Lattice-Based FHE as Secure as PKE

Zvika Brakerski[*] Vinod Vaikuntanathan[†]

Abstract

We show that (leveled) fully homomorphic encryption (FHE) can be based on the hardness of $\widetilde{O}(n^{1.5+\epsilon})$-approximation for lattice problems (such as GapSVP) under quantum reductions for any $\epsilon > 0$ (or $\widetilde{O}(n^{2+\epsilon})$-approximation under classical reductions). This matches the best known hardness for "regular" (non-homomorphic) lattice based public-key encryption up to the ϵ factor. A number of previous methods had hit a roadblock at quasipolynomial approximation. (As usual, a circular security assumption can be used to achieve a non-leveled FHE scheme.)

Our approach consists of three main ideas: Noise-bounded sequential evaluation of high fan-in operations; Circuit sequentialization using Barrington's Theorem; and finally, successive dimension-modulus reduction.

1 Introduction

Fully homomorphic encryption (FHE) allows us to convert an encryption of a message $\mathsf{Enc}(m)$ into an encryption of a related message $\mathsf{Enc}(f(m))$ for *any* efficient f, using only public information and without revealing anything about m itself. FHE has numerous theoretical and practical applications, the canonical one being to the problem of *outsourcing* computation to a remote server without compromising one's privacy.

Until 2008, FHE was considered practically science fiction as no constructions or even viable approaches were known. A breakthrough by Gentry [Gen09b, Gen10, Gen09a] presented the first plausible candidate construction. The security of Gentry's scheme relied on much stronger assumptions than standard (non homomorphic) public-key encryption (PKE), namely the hardness of problems on specially chosen ideal lattices as well as a new assumption called the sparse subset sum assumption. This state of affairs coincided with many researchers' intuition that FHE, being much more versatile, should naturally be harder to achieve and should require stronger assumptions than regular public-key encryption. Brakerski and Vaikuntanathan [BV11] subsequently constructed an FHE scheme[1] based on the worst-case hardness of approximating lattice problems such as GapSVP (a promise version of the shortest vector problem on lattices) which have been studied extensively and are by now considered standard cryptographic assumptions. However, they required that the problem is hard to approximate to within a subexponential factor (in the dimension of the underlying lattice). This is in contrast to standard lattice-based public-key encryption

[*]Stanford University, `zvika@stanford.edu`. Supported by a Simons Postdoctoral Fellowship and by DARPA.

[†]MIT and University of Toronto, `vinodv@mit.edu`. Supported by an NSERC Discovery Grant, DARPA Grant number FA8750-11-2-0225, Connaught New Researcher Award and an Alfred P. Sloan Fellowship.

[1]Here, and in the rest of the introduction, when we say FHE, we mean a leveled FHE scheme that can evaluate circuits of any a-priori bounded polynomial depth. The only known way to achieve non-leveled FHE schemes is to make a circular security assumption, in addition.

which can be based on the hardness of approximating the problem to within polynomial factors (explicitly $\widetilde{O}(n^{1.5})$ using quantum reductions [Reg05] or $\widetilde{O}(n^2)$ using classical reductions [Pei09]). Closing this gap has been a central goal in the study of FHE from both a theoretical and a practical perspective (since relying on a weaker assumption allows us to use shorter parameters resulting in better efficiency). Starting with [BGV12], several works using different approaches [Bra12, GSW13] have reduced the required factor of approximation to $n^{O(\log n)}$, which seemed to be a barrier for known methods.

In this work, we match the best known approximation factors up to any $\epsilon > 0$ and show that "science fiction" FHE can be as secure as any other lattice-based public-key encryption scheme. Furthermore, the keys and ciphertexts in our scheme (with the exception of the evaluation key which is only used for homomorphic evaluation) are identical to Regev's original lattice-based PKE [Reg05], with parameters that are optimal up to a factor of $1 + \epsilon$.

Our results are summarized in the following theorem.

