# Beyond LWE: a Lattice Framework for Homomorphic Encryption

Alberto Leporati[1], Lorenzo Rovida[1], and Wessel van Woerden[2]

[1] University of Milano-Bicocca, DISCo, Milan, Italy
{alberto.leporati, lorenzo.rovida}@unimib.it
[2] PQShield, Amsterdam, the Netherlands
wessel.vanwoerden@pqshield.com

**Abstract.** We suggest a generalization of homomorphic encryption (HE) schemes from a purely geometrical and lattice-based perspective. All the current reference HE schemes are based on the ring version of the Learning with Errors (LWE) problem. In this proposal, we first investigate LWE-based cryptosystems from a lattice point of view and present a framework that allows to obtain the same result, in geometrical terms, from any lattice – as long as it contains a sufficiently short trapdoor vector. More precisely, we generalize the classical BGV (Brakerski, Gentry and Vaikuntanathan, ITCS '12) and GSW (Gentry, Sahai and Waters, CRYPTO '14) schemes to purely lattice-based variants, which we call Lattice-BGV and Lattice-GSW. By abstracting away the particular hardness assumption, our lattice framework allows to be instantiated with a broader range of lattices and hardness assumptions. For example, LWE gives a natural trapdoor for random $q$-ary lattices, and when plugged into our framework one obtains the original BGV and GSW schemes, while in this work we will also consider an instantiation based on the Lattice Isomorphism Problem (LIP), leading to the first more advanced cryptographic scheme build from LIP [3]. Our framework also gives a geometrical and natural explanation of HE procedures and generalizes some properties, such as the ability to store many messages in a single ciphertext, one for each short trapdoor vector, without relying on any particular algebraic structure.

**Keywords:** Homomorphic encryption · Lattice-based cryptography · Lattice isomorphism problem

## 1 Introduction

Originally referred to as *privacy homomorphisms*, the theoretical concept of homomorphic encryption (HE) was introduced in 1978 by Rivest, Adleman and Dertouzos [RAD78]. A homomorphic encryption scheme allows for computations to be performed on encrypted data without having access to the secret key. The

[3] In a concurrent work Branco, Malavolta and Maradni [BMM25] propose an alternative LIP-based FHE construction.

first proposal of an actual *fully homomorphic encryption* (FHE) scheme able to evaluate circuits of any depth, was presented by Gentry in 2009 in its breakthrough work [Gen09], where the *bootstrapping* operation was introduced. Years later, many FHE constructions followed and improved this blueprint.

Today it is possible to find at least two main approaches to instantiate a fully homomorphic encryption scheme, both with security reductions to hard lattice problems: the BGV-like [BGV12] (then evolved towards, e.g., BFV [Bra12,FV12] and CKKS [CKKS17]) and the DM/CGGI-like [DM15,CGGI16] (evolved towards TFHE [CGGI20]) approach, based on GSW [GSW13].

Both these approaches are based on the same hardness assumption, that is the Learning with Errors (LWE) problem [Reg05] or structured versions thereof. Indeed, most of the more popular FHE schemes share the same key generation procedure. This begs the question: is LWE required to instantiate a HE scheme or can we also use other lattice assumptions? Or related, what are the minimum requirements to instantiate it? For the sake of clarity, we will make no difference between HE and FHE as we will not discuss bootstrapping further.

**A lattice framework for (fully) homomorphic encryption** To answer this question we go back to the core of these LWE-based HE schemes to better understand their workings. Since LWE can be seen as a lattice problem, we will take a lattice perspective to describe these schemes. This allows to have some geometrical intuition on both the key generation, the encryption/decryption, and the other FHE specific procedures. In particular, it makes clear what *lattice properties* these schemes are actually using, and what the requirements are on the lattices involved.

Following this perspective, we see that the LWE assumption is used in two ways for the underlying encryption scheme. Firstly, in the key generation, it is used to create a random $q$-ary lattice $q\mathbb{Z}^m \subset \mathcal{L} \subset \mathbb{Z}^m$ with a planted secret short vector $\boldsymbol{s} \in \mathcal{L}$. Secondly, for encryption, a message is encoded in a small error $\boldsymbol{e} \in \mathbb{Z}^m$ which is then hidden as a Bounded Distance Decoding (BDD) target $\boldsymbol{t} \in \boldsymbol{e} + q\mathcal{L}^*$ in the dual lattice $\mathcal{L}^*$ scaled up by $q$. By the LWE assumption, this BDD target is indistinguishable from a uniform target and thus the encoded message is securely hidden. To decrypt, one then uses the secret short vector $\boldsymbol{s} \in \mathcal{L}$ to partially decode the target, as $\langle \boldsymbol{t}, \boldsymbol{s} \rangle \equiv \langle \boldsymbol{e}, \boldsymbol{s} \rangle \bmod q$. Since $\langle \boldsymbol{e}, \boldsymbol{s} \rangle \in \mathbb{Z}$ is small, this value is recovered exactly, and we obtain information about the error $\boldsymbol{e}$ which, if well encoded, allows to recover the message.

It is folklore that the above strategy can also be generalized to construct a simple encryption scheme using different lattice families with different underlying security assumptions. It turns out however that these minimal requirements are also sufficient for HE. To support all common HE primitives, one only requires the key generation to produce a *public lattice* $\mathcal{L}$, for which decoding in the dual lattice $\mathcal{L}^*$ is hard, along with a *secret short vector* $\boldsymbol{s} \in \mathcal{L}$. We demonstrate this by presenting a *lattice framework* for HE. In particular, we present a generalization of the classical BGV and GSW schemes using this public lattice and the secret

short vector only. This is a proper generalization: if instantiated with an LWE-based key generation, we obtain the original schemes.

Recall that ciphertexts can be seen as BDD targets with respect to the lattice $q\mathcal{L}^*$. An interesting and fundamental question for making further HE primitives work, is to understand the nature of the ambient space $\mathcal{D} \supset q\mathcal{L}^*$, where these BDD targets and their associated error live. For LWE-based schemes we naturally have the ambient space $\mathcal{D} = \mathbb{Z}^m$ and $\mathcal{D}/q\mathcal{L}^* \cong (\mathbb{Z}/q\mathbb{Z})^n$. The integer lattice $\mathbb{Z}^m$ has an orthogonal basis making it easy to sample small errors, and the group structure of $\mathcal{D}/q\mathcal{L}^*$ is fundamental to make the more complex HE primitives work. For our more general construction, there does not exist such a natural ambient space, as the lattice $q\mathcal{L}^*$ does not even have to be an integer lattice. We therefore construct a similar ambient space $\mathcal{D} \supset q\mathcal{L}^*$ for any public lattice $\mathcal{L}$. Namely, we choose a (highly composite) modulus $q$ and construct an ambient space $\mathcal{D}$ with a sufficiently orthogonal basis and such that $q\mathcal{D} \subset q\mathcal{L}^* \subset \mathcal{D}$. The latter implies that $\mathcal{D}/q\mathcal{L}^*$ is isomorphic to a subgroup of $(\mathbb{Z}/q\mathbb{Z})^m$. Then, we show that this is sufficient to make all HE primitives work. Furthermore, we obtain a nice geometric interpretation of modulus switching as moving from one ambient space $\mathcal{D}$ to another ambient space $\mathcal{D}'$.

**The Lattice Isomorphism Problem** Our lattice framework can directly be instantiated with lattices constructed from LWE, SIS, NTRU and related assumptions. However, a new assumption of interest that fits well within our framework is the Lattice Isomorphism Problem (LIP), which was recently introduced by Ducas and van Woerden [DvW22] and Bennett, Ganju, Peetathawatchai and Stephens-Davidowitz [BGPSD23]. Two lattices $\mathcal{L}_1, \mathcal{L}_2 \subset \mathbb{R}^n$ are called *isomorphic* if there exists a linear isometry mapping one to the other, i.e., if there exists an orthonormal transformation $O \in \mathcal{O}_m(\mathbb{R})$ such that $\mathcal{L}_2 = O \cdot \mathcal{L}_1 = \{O\boldsymbol{v} : \boldsymbol{v} \in \mathcal{L}_1\}$. LIP then asks, given two isomorphic lattices, to compute such an orthonormal transformation mapping one lattice to the other.

The main idea is that one can use an isometry to hide the structure of a lattice. While [BGPSD23] gave an encryption scheme based on LIP with the lattice $\mathbb{Z}^n$, [DvW22] presented a general construction, presenting an identification, encryption and signature scheme based on LIP. This new hardness assumption gained particular attention after the efficient signature scheme HAWK [BBD+23], based on a module-lattice variant of LIP, that was submitted to NIST's additional call for signatures [DPPW22].

Since the usage of LIP as a hardness assumption is relatively new, there are at this moment only LIP-based schemes for basic cryptographic primitives, like identification, encryption and signature schemes. Given our lattice framework, we can however also construct a HE scheme based on LIP in a straightforward way. For this we can pick any lattice $\mathcal{L}'$ with some known sufficiently short vector $\boldsymbol{s}' \in \mathcal{L}'$. Then we hide the known structure by considering a bad public basis of a lattice $\mathcal{L} := O \cdot \mathcal{L}'$ isomorphic to $\mathcal{L}'$ by some *secret isometry* $O \in \mathcal{O}_m(\mathbb{R})$. Naturally, given the secret isometry $O$ one also knows a secret short vector $\boldsymbol{s} = O \cdot \boldsymbol{s}' \in \mathcal{L}$. Informally, with only the bad description of $\mathcal{L}$, and being hard

to recover the isometry, it is typically hard to decode a BDD instance in $q\mathcal{L}^*$ and therefore the scheme is secure. The latter can be formally proven based on a distinguishing-LIP assumption as in [DvW22]. We thus have all the ingredients to instantiate our lattice framework based on the LIP assumption.

**Related works** Highly related to our work is the work by Gama, Izabachene, Nguyen and Xie [GINX16] on *structural lattice reduction*. This work makes the generalization of LWE to *Group LWE* (GLWE); where LWE is a decoding problem in a lattice $\mathcal{L} \subset \mathbb{Z}^m$ and an ambient space $\mathbb{Z}^m$ such that $\mathbb{Z}^m/\mathcal{L} \cong \mathbb{Z}_q^n$, the GLWE problem considers the decoding problem in a lattice $\mathcal{L}$ such that $\mathbb{Z}^m/\mathcal{L} \cong G$ for any finite abelian group $G$. Furthermore, given a large enough group $G$, it is shown that for any lattice $\mathcal{L}$ (not necessarily integer) one can construct an ambient space $\mathcal{D}$ with $\mathcal{D}/\mathcal{L} \cong G$ along with a nearly orthogonal basis of $\mathcal{D}$. They use this to show reductions of worst-case lattice problems to GLWE, and they show how to construct a GSW-like scheme based on GLWE a lá [AP14,DM15].

In our work we come across similar problems and generalizations. For one, we also need to construct an appropriate ambient space and construct one in a similar approach as in [GINX16], with the main difference being that we do not fix the group $G$ in advance, but only restrict it to a subgroup of some $\mathbb{Z}_q^m$. This makes our approach simpler, and a bit more efficient in terms of the parameters we obtain, but we stress that one might as well use structural lattice reduction within our framework if one aims for a certain group structure. Secondly, the generalized modulus switching that we will consider is related to the *group switching* method in [GINX16]. Again however, we do not need the full generalization and present a simpler method that is sufficient for our construction.

As a main difference between these works is that our goal is to present and explain HE from a *lattice perspective*, while one could view [GINX16] as presenting it from a group structure or algebraic perspective. Furthermore, while we could instantiate the HE schemes we obtain with (the lattices occurring in) GLWE, our framework is more general and allows to be instantiated with *any lattice* with a sufficiently short secret vector.

*Connections with* [BMM25]. In a recent concurrent work Branco, Malavolta and Maradni [BMM25] propose an FHE scheme based on the hardness of LIP. While this does not coincide with the main goal of our work, we do want to shortly compare it to a LIP-based instantiation of our lattice framework. We will mostly focus on the main differences.

Firstly, the work of [BMM25] has a large focus on building an FHE scheme along with a security reduction to (distinguish-)LIP with a similar strategy as used in [DvW22]. As our main goal was to give a general lattice framework for (F)HE and its operations, we left such a security proof out of the scope of our work. We do believe the reduction of [DvW22] could also be adapted to our work.

Secondly, [BMM25] presents a private-key encryption scheme, contrary to our public-key encryption scheme. This is vital for the correctness of their ho-

momorphic multiplications. Following the strategy of [Rot11] one could turn their private-key FHE scheme into a public-key. Conversely, both our GSW and BGV-like multiplication approaches naturally work in a public-key setting, without resorting to extra constructions à la [Rot11].

Thirdly, the schemes we present naturally allow for the evaluation of circuits of depth more than one and we give an explicit connection between the shortness of the secret trapdoor vector and the depth that can be achieved before bootstrapping is required. In [BMM25] only branching programs are considered, where one of the two inputs of the homomorphic multiplication is a fresh ciphertext.

Lastly, while [BMM25] claims an *FHE* scheme our work claims an *HE* scheme. Concretely however, both works only present an HE scheme for bounded depth circuits, and argue that from there bootstrapping is possible by standard techniques. To conclude, we mostly consider the goals and execution of our work and [BMM25] as orthogonal.

**Structure of the paper** The rest of the paper is structured as follows. Section 2 will contain all the required preliminaries from lattices and the BGV/GSW definitions. Section 3 will introduce the lattice perspective on the fundamental components of the previous schemes, namely the key generation and encryption/decryption procedures. Section 4 will introduce a pure lattice-based version of both BGV and GSW. Section 5 will present a possible LIP-based instantiation of previously introduced schemes. Eventually, in Section 6, we consider some further extensions and open questions.

**Acknowledgements** A large part of this work was carried out while W. van Woerden was employed, and while L. Rovida was hosted, by IMB at the University of Bordeaux. There, they were both supported by the CHARM ANR-NSF grant (ANR-21-CE94-0003).

## 2 Preliminaries

In what follows, if not stated otherwise, lowercase letters are used to indicate numbers (e.g. $a$), bold-italic lowercase letters to indicate vectors (e.g. $\boldsymbol{v}$), and uppercase letters are used to indicate matrices (e.g. $A$).

Vectors are considered as column vectors, the $i$-th column of a matrix $A$ is referred to as $\boldsymbol{a}_i$ and the $i$-th element of a vector $\boldsymbol{v}$ as $v_i$. The symbol $\mathbb{Z}_q$ refers to the cyclic group $\mathbb{Z}/q\mathbb{Z}$ of integers modulo $q$, which we identify with $[-q/2, \ldots, q/2) \cap \mathbb{Z}$. The general linear group of invertible matrices with integer (or real) entries of dimension $m$ is referred to as $\mathrm{GL}_m(\mathbb{Z})$ (or $\mathrm{GL}_m(\mathbb{R})$, respectively).

The group of orthonormal matrices is referred to as $\mathcal{O}_m(\mathbb{R})$. Given a matrix $B$ composed of vectors $\boldsymbol{b}_i$, we write $\|B\| := \max_i \|\boldsymbol{b}_i\|$, and will refer to its Gram-Schmidt orthogonalization as $\tilde{B}$ and to the Gram-Schmidt vectors as $\tilde{\boldsymbol{b}}_i$.

