Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP

Zvika Brakerski*

Stanford University
zvika@stanford.edu

Abstract. We present a new tensoring technique for LWE-based fully homomorphic encryption. While in all previous works, the ciphertext noise grows quadratically ($B \to B^2 \cdot \mathrm{poly}(n)$) with every multiplication (before "refreshing"), our noise only grows linearly ($B \to B \cdot \mathrm{poly}(n)$).
We use this technique to construct a *scale-invariant* fully homomorphic encryption scheme, whose properties only depend on the ratio between the modulus q and the initial noise level B, and not on their absolute values.
Our scheme has a number of advantages over previous candidates: It uses the same modulus throughout the evaluation process (no need for "modulus switching"), and this modulus can take arbitrary form. In addition, security can be *classically* reduced from the worst-case hardness of the GapSVP problem (with quasi-polynomial approximation factor), whereas previous constructions could only exhibit a quantum reduction from GapSVP.

1 Introduction

Fully homomorphic encryption has been the focus of extensive study since the first candidate scheme was introduced by Gentry [9]. In a nutshell, fully homomorphic encryption allows to perform arbitrary computation on encrypted data. It can thus be used, for example, to outsource a computation to a remote server without compromising data privacy.

The first generation of fully homomorphic schemes [4,6,7,9,10,26] that started with Gentry's seminal work, all followed a similar and fairly complicated methodology, often relying on relatively strong computational assumptions. A second generation of schemes started with the work of Brakerski and Vaikuntanathan [5], who established full homomorphism in a simpler way, based on the *learning with errors* (LWE) assumption. Using known reductions [20, 22], the security of their construction is based on the (often quantum) hardness of approximating some short vector problems in worst-case lattices. Their scheme was then improved by Brakerski, Gentry and Vaikuntanathan [3], as we describe below.

In LWE-based schemes such as [3, 5], ciphertexts are represented as vectors in $\mathbb{Z}_q$, for some modulus q. The decryption process is essentially computing an

* Supported by a Simons Postdoctoral Fellowship and by DARPA.

inner product of the ciphertext and the secret key vector, which produces a noisy version of the message (the noise is added at encryption for security purposes). The noise increases with every homomorphic operation, and correct decryption is guaranteed if the final noise magnitude is below $q/4$. Homomorphic addition roughly doubles the noise, while homomorphic multiplication roughly squares it.

In the [5] scheme, after L levels of multiplication (e.g. evaluating a depth L multiplication tree), the noise grows from an initial magnitude of B, to B^{2^L}. Hence, to enable decryption, a very large modulus $q \approx B^{2^L}$ was required. This affected both efficiency and security (the security of the scheme depends inversely on the ratio q/B, so bigger q for the same B means less security).

The above was improved by [3], who suggested to *scale down* the ciphertext vector after every multiplication (they call this "modulus switching", see below).[1] That is, to go from a vector $\mathbf{c}$ over $\mathbb{Z}_q$, into the vector $\mathbf{c}/w$ over $\mathbb{Z}_{q/w}$ (for some scaling factor w). Scaling "switches" the modulus q to a smaller q/w, but also reduces the noise by the same factor (from B to B/w). To see why this change of scale is effective, consider scaling by a factor B after every multiplication (as indeed suggested by [3]): After the first multiplication, the noise goes up to B^2, but scaling brings it back down to B, at the cost of reducing the modulus to q/B. With more multiplications, the noise magnitude always goes back to B, but the modulus keeps reducing. After L levels of multiplication-and-scaling, the noise magnitude is still B, but the modulus is down to q/B^L. Therefore it is sufficient to use $q \approx B^{L+1}$, which is significantly lower than before. However, this process results in a complicated homomorphic evaluation process that "climbs down the ladder of moduli".

The success of the scaling methodology teaches us that *perspective* matters: scaling does not change the ratio between the modulus and noise, but it still manages the noise better by changing the perspective in which we view the ciphertext. In this work, we suggest to work in an *invariant perspective* where only the ratio q/B matters (and not the absolute values of q, B as in previous works). We derive a scheme that is superior to the previous best known in simplicity, noise management and security. Details follow.

1.1 Our Results

As explained above, we present a *scale invariant* scheme, by finding an invariant perspective. The idea is very natural based on the outlined motivation: if we scale down the ciphertext by a factor of q, we get a fractional ciphertext modulo 1, with noise magnitude B/q. In this perspective, all choices of q, B with the same B/q ratio will look the same. It turns out that in this perspective, homomorphic multiplication does not square the noise, but rather multiplies it by a polynomial factor $p(n)$ that depends only on the security parameter.[2] After L levels of

[1] A different scaling technique was already suggested in [5] as a way to simplify decryption and improve efficiency, but not to manage noise.

[2] More accurately, a polynomial $p(n, \log q)$, but w.l.o.g $q \leq 2^n$.

multiplication, the noise will grow from B/q to $(B/q) \cdot p(n)^L$, which means that we only need to use $q \approx B \cdot p(n)^L$.

Interestingly, the idea of working modulo 1 goes back to the early works of Ajtai and Dwork [1], and Regev [21], and to the first formulation of LWE [22]. In a sense, we are "going back to the roots" and showing that these early ideas are instrumental in the construction of homomorphic encryption.

For technical reasons, we don't implement the scheme over fractions, but rather mimic the invariant perspective over $\mathbb{Z}_q$ (see Section 1.2 for more details). Perhaps surprisingly, the resulting scheme is exactly Regev's original LWE-based scheme, with additional auxiliary information for the purpose of homomorphic evaluation. The properties of our scheme are summarized in the following theorem:

Theorem. *There exists a homomorphic encryption scheme for depth L circuits, based on the $\mathsf{DLWE}_{n,q,\chi}$ assumption (n-dimensional decision-LWE modulo q, with noise χ), so long as*

$$q/B \geq (O(n \log q))^{L+O(1)} \ ,$$

where B is a bound on the values of χ.

The resulting scheme has a number of interesting properties:

1. **Scale invariance.** Homomorphic properties only depend on q/B (as explained above).
2. **No modulus switching.** We work with a single modulus q. We don't need to switch moduli as in [3,5]. This leads to a simpler description of the scheme (and hopefully better implementations).
3. **No restrictions on the modulus.** Our modulus q can take any form (so long as it satisfies the size requirement). This is achieved by putting the message bit in the most significant bit of the ciphertext, rather than least significant as in previous homomorphic schemes (this can be interpreted as making the *message* scale invariant). We note that for odd q, the least and most significant bit representations are interchangeable.
   In particular, in our scheme q can be a power of 2, which can simplify implementation of arithmetics.[3] In previous schemes, such q could not be used for binary message spaces.[4]
   This, again, is going back to the roots: Early schemes such as [22], and in a sense also [1, 14], encoded ciphertexts in the most significant bits. Switching to least significant bit encoding was (perhaps ironically) motivated by improving homomorphism.
4. **No restrictions on the secret key distribution.** While [3] requires that the secret key is drawn from the noise distribution (LWE in Hermite normal form), our scheme works under any secret key distribution for which the LWE assumption holds.

[3] On the downside, such q might reduce efficiency when using ring-LWE (see below) due to FFT embedding issues.

[4] [11] gain on efficiency by using moduli that are "almost" a power of 2.

5. **Classical Reduction from GapSVP.** One of the appeals of LWE-based cryptography is the known quantum (Regev [22]) and classical (Peikert [20]) reductions from the worst case hardness of lattice problems. Specifically to GapSVP_γ, which is the problem of deciding, given an n dimensional lattice and a number d, between the following two cases: either the lattice has a vector shorter than d, or it doesn't have any vector shorter than $\gamma(n) \cdot d$. The value of γ depends on the ratio q/B (essentially $\gamma = (q/B) \cdot \tilde{O}(n)$), and the smaller γ is, the better the reduction ($\mathsf{GapSVP}_{2^{\Omega(n)}}$ is an easy problem). Peikert's classical reduction requires that $q \approx 2^{n/2}$, which makes his reduction unusable for previous homomorphic schemes, since γ becomes exponential. For example, in [3], $q/B = q/q^{1/(L+1)} = q^{1-1/(L+1)}$ which translates to $\gamma \approx 2^{n/2}$ for the required q.[5]
   In our scheme, this problem does not arise. We can instantiate our scheme with any q while hardly affecting the ratio q/B. We can therefore set $q \approx 2^{n/2}$ and get a classical reduction from $\mathsf{GapSVP}_{n^{O(\log n)}}$, which is currently solvable only in $2^{\tilde{\Omega}(n)}$ time. (This is mostly of theoretical interest, though, since efficiency considerations will favor the smallest possible q.)