Theorem 1.1. *For every $\epsilon > 0$, there exists a leveled fully homomorphic encryption scheme based on the $\mathsf{DLWE}_{n,q,\alpha}$ assumption (n-dimensional decisional LWE modulo q, with discrete Gaussian noise with parameter α), where $\alpha = 1/\widetilde{O}(n^{\epsilon} \cdot \sqrt{n \log(q)})$.*

Thus, the scheme is secure based on either the quantum *worst-case hardness of* $\mathsf{GapSVP}_{\widetilde{O}(n^{1.5+\epsilon})}$*, or the* classical *worst-case hardness of* $\mathsf{GapSVP}_{\widetilde{O}(n^{2+\epsilon})}$.

High Level Overview. Our starting point is a new LWE-based FHE scheme by Gentry, Sahai and Waters [GSW13]. They present an encryption scheme where the public key is identical to Regev's scheme, but the ciphertexts are square matrices rather than vectors. It was then possible to add and multiply ciphertexts using (roughly) matrix addition and multiplication. As in previous LWE-based FHE schemes, the ciphertext contains a "noise" element that grows with homomorphic operations and must be kept under a certain threshold in order for the ciphertext to be decryptable. The scheme is instantiated by a dimension n and modulus q, which correspond to the parameters of the LWE problem. The initial noise level is $\text{poly}(n)$ and the scheme is decryptable so long as the noise remains under (say) $q/8$. In order to base the scheme on the hardness of polynomial approximation to lattice problems, we would like to characterize the class of functions that can be homomorphically evaluated using $q = \text{poly}(n)$. The analysis of Gentry, Sahai and Waters [GSW13] shows that the evaluation of each Boolean gate increases the noise by a $\text{poly}(n)$ factor, and thus the class of functions that can be evaluated setting $q = \text{poly}(n)$ is NC^0.

Our first observation is that the asymmetric (namely, non-commutative) nature of matrix multiplication gives rise to an interesting phenomenon in the GSW scheme: when multiplying two ciphertexts with noise levels e_1 and e_2, the noise in the output turns out to be $e_1 + \text{poly}(n) \cdot e_2$. That is, the noise grows in an asymmetric manner. This means that if we want to multiply ℓ ciphertexts, for example, which all start with the same noise level, we can consecutively multiply them one after the other, and the final noise will only grow by a $\ell \cdot \text{poly}(n)$ factor. This is in contrast to the conventional wisdom that favors the use of a multiplication tree, which in this case would have resulted in a $\text{poly}(n)^{\log \ell}$ noise blowup. This observation already allows us to evaluate AC^0 circuits in a setting where the modulus $q = \text{poly}(n)$. (Using an additional trick, this can be extended to $\mathsf{AC}^0[\oplus]$, namely AC^0 circuits augmented with XOR gates).

Our second idea is to push this technique forward by "sequentializing" larger circuit classes. A particularly potent tool in this direction of thought is Barrington's Theorem [Bar89] which allows us to transform any NC^1 circuit into a polynomial length, width-5 permutation branching program.

Homomorphic evaluation of a length-ℓ branching program essentially requires homomorphically multiplying ℓ 5-by-5 encrypted permutation matrices, in contrast to the simple product operation on bits that we just accomplished. We show that this is in fact possible, namely a method of homomorphically multiplying ℓ permutation matrices that only increases the noise by an $\ell \cdot \mathrm{poly}(n)$ factor. This gives us a way to evaluate any NC^1 circuit in a setting where the modulus $q = \mathrm{poly}(n)$. In a high level, our technique here is reminiscent of Ishai and Paskin's method of evaluating branching programs on encrypted data [IP07].

Evaluating NC^1 circuits with low noise blowup is a highly sought-after goal in the study of FHE schemes. The reason is Gentry's bootstrapping theorem [Gen09b], which shows how to convert a scheme with some homomorphic properties into a fully homomorphic one, assuming that it can evaluate its own decryption circuit. Since the decryption circuit of the scheme in question lies in NC^1, we can apply the bootstrapping theorem and obtain and FHE scheme with $q = \mathrm{poly}(n)$, thus basing its security on the worst-case hardness of approximating lattice problems to within a (somewhat large) polynomial factor.