The notation $a \sim \chi$ indicates a value $a$ sampled from the probability distribution $\chi$. The notation $[a]_q : \mathbb{Z} \to \mathbb{Z}_q$ indicates the modulo reduction $a$ mod $q$ in the interval $[-q/2, q/2)$. The notation $\lfloor a \rceil_p : \mathbb{Z}_q \to \{0, \ldots, p-1\}$, indicates a function that returns $b$ if, modulo $q$, the argument is closest to $b \cdot q/p$ among $\{0, q/p, 2q/p, \ldots, (p-1)q/p\}$.

We write $\log := \log_2$ for the logarithm function with base two, i.e., $\log(2^k) = k$. Let $\texttt{Bits}(a) : \mathbb{Z}_q \to \{0,1\}^{\log(q)} \subset \mathbb{Z}_q^{\log(q)}$ be the bit decomposition of $a$, least to most significant bit, and let $\texttt{Pow}(a) : \mathbb{Z}_q \to \mathbb{Z}_q^{\log(q)} = (a, 2a, 4a, \ldots, 2^{\log(q)-1} \cdot a)$. By abuse of notation we allow both functions to be applied to vectors by evaluating the corresponding function independently on each entry, giving functions $\texttt{Bits}, \texttt{Pow} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n \log(q)}$.

### 2.1 Lattices and hard problems

For a basis $B \in \mathrm{GL}_m(\mathbb{R})$ we define the lattice $\mathcal{L}(B) \subset \mathbb{R}^m$ generated by $B$ as

$$\mathcal{L}(B) := B \cdot \mathbb{Z}^m = \{B\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^m\}.$$

Two bases $B, B' \in \mathrm{GL}_m(\mathbb{R})$ generate the same lattice if and only if there exists a unimodular matrix $U \in \mathrm{GL}_m(\mathbb{Z})$ such that $B' = B \cdot U$. The volume $\mathrm{vol}(\mathcal{L})$ of a lattice $\mathcal{L} = \mathcal{L}(B) \subset \mathbb{R}^m$ is given by $\mathrm{vol}(\mathbb{R}^m/\mathcal{L})$ and equals $|\det(B)|$ for any basis of the lattice. The first minimum is the length of the shortest non-zero vector of the lattice $\mathcal{L}$ and is denoted by $\lambda_1(\mathcal{L}) = \min \|\boldsymbol{v}\|$ for all nonzero $\boldsymbol{v} \in (\mathcal{L} \setminus \boldsymbol{0})$. A $q$-ary lattice is a lattice $\mathcal{L}$ that satisfies $q\mathbb{Z}^m \subset \mathcal{L} \subset \mathbb{Z}^m$. Given any (primal) lattice $\mathcal{L}$, it is possible to define its *dual* $\mathcal{L}^*$ as follows:

**Definition 2.1 (Dual lattice).** *Given a lattice $\mathcal{L}$, its dual is defined as the lattice $\mathcal{L}^*$ of all the vectors $\boldsymbol{x} \in \mathrm{span}(\mathcal{L})$ such that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}$ for all $\boldsymbol{y} \in \mathcal{L}$.*

For a lattice $\mathcal{L}(B)$ generated by a square basis $B \in \mathrm{GL}_m(\mathbb{R})$, its dual is generated by the basis $B^{-t}$, namely $\mathcal{L}(B)^* = \mathcal{L}(B^{-t})$. For a lattice $\mathcal{L}$ and any scalar $q \in \mathbb{R}_{>0}$ we will for convenience call $q\mathcal{L}^*$ the *$q$-dual* of $\mathcal{L}$. In particular, the $q$-dual of a $q$-ary lattice is again a $q$-ary lattice.

Two lattices $\mathcal{L}, \mathcal{L}'$ are said to be *isomorphic* if there exists an orthonormal transformation $O \in \mathcal{O}_m(\mathbb{R})$ such that $\mathcal{L}' = O \cdot \mathcal{L}$. This relation allows to introduce the *Lattice Isomorphism Problem* (LIP).

**Definition 2.2 (Lattice Isomorphism Problem (LIP) - search version).** *Given two isomorphic lattices $\mathcal{L}, \mathcal{L}' \subset \mathbb{R}^m$, find an orthonormal transformation $O \in \mathcal{O}_m(\mathbb{R})$ such that $\mathcal{L}' = O \cdot \mathcal{L}$.*

Since two bases generate the same lattice up to some unimodular transformation $U$, the Lattice Isomorphism Problem can also be stated in terms of the bases. Given that two bases $B', B \in \mathrm{GL}_m(\mathbb{R})$ generate the same lattice if there exists a unimodular transformation $U \in \mathrm{GL}_m(\mathbb{Z})$ such that $B' = B \cdot U$, LIP asks to find an orthonormal $O \in \mathcal{O}_m(\mathbb{R})$ and a unimodular $U \in \mathrm{GL}_m(\mathbb{Z})$ such that $B' = O \cdot B \cdot U$. The presence of both the orthonormal and the unimodular transformations is what makes this problem hard.

Lastly, we introduce two variants of the lattice decoding problem, along with the Learning with Errors (LWE) problem by Regev [Reg05].

**Definition 2.3 ($\alpha$-Bounded Distance Decoding ($\mathrm{BDD}_\alpha$)).** *Let $0 \leq \alpha < \frac{1}{2}$, $\mathcal{L}$ a lattice, and $\boldsymbol{t} = \boldsymbol{v} + \boldsymbol{e}$ where $\boldsymbol{v} \in \mathcal{L}$ and $\|\boldsymbol{e}\| < \alpha\lambda_1(\mathcal{L})$. The $\mathrm{BDD}_\alpha$ problem asks: given a basis $B$ of $\mathcal{L}$ and the target $\boldsymbol{t} = \boldsymbol{v} + \boldsymbol{e}$, compute the closest lattice point $\boldsymbol{v}$ and the error $\boldsymbol{e}$.*

Throughout the paper we will ignore the particular promise factor $\alpha$ and simply talk about a BDD instance $(\boldsymbol{t}, \mathcal{L})$ whenever $\mathrm{dist}(\boldsymbol{t}, \mathcal{L}) < \frac{1}{2}\lambda_1(\mathcal{L})$.

Note that the error computed in the BDD problem is invariant under translation of the target by a lattice point, i.e., any $\boldsymbol{t} \in \boldsymbol{e} + \mathcal{L}$ will decode to the same error. In certain cases one is only interested in this small error and it is sufficient to receive as input any *syndrome* $\boldsymbol{t} \equiv \boldsymbol{e} \in \mathrm{span}(\mathcal{L})/\mathcal{L}$ and we call recovering $\boldsymbol{e}$ the *syndrome decoding problem* (SDP). Typically, when a (public) basis $B$ is clear from the context we identify $\mathrm{span}(\mathcal{L})/\mathcal{L}$ with the fundamental parallelepiped $\mathcal{P}(B) := B \cdot [-\frac{1}{2}, \frac{1}{2})^m$ and thereby assume a canonical representative as input. If the error is not necessarily small enough to allow for unique decoding one can also consider the *approximate syndrome decoding problem* (approxSDP) where the goal is merely to return a small enough error in the coset.

**Definition 2.4 (Learning with Errors ($\mathrm{LWE}_{n,m,q,\chi}$)).** *For a security parameter $\lambda$, let $n = n(\lambda)$ be an integer dimension, let $q = q(\lambda) \geq 2$ be an integer, let $m = \mathcal{O}(n \log(q))$ be the number of samples, and let $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The $\mathrm{LWE}_{n,m,q,\chi}$ problem is to distinguish with non-negligible probability (in $\lambda$) the following two distributions both returning $(A, \boldsymbol{b})$: in the first one draws $A, \boldsymbol{b}$ uniformly from $\mathbb{Z}_q^{m \times n}$ and $\mathbb{Z}_q^m$ respectively. In the second distribution one first draws $\boldsymbol{x} \in \mathbb{Z}_q^n$ uniformly, then samples $A$ uniformly from $\mathbb{Z}_q^{m \times n}$ and $\boldsymbol{s} \in \mathbb{Z}_q^m$ with $s_i \sim \chi$, and sets $\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{s} = (\langle a_i, \boldsymbol{s} \rangle + s_i \bmod q)_i$. The $\mathrm{LWE}_{n,q,m,\chi}$ assumption is that the $\mathrm{LWE}_{n,m,q,\chi}$ problem is infeasible.*

We emphasize and stress that an LWE instance is defined using the notation introduced above, namely $\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{s} \bmod q$, where the "error" $\boldsymbol{s} \in \mathbb{Z}_q^m$ is a small vector from the distribution $\chi$. Although not standard, this notation will allow to have a similar notation between the original LWE-based schemes and our generalizations. We will later see that recovering the $(\boldsymbol{x}, \boldsymbol{s})$ pair from an LWE instance is essentially a syndrome decoding problem in a random $q$-ary lattice.

**Discrete Gaussian distributions** The Gaussian function $\rho_{\sigma,\boldsymbol{c}} : \mathbb{R}^m \to (0, 1]$ with width $\sigma \in \mathbb{R}$, centered at $\boldsymbol{c} \in \mathbb{R}^m$ (if not specified, $\boldsymbol{c} = \boldsymbol{0}$) is defined as $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) := \exp(-\pi \|\boldsymbol{x} - \boldsymbol{c}\|^2 / \sigma^2)$. The Gaussian function can be used to define the continuous Gaussian distribution, however we restrict its domain to a lattice and obtain the *discrete Gaussian distribution*.

**Definition 2.5 (Discrete Gaussian distribution).** *For a lattice $\mathcal{L} \subset \mathbb{R}^m$, center $\boldsymbol{c} \in \mathbb{R}^m$, and Gaussian width $\sigma > 0$, the discrete Gaussian distribution $\mathcal{G}_{\mathcal{L},\sigma,\boldsymbol{c}}$ is defined by:*

$$\Pr_{X \sim \mathcal{G}_{\mathcal{L},\sigma,\boldsymbol{c}}}[X = \boldsymbol{x}] := \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})}{\rho_{\sigma,\boldsymbol{c}}(\mathcal{L})}.$$

We write $\mathcal{G}_{\mathcal{L},\sigma} := \mathcal{G}_{\mathcal{L},\sigma,\mathbf{0}}$ and $\mathcal{G}_{\sigma,\boldsymbol{c}} := \mathcal{G}_{\mathbb{Z},\sigma,\boldsymbol{c}}$. Given a basis $B$ of a lattice $\mathcal{L}$ we can efficiently sample from the discrete Gaussian distribution over $\mathcal{L}$ for a Gaussian width $\sigma$ that is slightly larger than the Gram-Schmidt norm $\|\tilde{B}\|$. In particular, $\|\tilde{B}\| \leq \|B\|$ so a short basis allows to sample somewhat short lattice vectors.

**Lemma 2.1** (from [BLP+13, Lemma 2.3]). *There is a probabilistic polynomial time algorithm that, given a basis $B$ of a lattice $\mathcal{L}(B)$, a center $\boldsymbol{c} \in \mathbb{R}^m$ and a parameter $\sigma \geq \|\tilde{B}\| \cdot \sqrt{\ln(2m+4)/\pi}$, returns a sample distributed as $\mathcal{G}_{\mathcal{L}(B),\sigma,\boldsymbol{c}}$.*

### 2.2 Original BGV cryptosystem

Started by the Brakerski-Gentry-Vaikuntanathan (BGV) cryptosystem [BGV12], the BGV-like type of schemes allows for practical homomorphic circuit evaluation based on (Ring-)LWE. The adopted "leveled" approach allows to evaluate circuits of a fixed multiplicative depth – or levels – depending on the selected set of parameters.

We describe the public-key version of the BGV cryptosystem in $\mathrm{LWE}_{n-1,m,q,\chi}$ form. Although the ring version based on Ring-LWE [LPR10] is more efficient, we focus on the plain variant to simplify the geometric interpretation we will give later. Let $n, q \in \mathbb{N}$ be the LWE dimension and modulus respectively, and let $m = \mathcal{O}(n \log(q))$ be the number of LWE samples. We also define $t \ll q$ with $\gcd(t, q) = 1$ as the plaintext modulus and $\chi$ as a small distribution from $\mathbb{Z}_q$. The starting modulus $q$ is constructed as $q = \prod_{i=1}^{L+1} p_i$, where $L$ is the depth of the circuit to be evaluated and each $p_i$ is set to be $p_i \equiv 1 \bmod t$.

**Key generation** Given a uniformly sampled $\boldsymbol{x}' \in \mathbb{Z}_q^{n-1}$, the secret key is set as $\boldsymbol{x} := (1, -\boldsymbol{x}') \in \mathbb{Z}_q^n$. For the public key, a uniformly random matrix $B \leftarrow \mathbb{Z}_q^{m \times (n-1)}$ is sampled and an LWE instance is generated as $\boldsymbol{b} := B \cdot \boldsymbol{x}' + \boldsymbol{s} \in \mathbb{Z}_q^m$, where $\boldsymbol{s} \sim \chi^m$ is a small error term. The public key $A$ is obtained by appending $\boldsymbol{b}$ to $B$ as the first column:

$$A := \left[ \boldsymbol{b} \mid B \right] \in \mathbb{Z}_q^{m \times n}. \tag{2.1}$$

Note that the short vector $\boldsymbol{s}$ can be recovered by combining the public and the secret key as $A \cdot \boldsymbol{x} \equiv \boldsymbol{s} \in \mathbb{Z}_q^m$.