Using our scheme as a building block we achieve:

1. **Fully homomorphic encryption using bootstrapping.** Using Gentry's bootstrapping theorem, we present a leveled fully homomorphic scheme based on the classical worst case $\mathsf{GapSVP}_{n^{O(\log n)}}$ problem. As usual, an additional circular security assumption is required to get a non-leveled scheme.
2. **Leveled fully homomorphic encryption without bootstrapping.** As in [3], our scheme can be used to achieve leveled homomorphism without bootstrapping.
3. **Increased efficiency using ring-LWE (RLWE).** RLWE ([15]) is a version of LWE that works over polynomial rings rather than the integers. Its hardness is quantumly related to short vector problems in *ideal* lattices.
   RLWE is a stronger assumption than LWE, but it can dramatically improve the efficiency of schemes [3, 4, 12]. Our methods are readily portable to the RLWE world.

In summary, our construction carries conceptual significance in its simplicity and in a number of theoretical aspects. Its practical usefulness compared to other schemes is harder to quantify, though, since it will vary greatly with the specific implementation and optimizations chosen.

1.2 Our Techniques

Our starting point is Regev's public key encryption scheme. There, the encryption of a message $m \in \{0,1\}$ is an integer vector $\mathbf{c}$ such that $\langle \mathbf{c}, \mathbf{s} \rangle = \lfloor \frac{q}{2} \rfloor \cdot m + e + qI$, for an integer I and for $|e| \leq E$, for some bound $E < q/4$.

[5] Peikert suggests to classically base small-q LWE on a new lattice problem that he introduces.

The secret key vector $\mathbf{s}$ is also over the integers. We can assume w.l.o.g that the elements of $\mathbf{c}, \mathbf{s}$ are in the segment $(-q/2, q/2]$. (We note that previous homomorphic constructions used a different variant of Regev's scheme, where $\langle \mathbf{c}, \mathbf{s} \rangle = m + 2e + qI$.)

In this work, we take the invariant perspective on the scheme, and consider the fractional ciphertext $\tilde{\mathbf{c}} = \mathbf{c}/q$. It holds that $\langle \tilde{\mathbf{c}}, \mathbf{s} \rangle = \frac{1}{2} \cdot m + \tilde{e} + I$, where $I \in \mathbb{Z}$ and $|\tilde{e}| \leq E/q = \epsilon$. The elements of $\mathbf{c}$ are now rational numbers in $(-1/2, 1/2]$. Note that the secret key does not change and is still over $\mathbb{Z}$.

Additive homomorphism is immediate: if $\mathbf{c}_1$ encrypts m_1 and $\mathbf{c}_2$ encrypts m_2, then $\mathbf{c}_{\mathsf{add}} = \mathbf{c}_1 + \mathbf{c}_2$ encrypts $[m_1 + m_2]_2$. The noise grows from ϵ to $\approx 2\epsilon$. Multiplicative homomorphism is achieved by *tensoring* the input ciphertexts:

$$\mathbf{c}_{\mathsf{mult}} = 2 \cdot \mathbf{c}_1 \otimes \mathbf{c}_2 \ .$$

The tensored ciphertext can be decrypted using a tensored secret key because

$$\big\langle \underbrace{2 \cdot \mathbf{c}_1 \otimes \mathbf{c}_2}_{\mathbf{c}_{\mathsf{mult}}}, \mathbf{s} \otimes \mathbf{s} \big\rangle = 2 \cdot \langle \mathbf{c}_1, \mathbf{s} \rangle \cdot \langle \mathbf{c}_2, \mathbf{s} \rangle \ .$$

A "key switching" mechanism developed in [5] and generalized in [3] allows to switch back from a tensored secret key into a "normal" one without much additional noise. The details of this mechanism are immaterial for this discussion. We focus on the noise growth in the tensored ciphertext.

We want to show that $2 \cdot \langle \mathbf{c}_1, \mathbf{s} \rangle \cdot \langle \mathbf{c}_2, \mathbf{s} \rangle \approx \frac{1}{2} m_1 m_2 + e' + I'$, for a small e'. To do this, we let $I_1, I_2 \in \mathbb{Z}$ be integers such that $\langle \mathbf{c}_1, \mathbf{s} \rangle = \frac{1}{2} m_1 + e_1 + I_1$, and likewise for $\mathbf{c}_2$. It can be verified that $|I_1|, |I_2|$ are bounded by $\approx \|\mathbf{s}\|_1$. We therefore get:

$$\begin{aligned} 2 \cdot \langle \mathbf{c}_1, \mathbf{s} \rangle \cdot \langle \mathbf{c}_2, \mathbf{s} \rangle &= 2 \cdot (\tfrac{1}{2} m_1 + e_1 + I_1) \cdot (\tfrac{1}{2} m_2 + e_2 + I_2) \\ &= \tfrac{1}{2} m_1 m_2 + 2(e_1 I_2 + e_2 I_1) + e_1 m_2 + e_2 m_1 + 2 e_1 e_2 \\ &\qquad + \underbrace{(m_1 I_2 + m_2 I_1 + 2 I_1 I_2)}_{\in \mathbb{Z}} \ . \end{aligned}$$

Interestingly, the cross-term $e_1 e_2$ that was responsible for the squaring of the noise in previous schemes, is now practically insignificant since $\epsilon^2 \ll \epsilon$. The significant noise term in the above expression is $2(e_1 I_2 + e_2 I_1)$, which is bounded by $O(\|\mathbf{s}\|_1) \cdot \epsilon$. All that is left to show now is that $\|\mathbf{s}\|_1$ is independent of B, q and only depends on n (recall that we allow dependence on $\log q \leq n$).

On the face of it, $\|\mathbf{s}\|_1 \approx n \cdot q$, since the elements of $\mathbf{s}$ are integers in the segment $(-q/2, q/2]$. In order to reduce the norm, we use *binary decomposition* (which was used in [3,5] for different purposes). Let $\mathbf{s}^{(j)}$ denote the binary vector that contains the j^{th} bit from each element of $\mathbf{s}$. Namely $\mathbf{s} = \sum_j 2^j \mathbf{s}^{(j)}$. Then

$$\langle \mathbf{c}, \mathbf{s} \rangle = \sum_j 2^j \langle \mathbf{c}, \mathbf{s}^{(j)} \rangle = \big\langle (\mathbf{c}, 2\mathbf{c}, \ldots), (\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \ldots) \big\rangle \ .$$

This means that we can convert a ciphertext $\mathbf{c}$ that corresponds to a secret key $\mathbf{s}$ in $\mathbb{Z}$, into a modified ciphertext $(\mathbf{c}, 2\mathbf{c}, \ldots)$ that corresponds to a *binary*

secret key $(\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \ldots)$. The norm of the binary key is at most its dimension, which is polynomial in n as required.[6]

We point out that an alternative solution to the norm problem follows by using the dual-Regev scheme of [13] as the basic building block. There, the secret key is natively binary and of low norm. (In addition, as noticed in previous works, working with dual-Regev naturally implies a weak form of homomorphic identity based encryption.) However, the ciphertexts and some other parameters will need to grow.