To obtain the optimal approximation factor (up to an arbitrarily small ϵ), we employ our third idea, namely a variant of the *dimension-modulus reduction* technique, originating in [BV11]. Our noise analysis of the NC^1 scheme shows that in order to obtain parameters that are optimal up to ϵ, the decryption circuit of our scheme must have depth at most $\epsilon \cdot \log(n)/2$, which seems unachievable. After all, an NC^1 circuit with n inputs and depth less than $\log n$ cannot even look at all the inputs! To solve this conundrum, we apply the *dimension-modulus reduction* technique, which allows us to "shrink" the ciphertext into a "smaller copy" of the same scheme. We show that by applying this method consecutively several times (as opposed to a single time as was done in [BV11]), we can reduce the ciphertext to a small enough size that decrypting it becomes possible in depth $\epsilon \log(n)/2$. This allows us to obtain an FHE scheme based on the worst-case hardness of approximating GapSVP within a factor of $\widetilde{O}(n^{2+\epsilon})$ by classical algorithms, or a factor of $\widetilde{O}(n^{1.5+\epsilon})$ by quantum algorithms.

Organization of the Paper. We start with some background and preliminaries: the reader should consult section 2.1 for background on Gaussian distributions, section 2.2 for the learning with error problem, and section 2.5 for homomorphic encryption. Our main result is described in Section 3 where we construct a (leveled) FHE scheme secure under the polynomial LWE assumption. We conclude in Section 4 by showing how to show how to reduce and optimize the assumption to match the best known LWE assumption for lattice-based PKE.

2 Preliminaries

Matrices are denoted by bold-face capital letters, and vectors are denoted by bold-face small letters. All logarithms are taken to base 2, unless otherwise specified. For an integer q, we define the set $\mathbb{Z}_q \triangleq (-q/2, q/2] \cap \mathbb{Z}$. 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).

We let κ denote a security parameter. When we speak of a negligible function $\mathrm{negl}(\kappa)$, we mean a function that grows slower than $1/\kappa^c$ for any constant $c > 0$ and sufficiently large values of κ. When we say that an event happens with overwhelming probability, we mean that it happens with probability at least $1 - \mathrm{negl}(\kappa)$ for some negligible function $\mathrm{negl}(\kappa)$. We denote $y = \widetilde{O}_\kappa(x)$ if $y = O(x \cdot \mathrm{polylog}(\kappa))$, and $y = \widetilde{O}(x)$ if $y = \widetilde{O}_x(x)$. The notation $\widetilde{\Theta}_\kappa(\cdot)$, $\widetilde{\Omega}_\kappa(\cdot)$ is defined analogously.

The security parameter underlies all of our constructions. The parameters n, k etc. should all be considered to be a function of the security parameter κ, which is chosen according to the level of confidence desired by the user of the scheme. (The dimension of the LWE problem, defined below, should be considered to be polynomially related to the security parameter.)

2.1 Gaussians and Discrete Gaussians

In this work we will only consider one-dimensional Gaussians, and one-dimensional discrete Gaussians over the integers.

For $r > 0$, the (one-dimensional) Gaussian function $\rho_r : \mathbb{R} \to (0, 1]$ is defined as

$$\rho_r(x) \triangleq \exp(-\pi|x|^2/r^2).$$

The (spherical) continuous Gaussian distribution D_r is the distribution with density function proportional to ρ_r. The (one-dimensional, integer-coset) discrete Gaussian $D_{\mathbb{Z}-c,r}$ is the discrete distribution supported on $\mathbb{Z} - c$ for $c \in \mathbb{R}$, whose probability mass function is proportional to ρ_r.

Gaussian Rounding. To achieve the tightest results, we will need to use a simple Gaussian rounding procedure. The following is an immediate corollary of [BLP+13, Lemma 2.3].

Corollary 2.1. *There exists a randomized procedure $\lfloor\cdot\rceil_G$ such that given $x \in \mathbb{R}$, it holds that $y \leftarrow \lfloor x \rceil_G$ is such that $y - x \sim D_{\mathbb{Z}-x,1}$.*

In fact, a slightly smaller standard deviation is achievable, but we use 1 for the sake of simplicity.

Sum of Discrete Gaussians. We wish to bound the absolute value of a sum of discrete Gaussians. The following are immediate corollaries from [Reg09, Corollary 3.10] and [GPV08, Lemma 3.1].