**Encryption and decryption** In order to encrypt a message $\mu \in \mathbb{Z}_t$, first a binary vector $\boldsymbol{r} \in \{0,1\}^m$ is uniformly sampled. Then, a (scaled) Regev encryption [Reg05] of 0 is computed as $[t \cdot A^t \cdot \boldsymbol{r}]_q \in \mathbb{Z}_q^n$. Eventually, the message $\mu$ is embedded in a vector as $\boldsymbol{m} = (\mu, 0, \ldots, 0)$, and added to the product $t \cdot A^t \cdot \boldsymbol{r}$:

$$\mathrm{encrypt}(\mu, A) = [t \cdot A^t \cdot \boldsymbol{r} + \boldsymbol{m}]_q \in \mathbb{Z}_q^n. \tag{2.2}$$

The $A^t \cdot \boldsymbol{r}$ product, which refers to the error of the ciphertext, is scaled by the plaintext modulus $t$ so that "enough room" is created to store the message $\mu < t$

to be encrypted. To decrypt a ciphertext $\boldsymbol{c} \in \mathbb{Z}_q^n$, simply evaluate a rounded inner product with $\boldsymbol{x} \in \mathbb{Z}_q^n$ modulo $q$ followed by a reduction modulo $t$:

$$\begin{aligned}\text{decrypt}(\boldsymbol{c}, \boldsymbol{x}) &:= [[\langle \boldsymbol{c}, \boldsymbol{x} \rangle]_q]_t \equiv [[t \cdot \boldsymbol{r}^t \cdot A \cdot \boldsymbol{x} + x_1 \cdot \mu]_q]_t \equiv [[t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle + x_1 \cdot \mu]_q]_t \\ &\equiv [t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle + x_1 \cdot \mu]_t \equiv \mu \in \mathbb{Z}_t,\end{aligned}$$

remark that $A \cdot \boldsymbol{x} \equiv \boldsymbol{s}$ and that $x_1 = 1$. Note that the decryption works as long as the magnitude of the *decryption noise* $t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle$ is roughly smaller than $q/2$. The addition of two ciphertexts $\boldsymbol{c}^+ = \boldsymbol{c}_1 + \boldsymbol{c}_2$ produces a valid ciphertext $\boldsymbol{c}^+$ (assuming that $|t \cdot \langle \boldsymbol{r}^+, \boldsymbol{s} \rangle + \mu_1 + \mu_2| < q/2$) that can be decrypted using the distributive property of the inner product:

$$\begin{aligned}\text{decrypt}(\boldsymbol{c}^+, \boldsymbol{x}) :=& [[\langle \boldsymbol{c}_1 + \boldsymbol{c}_2, \boldsymbol{x} \rangle]_q]_t \equiv [[\langle \boldsymbol{c}_1, \boldsymbol{x} \rangle + \langle \boldsymbol{c}_2, \boldsymbol{x} \rangle]_q]_t \\ \equiv& [t \cdot \langle \boldsymbol{r}_1, \boldsymbol{s} \rangle + \mu_1 + t \cdot \langle \boldsymbol{r}_2, \boldsymbol{s} \rangle + \mu_2]_t \equiv \mu_1 + \mu_2 \in \mathbb{Z}_t.\end{aligned}$$

On the other hand, the multiplication between two ciphertexts $\boldsymbol{c}_1, \boldsymbol{c}_2$ is evaluated as a tensor product $\boldsymbol{c}^\times = \boldsymbol{c}_1 \otimes \boldsymbol{c}_2$, although a "tensored" version of the secret key is required in order to correctly decrypt $\boldsymbol{c}^\times$:

$$\begin{aligned}\text{decrypt}(\boldsymbol{c}^\times, \boldsymbol{x} \otimes \boldsymbol{x}) :=& [[\langle \boldsymbol{c}_1 \otimes \boldsymbol{c}_2, \boldsymbol{x} \otimes \boldsymbol{x} \rangle]_q]_t \equiv [[\langle \boldsymbol{c}_1, \boldsymbol{x} \rangle \cdot \langle \boldsymbol{c}_2, \boldsymbol{x} \rangle]_q]_t \\ \equiv& [(t \cdot \langle \boldsymbol{r}_1, \boldsymbol{s} \rangle + \mu_1) \cdot (t \cdot \langle \boldsymbol{r}_2, \boldsymbol{s} \rangle + \mu_2)]_t \equiv \mu_1 \cdot \mu_2 \in \mathbb{Z}_t.\end{aligned}$$

As before, the decryption noise must be smaller than $q/2$ in order to correctly decrypt, although with multiplications this tends to grow very quickly.

**Modulus switching** Assuming the initial decryption noise in a fresh ciphertext to be bounded by $b$, the output of an addition will be bounded by $2b$ and the multiplication by roughly $b^2$. By choosing $q = \Theta(b^{L+1})$, one would be allowed to approximately perform up to $\log(L)$ multiplications since the noise grows exponentially. A possible solution, firstly suggested in [BV11], is to perform a *modulus switching*. The informal idea is that the error growth is mostly determined by the absolute size of the errors, and one can reduce the absolute value of the error in the ciphertext by "cutting" part of the current modulus after each multiplication. Although the relative error with respect to the modulus remains fixed, the absolute error is reduced and this enables to perform more computations with a smaller error growth.

Say that a fresh ciphertext is encrypted using modulus $q_1$ such that $q_1 \equiv 1 \bmod t$, and let $q_2$ be the new modulus such that $q_2 \mid q_1$ and $q_2 \ll q_1$. Given a ciphertext $\boldsymbol{c} \in \mathbb{Z}_{q_1}^n$, the idea it to round it appropriately and to scale down by $q_1/q_2$; namely find the closest vector $\boldsymbol{c}' \in \mathbb{Z}_{q_1/q_2}^n$ to $\boldsymbol{c} \cdot (q_1/q_2)$ such that $\boldsymbol{c}_i \equiv (\boldsymbol{c}')_i \bmod t$ for all $i \in [1, n]$. It turns out that a simple solution is to use the following $\boldsymbol{\delta}$ term:

$$\boldsymbol{\delta} := t \cdot [\boldsymbol{c} \cdot t^{-1}]_{q_1/q_2},$$

where $t^{-1}$ is the inverse of $t$ modulo $q_1/q_2$.

The $\boldsymbol{\delta}$ term, added to $\boldsymbol{c}$, makes each term a multiple of $q_1/q_2$ and does not affect the decryption since each $\delta_i \equiv 0 \bmod t$. Finally, given the current and the

new modulus $q_1, q_2$, respectively, and a ciphertext $\boldsymbol{c} \in \mathbb{Z}_{q_1}^n$, the modulus switching is performed as a "round and scale" procedure:

$$\text{modSwitch}(\boldsymbol{c}, q_1, q_2) := [(\boldsymbol{c} - \boldsymbol{\delta}) \cdot (q_2/q_1)]_{q_2} \in \mathbb{Z}_{q_2}^n. \tag{2.3}$$

By applying this approach, the number of possible sequential multiplications, given a moduli chain $q_1 = b^{L+1}$, actually becomes roughly $L$, since after each multiplication the bound on the noise can be reduced from $b^2$ back to (approximately) $b$, while the modulus only decreases by a factor $b$, preventing an exponential growth.

**Key switching** After performing a tensor product between two ciphertexts, in addition to the noise growth, the dimension of the resulting ciphertext increases from $n$ to $n^2$. Given $\boldsymbol{c}^\times = \boldsymbol{c}_1 \otimes \boldsymbol{c}_2 \in \mathbb{Z}_q^{n^2}$, by the properties of the tensor product, $\boldsymbol{c}^\times$ can still be decrypted using a tensored version of the secret key, namely $\boldsymbol{x} \otimes \boldsymbol{x}$:

$$[\langle \boldsymbol{c}_1 \otimes \boldsymbol{c}_2, \boldsymbol{x} \otimes \boldsymbol{x} \rangle]_q = [\langle \boldsymbol{c}_1, \boldsymbol{x} \rangle \cdot \langle \boldsymbol{c}_2, \boldsymbol{x} \rangle]_q.$$

Nevertheless, this strategy is not sustainable in the long term, as the sizes of the ciphertexts and keys grow exponentially. A solution is to perform a *key switching* procedure. Given a ciphertext $\boldsymbol{c}$, encrypted under a secret key $\boldsymbol{x}_1$, the objective is to find a new ciphertext $\boldsymbol{c}'$, under a new secret key $\boldsymbol{x}_2$, such that $\text{decrypt}(\boldsymbol{c}, \boldsymbol{x}_1) = \text{decrypt}(\boldsymbol{c}', \boldsymbol{x}_2)$. In order to enable this procedure, the key generator must produce a *switching key* $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$ that will be publicly shared. The abstract idea is that this new key is some sort of an encryption of the old secret key $\boldsymbol{x}_1$ under $\boldsymbol{x}_2$, and because $\boldsymbol{c}$ is public we can compute an encryption of $[\langle \boldsymbol{c}, \boldsymbol{x}_1 \rangle]_q$ under $\boldsymbol{x}_2$ using only homomorphic scalings and additions. Note that the main difference with a full bootstrapping procedure is that there is no further removal of the decryption noise, allowing for a much simpler procedure.

Given two secret keys $\boldsymbol{x}_1, \boldsymbol{x}_2$, their respective dimensions $n_1, n_2$ and the current modulus $q$, the key generation procedure for key switching first generates a matrix $A_{\text{ks}} \in \mathbb{Z}_q^{n_1 \times n_2}$ following the public key generation procedure with the secret ket $\boldsymbol{x}_2$. While generated by the public key generation procedure, $A_{\text{ks}}$ is not made public and should be seen as a matrix containing $n_2$ Regev encryptions [Reg05] of zero, one for each row. To conclude the switching key generation phase, the first secret key $\boldsymbol{x}_1$ is added component-wise in the first column of $A_{\text{ks}}$. The final matrix is, considering the addition over the first column, equal to $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2} = A_{\text{ks}} + \boldsymbol{x}_1 \in \mathbb{Z}_q^{n_1 \times n_2}$. Since $A_{\text{ks}}$ is uniform-looking by LWE, the secret $\boldsymbol{x}_1$ is hard to be recovered from $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$.

At this point, the key switching can be performed by multiplying the (transposed) matrix $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$ with the ciphertext $\boldsymbol{c}^\times$. Intuitively, the presence of $\boldsymbol{x}_1$ in the first column of $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$ decrypts the message contained in the first coordinate of $\boldsymbol{c}^\times$, but at the same time this message is re-encrypted by adding to the latter the new encryptions of zero under $\boldsymbol{x}_2$:

$$\text{keySwitch}(\boldsymbol{c}, \tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}) := \tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}^t \cdot \boldsymbol{c} = ([\langle \boldsymbol{c}, \boldsymbol{x}_1 \rangle]_q, 0, \ldots, 0) + (A_{\text{ks}})^t \cdot \boldsymbol{c} \in \mathbb{Z}_q^{n_2}.$$

We observe that applying this approach as is might not produce a valid ciphertext. The problem is that $\tau^t_{\boldsymbol{x}_1 \to \boldsymbol{x}_2} \cdot \boldsymbol{c} \in \mathbb{Z}_q^{n_2}$ produces a new encryption of the old message by adding $n_1$ fresh ciphertexts to it, each scaled by $c_i$. As each $c_i$ can be as large as $q/2$ the result will contain a significant amount of noise that easily exceeds the $q/2$ threshold. There are different approaches to tackle this issue: BV [BV11], GHS and HYBRID [GHS12].

We will stick to BV: it consists in decomposing the ciphertext $\boldsymbol{c}$ as a longer binary vector using $\texttt{Bits}(\boldsymbol{c})$ and decomposing the secret key $\boldsymbol{x}_1$ as $\texttt{Pow}(\boldsymbol{x}_1)$; this implies choosing $n_1 = n^2 \log(q)$. The noise contained in the resulting ciphertext will be smaller, since the encryptions of 0 contained in $A_{\text{ks}}$ will be multiplied with a vector of smaller magnitude, that is $\texttt{Bits}(\boldsymbol{c})$; the partial decryption will still be valid since $\langle \texttt{Bits}(\boldsymbol{c}), \texttt{Pow}(\boldsymbol{x}_1) \rangle = \langle \boldsymbol{c}, \boldsymbol{x}_1 \rangle$.

### 2.3 Original GSW cryptosystem

One year after the introduction of the BGV cryptosystem, the Gentry-Sahai-Waters (GSW) [GSW13] scheme was proposed. Initially, it was presented as a "conceptually simple" alternative to the already available HE schemes. Today it is considered a fundamental building block to bootstrap advanced constructions such as TFHE. In particular, homomorphic operations are performed "naturally" between ciphertexts as they are encoded in square matrices[4]. As a consequence, multiplications are performed as matrix multiplications without requiring any key switching procedure. The noise growth is reduced by encoding the message to be encrypted in a so-called gadget matrix $G$; a multiplication between two ciphertexts $C_1$ and $C_2$ is indeed evaluated as $C_1 \cdot G^{-1}(C_2)$. The GSW cryptosystem is defined by the LWE dimension and modulus $n, q$ and by the number of samples $m = n \log(q)$. We assume $q$ to be some power of two. Additionally, we define $\chi$ as some small distribution over the integers. A plaintext message is set to be in $\{0, 1\} \subset \mathbb{Z}$.

The key generation phase is exactly the same as the one in the BGV cryptosystem, we thus have $\boldsymbol{x} = (1, -\boldsymbol{x}') \in \mathbb{Z}_q^n$ as the secret key and $A := \left[ \boldsymbol{b} \mid B \right] \in \mathbb{Z}_q^{m \times n}$, with $\boldsymbol{b} := B \cdot \boldsymbol{x}' + \boldsymbol{s} \in \mathbb{Z}_q^m$.

*Gadget decompositions.* Gadget matrices were firstly (implicitly) introduced by Ajtai [Ajt96] and formalized by Micciancio and Peikert [MP12] (although referred to as primitive matrices) as lattice trapdoors. Refer to [GMP19, Def. 4] for a precise definition; for our needs, we only require the basic $\mathbb{Z}_q$-gadget matrix of size $w = 2$, constructed as $\boldsymbol{g} \otimes I_n$, where $\boldsymbol{g} := \texttt{Pow}(1) = (1, 2, 4, \ldots, 2^{\log(q)-1})$. As a result, we obtain the following block matrix $G$.

$$G = \begin{pmatrix} 1\,2\,4 \ldots 2^{\log(q)-1} & & & \\ & 1\,2\,4 \ldots 2^{\log(q)-1} & & \\ & & \ddots & \\ & & & 1\,2\,4 \ldots 2^{\log(q)-1} \end{pmatrix} \in \mathbb{Z}_q^{n \times m} \quad (2.4)$$

[4] Note that we will follow the notation from [AP14, Section 3], where ciphertexts are not square matrices, but the two representations are semantically equivalent.

We also introduce the $\boldsymbol{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}_2^{\log(q)}$ function that performs the binary decomposition of a value such that $\langle \boldsymbol{g}, \boldsymbol{g}^{-1}(a) \rangle = a$. By abuse of notation, we define $G^{-1} : \mathbb{Z}_q^{n \times m} \to \mathbb{Z}_2^{n \log(q) \times m}$ to be the application of $\boldsymbol{g}^{-1}$ over each entry of the input matrix.

**Encryption and decryption** In order to encrypt a message $\mu \in \{0, 1\}$, a binary matrix $R \in \{0, 1\}^{m \times m}$ is first uniformly sampled. The first part of the encryption is done by generating $m$ different subset sums of columns of $A$ using $R$ as

$$[A^t R]_q \in \mathbb{Z}_q^{n \times m}.$$

Each column of this product can be indeed seen as a BGV encryption of zero (i.e., $[t \cdot A^t \cdot \boldsymbol{r}_j]_q$) with an error scale $t = 1$. To this product, we add the message $\mu$, encoded as the gadget matrix $G$:

$$\text{encrypt}(A, \mu) := [A^t R + G\mu]_q \in \mathbb{Z}_q^{n \times m}, \tag{2.5}$$

In order to decrypt the message contained in the ciphertext $C \in \mathbb{Z}_q^{n \times m}$ using the secret key $\boldsymbol{x} \in \mathbb{Z}_q^n$, consider the $j$-th column of $C$, with $j = \log(q)$, where the value $2^{\log(q)-1}$ has been added in the first component, and evaluate the following rounded inner product:

$$\begin{aligned} \text{decrypt}(C, \boldsymbol{x}) := \lfloor [\langle \boldsymbol{c}_j, \boldsymbol{x} \rangle]_q \rceil_2 &\equiv \lfloor [\langle A^t \cdot \boldsymbol{r}_j, \boldsymbol{x} \rangle + \langle (\frac{q}{2}\mu, 0, \ldots, 0), \boldsymbol{x} \rangle]_q \rceil_2 \\ &\equiv \lfloor [\langle \boldsymbol{r}_j, \boldsymbol{s} \rangle + (q/2)\mu]_q \rceil_2 \equiv \mu \in \{0, 1\}, \end{aligned}$$

where $\boldsymbol{r}_j \in \{0, 1\}^m$ is the $j$-th column of $R$ and $[\langle \boldsymbol{r}, \boldsymbol{s} \rangle]_q$ is the length of the short error vector $A \cdot \boldsymbol{x} \equiv \boldsymbol{s} \bmod q$, multiplied by $\boldsymbol{r}_j$. In general, decryption is correct as long as the decryption noise $|\langle \boldsymbol{r}_j, \boldsymbol{s} \rangle| < q/4$, otherwise the function $\lfloor \cdot \rceil_2$ would yield an incorrect result. Refer to [AP14, Section 3] for a more detailed analysis.