Finally, working with fractional ciphertexts brings about issues of precision in representation and other problems. We thus implement our scheme over $\mathbb{Z}$ with appropriate scaling: Each rational number x in the above description will be represented by the integer $y = \lfloor qx \rceil$ (which determines x up to an additive factor of $1/2q$). The addition operation $x_1 + x_2$ is mimicked by $y_1 + y_2 \approx \lfloor q(x_1 + x_2) \rceil$. To mimic multiplication, we take $\lfloor (y_1 \cdot y_2)/q \rceil \approx \lfloor x_1 \cdot x_2 \cdot q \rceil$. Our tensored ciphertext for multiplication will thus be defined as $\left\lfloor \frac{2}{q} \cdot \mathbf{c}_1 \otimes \mathbf{c}_2 \right\rceil$, where $\mathbf{c}_1, \mathbf{c}_2$ are integer vectors and the tensoring operation is over the integers. In this representation, encryption and decryption become identical to Regev's original scheme.

1.3 Notation

For an integer q, we define the set $\mathbb{Z}_q \triangleq (-q/2, q/2] \cap \mathbb{Z}$. We stress that in this work, $\mathbb{Z}_q$ is *not synonymous* with the ring $\mathbb{Z}/q\mathbb{Z}$. In particular, all arithmetics is performed over $\mathbb{Z}$ (or $\mathbb{Q}$ when division is used) and not over any sub-ring. For any $x \in \mathbb{Q}$, we let $y = [x]_q$ denote the unique value $y \in (-q/2, q/2]$ such that $y = x \pmod q$ (i.e. y is congruent to x modulo q).[7]

A distribution χ over the integers is B-bounded, denoted $|\chi| \leq B$, if χ is only supported on $[-B, B]$.

We denote vectors in bold lowercase ($\mathbf{x}$) and matrices in bold uppercase ($\mathbf{A}$). Slightly abusing notation, we denote the concatenation of vectors $\mathbf{x}, \mathbf{y}$ by $(\mathbf{x}, \mathbf{y})$. The tensor product of two vectors $\mathbf{v}, \mathbf{w}$ of dimension n, denoted $\mathbf{v} \otimes \mathbf{w}$, is the n^2 dimensional vector containing all elements of the form $\mathbf{v}[i]\mathbf{w}[j]$. Note that $\langle \mathbf{v} \otimes \mathbf{w}, \mathbf{x} \otimes \mathbf{y} \rangle = \langle \mathbf{v}, \mathbf{x} \rangle \cdot \langle \mathbf{w}, \mathbf{y} \rangle$.

1.4 Paper Organization

Section 2 introduces the LWE assumption and defines homomorphic encryption and related terms. Section 3 introduces our building blocks: Regev's encryption scheme, binary decomposition of vectors and the key switching mechanism. Finally, in Section 4 we present and analyze our scheme, and discuss several possible optimizations.

[6] Reducing the norm of $\mathbf{s}$ was also an issue in [3]. There it was resolved by using LWE in Hermite normal form, where $\mathbf{s}$ is sampled from the noise distribution and thus $\|\mathbf{s}\|_1 \approx n \cdot B$. This suffices when B must be very small, as in [3], but not in our setting.

[7] For example, if $x = 2, y = -3 \in \mathbb{Z}_7$, then $x \cdot y = -6 \notin \mathbb{Z}_7$, however $[x \cdot y]_7 = 1 \in \mathbb{Z}_7$.

2 Preliminaries

2.1 Learning With Errors (LWE)

The LWE problem was introduced by Regev [22] as a generalization of "learning parity with noise". For positive integers n and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on $\mathbb{Z}$, let $A_{\mathbf{s},\chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly at random and a noise term $e \xleftarrow{\$} \chi$, and outputting $(\mathbf{a}, [\langle \mathbf{a}, \mathbf{s} \rangle + e]_q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Decisional LWE (DLWE) is defined as follows.

Definition 2.1 (DLWE). *For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}$, the (average-case) decision learning with errors problem, denoted $\mathsf{DLWE}_{n,m,q,\chi}$, is to distinguish (with non-negligible advantage) m samples chosen according to $A_{\mathbf{s},\chi}$ (for uniformly random $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We denote by $\mathsf{DLWE}_{n,q,\chi}$ the variant where the adversary gets oracle access to $A_{\mathbf{s},\chi}$, and is not a-priori bounded in the number of samples.*

There are known quantum (Regev [22]) and classical (Peikert [20]) reductions between $\mathsf{DLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be (discretized versions of) the Gaussian distribution, which is statistically indistinguishable from B-bounded, for an appropriate B. Since the exact distribution χ does not matter for our results, we state a corollary of the results of [20, 22] (in conjunction with the search to decision reduction of Micciancio and Mol [16] and Micciancio and Peikert [17]) in terms of the bound B. These results also extend to additional forms of q (see [16, 17]).

Corollary 2.2 ([16, 17, 20, 22]). *Let $q = q(n) \in \mathbb{N}$ be either a prime power $q = p^r$, or a product of co-prime numbers $q = \prod q_i$ such that for all i, $q_i = \mathrm{poly}(n)$, and let $B \geq \omega(\log n) \cdot \sqrt{n}$. Then there exists an efficiently sampleable B-bounded distribution χ such that if there is an efficient algorithm that solves the (average-case) $\mathsf{DLWE}_{n,q,\chi}$ problem. Then:*

- *There is an efficient quantum algorithm that solves $\mathsf{GapSVP}_{\widetilde{O}(n \cdot q/B)}$ (and $\mathsf{SIVP}_{\widetilde{O}(n \cdot q/B)}$) on any n-dimensional lattice.*
- *If in addition $q \geq \tilde{O}(2^{n/2})$, then there is an efficient classical algorithm for $\mathsf{GapSVP}_{\tilde{O}(n \cdot q/B)}$ on any n-dimensional lattice.*

In both cases, if one also considers distinguishers with sub-polynomial advantage, then we require $B \geq \widetilde{O}(n)$ and the resulting approximation factor is slightly larger $\widetilde{O}(n\sqrt{n} \cdot q/B)$.

Recall that GapSVP_γ is the (promise) problem of distinguishing, given a basis for a lattice and a parameter d, between the case where the lattice has a vector shorter than d, and the case where the lattice doesn't have any vector shorter than $\gamma \cdot d$. SIVP is the search problem of finding a set of "short" vectors. We refer the reader to [20, 22] for more information.

The best known algorithms for GapSVP_γ ([18,25]) require at least $2^{\tilde{\Omega}(n/\log\gamma)}$ time. The scheme we present in this work reduces from $\gamma = n^{O(\log n)}$, for which the best known algorithms run in time $2^{\tilde{\Omega}(n)}$.

As a final remark, we mention that Peikert also shows a classical reduction in the case of small values of q, but this reduction is from a newly defined "ζ-to-γ decisional shortest vector problem", which is not as extensively studied as GapSVP.

2.2 Homomorphic Encryption and Bootstrapping

We now define homomorphic encryption and introduce Gentry's bootstrapping theorem. Our definitions are mostly taken from [3, 5].

A homomorphic (public-key) encryption scheme $\mathsf{HE} = (\mathsf{HE.Keygen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ is a quadruple of PPT algorithms as follows (n is the security parameter):

- **Key generation** $(pk, evk, sk) \leftarrow \mathsf{HE.Keygen}(1^n)$**:** Outputs a public encryption key pk, a public evaluation key evk and a secret decryption key sk.[8]
- **Encryption** $c \leftarrow \mathsf{HE.Enc}_{pk}(m)$**:** Using the public key pk, encrypts a single bit message $m \in \{0,1\}$ into a ciphertext c.
- **Decryption** $m \leftarrow \mathsf{HE.Dec}_{sk}(c)$**:** Using the secret key sk, decrypts a ciphertext c to recover the message $m \in \{0,1\}$.
- **Homomorphic evaluation** $c_f \leftarrow \mathsf{HE.Eval}_{evk}(f, c_1, \ldots, c_\ell)$**:** Using the evaluation key evk, applies a function $f : \{0,1\}^\ell \to \{0,1\}$ to $c_1, \ldots, c_\ell$, and outputs a ciphertext c_f.
  As in previous works, we represent f as an arithmetic circuit over GF(2) with addition and multiplication gates. Thus it is customary to "break" $\mathsf{HE.Eval}$ into homomorphic addition $c_{\mathsf{add}} \leftarrow \mathsf{HE.Add}_{evk}(c_1, c_2)$ and homomorphic multiplication $c_{\mathsf{mult}} \leftarrow \mathsf{HE.Mult}_{evk}(c_1, c_2)$.