Proposition 2.2. *Let $\kappa \in \mathbb{N}$ be a security parameter. Then with all but $\mathrm{negl}(\kappa)$ probability, if $x \sim D_r$, then $|x| \leq r \cdot \omega(\sqrt{\log \kappa})$. Similarly, if $x \sim D_{\mathbb{Z}^n - c, r}$ then with all but negligible probability, $|x| \leq \max\{r, \omega(\sqrt{\log \kappa})\} \cdot \omega(\sqrt{\log \kappa})$.*

Proposition 2.3. *Let $\kappa \in \mathbb{N}$ be a security parameter. Let $n \in \mathbb{N}$, let $\mathbf{z} \in \{0,1\}^n$ and $\mathbf{c} \in \mathbb{R}^n$ be arbitrary, and let $\mathbf{e} \sim D_{\mathbb{Z}^n - \mathbf{c}, r}$. Then with all but negligible probability*

$$|\langle \mathbf{z}, \mathbf{e} \rangle| \leq \sqrt{n} \cdot \max\{r, \omega(\sqrt{\log \kappa})\} \cdot \omega(\sqrt{\log \kappa}) = \widetilde{O}_\kappa(\sqrt{n}) \cdot r\ .$$

Proposition 2.4. *Let $\kappa \in \mathbb{N}$ be a security parameter. Let $n \in \mathbb{N}$, let $\mathbf{c} \in \mathbb{R}^n$ be arbitrary, let $\mathbf{e} \sim D_{\mathbb{Z} - \mathbf{c}, r}$, and let $\mathbf{z} \in \{0,1\}^n$ be possibly dependent on $\mathbf{e}$. Then with all but negligible probability*

$$|\langle \mathbf{z}, \mathbf{e} \rangle| \leq n \cdot \max\{r, \omega(\sqrt{\log \kappa})\} \cdot \omega(\sqrt{\log \kappa}) = \widetilde{O}_\kappa(n) \cdot r\ .$$

2.2 Learning With Errors (LWE)

The LWE problem was introduced by Regev [Reg05] 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 2.5 (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 $\mathbf{s} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n$), 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 [Reg05]) and classical (Peikert [Pei09]) reductions between $\mathsf{DLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be a discrete Gaussian distribution $D_{\mathbb{Z},\alpha q}$ for some $\alpha < 1$. We sometimes write $\mathsf{DLWE}_{n,m,q,\alpha}$ (resp. $\mathsf{DLWE}_{n,q,\alpha}$) to indicate this instantiation (it will be clear from the context when we use a distribution χ and when a Gaussian parameter α). We now state a corollary of the results of [Reg05, Pei09] (in conjunction with the search to decision reduction of Micciancio and Mol [MM11] and Micciancio and Peikert [MP11]). These results also extend to additional forms of q (see [MM11, MP11]).

Corollary 2.6 ([Reg05, Pei09, MM11, MP11])**.** *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 $\alpha \geq \sqrt{n}/q$. If there is an efficient algorithm that solves the (average-case) $\mathsf{DLWE}_{n,q,\alpha}$ problem, then:*

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

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 [Reg05, Pei09] for more information.

The best known algorithms for GapSVP_γ ([Sch87]) require at least $2^{\tilde{\Omega}(n/\log\gamma)}$ time.

In this work, we will only consider the case where $q \leq 2^n$. Furthermore, the underlying security parameter κ is assumed to be polynomially related to the dimension n.

2.3 Vector Decomposition and Key Switching

We show how to decompose vectors in a way that makes their norm smaller, and yet preserves certain inner products. Our notation is generally adopted from [BGV12].

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 $w_{i,j} \in \{0,1\}$ be such that $\mathbf{x}[i] = \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j \cdot w_{i,j} \pmod q$. Output the vector
$$(w_{1,\lceil \log q \rceil -1}, \ldots, w_{1,0}, \ldots, w_{n,\lceil \log q \rceil -1}, \ldots, w_{n,0}) \in \{0,1\}^{n \cdot \lceil \log q \rceil} .$$
- $\mathsf{PowersOfTwo}_q(\mathbf{y})$: For $\mathbf{y} \in \mathbb{Z}^n$, output
$$\left[(2^{\lceil \log q \rceil - 1}\mathbf{y}[1], \ldots, 2\mathbf{y}[1], \mathbf{y}[1], \ldots, 2^{\lceil \log q \rceil - 1} \cdot \mathbf{y}[n], \ldots, 2\mathbf{y}[n], \mathbf{y}[n])\right]_q \in \mathbb{Z}_q^{n \cdot \lceil \log q \rceil} .$$