It is easy to see that the decryption of an addition $C^+ = C_1 + C_2$ between two ciphertexts $C_1, C_2 \in \mathbb{Z}_q^{n \times m}$ is correct. Furthermore, the decryption of $C^\times = C_1 \cdot G^{-1}(C_2)$ effectively shows the role of the gadget matrix in the asymmetric and *quasi-additive* error growth in multiplication:

$$\begin{aligned} \boldsymbol{x}^t (C_1 \cdot C_2) &\equiv \boldsymbol{x}^t (C_1 \cdot G^{-1}(C_2)) \\ &\equiv (\boldsymbol{e}_1^t + \mu_1 \cdot \boldsymbol{x}^t G) \cdot G^{-1}(C_2) \\ &\equiv \boldsymbol{e}_1^t \cdot G^{-1}(C_2) + \mu_1 (\boldsymbol{x}^t G \cdot G^{-1}(C_2)) \\ &\equiv \boldsymbol{e}_1^t \cdot G^{-1}(C_2) + \mu_1 (\mu_2 \cdot \boldsymbol{x}^t G + \boldsymbol{e}_2^t) \\ &\equiv \boldsymbol{e}_1^t \cdot \underbrace{G^{-1}(C_2)}_{\text{Small error}} + \mu_1 \boldsymbol{e}_2^t + \boldsymbol{x}^t \underbrace{G \mu_1 \mu_2}_{\text{Large message}}. \end{aligned} \tag{2.6}$$

## 3 A lattice framework for HE

Until now, we simply described the cryptosystems without considering a lattice perspective. Now we provide a geometric intuition of the processes involved,

in lattice terms, when using an LWE-based cryptosystem. Then, we generalize these to any lattice.

### 3.1 Interpreting LWE from a lattice perspective

In LWE-based cryptosystems, the hardness assumption is used twice: for the key generation and for the encryption phases. Let us start from a geometric point of view: LWE has a reduction to hard problems over (random) $q$-ary lattices. More precisely, let $B \in \mathbb{Z}_q^{m\times n}$, we consider the $q$-ary lattice

$$\mathcal{L}_q(B) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{y} \equiv B\boldsymbol{x} \bmod q, \text{ for some } \boldsymbol{x} \in \mathbb{Z}_q^n\}. \tag{3.1}$$

We call $B$ a *$q$-ary basis* of $\mathcal{L}_q(B)$. Note that the lattice $\mathcal{L}_q(B)$ is generated by the columns of $B$ and the *$q$-vectors* $q\boldsymbol{e}_i \in q\mathbb{Z}^m$ where $\boldsymbol{e}_i$ is a unit vector. The $q$-dual lattice of $\mathcal{L}_q(B)$, also known as a parity check or SIS lattice, is given by

$$\mathcal{L}_q^\perp(B) = \{\boldsymbol{y} \in \mathbb{Z}^n : B^t\boldsymbol{y} \equiv \boldsymbol{0} \bmod q\}. \tag{3.2}$$

While for $\mathcal{L}_q(B)$ we already mentioned a generating set of $m+n$ vectors, writing down a basis for these lattices can be a bit more involved. Typically, we can write $B^t = [B_1^t \mid B_2^t]$ where $B_2 \in \mathbb{Z}_q^{n\times n}$ is invertible, in this case $\mathcal{L}_q(B)$ and $\mathcal{L}_q^\perp(B)$ have a *lattice basis* of the form

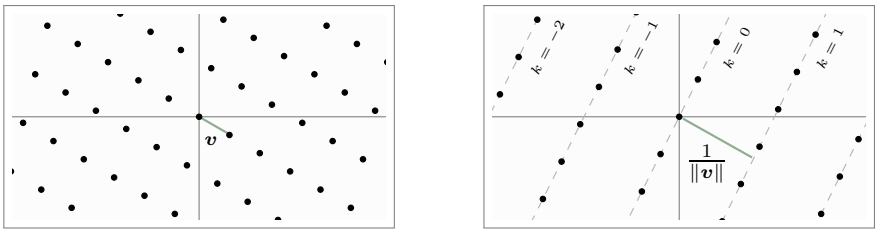
$$\begin{pmatrix} qI_{m-n} & B_1B_2^{-1} \\ 0 & I_n \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & I_{m-n} \\ qI_n & -B_2^{-t}B_1^t \end{pmatrix}, \tag{3.3}$$

respectively. When considering the $q$-dual, it naturally leads to another $q$-ary lattice, since we again obtain a lattice satisfying $q\mathbb{Z}^m \subset \mathcal{L}_q^\perp(B) \subset \mathbb{Z}^m$. For the normal dual $\mathcal{L}^*$ we have $\mathcal{L}_q(B)^* = 1/q \cdot \mathcal{L}_q^\perp(B)$. When $B \in \mathbb{Z}_q^{m\times n}$ is uniformly random we call $\mathcal{L}_q(B)$ a random $q$-ary lattice. Note that if $\mathcal{L}_q(B)$ is a random $q$-ary lattice, then its dual $\mathcal{L}_q^\perp(B)$ is also a random $q$-ary lattice.

For the key generation, the LWE assumption is used to safely hide a short vector in such a random $q$-ary lattice. To see this, let us consider an LWE instance $(B, \boldsymbol{b} = B\boldsymbol{x}' + \boldsymbol{s} \bmod q)$, where $B \in \mathbb{Z}_q^{m\times(n-1)}$ and $\boldsymbol{x}' \in \mathbb{Z}_q^{n-1}$ are uniformly random, and $\boldsymbol{s} \in \mathbb{Z}_q^m$ is a short vector sampled from some distribution $s_i \sim \chi$. Firstly, note that $\boldsymbol{b}$ (when lifted to $\mathbb{Z}^m$) is in fact a BDD instance in the random $q$-ary lattice $\mathcal{L}_q(B)$, as, up to addition by $q$-vectors, $\boldsymbol{b}$ has distance only $\|\boldsymbol{s}\|$ to the lattice point $B\boldsymbol{x}$. While this interpretation will be of interest later, we will turn this BDD instance in a planted short vector, focusing on the key generation. By the LWE assumption the vector $\boldsymbol{b} \in \mathbb{Z}_q^m$ is indistinguishable from a uniformly random vector, and thus in particular we can append it as a column to $B$ to construct $A := \left[\ \boldsymbol{b} \mid B\ \right] \in \mathbb{Z}_q^{m\times n}$, that is indistinguishable from a random $q$-ary basis.

The lattice $\mathcal{L}_q(A)$ with $q$-ary basis $A$ is thus indistinguishable from a random $q$-ary lattice, however, the knowledge of the secret $\boldsymbol{x} := (1, -\boldsymbol{x}')$, allows one to find the *secret short vector* $\boldsymbol{s} \in \mathcal{L}_q(A)$ by computing $A \cdot \boldsymbol{x} \equiv \boldsymbol{s} - B \cdot \boldsymbol{x}' \equiv \boldsymbol{s} \in \mathbb{Z}_q^m$.

**Dual vectors and partitions** Recall that the dual lattice $\mathcal{L}^*$ consists of all vectors having an integer inner product with all primal vectors $\boldsymbol{v} \in \mathcal{L}$. As a result, any primal vector $\boldsymbol{v} \in \mathcal{L}$ naturally partitions the dual lattice $\mathcal{L}^*$ into layers $\{\boldsymbol{y} \in \mathcal{L}^* : \langle \boldsymbol{v}, \boldsymbol{y} \rangle = k\}$, for $k \in \mathbb{Z}$, orthogonal to $\boldsymbol{v}$, as illustrated in Fig. 3.1.




(a) $\mathcal{L}$ and its shortest vector $\boldsymbol{v}$ (b) $\mathcal{L}^*$ with partitions given by $\boldsymbol{v}$

Fig. 3.1: The relation between short vectors in $\mathcal{L}$ and partitions in $\mathcal{L}^*$

The distance between these layers is $1/\|\boldsymbol{v}\|$; the shorter the vector, the larger the separation. By considering the shortest vector in the primal, one can generate layers of points in the dual lattice that are at the maximum distance between each other. This space is precisely where a message can be encoded and hidden.

Let us consider a short vector $\boldsymbol{v} \in \mathcal{L}$ and a BDD instance $\boldsymbol{t} = \boldsymbol{y} + \boldsymbol{e} \in \mathbb{R}^m$ where $\boldsymbol{y} \in \mathcal{L}^*$ and $\boldsymbol{e} \in \mathbb{R}^m$ is small. Note that, given the target $\boldsymbol{t}$, it is generally hard to recover the error $\boldsymbol{e}$. However, since we know that $\langle \boldsymbol{v}, \boldsymbol{y} \rangle$ is integer, we can, knowing $\boldsymbol{v} \in \mathcal{L}$, compute

$$\langle \boldsymbol{e}, \boldsymbol{v} \rangle \equiv \langle \boldsymbol{t}, \boldsymbol{v} \rangle \bmod 1 \in [-\tfrac{1}{2}, \tfrac{1}{2}).$$

At this point, if the distance $1/\|\boldsymbol{v}\|$ between the partitions is large and the error $\boldsymbol{e}$ is small, one can hope that the closest partition to $\boldsymbol{t}$ is the one with the lattice point $\boldsymbol{y}$, or equivalently, $\langle \boldsymbol{e}, \boldsymbol{v} \rangle < \frac{1}{2}$. In particular we have $\langle \boldsymbol{e}, \boldsymbol{v} \rangle \leq \|\boldsymbol{e}\| \cdot \|\boldsymbol{v}\|$. If this is the case, we do not solely recover $\langle \boldsymbol{e}, \boldsymbol{v} \rangle$ modulo 1, but we recover the exact value $\langle \boldsymbol{e}, \boldsymbol{v} \rangle \in \mathbb{R}$, and therefore gain partial information about the error $\boldsymbol{e}$. We can therefore try to encode the message in this error so that one can recover it from this partial information. Moreover, note that we only have interest in the error here and thus might as well consider only the syndrome $\boldsymbol{t} \equiv \boldsymbol{e} \in \mathbb{R}^m / \mathcal{L}^*$.

Note that in the LWE setting, as will we see later, one considers the $q$-dual $q\mathcal{L}^* = \mathcal{L}_q^\perp(A)$ of $\mathcal{L} = \mathcal{L}_q(A)$ and thus instead of considering the previous inner products modulo 1, one has to consider them modulo $q$. Because $\mathcal{L}_q^\perp(A) \subset \mathbb{Z}^m$ this conveniently gives an integer space for the syndromes and allows one to consider the inner products modulo $q$ as elements of $\mathbb{Z}_q$.

**Visualizing an LWE ciphertext** As already spoiled in the previous subsection, an LWE ciphertext can be interpreted as a BDD instance, or more precisely

a syndrome, with respect of the $q$-dual lattice $\mathcal{L}_q^\perp(A)$ of the trapdoor lattice $\mathcal{L}_q(A)$. To observe this we consider a BGV ciphertext, which (for now, assume a message $\mu = 0$) is sampled as $\boldsymbol{c} := [t \cdot A^t \cdot \boldsymbol{r}]_q \in \mathbb{Z}_q^n$, with small $\boldsymbol{r} \in \{0,1\}^m$ and $t \geq 1$. A GSW ciphertext, for $\mu = 0$ can be considered as a collection of $n \log(q)$ different BGV ciphertexts using $t = 1$. We thus focus on the BGV case.

In LWE terms, the ciphertext is a random subset sum (controlled by $t \cdot \boldsymbol{r}$) of the LWE samples contained in the public key, essentially forming a new LWE sample. We claim however that $[A^t \cdot (t \cdot \boldsymbol{r})]_q$ represents the coset $t \cdot \boldsymbol{r} + \mathcal{L}_q^\perp(A)$, or more precisely the syndrome $t \cdot \boldsymbol{r} \in \mathbb{Z}^m / \mathcal{L}_q^\perp(A)$, as $t \cdot \boldsymbol{r}$ is a small error (refer to Fig. 3.2). Indeed, for any $\boldsymbol{y} \in \mathcal{L}_q^\perp(A)$, we have by definition that $A^t \boldsymbol{y} \equiv 0 \bmod q$, and thus

$$[A^t(t \cdot \boldsymbol{r} + \boldsymbol{y})]_q \equiv [A^t \boldsymbol{y} + A^t(t \cdot \boldsymbol{r})]_q \equiv [\boldsymbol{0} + A^t(t \cdot \boldsymbol{r})]_q \equiv [A^t(t \cdot \boldsymbol{r})]_q.$$

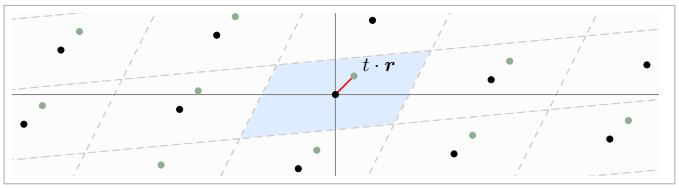



Fig. 3.2: Example of a lattice $\mathcal{L}^*$ (black points) and the coset $t \cdot \boldsymbol{r} + \mathcal{L}^*$ (green)

The LWE assumption precisely says that this syndrome is indistinguishable from a uniformly random syndrome in the (seemingly) random $q$-ary lattice $\mathcal{L}_q^\perp(A)$. For the decryption procedure, consider the secret short vector $\boldsymbol{s} \equiv A\boldsymbol{x} \in \mathcal{L}_q(A)$. Note that BGV first computes the modulo-$q$ inner product $[\langle \boldsymbol{c}, \boldsymbol{x} \rangle]_q$. Again, we claim that this is precisely the inner product between the short primal vector $\boldsymbol{s}$, partitioning $\mathcal{L}_q^\perp(A)$, and the syndrome $t \cdot \boldsymbol{r} \in \mathbb{Z}^m / \mathcal{L}_q^\perp(A)$. Indeed

$$[\langle \boldsymbol{c}, \boldsymbol{x} \rangle]_q = [\langle A^t \cdot (t \cdot \boldsymbol{r}), \boldsymbol{x} \rangle]_q = [\langle t \cdot \boldsymbol{r}, A \cdot \boldsymbol{x} \rangle]_q = [\langle t \cdot \boldsymbol{r}, \boldsymbol{s} \rangle]_q, \tag{3.4}$$

and assuming that the error $t \cdot \boldsymbol{r}$ and the short vector $\boldsymbol{s} \in \mathcal{L}_q(A)$ are small enough, i.e., $\langle t \cdot \boldsymbol{r}, \boldsymbol{s} \rangle < q/2$, we obtain precisely the inner product $t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle \in \mathbb{Z}$.