A homomorphic encryption scheme is said to be secure if it is semantically secure (note that the adversary is given both pk and evk).

Homomorphism w.r.t depth-bounded circuits and full homomorphism are defined next:

Definition 2.3 (L-homomorphism). *A scheme* HE *is L-homomorphic, for $L = L(n)$, if for any depth L arithmetic circuit f (over GF(2)) and any set of inputs $m_1, \ldots, m_\ell$, it holds that*

$$\Pr\left[\mathsf{HE.Dec}_{sk}(\mathsf{HE.Eval}_{evk}(f, c_1, \ldots, c_\ell)) \neq f(m_1, \ldots, m_\ell)\right] = \mathrm{negl}(n) \ ,$$

where $(pk, evk, sk) \leftarrow \mathsf{HE.Keygen}(1^n)$ and $c_i \leftarrow \mathsf{HE.Enc}_{pk}(m_i)$.

[8] We adopt the terminology of [5] that treats the evaluation key as a separate entity from the public key.

Definition 2.4 (compactness and full homomorphism). *A homomorphic scheme is compact if its decryption circuit is independent of the evaluated function. A compact scheme is fully homomorphic if it is L-homomorphic for any polynomial L. The scheme is leveled fully homomorphic if it takes 1^L as additional input in key generation.*

Gentry's bootstrapping theorem shows how to go from L-homomorphism to full homomorphism:

Theorem 2.5 (bootstrapping [8, 9]). *If there is an L-homomorphic scheme whose decryption circuit depth is less than L, then there exists a leveled fully homomorphic encryption scheme.*

Furthermore, if the aforementioned L-homomorphic scheme is also weak circular secure (remains secure even against an adversary who gets encryptions of the bits of the secret key), then there exists a fully homomorphic encryption scheme.

3 Building Blocks

In this section, we present building blocks from previous works that are used in our construction. Specifically, like all LWE-based fully homomorphic schemes, we rely on Regev's [22] basic public-key encryption scheme (Section 3.1). We also use the key-switching methodology of [3, 5] (Section 3.2).

3.1 Regev's Encryption Scheme

Let $q = q(n)$ be an integer function and let $\chi = \chi(n)$ be a distribution ensemble over $\mathbb{Z}$. The scheme Regev is defined as follows:

- $\mathsf{Regev.SecretKeygen}(1^n)$: Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. Output $sk = \mathbf{s}$.
- $\mathsf{Regev.PublicKeygen}(\mathbf{s})$: Let $N \triangleq (n+1) \cdot (\log q + O(1))$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{N \times n}$ and $\mathbf{e} \xleftarrow{\$} \chi^N$. Compute $\mathbf{b} := [\mathbf{A} \cdot \mathbf{s} + \mathbf{e}]_q$, and define
$$\mathbf{P} := [\mathbf{b} \| -\mathbf{A}] \in \mathbb{Z}_q^{N \times (n+1)} \ .$$
Output $pk = \mathbf{P}$.
- $\mathsf{Regev.Enc}_{pk}(m)$: To encrypt a message $m \in \{0,1\}$ using $pk = \mathbf{P}$, sample $\mathbf{r} \in \{0,1\}^N$ and output ciphertext
$$\mathbf{c} := \left[\mathbf{P}^T \cdot \mathbf{r} + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathbf{m}\right]_q \in \mathbb{Z}_q^{n+1} \ ,$$
where $\mathbf{m} \triangleq (m, 0, \ldots, 0) \in \{0,1\}^{n+1}$.
- $\mathsf{Regev.Dec}_{sk}(\mathbf{c})$: To decrypt $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ using secret key $sk = \mathbf{s}$, compute
$$m := \left[\left\lfloor 2 \cdot \frac{[\langle \mathbf{c}, (1, \mathbf{s}) \rangle]_q}{q} \right\rfloor\right]_2 \ .$$

Correctness. We analyze the noise magnitude at encryption and decryption. We start with a lemma regarding the noise magnitude of properly encrypted ciphertexts (the proof is omitted).

Lemma 3.1 (encryption noise). *Let $q, n, N, |\chi| \leq B$ be parameters for* Regev. *Let $\mathbf{s} \in \mathbb{Z}^n$ be any vector and $m \in \{0,1\}$ be a bit. Set $\mathbf{P} \leftarrow \mathsf{Regev.PublicKeygen}(\mathbf{s})$ and $\mathbf{c} \leftarrow \mathsf{Regev.Enc}_{\mathbf{P}}(m)$. Then for some e with $|e| \leq N \cdot B$ it holds that*

$$\langle \mathbf{c}, (1, \mathbf{s}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor \cdot m + e \pmod{q} \ .$$

We proceed to state the correctness of decryption for low-noise ciphertexts. The proof easily follows by assignment into the definition of Regev.Dec and is omitted.

Lemma 3.2 (decryption noise). *Let $\mathbf{s} \in \mathbb{Z}^n$ be some vector, and let $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ be such that*

$$\langle \mathbf{c}, (1, \mathbf{s}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor \cdot m + e \pmod{q} \ ,$$

with $m \in \{0,1\}$ and $|e| < \lfloor q/2 \rfloor / 2$. Then

$$\mathsf{Regev.Dec}_{\mathbf{s}}(\mathbf{c}) = m \ .$$

Security. The following lemma states the security of Regev. The proof is standard (see e.g. [22]) and is omitted.

Lemma 3.3. *Let n, q, χ be some parameters such that $\mathsf{DLWE}_{n,q,\chi}$ holds. Then for any $m \in \{0,1\}$, if $\mathbf{s} \leftarrow \mathsf{Regev.SecretKeygen}(1^n)$, $\mathbf{P} \leftarrow \mathsf{Regev.PublicKeygen}(\mathbf{s})$, $\mathbf{c} \leftarrow \mathsf{Regev.Enc}_{\mathbf{P}}(m)$, it holds that the joint distribution $(\mathbf{P}, \mathbf{c})$ is computationally indistinguishable from uniform over $\mathbb{Z}_q^{N \times (n+1)} \times \mathbb{Z}_q^{n+1}$.*

3.2 Vector Decomposition and Key Switching

We show how to decompose vectors in a way that preserves inner product and how to generate and use key switching parameters. Our notation is generally adopted from [3].

Vector Decomposition. We often break vectors into their bit representations as defined below:

- $\mathsf{BitDecomp}_q(\mathbf{x})$: For $\mathbf{x} \in \mathbb{Z}^n$, let $\mathbf{w}_i \in \{0,1\}^n$ be such that $\mathbf{x} = \sum_{i=0}^{\lceil \log q \rceil - 1} 2^i \cdot \mathbf{w}_i \pmod{q}$. Output the vector

$$(\mathbf{w}_0, \ldots, \mathbf{w}_{\lceil \log q \rceil - 1}) \in \{0,1\}^{n \cdot \lceil \log q \rceil} \ .$$

- $\mathsf{PowersOfTwo}_q(\mathbf{y})$: For $\mathbf{y} \in \mathbb{Z}^n$, output

$$\left[(\mathbf{y}, 2 \cdot \mathbf{y}, \ldots, 2^{\lceil \log q \rceil - 1} \cdot \mathbf{y}) \right]_q \in \mathbb{Z}_q^{n \cdot \lceil \log q \rceil} \ .$$

We will usually omit the subscript q when it is clear from the context.

