chris and Waters, Brent. "Lossy trapdoor functions and their applications", *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 2008, pp. 187–196. https://eprint.iacr.org/2007/279.

[25] Micciancio, Daniele. "On the hardness of learning with errors with binary secrets", *Theory of Computing*, Vol. 14 No. 1, (2018), pp. 1–17. https://theoryofcomputing.org/articles/v014a013.

[26] Gentry, Craig, Halevi, Shai, and Vaikuntanathan, Vinod. "A simple BGN-type cryptosystem from LWE", *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2010, pp. 506–522. https://eprint.iacr.org/2010/182.

[27] Gentry, Craig, Peikert, Chris, and Vaikuntanathan, Vinod. "Trapdoors for hard lattices and new cryptographic constructions", *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 2008, pp. 197–206.

[28] Park, Jai Hyun. "Ciphertext-Ciphertext Matrix Multiplication: Fast for Large Matrices", *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2025, pp. 153–180. https://eprint.iacr.org/2025/448.

[29] Hiromasa, Ryo, Abe, Masayuki, and Okamoto, Tatsuaki. "Packing messages and optimizing bootstrapping in GSW-FHE", *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 99 No. 1, (2016), pp. 73–82. https://iacr.org/archive/pkc2015/90200146/90200146.pdf.

[30] Lyubashevsky, Vadim and Micciancio, Daniele. "Generalized compact knapsacks are collision resistant", *International Colloquium on Automata, Languages, and Programming*. Springer. 2006, pp. 144–155. https://cseweb.ucsd.edu/~daniele/papers/IdealHash.pdf.

[31] Halevi, Shai and Shoup, Victor. "Faster homomorphic linear transformations in HElib", *Annual International Cryptology Conference*. Springer. 2018, pp. 93–120. https://eprint.iacr.org/2018/244.

[32] Langlois, Adeline and Stehlé, Damien. "Worst-case to average-case reductions for module lattices", *Designs, Codes and Cryptography*, Vol. 75 No. 3, (2015), pp. 565–599. https://eprint.iacr.org/2012/090.

[33] Albrecht, Martin R and Deo, Amit. "Large modulus ring-LWE $\geq$ module-LWE", *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 267–296. https://eprint.iacr.org/2017/612.

[34] Albrecht, Martin R *et al.*, "Estimate all the {LWE, NTRU} schemes!", *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings 11*. Springer. 2018, pp. 351–367. https://eprint.iacr.org/2018/331.

[35] Albrecht, Martin, Bai, Shi, and Ducas, Léo. "A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and graded encoding schemes", *Annual international cryptology conference*. Springer. 2016, pp. 153–178. https://eprint.iacr.org/2016/127.

[36] Peikert, Chris. "How (not) to instantiate ring-LWE", *International Conference on Security and Cryptography for Networks*. Springer. 2016, pp. 411–430. https://eprint.iacr.org/2016/351.

[37] Halevi, Shai. "Homomorphic encryption", *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Springer, 2017, pp. 219–276. https://shaih.github.io/pubs/he-chapter.pdf.