Since $t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle \equiv 0 \bmod t$, BGV encodes a message $\mu \in \{0, \ldots, t-1\}$ by adding $\mu \boldsymbol{e}_1$ to $\boldsymbol{c}$, using that $\langle \mu \boldsymbol{e}_1, \boldsymbol{x} \rangle = \mu \equiv \mu \bmod t$. Essentially, this corresponds to the addition of some vector $\mu \cdot \boldsymbol{d}$ to $t \cdot \boldsymbol{r}$ for which we know that $\langle \boldsymbol{d}, \boldsymbol{s} \rangle$ is small and satisfies $\langle \boldsymbol{d}, \boldsymbol{s} \rangle \equiv 1 \bmod t$.

### 3.2 Generalizing key generation: a (trapdoor) lattice

We have seen that the key generation of LWE-based encryption schemes constructs a *public* lattice $\mathcal{L}$ along with some short *secret* vector $\boldsymbol{s} \in \mathcal{L}$. This will

be the starting point of our generalization. For LWE, the lattice $\mathcal{L}$ is a (seemingly) random $q$-ary lattice $q\mathbb{Z}^m \subset \mathcal{L} \subset \mathbb{Z}^m$ of dimension $m$ along with a planted short vector. In our framework we will allow $\mathcal{L}$ to be any lattice with a planted short vector $\boldsymbol{s} \in \mathcal{L}$, and we leave the specifics of this to the instantiation of our framework. We assume that the public lattice $\mathcal{L}$ is represented by some (bad) basis $B$, and that the short vector $\boldsymbol{s} \in \mathcal{L}$ is secret. The basis $B$ will be our public key and the short vector $\boldsymbol{s}$ the secret key. As all the other steps will be generic to any lattice, one essentially only has to change the instantiation of the key generation.

The LWE-based encryption schemes we discussed encrypt by constructing a syndrome decoding instance in the $q$-dual lattice $q\mathcal{L}^*$ of $\mathcal{L}$. For decryption, the short primal vector gives a partition of the $q$-dual lattice as seen in Figure 3.1, which allows to partially decode or distinguish the syndrome.

For security reasons, the decryption procedure must be hard given only the public basis $B$ of $\mathcal{L}$. Several choices can be made here. For LWE-based schemes one usually uses the decision-LWE assumption which with our interpretation essentially says (ignoring the particular ambient space) that a syndrome in a random $q$-ary lattice is indistinguishable from a random target. Now we informally generalize this assumption, assuming an ambient space to be defined later $\mathcal{D} \supset q\mathcal{L}^*$ such that $\mathcal{D}/q\mathcal{L}^*$ is finite.

**Security Assumption 1 (Informal)** *Let $B, \boldsymbol{s} \in \mathcal{L}(B) \leftarrow$ `Keygen`. Given $B$ we assume that it is hard to distinguish a syndrome (as generated by the encryption procedure) in $\mathcal{D}/q\mathcal{L}^* \subset \mathbb{R}^m/q\mathcal{L}^*$ from a uniformly random one.*

Note that the goal of this work is not to give security proofs under this assumption, but merely to give a general lattice framework for HE.

*Remark 1.* The choice for an unstructured lattice $\mathcal{L}$ and a single secret short vector $\boldsymbol{s} \in \mathcal{L}$ was made to closely follow the classical LWE-based HE schemes, and to present the framework in its simplest form. One could easily replace this by allowing multiple short vectors, or even specific decoding procedures that depend on the lattice $q\mathcal{L}^*$. This could for instance improve the error resistance and the number of messages that can be packed in a single ciphertext. Moreover, one could simply replace the lattices here by structured (module-)lattices and the main steps of our approach should still work with minor changes.

### 3.3 The ambient space

Let $\mathcal{L}$ and $q\mathcal{L}^*$ be our primal and $q$-dual lattice, respectively, as generated by the key generation procedure, and let $B$ be the public basis of $\mathcal{L}$. Recall that in LWE-based schemes, encryptions are BDD instances $\boldsymbol{t} = \boldsymbol{v} + \boldsymbol{e} \in \mathbb{Z}^m$, or more precisely, their syndrome $\boldsymbol{t} \bmod q\mathcal{L}^* \in \mathbb{Z}^m/q\mathcal{L}^* \cong \mathbb{Z}_q^n$. Therefore, these BDD instances, and their errors, live in the ambient space $\mathbb{Z}^m$, which $q\mathcal{L}^*$ is naturally contained in as it is a $q$-ary lattice. Having $\mathbb{Z}^m$ as an ambient space is useful, as we know how to sample small errors $\boldsymbol{e} \in \mathbb{Z}^m$, needed for the public-key encryptions, and because the syndrome space $\mathbb{Z}^m/q\mathcal{L}^*$ is finite and naturally has a nice group

structure to make the homomorphic operations work. Generally however, we do not have that $q\mathcal{L}^*$ lies in $\mathbb{Z}^m$, as we do not restrict ourselves merely to cases were $q\mathcal{L}^*$ is integral. We thus have to construct some appropriate ambient space $\mathcal{D} \supset q\mathcal{L}^*$ with similar properties. We therefore aim to construct a generalized ambient space that satisfies the following two properties:

1. It has a good public basis $D$ that allows to sample small errors $\boldsymbol{e} \in \mathcal{D}$.
2. It satisfies $q\mathcal{D} \subset q\mathcal{L}^* \subset \mathcal{D}$.

The first point generalizes that the integer lattice $\mathbb{Z}^m$ has an orthogonal basis which allows to efficiently sample small errors. The second point is a generalization of the $q$-ary property, and forces $\mathcal{D}/q\mathcal{L}^*$ to be a subgroup of $\mathbb{Z}_q^m$, which will later be important to make the homomorphic operations work.

Let $C := B^{-t}$ be the basis of $\mathcal{L}^*$ dual to $B$, so that $qC$ is a basis of $q\mathcal{L}^*$. Before presenting our generalized construction we make a further observation about LWE-based schemes, where we assume that the basis $qC$ of $\mathcal{L}_q^\perp(A)$ has the block structure as in Eq. 3.3. Namely, one can observe that the Gram-Schmidt profile $q(\|\tilde{\boldsymbol{c}}_1\|, \ldots, \|\tilde{\boldsymbol{c}}_m\|)$ of $qC$ has a Z-shape: it starts with vectors of norm $q$, and ends with vectors of norm 1. Now, if one scales down these first $q$-vectors by $q$, they would obtain the basis $D$ with vectors $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n, q\boldsymbol{c}_{n+1}, \ldots, q\boldsymbol{c}_m$. Now observe two things: firstly, the Gram-Schmidt profile of $D$ is entirely flat, all Gram-Schmidt vectors have norm 1, and secondly, by construction $q\mathcal{L}^* \subset \mathcal{L}(D) \subset \mathcal{L}^*$, in fact $D$ is a basis of the ambient space $\mathbb{Z}^m$. Furthermore, after size-reducing $D$ one precisely obtains a basis of orthogonal unit vectors.

We will now generalize this construction. Generally, the Gram-Schmidt profile of $C$ does not necessarily have such a Z-shape, so more work is needed. Note that so far we have not specified the value of $q$, thus we still have the freedom to do so. Let us therefore assume that $q$ is of the form $q = p^k$ for some small prime $p$. Instead of scaling vectors by $q$ (or 1) only, we will allow any scaling of the form $p^i$ for $0 \leq i \leq k$. This choice is made for simplicity, for example we could also allow $q$ to be a product of many small primes, or another value that works well for the specific basis we have as input.

**Algorithm 3.1** Ambient space construction

**Input:** A basis $C$ of some lattice $\mathcal{L}^* \subset \mathbb{R}^m$, and moduli $p, q = p^k$ for some $k > 0$.
**Output:** A basis $D$ such that $q\mathcal{L}^* \subset \mathcal{L}(D) \subset \mathcal{L}^*$ and $\max_i \|\tilde{\boldsymbol{d}}_i\| \leq \max_i \|\tilde{\boldsymbol{c}}_i\|$

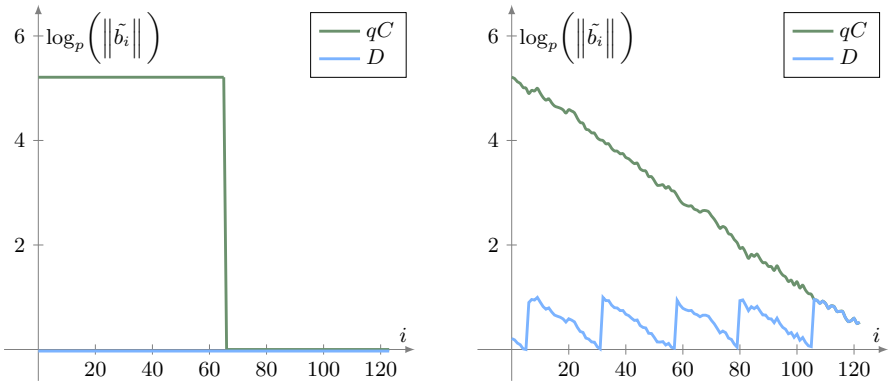
1: **procedure** ConstructAmbientSpace($C, p, k$)
2: Compute the Gram-Schmidt orthogonalisation $\tilde{\boldsymbol{c}}_1, \ldots, \tilde{\boldsymbol{c}}_m$ of $C$.
3: $q \leftarrow p^k$, $\ell_{\max} \leftarrow \max_i \|\tilde{\boldsymbol{c}}_i\|$
4: $j_i = \min\{k, \lfloor \log_p(\ell_{\max} / \|\tilde{\boldsymbol{c}}_i\|) \rfloor\}$ for all $i = 1, \ldots, m$
5: $\boldsymbol{d}_i \leftarrow p^{j_i} \cdot \boldsymbol{c}_i$ for all $i = 1, \ldots, m$
6: **return** $D = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_m)$
7: **end procedure**

To construct such an ambient space $\mathcal{D}$, we will use the procedure given in Algorithm 3.1 that generates an appropriate $\mathcal{D}$ such that $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$ along with a good basis $D$ (see Figure 3.3).

**Lemma 3.1.** *Let $C$ be a basis of some lattice $\mathcal{L}^* \subset \mathbb{R}^m$ and let $p > 0, q = p^k$ be some moduli for some $k > 0$. Let $\ell_{\min} = \min_i \|\tilde{\boldsymbol{c}}_i\|$ and $\ell_{\max} = \max_i \|\tilde{\boldsymbol{c}}_i\|$. Then Algorithm 3.1 returns a basis $D$ of an ambient space $\mathcal{D}$ such that $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$ and such that for all $i = 1, \ldots, m$ we have*

$$\min\{\ell_{max}/p, q\ell_{\min}\} \leq \|\tilde{\boldsymbol{d}}_i\| \leq \ell_{\max}.$$

*Proof.* Consider the variables used in Algorithm 3.1. Note that each $j_i$ satisfies $0 \leq j_i \leq k$, and thus $1 \leq p^{j_i} \leq q$ for all $i = 1, \ldots, m$. So by construction $q\mathcal{L}^* \subset \mathcal{L}(D) \subset \mathcal{L}^*$. For the bounds note that $\|\tilde{\boldsymbol{d}}_i\| = p^{j_i} \cdot \|\tilde{\boldsymbol{c}}_i\|$. The upper bound then follows immediately from the fact that $j_i \leq \log_p(\ell_{\max}/\|\tilde{\boldsymbol{c}}_i\|)$. For the lower bound we consider two cases, first suppose that $\log_p(\ell_{\max}/\|\tilde{\boldsymbol{c}}_i\|) < k + 1$, then $j_i > \log_p(\ell_{\max}/\|\tilde{\boldsymbol{c}}_i\|) - 1$ and thus $\|\tilde{\boldsymbol{d}}_i\| > \ell_{\max}/p$. For the other case we have that $j_i = k$ and we obtain $\|\tilde{\boldsymbol{d}}_i\| = q \cdot \|\tilde{\boldsymbol{c}}_i\| \geq q\ell_{\min}$. □




(a) $qC$ is the public basis of a random $q$-ary lattice, with $p = q = 123$

(b) $qC$ is an LLL-reduced basis of (a rotation of ) $\mathbb{Z}^m$, with $p = 3$ and $q = p^5$

Fig. 3.3: Gram-Schmidt profiles of the bad basis $qC$ of $q\mathcal{L}^*$ and the good basis $D$ of $\mathcal{D}$ as given by Algorithm 3.1

**Concrete parameters and requirements** The basis $D$ of the ambient space $\mathcal{D}$ obtained from Lemma 3.1, combined with Lemma 2.1, allows to sample small errors in $\mathcal{D}$ from a discrete Gaussian with any parameter $\sigma_{\text{enc}} \geq \ell_{\max} \cdot \sqrt{\ln(2m+4)/\pi}$. For a small error $\boldsymbol{r}$, sampled from this distribution, the decryption error $\langle \boldsymbol{r}, \boldsymbol{s} \rangle$ with the short secret vector $\boldsymbol{s} \in \mathcal{L}$ will roughly be of size $\sigma_{\text{enc}} \cdot \|\boldsymbol{s}\|$, which we have to compare to the modulus $q$.

A choice for $q$ would be a value $1 \leq q \leq \ell_{\max}/\ell_{\min}$ which guarantees that $\sigma_{\text{enc}} > q\ell_{\min}/2$, that is a minimum requirement to make the resulting syndrome decoding problem with respect to $q\mathcal{L}^*$ non-trivial to solve[5]. By increasing $q$ we can thus make the lattice $q\mathcal{L}^*$ sparser and allow for a larger decryption noise, so naturally one would choose the maximum value $q \approx \ell_{\max}/\ell_{\min}$.

Notice that this is also precisely the choice that is made for LWE-schemes, where $q = \ell_{\max}/\ell_{\min}$. Furthermore, notice that for $q > \ell_{\max}/(p\ell_{\min})$ we obtain that all Gram-Schmidt norms of $D$ are within a factor $p$ of each other, i.e., if $p$ is small and $q$ is large enough the profile of $D$ is essentially flat and $D$ is near-optimally reduced.

We thus observe that a smaller secret vector $\boldsymbol{s} \in \mathcal{L}$ reduces the decryption error while a worse public basis $C$ of $\mathcal{L}^*$ (typcially giving larger $\ell_{\max}$ and smaller $\ell_{\min}$) increases it but can increase the modulus $q$ even more. Recall that for the BGV scheme the decryption noise is roughly squared after a single multiplication, so the minimum requirement on $\|\boldsymbol{s}\|$ would be that $(\|\boldsymbol{s}\| \cdot \sigma_{\text{enc}})^2 < \mathcal{O}(\sqrt{q})$.