Claim 3.4. *For all $q \in \mathbb{Z}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, it holds that*

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathsf{BitDecomp}_q(\mathbf{x}), \mathsf{PowersOfTwo}_q(\mathbf{y}) \rangle \pmod{q} \ .$$

Key Switching. In the functions below, q is an integer and χ is a distribution over $\mathbb{Z}$:

- $\mathsf{SwitchKeyGen}_{q,\chi}(\mathbf{s},\mathbf{t})$: For a "source" key $\mathbf{s}\in\mathbb{Z}^{n_s}$ and "target" key $\mathbf{t}\in\mathbb{Z}^{n_t}$, we define a set of parameters that allow to switch ciphertexts under $\mathbf{s}$ into ciphertexts under $(1,\mathbf{t})$.
  Let $\hat{n}_s \triangleq n_s\cdot\lceil\log q\rceil$ be the dimension of $\mathsf{PowersOfTwo}_q(\mathbf{s})$. Sample a uniform matrix $\mathbf{A}_{\mathbf{s}:\mathbf{t}} \xleftarrow{\$} \mathbb{Z}_q^{\hat{n}_s\times n_t}$ and a noise vector $\mathbf{e}\xleftarrow{\$}\chi^{\hat{n}_s}$. The function's output is a matrix
  $$\mathbf{P}_{\mathbf{s}:\mathbf{t}} = [\mathbf{b}_{\mathbf{s}:\mathbf{t}} \| -\mathbf{A}_{\mathbf{s}:\mathbf{t}}] \in \mathbb{Z}_q^{\hat{n}_s\times(n_t+1)} ,$$
  where
  $$\mathbf{b}_{\mathbf{s}:\mathbf{t}} := \left[\mathbf{A}_{\mathbf{s}:\mathbf{t}}\cdot\mathbf{t}+\mathbf{e}_{\mathbf{s}:\mathbf{t}}+\mathsf{PowersOfTwo}_q(\mathbf{s})\right]_q \in \mathbb{Z}_q^{\hat{n}_s} .$$
  This is similar, although not identical, to encrypting $\mathsf{PowersOfTwo}_q(\mathbf{s})$ (the difference is that $\mathsf{PowersOfTwo}_q(\mathbf{s})$ contains non-binary values).
- $\mathsf{SwitchKey}_q(\mathbf{P}_{\mathbf{s}:\mathbf{t}},\mathbf{c}_s)$: To switch a ciphertext from a secret key $\mathbf{s}$ to $(1,\mathbf{t})$, output
  $$\mathbf{c}_t := \left[\mathbf{P}_{\mathbf{s}:\mathbf{t}}^T\cdot\mathsf{BitDecomp}_q(\mathbf{c}_s)\right]_q .$$

Again, we usually omit the subscripts when they are clear from the context. Correctness and security are stated below, the proofs are by definition.

Lemma 3.5 (correctness). *Let $\mathbf{s}\in\mathbb{Z}^{n_s}$, $\mathbf{t}\in\mathbb{Z}^{n_t}$ and $\mathbf{c}_s\in\mathbb{Z}_q^{n_s}$ be any vectors. Let $\mathbf{P}_{\mathbf{s}:\mathbf{t}}\leftarrow\mathsf{SwitchKeyGen}(\mathbf{s},\mathbf{t})$ and set $\mathbf{c}_t\leftarrow\mathsf{SwitchKey}(\mathbf{P}_{\mathbf{s}:\mathbf{t}},\mathbf{c}_s)$. Then*
$$\langle\mathbf{c}_s,\mathbf{s}\rangle = \langle\mathbf{c}_t,(1,\mathbf{t})\rangle - \langle\mathsf{BitDecomp}_q(\mathbf{c}_s),\mathbf{e}_{\mathbf{s}:\mathbf{t}}\rangle \pmod q .$$

Lemma 3.6 (security). *Let $\mathbf{s}\in\mathbb{Z}^{n_s}$ be arbitrary, $\mathbf{t}\leftarrow\mathsf{Regev.SecretKeygen}(1^n)$ and $\mathbf{P}\leftarrow\mathsf{SwitchKeyGen}_{q,\chi}(\mathbf{s},\mathbf{t})$. Then $\mathbf{P}$ is computationally indistinguishable from uniform over $\mathbb{Z}_q^{\hat{n}_s\times(n_t+1)}$, assuming $\mathsf{DLWE}_{n,q,\chi}$.*

4 A Scale Invariant Homomorphic Encryption Scheme

We present our scale invariant L-homomorphic scheme as outlined in Section 1.2. Homomorphic properties are discussed in Section 4.1, implications and optimizations are discussed in Section 4.2.

Let $q=q(n)$ be an integer function, let $L=L(n)$ be a polynomial and let $\chi=\chi(n)$ be a distribution ensemble over $\mathbb{Z}$. The scheme $\mathsf{SI\text{-}HE}$ is defined as follows:

- $\mathsf{SI\text{-}HE.Keygen}(1^L,1^n)$: Sample vectors $\mathbf{s}_0,\ldots,\mathbf{s}_L\leftarrow\mathsf{Regev.SecretKeygen}(1^n)$. Compute a Regev public key for the first one: $\mathbf{P}_0\leftarrow\mathsf{Regev.PublicKeygen}(\mathbf{s}_0)$. For all $i\in[L]$, define
  $$\tilde{\mathbf{s}}_{i-1} := \mathsf{BitDecomp}((1,\mathbf{s}_{i-1}))\otimes\mathsf{BitDecomp}((1,\mathbf{s}_{i-1})) \in \{0,1\}^{((n+1)\lceil\log q\rceil)^2} .$$
  and compute
  $$\mathbf{P}_{(i-1):i}\leftarrow\mathsf{SwitchKeyGen}\left(\tilde{\mathbf{s}}_{i-1},\mathbf{s}_i\right) .$$
  Output $pk=\mathbf{P}_0$, $evk=\{\mathbf{P}_{(i-1):i}\}_{i\in[L]}$ and $sk=\mathbf{s}_L$.

- $\mathsf{SI\text{-}HE.Enc}_{pk}(m)$: Identical to Regev's, output $\mathbf{c} \leftarrow \mathsf{Regev.Enc}_{pk}(m)$.
- $\mathsf{SI\text{-}HE.Eval}_{evk}(\cdot)$: As usual, we describe homomorphic addition and multiplication over GF(2), which allows to evaluate depth L arithmetic circuits in a gate-by-gate manner. The convention for a gate at level i of the circuit is that the operand ciphertexts are decryptable using $\mathbf{s}_{i-1}$, and the output of the homomorphic operation is decryptable using $\mathbf{s}_i$.
  Since evk contains key switching parameters from $\tilde{\mathbf{s}}_{i-1}$ to $\mathbf{s}_i$, homomorphic addition and multiplication both first produce an intermediate output $\tilde{\mathbf{c}}$ that corresponds to $\tilde{\mathbf{s}}_{i-1}$, and then use key switching to obtain the final output.[9]
  - $\mathsf{SI\text{-}HE.Add}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$: Assume w.l.o.g that both input ciphertexts are encrypted under the same secret key $\mathbf{s}_{i-1}$. First compute
  $$\tilde{\mathbf{c}}_{\mathsf{add}} := \mathsf{PowersOfTwo}(\mathbf{c}_1 + \mathbf{c}_2) \otimes \mathsf{PowersOfTwo}((1, 0, \ldots, 0)) \ ,$$
  then output
  $$\mathbf{c}_{\mathsf{add}} \leftarrow \mathsf{SwitchKey}(\mathbf{P}_{(i-1):i}, \tilde{\mathbf{c}}_{\mathsf{add}}) \ \in \mathbb{Z}_q^{n+1} \ .$$
  Let us explain what we did: We first added the ciphertext vectors (as expected) to obtain $\mathbf{c}_1 + \mathbf{c}_2$. This already implements the homomorphic addition, but provides an output that corresponds to $\mathbf{s}_{i-1}$ and not $\mathbf{s}_i$ as required. We thus generate $\tilde{\mathbf{c}}_{\mathsf{add}}$ by tensoring with a "trivial" ciphertext. The result corresponds to $\tilde{\mathbf{s}}_{i-1}$, and allows to finally use key switching to obtain an output corresponding to $\mathbf{s}_i$. We use powers-of-two representation in order to control the norm of the secret key (as we explain in Section 1.2).
  - $\mathsf{SI\text{-}HE.Mult}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$: Assume w.l.o.g that both input ciphertexts are encrypted under the same secret key $\mathbf{s}_{i-1}$. First compute
  $$\tilde{\mathbf{c}}_{\mathsf{mult}} := \left\lfloor \frac{2}{q} \cdot \Big(\mathsf{PowersOfTwo}(\mathbf{c}_1) \otimes \mathsf{PowersOfTwo}(\mathbf{c}_2) \Big) \right\rceil \ ,$$
  then output
  $$\mathbf{c}_{\mathsf{mult}} \leftarrow \mathsf{SwitchKey}(\mathbf{P}_{(i-1):i}, \tilde{\mathbf{c}}_{\mathsf{mult}}) \ \in \mathbb{Z}_q^{n+1} \ .$$
  As we explain in Section 1.2, The tensored ciphertext $\tilde{\mathbf{c}}_{\mathsf{mult}}$ mimics tensoring in the "invariant perspective", which produces an encryption of the product of the plaintexts under the tensored secret key $\tilde{\mathbf{s}}_{i-1}$. We then switch keys to obtain an output corresponding to $\mathbf{s}_i$.
- Decryption $\mathsf{SI\text{-}HE.Dec}_{sk}(\mathbf{c})$: Assume w.l.o.g that $\mathbf{c}$ is a ciphertext that corresponds to $\mathbf{s}_L$ $(=sk)$. Then decryption is again identical to Regev's, output
$$m \leftarrow \mathsf{Regev.Dec}_{sk}(\mathbf{c}) \ .$$