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

Claim 2.7. *For all $q \in \mathbb{N}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, it holds that*

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

Additionally, we define the procedure $\mathsf{Flatten}$ following [GSW13], along with the procedure $\mathsf{Combine}$. Let $\mathbf{g} = (2^{\lceil \log(q) \rceil - 1}, 2^{\lceil \log(q) \rceil - 2} \ldots, 4, 2, 1) \in \mathbb{Z}^{\lceil \log q \rceil}$ and let $\mathbf{G} := \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}^{n \times (n \cdot \lceil \log q \rceil)}$ denote the tensor product of $\mathbf{g}$ with the n-by-n identity matrix $\mathbf{I}_n$.

- $\mathsf{Combine}_q(\mathbf{z})$: For $\mathbf{z} \in \mathbb{Z}^{n \cdot \lceil \log q \rceil}$, output $[\mathbf{G} \cdot \mathbf{z}]_q \in \mathbb{Z}_q^n$.
- $\mathsf{Flatten}_q(\mathbf{z})$: For $\mathbf{z} \in \mathbb{Z}^{n \cdot \lceil \log q \rceil}$, output $\mathsf{BitDecomp}_q(\mathsf{Combine}(\mathbf{z})) \in \{0, 1\}^{n \cdot \lceil \log q \rceil}$.

Claim 2.8. *For all $q \in \mathbb{N}$, and $\mathbf{x}, \mathbf{z} \in \mathbb{Z}^{n \cdot \lceil \log q \rceil}$, it holds that*

$$\langle \mathsf{PowersOfTwo}(\mathbf{x}), \mathbf{z} \rangle = \langle \mathsf{PowersOfTwo}(\mathbf{x}), \mathsf{Flatten}(\mathbf{z}) \rangle \pmod{q} \ .$$

2.4 Partial Randomization Using LWE

We describe a procedure that allows us to partially randomize vectors while preserving their inner product with an LWE secret $\mathbf{s}$. This procedure will be useful to us when trying to manipulate ciphertexts that are a result of a homomorphic operation (and thus may have arbitrary dependence on the public parameters).

Let n, q, α be parameters for the DLWE problem, let $\chi = D_{\mathbb{Z}, \alpha q}$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ be some (arbitrary) vector.

- $\mathsf{RandParam}(\mathbf{s})$: Let $m \triangleq (n+1) \cdot (\log q + O(1))$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and $\mathbf{e} \xleftarrow{\$} \chi^m$. Compute $\mathbf{b} := [\mathbf{A} \cdot \mathbf{s} + \mathbf{e}]_q$, and define
$$\mathbf{P}_{\mathsf{rand}} := [\mathbf{b} \| - \mathbf{A}] \in \mathbb{Z}_q^{m \times (n+1)} \ .$$
Output $\mathbf{P}_{\mathsf{rand}}$.

  We note that this is identical to the public key generation process in Regev's encryption scheme.

- $\mathsf{Rand}(\mathbf{P}_{\mathsf{rand}}, \mathbf{c})$: For $\mathbf{c} \in \mathbb{Z}_q^{n+1}$, sample $\mathbf{r} \xleftarrow{\$} \{0, 1\}^m$, and compute
$$\mathbf{c}_{\mathsf{rand}} := \left[\mathbf{c} + \mathbf{r}^T \mathbf{P}_{\mathsf{rand}}\right]_q \ .$$
Output $\mathbf{c}_{\mathsf{rand}}$.

The properties of this process are summarized below.

First, we state the security of our procedure, namely that $\mathbf{P}_{\mathsf{rand}}$ does not reveal information about $\mathbf{s}$. The proof is straightforward and omitted.

Lemma 2.9. *If $\mathbf{s}$ is uniformly sampled and $\mathbf{P}_{\mathsf{rand}} \leftarrow \mathsf{RandParam}(\mathbf{s})$, then under the $\mathsf{DLWE}_{n,q,\alpha}$ assumption, $\mathbf{P}_{\mathsf{rand}}$ is computationally indistinguishable from uniform.*

Next, we state that the inner product of the randomized vector with $(1, \mathbf{s})$ does not change by much.