To allow for many homomorphic multiplications, however, the decryption noise of fresh ciphertexts should be roughly constant with respect to $q$. In particular, one would preferably have $\|\boldsymbol{s}\| \sim \ell_{\max}^{-1}$. Notice that for the LWE case we have precisely this, as $\boldsymbol{s}$ is binary and $\ell_{\max} = 1$. In the rest of this work we assume to have a small enough secret $\boldsymbol{s} \in \mathcal{L}$ such that we can assume that the decoding noise of fresh ciphertexts is roughly constant relative to the modulus $q$. In our instantiation based on LIP (Section 5), we will shortly explain how to achieve this.

**Connections with structural lattice reduction** Our goal and method of creating an ambient space $\mathcal{D} \supset q\mathcal{L}^*$ along with a nearly orthogonal basis $D$ satisfying $\mathcal{D}/q\mathcal{L}^* \cong G$ for some finite abelian group $G$ is similar to the work [GINX16]. They introduce a method called *structural lattice reduction*, which given any sufficiently large group $G$ (and some minor additional constraints), creates an ambient space as above for this specific group. Contrary to our method, they fix some group $G$ and adapt the ambient space to it, while we allow $G$ to be determined by the procedure, only restricting that $G$ must be a subgroup of $\mathbb{Z}_q^m$ (where $q$ is a product of small elements). We show later that this property is sufficient for the homomorphic operations that we will define. This extra freedom allows our method to be simpler and a bit more efficient, in particular, a direct comparison shows that our group can be up to a factor $O(m^m)$ smaller, and the modulus $q$ can be up to a factor $O(m)$ smaller, for the same bound on $\|\tilde{D}\|$. Fixing a particular group $G$ however, could come with some additional benefits and our framework could as well be instantiated using the structural lattice reduction procedure.

**Coefficient representation** So far, to keep the geometric intuition, we have presented our ambient space $\mathcal{D}$ as a lattice with a good basis $D$, and small errors

[5] For a secure instantiation, and depending on the precise basis $qC$ of $q\mathcal{L}^*$, one could need a slightly larger margin, for simplicity we will ignore this minor dependency.

$\boldsymbol{e} \in \mathcal{D}$ and syndromes $\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$ are simply assumed to be represented by lattice vectors. One of the crucial aspects of the key switching procedure using the `Bits` and `Pow` functions in BGV and the gadget matrix used in the GSW scheme is however that we are working with integer vectors (or say in $\mathbb{Z}_q^m$). Furthermore, working with such integer vectors is more useful in concrete implementations. This is, however, not a problem in our more general setting, as in fact a lattice vector $\boldsymbol{v} \in \mathcal{D}$ can still be represented by their integer coefficient vector $\boldsymbol{y} \in \mathbb{Z}^m$ in the basis $D$ such that $\boldsymbol{v} = D \cdot \boldsymbol{y}$. Similarly, the isomorphism $\mathcal{D}/q\mathcal{L}^* \cong G \subset \mathbb{Z}_q^m$ where $G$ is a subgroup of $\mathbb{Z}_q^m$ can be made explicit by the map $\boldsymbol{c} \mapsto D^{-1}\boldsymbol{c}$. Later we will see that applying the `Bits`, `Pow` and gadget matrix can simply be applied to this coefficient representation. E.g. for $\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$ and $\boldsymbol{s} \in \mathcal{L}$ we have that $D^t\boldsymbol{s} \in \mathbb{Z}^m$, and we have that

$$\langle \boldsymbol{c}, \boldsymbol{s} \rangle \equiv \langle D^{-t}\boldsymbol{c}, D^t\boldsymbol{s} \rangle \equiv \langle \texttt{Bits}(D^{-t}\boldsymbol{c}), \texttt{Pow}(D^t\boldsymbol{s}) \rangle.$$

## 4 Abstract HE cryptosystems from lattices

Following the key generation and ambient space construction in the previous section, we now present some lattice-based generalizations of the BGV [BGV12] and GSW [GSW13] homomorphic encryption cryptosystems.

### 4.1 Lattice-BGV

The Lattice-BGV scheme is a lattice generalization of BGV that can in principle be instantiated, in public key mode, using any hardness assumption that hides the secret shortest vector of a lattice (e.g., LWE, NTRU, LIP and so on), and for which decoding in the dual is hard.

**Key generation** Given a lattice dimension $m$, let $\mathcal{L}$ and $q\mathcal{L}^*$ be our primal and $q$-dual lattice, respectively, and let $D$ be the basis of the ambient space $\mathcal{D}$ as generated by Algorithm 3.1 such that $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$. The key generation follows from the chosen hardness assumption. It is expected for the output of this procedure to be a public bad basis $B$ of some lattice $\mathcal{L}$ along with a sufficiently short secret vector $\boldsymbol{s} \in \mathcal{L}$. Recall that for the LWE schemes we have $\mathcal{L} = \mathcal{L}_q(A)$ and the secret key $\boldsymbol{x} = (1, -\boldsymbol{x}') \in \mathbb{Z}_q^n$ represents the coefficients of the shortest vector $\boldsymbol{s} \in \mathcal{L}$ with respect to the $q$-ary basis $A$, since $A \cdot \boldsymbol{x} \equiv \boldsymbol{s} \bmod q$.

**Encrypting and decrypting** As introduced in Section 3.1, the encryption process in BGV samples a small error $\boldsymbol{e} \in \mathcal{D}$, containing an encoded message, which is then canonically represented by some syndrome $\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$. The secret short vector $\boldsymbol{s} \in \mathcal{L}_q$ will be used to partially decode it and recover the encoded message. To sample a small error in the ambient space $\mathcal{D}$ we precisely use its public good basis $D$ that we constructed in Section 3.3. We assume, without loss

of generality, that the first basis vector $\boldsymbol{d}_1$ of $D$ has order precisely $q$ in $\mathcal{D}/q\mathcal{L}^*$. The goal is to generalize the BGV encryption procedure, given by

$$[t \cdot A^t \cdot \boldsymbol{r} + (\mu, 0, \ldots, 0)^t]_q.$$

We start by considering a (scaled) small sample $\boldsymbol{r} \sim \mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$ in the ambient space $\mathcal{D}$, with $\sigma_{\text{enc}}$ chosen as explained in Section 3.3. Then, the message is encoded in the small error $\boldsymbol{e} \in \mathcal{D}$ as follows:

$$\boldsymbol{e} := t \cdot \boldsymbol{r} + D \cdot (\mu, 0, \ldots, 0)^t \in \mathcal{D}, \tag{4.1}$$

where $t \in \mathbb{N}$ is the plaintext modulus, $\mu \in \mathbb{Z}_t$ is the message, and $D$ is the good basis of $\mathcal{D}$. The discrete Gaussian sampler $\mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$ returns a point $\boldsymbol{r}$ that is close to the origin in the ambient space $\mathcal{D}$, and thus $\boldsymbol{e} \approx t \cdot \boldsymbol{r}$ is also small. The following step is to hide the sampled error $\boldsymbol{e}$, containing the message $\mu$, in a canonical coset representative $\boldsymbol{c} \in \boldsymbol{e} + q\mathcal{L}^*$. For this we use the fundamental region of $q\mathcal{L}^*$ defined by the bad basis $qC = qB^{-t}$:

$$\boldsymbol{c} \equiv \boldsymbol{e} \bmod \mathcal{P}(qC) \in \mathcal{D}/q\mathcal{L}^*. \tag{4.2}$$

This process is represented in Fig. 4.1. Because $\mathcal{L}^* \subset \mathcal{D} \subset q\mathcal{L}^*$ one can represent $\boldsymbol{e}$ and $\boldsymbol{c}$ as integer vectors with respect to the basis $C$ and thus the above operation can be simply implemented with modular arithmetic.

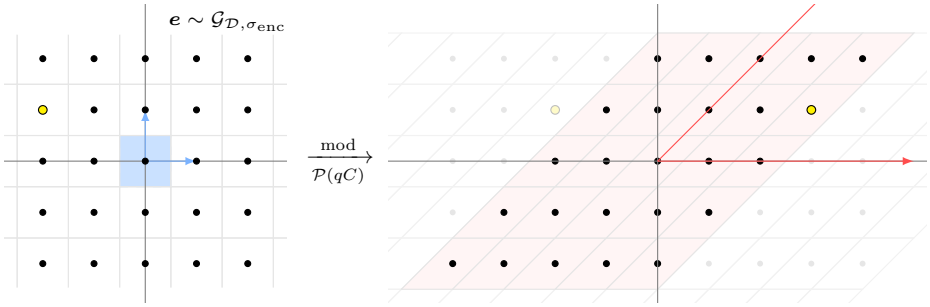



Fig. 4.1: Hiding a sample $\boldsymbol{e} \sim \mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$ (in yellow) within the fundamental region of $\mathcal{D}/q\mathcal{L}^*$, assuming $\mathcal{L}^* = \mathbb{Z}^n$, with $q = 5$

Recall that the secret key $\boldsymbol{s}$ is a short vector in the primal lattice $\mathcal{L}$. The decryption therefore follows the arguments from Section 3, i.e., we have

$$\begin{aligned} [[\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_q]_t &\equiv [[\langle t \cdot \boldsymbol{r} + D \cdot (\mu, 0, \ldots, 0)^t, \boldsymbol{s} \rangle]_q]_t \\ &\equiv [t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle + \langle \boldsymbol{d}_1, \boldsymbol{s} \rangle \cdot \mu]_t \equiv \mu \in \mathbb{Z}_t, \end{aligned} \tag{4.3}$$

assuming that $|t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle + \langle \boldsymbol{d}_1, \boldsymbol{s} \rangle \cdot \mu| < q/2$ and $\langle \boldsymbol{d}_1, \boldsymbol{s} \rangle \equiv 1 \bmod t$. As discussed in Section 3.3, we assumed that, for appropriate parameters, the decryption noise $t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle$ is roughly constant relative to $q$ and thus the correct decryption follows.

**Homomorphic multiplications** Homomorphic multiplications can be evaluated as in the standard BGV scheme, and their correctness follows from the fact that the inner product of two tensor products becomes the product of two inner products. Given two ciphertexts $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{D}/q\mathcal{L}^*$, their tensor product $\boldsymbol{c}^\times = \boldsymbol{c}_1 \otimes \boldsymbol{c}_2 \in (\mathcal{D}/q\mathcal{L}^*) \otimes (\mathcal{D}/q\mathcal{L}^*)$ is a correct ciphertext under the new secret key $\boldsymbol{s}^\times = \boldsymbol{s} \otimes \boldsymbol{s} \in \mathcal{L} \otimes \mathcal{L}$ representing the product of the encrypted values $\mu_1$ and $\mu_2$:

$$\begin{aligned}[[\langle \boldsymbol{c}^\times, \boldsymbol{s}^\times \rangle]_q]_t &= [[\langle \boldsymbol{c}_1 \otimes \boldsymbol{c}_2, \boldsymbol{s} \otimes \boldsymbol{s} \rangle]_q]_t = [[\langle \boldsymbol{c}_1, \boldsymbol{s} \rangle \cdot \langle \boldsymbol{c}_2, \boldsymbol{s} \rangle]_q]_t \\ &= [[(t \cdot \langle \boldsymbol{r}_1, \boldsymbol{s} \rangle + \langle \boldsymbol{d}_1, \boldsymbol{s} \rangle \cdot \mu_1) \cdot (t \cdot \langle \boldsymbol{r}_2, \boldsymbol{s} \rangle + \langle \boldsymbol{d}_1, \boldsymbol{s} \rangle \cdot \mu_2)]_q]_t \\ &= [[t \cdot e + \langle \boldsymbol{d}_1, \boldsymbol{s} \rangle^2 \mu_1 \mu_2]_q]_t = [t \cdot e + \langle \boldsymbol{d}_1, \boldsymbol{s} \rangle^2 \mu_1 \mu_2]_t = \mu_1 \cdot \mu_2 \in \mathbb{Z}_t,\end{aligned}$$

assuming that $|t \cdot e + \langle \boldsymbol{d}_1, \boldsymbol{s} \rangle^2 \mu_1 \mu_2| < q/2$ where we use $t \cdot e \in \mathbb{Z}$ to refer to the decryption noise coming from the other terms to ease readability. Just as for the original BGV scheme, the decryption noise $t \cdot e$ of the product is roughly the square of that of the input ciphertexts. Notice that, again, with a right instantiation as discussed in Section 3.3, the original decryption noise is roughly constant relative to $q$ and thus for $q$ sufficiently large the above multiplication can be correctly decrypted.

While the noise is bounded independently of $q$, it still is roughly squared after a single multiplication, and therefore grows exponentially on repeated multiplications, allowing us to do at most $O(\log(q))$ of them. Similarly, the dimension of the ciphertext space and the lattice involved is squared as $\dim(\mathcal{L} \otimes \mathcal{L}) = \dim(\mathcal{L})^2$. Just as in the LWE case we can solve these two problems by modulus switching and key switching, respectively.

**Modulus switching** We shortly recall the modulus switching step in the original BGV scheme. Given starting and final moduli $q_1, q_2$ such that $q_2 \,|\, q_1$ and $q_1 \equiv q_2 \equiv 1 \bmod t$, the LWE modulus switching proceeds in two steps. First the ciphertext is shifted by some $\boldsymbol{\delta} \in \mathbb{Z}^m/q_1\mathcal{L}^* \cong \mathbb{Z}_q^n$ term, with the objective to make it divisible by $q_1/q_2$, after which it is divided by this factor.

The $\boldsymbol{\delta}$ term may add a small amount of noise, but the subsequent scaling will drastically reduce it. More precisely, given a ciphertext $\boldsymbol{c}_1 \in \mathbb{Z}^m/q_1\mathcal{L}^*$, the modulus switching procedure first moves $\boldsymbol{c}_1$, and therefore the error, by some $\boldsymbol{\delta}$ term such that

$$\boldsymbol{c}' := (\boldsymbol{c} - \boldsymbol{\delta}) \in q_1/q_2 \cdot \mathbb{Z}^m/q_1\mathcal{L}^* \cong q_1/q_2 \cdot \mathbb{Z}_{q_1}^n.$$

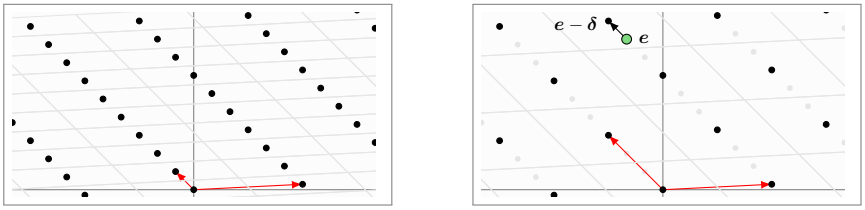
Then, given $\boldsymbol{c}' \in q_1/q_2 \cdot \mathbb{Z}_{q_1}^n$ we divide to obtain the new ciphertext $\boldsymbol{c}_2 := \frac{q_2}{q_1}\boldsymbol{c}' \in \mathbb{Z}_{q_2}^n \cong \mathbb{Z}^m/q_2\mathcal{L}^*$ which is interpreted as a syndrome with respect to $q_2\mathcal{L}^*$. Note that in terms for the corresponding BDD errors $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{Z}^m$, we similarly have $\boldsymbol{e}_2 = (\boldsymbol{e}_1 - \boldsymbol{\delta})/(q_1/q_2)$, and thus the absolute norm $\|\boldsymbol{e}_2\| \approx q_2/q_1 \cdot \|\boldsymbol{e}_1\|$ is significantly reduced, assuming $\boldsymbol{\delta}$ is sufficiently small. In the original BGV scheme this small term is defined as:

$$\boldsymbol{\delta} = t \cdot [\boldsymbol{c} \cdot t^{-1}]_{q_1/q_2} \in \mathbb{Z}_{q_1}^n \cong \mathbb{Z}^m/q_1\mathcal{L}^*,$$

where $t^{-1}$ is the inverse of $t$ mod $q_1/q_2$, and it is required to keep the encoding of the message during the scale procedure. Notice that this definition respects the requirements for smallness ($\boldsymbol{\delta}$ is small), scalability (the ciphertext becomes divisible by $q_1/q_2$), and correctness (the encrypted message is kept) since:

- Each coefficient of $\boldsymbol{\delta}$ is bounded by $\frac{1}{2}t \cdot q_1/q_2$.
- $\boldsymbol{c} - \boldsymbol{\delta} \equiv \mathbf{0} \bmod q_1/q_2$ allows to make the ciphertext divisible by $q_1/q_2$.
- $\boldsymbol{\delta} \equiv \mathbf{0} \bmod t$ allows to keep the encoding of the message.