[9] The final key switching replaces the more complicated "refresh" operation of [3].

Security. The security of the scheme follows in a straightforward way, very similarly to the proof of [5, Theorem 4.1]. The proof is omitted.

Lemma 4.1. *Let n, q, χ be some parameters such that $\mathsf{DLWE}_{n,q,\chi}$ holds, and let $L = L(n)$ be polynomially bounded. Let $(pk, evk, sk) \leftarrow \mathsf{SI\text{-}HE.Keygen}(1^L, 1^n)$, $\mathbf{c} \leftarrow \mathsf{SI\text{-}HE.Enc}_{pk}(m)$ (for any $m \in \{0,1\}$), then it holds that the joint distribution $(pk, evk, \mathbf{c})$ is computationally indistinguishable from uniform.*

4.1 Homomorphic Properties of SI-HE

The following theorem summarizes the homomorphic properties of our scheme.

Theorem 4.2. *The scheme* SI-HE *with parameters $n, q, |\chi| \leq B, L$ for which*

$$q/B \geq (O(n \log q))^{L+O(1)} \ ,$$

is L-homomorphic.

The theorem is proven using the following lemma, which bounds the growth of the noise in gate evaluation.

Lemma 4.3. *Let $q, n, |\chi| \leq B, L$ be parameters for* SI-HE*. Set (pk, evk, sk) by calling $\mathsf{SI\text{-}HE.Keygen}(1^L, 1^n)$ and let $\mathbf{c}_1, \mathbf{c}_2$ be such that*

$$\begin{aligned} \langle \mathbf{c}_1, (1, \mathbf{s}_{i-1}) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_1 \pmod q \\ \langle \mathbf{c}_2, (1, \mathbf{s}_{i-1}) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_2 \pmod q \ , \end{aligned} \tag{1}$$

with $|e_1|, |e_2| \leq E < \lfloor q/2 \rfloor / 2$. Consider ciphertexts $\mathbf{c}_{\mathsf{add}} \leftarrow \mathsf{SI\text{-}HE.Add}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$, $\mathbf{c}_{\mathsf{mult}} \leftarrow \mathsf{SI\text{-}HE.Mult}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$. Then

$$\begin{aligned} \langle \mathbf{c}_{\mathsf{add}}, (1, \mathbf{s}_i) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot \left([m_1 + m_2]_2 \right) + e_{\mathsf{add}} \pmod q \\ \langle \mathbf{c}_{\mathsf{mult}}, (1, \mathbf{s}_i) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 m_2 + e_{\mathsf{mult}} \pmod q \ , \end{aligned}$$

where

$$|e_{\mathsf{add}}|, |e_{\mathsf{mult}}| \leq O(n \log q) \cdot \max \left\{ E, (n \log^2 q) \cdot B \right\} \ .$$

We remark that, as usual, homomorphic addition increases noise much more moderately than multiplication, but the coarse bound we show in the lemma is sufficient for our purposes.

Theorem 4.2 follows from Lemma 4.3 in a straightforward manner. The proof of Lemma 4.3 appears in Section 4.3.

4.2 Implications and Optimizations

Fully Homomorphic Encryption using Bootstrapping. Fully homomorphic encryption follows using the bootstrapping theorem (Theorem 2.5). In order to use bootstrapping, we need to bound the depth of the decryption circuit. The following lemma has been proven in a number of previous works (e.g. [5, Lemma 4.5]):

Lemma 4.4. *For all* $\mathbf{c}$*, the function* $f_{\mathbf{c}}(\mathbf{s}) = \mathsf{SI\text{-}HE.Dec}_{\mathbf{s}}(\mathbf{c})$ *can be implemented by a circuit of depth* $O(\log n + \log\log q)$*.*

An immediate corollary follows from Theorem 2.5, Theorem 4.2, Lemma 4.4:

Corollary 4.5. *Let* $n, q, |\chi| \leq B$ *be such that* $q/B \geq (n \log q)^{O(\log n + \log\log q)}$*. Then there exists a (leveled) fully homomorphic encryption scheme based on the* $\mathsf{DLWE}_{n,q,\chi}$ *assumption.*

Furthermore, if SI-HE *is weak circular secure, then the same assumption implies full (non leveled) homomorphism.*

Finally, we can classically reduce security from GapSVP using Corollary 2.2, by choosing $q = \tilde{O}(2^{n/2})$ and $B = q/(n \log q)^{O(\log n + \log\log q)} = q/n^{O(\log n)}$:

Corollary 4.6. *There exists a (leveled) fully-homomorphic encryption scheme based on the classical worst case hardness of the* $\mathsf{GapSVP}_{n^{O(\log n)}}$ *problem.*

(Leveled) Fully Homomorphic Encryption without Bootstrapping. As in [3], our scheme implies a leveled fully homomorphic encryption without bootstrapping. Plugging our scheme into the [3] framework, we obtain a (leveled) fully homomorphic encryption without bootstrapping, based on the classical worst case hardness of $\mathsf{GapSVP}_{2^{n^\epsilon}}$, for any $\epsilon > 0$.

Optimizations. So far, we chose to present our scheme in the cleanest possible way. However, there are a few techniques that can somewhat improve performance. While the asymptotic advantage of some of these methods is not great, a real life implementation can benefit from them.