**Towards a general modulus switching** Having a clear view and more freedom in terms of lattices allows us to define a somehow more generic approach to modulus switching. In this more general setting, modulus switching has the goal to switch from some initial modulus $q_1$ to a smaller modulus $q_2$ such that $q_2 \,|\, q_1$. In particular, given $q_1\mathcal{L}^* \subset \mathcal{D}_1$ and $q_2\mathcal{L}^* \subset \mathcal{D}_2$ with their respective ambient spaces, the aim is to switch a ciphertext $\boldsymbol{c}_1 \in \mathcal{D}_1/q_1\mathcal{L}^*$ to a ciphertext $\boldsymbol{c}_2 \in \mathcal{D}_2/q_2\mathcal{L}^*$, while reducing the error by roughly a factor $q_1/q_2$ and while keeping the encoded message intact. We assume that $\mathcal{D}_1$ and $\mathcal{D}_2$ are both generated by our ambient space construction using the moduli $q_1$ and $q_2$ respectively. Note that by construction we have that $q_1/q_2 \cdot \mathcal{D}_2 \subset \mathcal{D}_1 \subset \mathcal{D}_2$, so $(q_1/q_2)\mathcal{D}_2$ is a sparsification of $\mathcal{D}_1$ (see Fig. 4.2).




(a) An ambient space $\mathcal{D}_1$ (b) Rounding the error in $q_1/q_2\mathcal{D}_2$

Fig. 4.2: Defining a sparser sublattice $q_1/q_2\mathcal{D}_2 \subset \mathcal{D}_1$ from $\mathcal{D}_1$, assuming $q_1/q_2 = 3$ and $\mathcal{D}_1 = \mathcal{D}_2$ for simplicity. Notice that all the points in $q_1/q_2\mathcal{D}_2$ are divisible by $q_1/q_2$ by definition

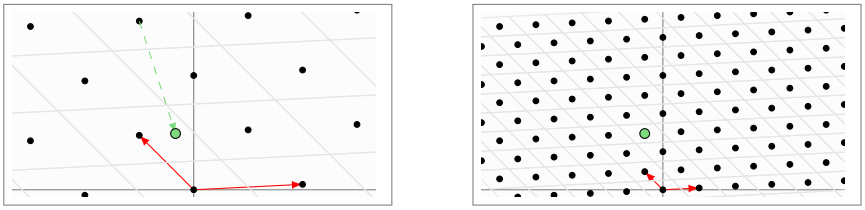
Similarly to the original LWE case, we proceed in two steps which can be summarized by the following:

$$\mathcal{D}_1/q_1\mathcal{L}^* \xrightarrow[\text{round}]{-\boldsymbol{\delta}} (q_1/q_2)\mathcal{D}_2/q_1\mathcal{L}^* \xrightarrow[\text{scale}]{\times q_2/q_1} \mathcal{D}_2/q_2\mathcal{L}^*,$$

where in the first step we *round* to the sparsification $(q_1/q_2)\mathcal{D}_2$ of the ambient space $\mathcal{D}_1$, and then *scale* down to the new ambient space $\mathcal{D}_2$ with respect to the lattice $q_2\mathcal{L}^*$. We first focus on the rounding part, where, given some ciphertext $\boldsymbol{c}_1 \in \mathcal{D}_1/q_1\mathcal{L}^*$, we need to find some small $\boldsymbol{\delta}$ such that $\boldsymbol{c}_1 - \boldsymbol{\delta} \in (q_1/q_2)\mathcal{D}_2/q_1\mathcal{L}^*$. Note that this is precisely an (approximate) syndrome decoding problem in

$\mathcal{D}_1/(q_1/q_2)\mathcal{D}_2$, where $\boldsymbol{\delta}$ is a small error in the coset $\boldsymbol{c}_1 + (q_1/q_2)\mathcal{D}_2/q_1\mathcal{L}^*$. In particular, the better we can decode in $(q_1/q_2)\mathcal{D}_2$, the smaller $\boldsymbol{\delta}$ can be.

To decode we can again use the good basis that our ambient space $\mathcal{D}_2$, and thus $(q_1/q_2)\mathcal{D}_2$ has, generalizing the LWE setting where decoding in $(q_1/q_2)\mathbb{Z}^m$ is also possible due to a good (orthogonal) basis. In particular, we can use Babai's decoding algorithm which always finds a $\boldsymbol{\delta}$ such that $\|\boldsymbol{\delta}\|^2 \leq \frac{q_1}{4q_2}\sum_{i=1}^{m}\|\tilde{\boldsymbol{d}}_i\|^2$, where the latter are the Gram-Schmidt norms of the basis $D$, which are bounded independently of $q$ by Lemma 3.1. The dependency on $q_1/q_2$ is natural as it directly determines the sparsity of $q_1/q_2\mathcal{D}_2$, and that factor will be removed by the scaling step. Now that the $\boldsymbol{c} - \boldsymbol{\delta}$ point lies in $q_1/q_2\mathcal{D}_2 \subset \mathcal{D}_1$, it is possible to scale it down, along with the lattice $q_1\mathcal{L}^*$, to obtain a new ciphertext with a smaller error (Fig. 4.3) with respect to the lattice $q_1\mathcal{L}^*/(q_1/q_2) = q_2\mathcal{L}^*$.



(a) Scaling the error (b) The scaled error in $\mathcal{D}_2$

Fig. 4.3: Given a "rounded" ciphertext in $q_1/q_2\mathcal{D}_2$, we can scale it down by the factor $q_1/q_2$

As a result, we obtain a point that, although scaled from the original lattice $q_1\mathcal{L}^*$ to $q_2\mathcal{L}^*$, has an error that, in absolute terms, is smaller. Ignoring the additional error introduced by the by $\boldsymbol{\delta}$, the reduction is by a factor $q_1/q_2$.

Despite this nice procedure, we still require caution as this operation could erase the content of the message stored in the first component. In particular, the decoding algorithm could return a point that decrypts to a different message than the one originally contained in $\boldsymbol{c}$. In LWE, this is tackled by using a $\boldsymbol{\delta}$ term that is a multiple of the plaintext modulus $t$, so that during decryption can easily be removed.

The difference is how the rounding is performed: in LWE, the delta to the closest vector in the sparse lattice is given by $t \cdot [\boldsymbol{c} \cdot t^{-1}]_{q_1/q_2}$, which implicitly acts as a syndrome decoding algorithm in $t\mathcal{D}_1/(q_1/q_2)\mathcal{D}_2 \cong \mathbb{Z}^n_{q_1/q_2}$, returning a small $\delta \in t\mathcal{D}_1$. Mimicking this, combined with our more general syndrome decoding approach, we can therefore compute the $\boldsymbol{\delta}$ term as follows:

$$\boldsymbol{\delta} = t \cdot \text{approxSDP}(\boldsymbol{c} \cdot t^{-1}, (q_1/q_2)\mathcal{D}_2) \in t\mathcal{D}_1. \tag{4.4}$$

**Key switching**

Given two public lattices $\mathcal{L}_1, \mathcal{L}_2$ with secret short vectors $\boldsymbol{s}_1, \boldsymbol{s}_2$, dimensions $m_1, m_2 \in \mathbb{N}$ and ambient spaces $\mathcal{D}_1, \mathcal{D}_2$ for $q\mathcal{L}_1^*, q\mathcal{L}_2^*$ respectively, the goal of the key switching procedure is to transform a ciphertext $\boldsymbol{c} \in \mathcal{D}_1/q\mathcal{L}_1^*$, with respect to the secret key $\boldsymbol{s}_1$, to a ciphertext in $\mathcal{D}_2/q\mathcal{L}_2^*$ containing the same message with respect to the secret key $\boldsymbol{s}_2$. In particular, this allows us to switch from a larger dimensional lattice, say $\mathcal{L}_1 = \mathcal{L} \otimes \mathcal{L}$ obtained after a multiplication, back to the starting dimension. As the key switching procedure can essentially be generalized to any additive homomorphic scheme, without using any specific lattice properties, we will be rather brief about it.

Recall from the LWE case that the switching key $\tau_{\boldsymbol{s}_1 \to \boldsymbol{s}_2}$ essentially consists of encryptions $D_2(\boldsymbol{c}_i^{(0)} + ((D_1^t \boldsymbol{s}_1)_i, 0, \ldots, 0)^t) \in \mathcal{D}_2/q\mathcal{L}_2^*$ of each coefficient of $D_1^t \boldsymbol{s}_1$ under $\boldsymbol{s}_2$ (ignoring the modulo $t$ encoding), where $D_2\boldsymbol{c}_i^{(0)}$ is an encryption of 0. Given the public ciphertext $\boldsymbol{c} = D_1(c_1, \ldots, c_{m_1})^t \in \mathcal{D}_1/q\mathcal{L}_1^*$ we simply scale and sum over these encryptions to get

$$\begin{aligned}\boldsymbol{c}' &:= D_2 \cdot \sum_{i=1}^{m_1} c_i \cdot \left(\boldsymbol{c}_i^{(0)} + ((D_1^t \boldsymbol{s}_1)_i, 0, \ldots, 0)^t\right) \\ &= \left(\sum_{i=1}^{m_1} c_i \cdot D_2 \boldsymbol{c}_i^{(0)}\right) + D_2 \cdot ([\langle \boldsymbol{c}, \boldsymbol{s}_1 \rangle]_q, 0, \ldots, 0).\end{aligned}$$

We thus obtain a scaled sum of encryptions of 0 which masks the message $[\langle \boldsymbol{c}, \boldsymbol{s}_1 \rangle]_q$. Assuming $\boldsymbol{c}$ is a valid ciphertext we have $[\langle \boldsymbol{c}, \boldsymbol{s}_1 \rangle]_q = te + \mu$ for the message $\mu$ and some *small* decryption noise $e$. Assuming the $c_i$ are small, the errors coming from the sum of encryptions of 0, and the error $D_2 \cdot (te, 0, \ldots, 0)^t$, should be small enough for the new ciphertext $\boldsymbol{c}'$ to correctly decrypt to $\mu$ under $\boldsymbol{s}_2$. For the $c_i$ to be small enough we can use the BV binary decomposition trick.

*Remark 2.* The encryptions of 0 used to construct the switching key in the LWE case are generated using the public key generation procedure. This must be seen as an artifact of the specific key generation procedure under LWE. More generally, this could be interpreted as a *secret key encryption* procedure, i.e., the valid encryptions are constructed with knowledge of the secret key. This allows to make the error smaller in the direction of $\boldsymbol{s}_2$, while making it uniform in the direction orthogonal to it to compensate for any concrete security loss. One could easily generalize this to our lattice setting, or alternatively just use the *public key encryption* procedure (albeit leading to a larger decryption error).

Summarized we obtain the following key switching procedure.

1. Sample $m_1 \cdot \lceil \log(q) \rceil$ encryptions $D_2 \cdot \boldsymbol{c}_{(i,j)}^{(0)} \in \mathcal{D}_2/q\mathcal{L}_2^*$ of zero indexed by the pair $(i, j)$ for $i = 1, \ldots, m_1$ and $j = 1, \ldots, \lceil \log(q) \rceil$ via a public or secret key encryption procedure.
2. Compute the switching key $\tau_{\boldsymbol{s}_1 \to \boldsymbol{s}_2} = (\boldsymbol{c}_{(i,j)}^{(0)} + (\mathtt{Pow}((D_1^t \boldsymbol{s}_1)_i)_j, 0, \ldots, 0)^t)_{(i,j)}$.

3. To switch from $\boldsymbol{c}$ to $\boldsymbol{c}'$ given $\tau := \tau_{\boldsymbol{s}_1 \to \boldsymbol{s}_2}$ compute

$$\boldsymbol{c}' = D_2 \cdot \sum_{i=1}^{m_1} \sum_{j=1}^{\lceil \log(q) \rceil} (\texttt{Bits}((D_1^{-t}\boldsymbol{c})_i)_j \cdot \tau_{(i,j)}.$$

### 4.2 Lattice-GSW

We can now use previous results to define the Lattice-GSW scheme. The key generation is the same as in the previous scheme: the public key is a bad basis $B$ of a lattice $\mathcal{L}$ along with a good basis $D$ of the ambient space $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$ (where $q = p^k$), and the secret key $\boldsymbol{s} \in \mathcal{L}$ is a sufficiently short vector of $\mathcal{L}$. For the GSW scheme and its operations, it is useful to express elements more explicitly in their coefficient representation, which we will therefore do.

**Encrypting and decrypting** As observed in Section 2.3, the encryption process of GSW is slightly different from the BGV one. In particular, a ciphertext will now be a matrix consisting of $m \log_p(q)$ encryptions of zero from Lattice-BGV (with $t = 1$), one for each column. So we first define the error matrix $E$ with columns in $\mathcal{D}$ such that:

$$E := D \cdot \begin{pmatrix} | & | & & | \\ \boldsymbol{e}_1 & \boldsymbol{e}_2 & \dots & \boldsymbol{e}_n \\ | & | & & | \end{pmatrix} \text{ where each } D\boldsymbol{e}_i \sim \mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}.$$

We identify this matrix space with $\mathcal{D}^{m \log_p(q)}$. Then the syndrome matrix $E$ mod $q\mathcal{L}^* \in (\mathcal{D}/q\mathcal{L}^*)^{m \log_p(q)}$ represents $m \log_p(q)$ encryptions of zero (equivalent to $[A^t R]_q$ in LWE-based GSW). Without loss of generality we assume a plaintext space $\mathbb{Z}_p$, where $q = p^k$. In order to encrypt a message $\mu \in \mathbb{Z}_p$, we simply add a multiple of the gadget matrix $\mu \cdot G$ to the coefficient representation of the just crafted matrix $E$. We just have to be careful as the gadget matrix must be constructed with respect to the base $p$ and to match the group structure of $\mathcal{D}/q\mathcal{L}^*$, defined during the generation of $D$ (Algorithm 3.1). One can always cover the larger group $\mathbb{Z}_q^m$ by taking

$$G := (1, p, p^2, \dots, p^{\log(q)-1}) \otimes I_m \in \mathbb{Z}_q^{m \times m \log(q)},$$

however one can further optimize this and reduce the $m \log_p(q)$ ciphertexts down to $\log_p(|\mathcal{D}/q\mathcal{L}^*|)$ by noting that $D \cdot G \bmod q\mathcal{L}^*$ has many zero columns which can simply be removed (leading to $n \log_p(q)$ ciphertexts in the LWE case). To simplify notation we will continue to use the full gadget matrix $G$. Encryption is thus defined as:

$$\text{encrypt}(\mu) := D \cdot \left( \begin{pmatrix} | & | & & | \\ \boldsymbol{e}_1 & \boldsymbol{e}_2 & \dots & \boldsymbol{e}_n \\ | & | & & | \end{pmatrix} + \mu \cdot G \right) \in (\mathcal{D}/q\mathcal{L}^*)^{m \log(q)}.$$

Decryption follows from GSW, it is possible to extract the $j$-th column of $C$, with $j = \log(q)$, and use the secret key $\boldsymbol{s}$ as follows:

$$\begin{aligned}\text{decrypt}(C, \boldsymbol{s}) &:= \lfloor [\langle \boldsymbol{c}_j, \boldsymbol{s}\rangle]_q \rceil_p \equiv \lfloor [\langle D\boldsymbol{e}_j, \boldsymbol{s}\rangle + \langle D((q/p)\mu, 0, \ldots, 0)^t, \boldsymbol{s}\rangle]_q \rceil_p \\ &\equiv \lfloor [e + \langle \boldsymbol{d}_1, \boldsymbol{s}\rangle (q/p)\mu]_q \rceil_p \equiv \mu \in \{0, \ldots, p-1\}\end{aligned}$$

where, as in Lattice-BGV, we have that $D\boldsymbol{e}_j \sim \mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$. and we assume that $\langle \boldsymbol{d}_1, \boldsymbol{s}\rangle \equiv 1 \bmod p$. Notice that the decryption noise $e = \langle D\boldsymbol{e}_j, \boldsymbol{s}\rangle$ must be roughly smaller than $q/2p$ to allow for correct decryption.

**Homomorphic multiplications** The multiplication between two ciphertexts almost immediately follows the original GSW one. We re-define the decomposition function $G^{-1}$ accordingly with base $p$. Given two ciphertexts $C_1, C_2 \in (\mathcal{D}/q\mathcal{L}^*)^{m \log_p(q)}$, – encrypting respectively $\mu_1$ and $\mu_2$ – we define the multiplication as $C^\times := C_1 \cdot G^{-1}(D^{-1}C_2)$. Note the transformation $D^{-t}C_2$ to the coefficient representation which is required to apply $G^{-1}$. We now show that the decryption of multiplications is correct, as long as the noise is sufficiently small. For the short secret $\boldsymbol{s} \in \mathcal{L}$ we have

$$\begin{aligned}\boldsymbol{s}^t \cdot C^\times &\equiv \boldsymbol{s}^t (C_1 \cdot G^{-1}(D^{-1}C_2)) \\ &\equiv (\boldsymbol{e}^t + \mu_1 \cdot \boldsymbol{s}^t DG) \cdot G^{-1}(D^{-1}C_2) \\ &\equiv \boldsymbol{e}^t \cdot G^{-1}(D^{-1}C_2) + \mu_1 \cdot \boldsymbol{s}^t DG \cdot G^{-1}(D^{-1}C_2) \\ &\equiv \boldsymbol{e}^t \cdot G^{-1}(D^{-1}C_2) + \mu_1 \cdot ((\boldsymbol{e}')^t + \mu_2 \cdot \boldsymbol{s}^t DG) \\ &\equiv \boldsymbol{e}^t \cdot G^{-1}(D^{-1}C_2) + \mu_1 \boldsymbol{e}_2^t + \mu_1\mu_2 \cdot \boldsymbol{s}^t DG \in \mathbb{Z}_q^n,\end{aligned} \tag{4.5}$$

where $\boldsymbol{e}, \boldsymbol{e}'$ are the vectors containing all the small decoding errors $[\langle \boldsymbol{e}_i, \boldsymbol{s}\rangle]_q$, for each column $i$ of $C_1$ and $C_2$, respectively.

If the decryption noise $(\boldsymbol{e}^t \cdot G^{-1}(C_2) + \mu_1(\boldsymbol{e}')^t)_j$, where $j = \log(q)$, is not larger than $q/2p$, then the decryption can be performed by considering the $j$-th element of the vector $(C^\times)^t \boldsymbol{s}$ as in the decryption procedure.

## 5 A possible instantiation using LIP

When instantiating our Lattice-BGV and Lattice-GSW scheme with an LWE-based key generation we precisely obtain the original BGV and GSW schemes respectively. We now show, as an example to illustrate the flexibility of our framework, a possible instantiation based on the Lattice Isomorphism Problem.

**Key generation** Recall that we only need to instantiate the key generation step and that the output should simply be a public bad description of a primal lattice $\mathcal{L}$ and a sufficiently short secret vector $\boldsymbol{s} \in \mathcal{L}$. A LIP instance is defined as:

$$B' = O \cdot B \cdot U \in \text{GL}_m(\mathbb{R})$$

where $B, B'$ are public material, and refer to the basis of the chosen lattice and a public rotation of it, respectively, and $O \in \mathcal{O}_m(\mathbb{R})$ or $U \in \mathrm{GL}_m(\mathbb{Z})$ are the secret transformations. The problem asks to recover either $O$ or equivalently $U$ (up to an automorphism). Notice that it is possible to use any good basis $B$ – generating a lattice with nice decoding properties – even $B = I_m$ to generate a rotation of $\mathbb{Z}^m$, which is what is essentially used by the HAWK scheme [DPPW22]. For now we will simply assume that the first vector $\boldsymbol{b}_1$ in $B$ is a sufficiently short vector, but note that this also easily allows to have multiple short vectors. This fits our setting well, since we naturally have all the required ingredients:

- $B' = O \cdot B \cdot U \in \mathrm{GL}_m(\mathbb{R})$: bad description of the lattice $\mathcal{L} = \mathcal{L}(B')$.
- $\boldsymbol{s} := O \cdot \boldsymbol{b}_1 \in \mathcal{O}_m(\mathbb{R})$: a short vector $\boldsymbol{s} \in O \cdot \mathcal{L}(B) = \mathcal{L}(B')$.

Notice that one can securely sample the basis $B'$ of a certain quality following [DvW22, Algorithm 1], or by applying the procedure to the dual to create a basis $C$ of the dual lattice $\mathcal{L}^* := \mathcal{L}(B')^*$ of a certain quality. The next step is to generate a basis $D \in \mathbb{R}^{m \times m}$ for the ambient space according to Algorithm 3.1.

**Fixing a concrete set of parameters** Ignoring some further technical details we assume that the lattice $\mathcal{L}$ is normalized to have volume 1 and that we can sample a dual basis $C$ of $\mathcal{L}^*$ such that $\|C\| \sim \sigma_{\mathrm{pk}}$ for some sufficiently large parameter $\sigma_{\mathrm{pk}} > 0$. Following Lemma 3.1 there is a strong dependency between the $\sigma_{\mathrm{pk}} > 0$ value used to sample $C$, and the allowed modulus $q$ of the scheme. Intuitively, larger values of $\sigma_{\mathrm{pk}}$ correspond to a worse basis, thus in larger spread $\ell_{\max}/\ell_{\min}$ in the Gram-Schmidt norms of $C$. This can be visualized in a profile that spreads along a larger range. By analyzing the profile of $C$ as a straight line, we roughly expect that $\ell_{\max} \sim \sigma_{\mathrm{pk}}$ and $\ell_{\min} \sim \sigma_{\mathrm{pk}}^{-1}$, and thus $(\ell_{\max}/\ell_{\min}) \sim \sigma_{\mathrm{pk}}^2$ for their ratio.

By setting $q = p^k \leq \ell_{\max}/\ell_{\min}$ with $k = \lfloor \log_p(\ell_{\max}/\ell_{\min}) \rfloor$ we can have a modulus $q$ up to roughly $\sigma_{\mathrm{pk}}^2$ as well while creating a good basis $D$ for the ambient space $\mathcal{D}$ where decoding BDD instances is non trivial. The value of $\sigma_{\mathrm{enc}} > 0$ used to sample encryptions, on the other hand, depends on the quality of the basis $D$. In particular, because of Lemma 2.1, and for security reasons, we require $\sigma_{\mathrm{enc}}$ to be slightly larger than $\max_i \|\tilde{\boldsymbol{d}}_i\| = \max_i \|\tilde{\boldsymbol{c}}_i\| \sim \sigma_{\mathrm{pk}}$. Note that while $\sigma_{\mathrm{enc}}$ used to sample an encryption error $\boldsymbol{e}$ might be large, what matters is the decryption noise coming from $\langle \boldsymbol{e}, \boldsymbol{s} \rangle$ which is roughly of size $\|\boldsymbol{s}\| \cdot \sigma_{\mathrm{enc}} \sim \|\boldsymbol{s}\| \cdot \sigma_{\mathrm{pk}}$, in contrast to the modulus $q \sim \sigma_{\mathrm{pk}}^2$. In particular, after a BGV-multiplication one would obtain a decryption noise of roughly $(\|\boldsymbol{s}\| \cdot \sigma_{\mathrm{pk}})^2$ which needs to be smaller than $q$. To allow for many multiplications we would therefore preferably have an exceptionally short vector $\boldsymbol{s} \in \mathcal{L}$, or equivalently, start with a lattice such that $\boldsymbol{b}_1 \in \mathcal{L}$ is roughly of norm $\sigma_{\mathrm{pk}}^{-1}$. Note that this matches the LWE case where after normalizing the volume $\|\boldsymbol{s}\| \sim 1/\sqrt{q} \approx \sigma_{\mathrm{pk}}^{-1}$. One way to obtain this is similar to the instantiation proposed in [DvW22], picking $\mathcal{L} := \frac{1}{\lambda}\mathcal{L}' \oplus \lambda\mathcal{L}'$ for one's favorite lattice $\mathcal{L}'$, where $\lambda$ allows to arbitrarily decrease its first minimum.

## 6 Further extensions and open questions

We presented a lattice-based perspective of two well-known homomorphic encryption schemes: BGV and GSW. This opens up a lot of possibilities in terms of constructions and hardness assumptions. We therefore consider some further extensions and open questions.

*Multiple message packing.* We encoded a single message in each ciphertext and used a single short vector $\boldsymbol{s} \in \mathcal{L}$ to decrypt it. The generalization to multiple messages is rather straightforward by going from one short secret vector to $k > 1$ short vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k \in \mathcal{L}$ to pack $k$ messages in a ciphertext at once. One does have to assume some additional modular constraints between the ambient basis $D$ and these vectors, but these can be accounted for in the key generation.

*Structured versions.* We presented the schemes for plain lattices to focus on the geometric intuition and ideas. This makes the schemes rather impractical in their current form. One could significantly improve this by using structured *module lattices*, and this generalization follows quite immediately by plugging in such a lattice. This also directly explains why RLWE-based schemes allow to pack multiple messages, as for example in a module lattice over a cyclotomic number field one could multiply the short vector $\boldsymbol{s}$ by a root of unity to obtain another short vector, leading to $d$ short vectors where $d$ is the degree of the number field.

*Better decoding.* We tried to present our lattice framework for HE in the simplest way possible where we only assume a secret vector $\boldsymbol{s} \in \mathcal{L}$ to partially decode the ciphertexts with respect of the dual lattice. Moreover we use the simple Babai nearest plane algorithm and a good basis for switching between different ambient spaces. There are however more advanced decoding techniques and one could construct a similar scheme where the dual lattice $\mathcal{L}^*$ and/or the ambient space $\mathcal{D}$ have a special structure that allows to decode more efficiently (with and without secret information respectively). This could further reduce the decoding noise, or allow for more efficient encodings of the messages.

*Key switching and bootstrapping.* We did not discuss the bootstrapping procedure, although we conjecture that standard techniques can be easily be applied to our schemes, since the decryption circuit is the same as in LWE. The bootstrapping and key switching method are mostly making use of the homomorphic properties and are not really specific to lattices. This opens the question: can key switching and/or bootstrapping be performed differently, considering specific families of lattices or specific lattice operations? For example, the main goal of key switching is to move from a large dimensional lattice to a lower dimensional one. For lattices such an operation is naturally associated with projections, and in general, BDD instances can still be BDD instances after an appropriate projection. Could we therefore replace or rephrase key switching as some lattice projection?

## References


Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, 1996.

AP14. Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology – CRYPTO 2014*, pages 297–314, Berlin, Heidelberg, 2014.

BBD+23. Joppe W Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W Postlethwaite, Thomas Prest, Ludo N Pulles, and Wessel van Woerden. Hawk. Technical report, National Institute of Standards and Technology, 2023.

BGPSD23. Huck Bennett, Atul Ganju, Pura Peetathawatchai, and Noah Stephens-Davidowitz. Just how hard are rotations of $\mathbb{Z}^n$? Algorithms and cryptography with the simplest lattice. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 252–281. Springer, 2023.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, 2012.

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of Learning with Errors. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 575–584, 2013.

BMM25. Pedro Branco, Giulio Malavolta, and Zayd Maradni. Fully-homomorphic encryption from lattice isomorphism. Cryptology ePrint Archive, Paper 2025/993, 2025.

Bra12. Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *Advances in Cryptology – CRYPTO 2012*, pages 868–886, 2012.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, 2011.

CGGI16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In *Advances in Cryptology – ASIACRYPT 2016*, pages 3–33, 2016.

CGGI20. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 1 2020.

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, 2017.

DM15. Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology – EUROCRYPT 2015*, pages 617–640, 2015.

DPPW22. Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel van Woerden. Hawk: Module LIP Makes Lattice Signatures Fast, Compact and Simple. In *Advances in Cryptology – ASIACRYPT 2022*, pages 65–94, 2022.

DvW22. Léo Ducas and Wessel van Woerden. On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography. In *Advances in Cryptology – EUROCRYPT 2022*, pages 643–673, 2022.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, Paper 2012/144, 2012.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, 2009.

GHS12. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic Evaluation of the AES Circuit. In *Advances in Cryptology – CRYPTO 2012*, pages 850–867, 2012.

GINX16. Nicolas Gama, Malika Izabachene, Phong Q Nguyen, and Xiang Xie. Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 528–558. Springer, 2016.

GMP19. Nicholas Genise, Daniele Micciancio, and Yuriy Polyakov. Building an Efficient Lattice Gadget Toolkit: Subgaussian Sampling and More. In *Advances in Cryptology – EUROCRYPT 2019*, pages 655–684, 2019.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology – CRYPTO 2013*, pages 75–92, 2013.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology – EUROCRYPT 2010*, pages 1–23, 2010.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, 2012.

RAD78. Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, 2005.

Rot11. Ron Rothblum. Homomorphic Encryption: From Private-Key to Public-Key. In *Theory of Cryptography – TTC 2011*, pages 219–234, 2011.