1. Our tensored secret key $\tilde{\mathbf{s}}_{i-1}$ is obtained by tensoring a vector with itself. Such a vector can be represented by only $\binom{n_s}{2}$ (as opposed to our n_s^2), saving a factor of (almost) 2 in the representation length.
2. When $B \ll q$, some improvement can be achieved by using LWE in Hermite normal form. It is known (see e.g. [2]) that the hardness of LWE remains essentially the same if we sample $\mathbf{s} \xleftarrow{\$} \chi^n$ (instead of uniformly in $\mathbb{Z}_q^n$). Sampling our keys this way, we only need $O(n \log B)$ bits to represent $\mathsf{BitDecomp}(\mathbf{s})$, and its norm goes down accordingly.
   We can therefore reduce the size of the evaluation key (which depends quadratically on the bit length of the secret key), and more importantly, we can prove a tighter version of Lemma 4.3. When using Hermite normal form, the noise grows from E to $O(n \log B) \cdot \max\{E, (n \log B \log q) \cdot B\}$. Therefore, L-homomorphism is achieved whenever
   $$q/B \geq (O(n \log B))^{L+O(1)} \cdot \log q \ .$$

3. The least significant bits of the ciphertext can sometimes be truncated without much harm, which can lead to significant saving in ciphertext and key length, especially when B/q is large: Let $\mathbf{c}$ be an integer ciphertext vector and define $\mathbf{c}' = \lfloor 2^{-i} \cdot \mathbf{c} \rceil$. Then $\mathbf{c}'$, which can be represented with $n \cdot i$ fewer bits than $\mathbf{c}$, implies a good approximation for $\mathbf{c}$ since

$$\left|\langle \mathbf{c}, \mathbf{s} \rangle - \langle 2^i \cdot \mathbf{c}', \mathbf{s} \rangle\right| \leq 2^{i-1} \left\|\mathbf{s}\right\|_1 \ .$$

This means that $2^i \cdot \mathbf{c}'$ can be used instead of $\mathbf{c}$, at the cost of an additive increase in the noise magnitude.
Consider a case where $q, B \gg q/B$ (which occurs when we artificially increase q in order for the classical reduction to work). Recall that $\left\|\mathbf{s}\right\|_1 \approx n \log q$ and consider truncating with $i \approx \log(B/(n \log q))$. Then the additional noise incurred by using $\mathbf{c}'$ instead of $\mathbf{c}$ is only an insignificant $\approx B$. The number of bits required to represent each element in $\mathbf{c}'$ however now becomes $\log q - i \approx \log(q/B) + \log(n \log q)$. In conclusion, we hardly lose anything in ciphertext length compared to the case of working with smaller q, B to begin with (with similar q/B ratio). The ciphertext length can, therefore, be made invariant to the absolute values of q, B, and depend only on their ratio. This of course applies also to the vectors in evk.

4.3 Proof of Lemma 4.3

We start with the analysis for addition, which is simpler and will also serve as good warm-up towards the analysis for multiplication.

Analysis for Addition. By Lemma 3.5, it holds that

$$\langle \mathbf{c}_{\mathsf{add}}, (1, \mathbf{s}_i) \rangle = \langle \tilde{\mathbf{c}}_{\mathsf{add}}, \tilde{\mathbf{s}}_i \rangle + \underbrace{\langle \mathsf{BitDecomp}(\tilde{\mathbf{c}}), \mathbf{e}_{i-1:i} \rangle}_{\triangleq \delta_1} \pmod q \ .$$

where $\mathbf{e}_{i-1:i} \sim \chi^{(n+1)^2 \cdot (\lceil \log q \rceil)^3}$. That is, δ_1 is the noise inflicted by the key switching process.

We bound $|\delta_1|$ using the bound on χ:

$$|\delta_1| = |\langle \mathsf{BitDecomp}(\tilde{\mathbf{c}}_{\mathsf{add}}), \mathbf{e}_{i-1:i} \rangle| \leq (n+1)^2 \cdot (\lceil \log q \rceil)^3 \cdot B = O(n^2 \log^3 q) \cdot B \ .$$

Next, we expand the term $\langle \tilde{\mathbf{c}}_{\mathsf{add}}, \tilde{\mathbf{s}}_i \rangle$, by breaking an inner product of tensors into a product of inner products (one of which is trivially equal to 1):

$$\begin{aligned}
\langle \tilde{\mathbf{c}}_{\mathsf{add}}, \tilde{\mathbf{s}}_i \rangle &= \Big\langle \mathsf{PowersOfTwo}(\mathbf{c}_1 + \mathbf{c}_2) \otimes \mathsf{PowersOfTwo}((1, 0, \ldots, 0)), \\
&\qquad\qquad \mathsf{BitDecomp}((1, \mathbf{s}_{i-1})) \otimes \mathsf{BitDecomp}((1, \mathbf{s}_{i-1})) \Big\rangle \\
&= \big\langle \mathsf{PowersOfTwo}(\mathbf{c}_1 + \mathbf{c}_2), \mathsf{BitDecomp}((1, \mathbf{s}_{i-1})) \big\rangle \cdot 1 \\
&= \big\langle (\mathbf{c}_1 + \mathbf{c}_2), (1, \mathbf{s}_{i-1}) \big\rangle \qquad\qquad \pmod q \\
&= \big\langle \mathbf{c}_1, (1, \mathbf{s}_{i-1}) \big\rangle + \big\langle \mathbf{c}_2, (1, \mathbf{s}_{i-1}) \big\rangle \qquad\qquad \pmod q \ .
\end{aligned}$$

We can now plug in what we know about $\mathbf{c}_1, \mathbf{c}_2$ from Eq. (1) in the lemma statement:

$$\begin{aligned}\langle \tilde{\mathbf{c}}_{\mathsf{add}}, \tilde{\mathbf{s}}_i \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_1 + \left\lfloor \frac{q}{2} \right\rfloor \cdot m_2 + e_2 \qquad (\text{mod } q) \\ &= \left\lfloor \frac{q}{2} \right\rfloor \cdot [m_1 + m_2]_2 \underbrace{-\tilde{m} + e_1 + e_2}_{\triangleq \delta_2} \ , \quad (\text{mod } q)\end{aligned}$$

where $\tilde{m} \in \{0,1\}$ is defined as:

$$\tilde{m} \triangleq \begin{cases} 0, & \text{if } q \text{ is even,} \\ \frac{1}{2} \cdot (m_1 + m_2 - [m_1 + m_2]_2), & \text{if } q \text{ is odd,} \end{cases}$$

and $|\delta_2| \leq 1 + 2E$.

Putting it all together,

$$\langle \mathbf{c}_{\mathsf{add}}, (1, \mathbf{s}_i) \rangle = \left\lfloor \frac{q}{2} \right\rfloor \cdot [m_1 + m_2]_2 + \underbrace{\delta_1 + \delta_2}_{= e_{\mathsf{add}}} \pmod q \ .$$

Where the bound on e_{add} is

$$|e_{\mathsf{add}}| = |\delta_1 + \delta_2| \leq O(n^2 \log^3 q) \cdot B + O(1) \cdot E \leq O(n \log q) \cdot \max\left\{E, (n \log^2 q) \cdot B\right\}$$

which finishes the argument for addition.

Analysis for Multiplication. The analysis for multiplication starts very similarly to addition:

$$\langle \mathbf{c}_{\mathsf{mult}}, (1, \mathbf{s}_i) \rangle = \langle \tilde{\mathbf{c}}_{\mathsf{mult}}, \tilde{\mathbf{s}}_i \rangle + \underbrace{\langle \mathsf{BitDecomp}(\tilde{\mathbf{c}}_{\mathsf{mult}}), \mathbf{e}_{i-1:i} \rangle}_{\triangleq \delta_1} \pmod q \ ,$$

and as before

$$|\delta_1| = O(n^2 \log^3 q) \cdot B \ .$$

Let us now focus on $\langle \tilde{\mathbf{c}}_{\mathsf{mult}}, \tilde{\mathbf{s}}_i \rangle$. We want to use the properties of tensoring to break the inner product into two smaller inner products, as we did before. This time, however, $\tilde{\mathbf{c}}_{\mathsf{mult}}$ is a *rounded* tensor:

$$\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}_i \rangle = \left\langle \left\lfloor \frac{2}{q} \cdot (\mathsf{PowersOfTwo}(\mathbf{c}_1) \otimes \mathsf{PowersOfTwo}(\mathbf{c}_2)) \right\rceil, \tilde{\mathbf{s}}_{i-1} \right\rangle \pmod q \ .$$

We start by showing that the rounding does not add much noise. Intuitively this is because $\tilde{\mathbf{s}}_{i-1}$ is a binary vector and thus has low norm. We define

$$\begin{aligned}\delta_2 \triangleq \left\langle \left\lfloor \frac{2}{q} \cdot (\mathsf{PowersOfTwo}(\mathbf{c}_1) \otimes \mathsf{PowersOfTwo}(\mathbf{c}_2)) \right\rceil, \tilde{\mathbf{s}}_{i-1} \right\rangle \\ - \left\langle \frac{2}{q} \cdot (\mathsf{PowersOfTwo}(\mathbf{c}_1) \otimes \mathsf{PowersOfTwo}(\mathbf{c}_2)), \tilde{\mathbf{s}}_{i-1} \right\rangle \ ,\end{aligned}$$

and for convenience we also define

$$\mathbf{c}' \triangleq \left\lfloor \frac{2}{q} \cdot (\mathsf{PowersOfTwo}(\mathbf{c}_1) \otimes \mathsf{PowersOfTwo}(\mathbf{c}_2)) \right\rceil - \frac{2}{q} \cdot (\mathsf{PowersOfTwo}(\mathbf{c}_1) \otimes \mathsf{PowersOfTwo}(\mathbf{c}_2)) \ .$$

By definition, $\delta_2 = \langle \mathbf{c}', \tilde{\mathbf{s}}_{i-1} \rangle$. We know that $\|\mathbf{c}'\|_\infty \leq 1/2$ and $\|\tilde{\mathbf{s}}_{i-1}\|_1 \leq ((n+1)\lceil \log q \rceil)^2 = O(n^2 \log^2 q)$, and therefore

$$|\delta_2| \leq \|\mathbf{c}'\|_\infty \cdot \|\tilde{\mathbf{s}}_{i-1}\|_1 = O(n^2 \log^2 q) \ .$$

We can now break the inner product using the properties of tensoring:

$$\langle \tilde{\mathbf{c}}_{\mathsf{mult}}, \tilde{\mathbf{s}}_{i-1} \rangle - \delta_2 = \frac{2}{q} \cdot \left\langle \mathsf{PowersOfTwo}(\mathbf{c}_1), \mathsf{BitDecomp}((1, \mathbf{s}_{i-1})) \right\rangle \cdot \left\langle \mathsf{PowersOfTwo}(\mathbf{c}_2), \mathsf{BitDecomp}((1, \mathbf{s}_{i-1})) \right\rangle \ . \quad (2)$$

We keep $\mathbf{c}_1, \mathbf{c}_2$ in powers-of-two form on purpose (it is useful to have ciphertexts that relate to low-norm secrets).

Going back to Eq. (1) from the lemma statement and using Claim 3.4, there exist $I_1, I_2 \in \mathbb{Z}$ such that

$$\begin{aligned} \langle \mathsf{PowersOfTwo}(\mathbf{c}_1), \mathsf{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_1 + q \cdot I_1 \\ \langle \mathsf{PowersOfTwo}(\mathbf{c}_2), \mathsf{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_2 + e_2 + q \cdot I_2 \ . \end{aligned} \quad (3)$$

Let us bound the absolute value of I_1 (obviously the same holds for I_2):

$$\begin{aligned} |I_1| &= \frac{\left| \langle \mathsf{PowersOfTwo}(\mathbf{c}_1), \mathsf{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle - \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 - e_1 \right|}{q} \\ &\leq \frac{\left| \langle \mathsf{PowersOfTwo}(\mathbf{c}_1), \mathsf{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle \right|}{q} + 1 \\ &\leq \frac{\|\mathsf{PowersOfTwo}(\mathbf{c}_1)\|_\infty}{q} \cdot \|\mathsf{BitDecomp}(1, \mathbf{s}_{i-1})\|_1 + 1 \\ &\leq \frac{1}{2} \cdot \|\mathsf{BitDecomp}(1, \mathbf{s}_{i-1})\|_1 + 1 \\ &\leq \frac{1}{2} \cdot (n+1)\lceil \log q \rceil + 1 \\ &= O(n \log q) \ . \end{aligned} \quad (4)$$

Plugging Eq. (3) into Eq. (2), we get

$$\begin{aligned} \langle \tilde{\mathbf{c}}_{\mathsf{mult}}, \tilde{\mathbf{s}}_{i-1} \rangle - \delta_2 &= \frac{2}{q} \cdot \left(\left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_1 + q \cdot I_1 \right) \cdot \left(\left\lfloor \frac{q}{2} \right\rfloor \cdot m_2 + e_2 + q \cdot I_2 \right) \\ &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 \cdot m_2 + \delta_3 + q \cdot (m_1 I_2 + m_2 I_1 + 2 I_1 I_2) \ , \end{aligned}$$

where δ_3 is defined as

$$\delta_3 \triangleq \begin{cases} 2e_2 \cdot I_1 + 2e_1 \cdot I_2 + (e_1 m_2 + e_2 m_1) + \frac{2e_1 \cdot e_2}{q} \ , & \text{if } q \text{ is even,} \\ (2e_2 - m_2) \cdot I_1 + (2e_1 - m_1) \cdot I_2 & \\ \qquad + \frac{q-1}{q} \cdot (e_1 m_2 + e_2 m_1) - \frac{m_1 \cdot m_2}{2q} + \frac{2e_1 \cdot e_2}{q} \ , & \text{if } q \text{ is odd.} \end{cases}$$

In particular (recall that $E < \lfloor q/2 \rfloor / 2 \leq q/4$):

$$|\delta_3| \leq 2(2E+1)O(n \log q) + 2E + \frac{1 + 2E^2}{q} = O(n \log q) \cdot E \ .$$

Putting everything together, we get that

$$\langle \mathbf{c}_{\mathsf{mult}}, (1, \mathbf{s}_i) \rangle = \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 m_2 + \underbrace{\delta_1 + \delta_2 + \delta_3}_{= e_{\mathsf{mult}}} \pmod q \ ,$$

where

$$|e_{\mathsf{mult}}| = |\delta_1 + \delta_2 + \delta_3| \leq O(n \log q) \cdot E + O(n^2 \log^3 q) \cdot B \ ,$$

and the lemma follows. □

Acknowledgments

We thank Vinod Vaikuntanathan for fruitful discussions and advice, and Dan Boneh for his comments on an earlier version of this manuscript. We thank the reviewers of CRYPTO 2012 for their constructive comments. In addition, we thank various readers for pointing out typos in earlier versions of this manuscript.

References

1. M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In F. T. Leighton and P. W. Shor, editors, *STOC*, pages 284–293. ACM, 1997.
2. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.
3. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. ITCS, 2012. See also `http://eprint.iacr.org/2011/277`.
4. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, volume 6841, page 501, 2011.
5. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Ostrovsky [19], pages 97–106. References are to full version: `http://eprint.iacr.org/2011/344`.
6. J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Rogaway [24], pages 487–504.

7. M. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.
8. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.
9. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
10. C. Gentry and S. Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Ostrovsky [19], pages 107–109.
11. C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:680, 2011.
12. C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. *IACR Cryptology ePrint Archive*, 2011:566, 2011.
13. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In C. Dwork, editor, *STOC*, pages 197–206. ACM, 2008.
14. O. Goldreich, S. Goldwasser, and S. Halevi. Eliminating decryption errors in the ajtai-dwork cryptosystem. In B. S. K. Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 105–111. Springer, 1997.
15. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010. Draft of full version was provided by the authors.
16. D. Micciancio and P. Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In Rogaway [24], pages 465–484.
17. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.
18. D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In L. J. Schulman, editor, *STOC*, pages 351–358. ACM, 2010.
19. R. Ostrovsky, editor. *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. IEEE, 2011.
20. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.
21. O. Regev. New lattice based cryptographic constructions. In L. L. Larmore and M. X. Goemans, editors, *STOC*, pages 407–416. ACM, 2003. Full version in J. ACM 51(6), 2004.
22. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 84–93. ACM, 2005. Full version in J. ACM 56(6), 2009.
23. R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.
24. P. Rogaway, editor. *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*. Springer, 2011.
25. C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.
26. N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In P. Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.