Lemma 2.10. *Let* $\mathbf{s} \in \mathbb{Z}_q^n$ *be arbitrary, and let* $\mathbf{P}_{\mathsf{rand}} \leftarrow \mathsf{RandParam}(\mathbf{s})$*. Let* $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ *be arbitrary, and* $\mathbf{c}_{\mathsf{rand}} \leftarrow \mathsf{Rand}(\mathbf{P}_{\mathsf{rand}}, \mathbf{c})$*, then there exists* δ *such that*

$$\langle \mathbf{c}, (1, \mathbf{s}) \rangle - \langle \mathbf{c}_{\mathsf{rand}}, (1, \mathbf{s}) \rangle = \delta \pmod{q} \ ,$$

and $|\delta| \le \widetilde{O}_\kappa(\sqrt{n \log(q)}) \cdot \alpha q$ *with all but* $\text{negl}(\kappa)$ *probability.*

Proof. We start by noting that

$$\langle \mathbf{r}^T \mathbf{P}_{\mathsf{rand}}, (1, \mathbf{s}) \rangle = \langle \mathbf{r}, \mathbf{e} \rangle \pmod{q} \ ,$$

where $\mathbf{e}$ is the noise used to generate $\mathbf{P}_{\mathsf{rand}}$. Using Proposition 2.3, the result follows. □

Finally, we state the randomization property of our procedure.

Lemma 2.11. *Let* $q \le 2^n$*. Let* $\mathbf{s} \in \mathbb{Z}_q^n$ *be arbitrary, and let* $\mathbf{P}_{\mathsf{rand}} \leftarrow \mathsf{RandParam}(\mathbf{s})$*. Let* $\mathbf{f} \overset{\$}{\leftarrow} D_{\mathbb{Z},t}^{(n+1)\cdot\lceil \log(q) \rceil}$ *for some* t*, and let* $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ *be arbitrary (possibly dependent on* $\mathbf{f}$*). Finally, let* $\mathbf{c}_{\mathsf{rand}} \leftarrow \mathsf{Rand}(\mathbf{P}_{\mathsf{rand}}, \mathbf{c})$*, then*

$$\left| \langle \mathsf{BitDecomp}_q(\mathbf{c}_{\mathsf{rand}}), \mathbf{f} \rangle \right| \le \widetilde{O}_\kappa(\sqrt{n \log(q)}) \cdot t \ ,$$

with all but $\text{negl}(\kappa)$ *probability.*

Proof. By the leftover hash lemma, the last n coordinates of $\mathbf{c}_{\mathsf{rand}}$ are distributed uniformly, and independently of $\mathbf{f}, \mathbf{c}$. By Proposition 2.3, this part of $\mathbf{c}_{\mathsf{rand}}$ contributes $\widetilde{O}_\kappa(\sqrt{n \log(q)}) \cdot t$ to the inner product (with all but negligible probability).

The first coordinate of $\mathbf{c}_{\mathsf{rand}}$ may have dependence on $\mathbf{f}$, but it only decomposes to $O(\log q)$ bits, and therefore by Proposition 2.4, its contribution to the inner product is at most $\widetilde{O}_\kappa(\log(q)) \cdot t$ with all but negligible probability. Recalling that $q \le 2^n$, this is at most $\widetilde{O}_\kappa(\sqrt{n \log(q)}) \cdot t$.

The union bound completes the proof. □

2.5 Homomorphic Encryption and Bootstrapping

We now define homomorphic encryption and introduce Gentry's bootstrapping theorem. Our definitions are mostly taken from [BV11, BGV12].

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 (κ is the security parameter):

- **Key generation** $(pk, evk, sk) \leftarrow \mathsf{HE.Keygen}(1^\kappa)$**:** Outputs a public encryption key pk, a public evaluation key evk and a secret decryption key sk.[2]

- **Encryption** $c \leftarrow \mathsf{HE.Enc}_{pk}(\mu)$**:** Using the public key pk, encrypts a single bit message $\mu \in \{0, 1\}$ into a ciphertext c.

- **Decryption** $\mu \leftarrow \mathsf{HE.Dec}_{sk}(c)$**:** Using the secret key sk, decrypts a ciphertext c to recover the message $\mu \in \{0, 1\}$.

- **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.

[2]We adopt the terminology of [BV11] that treats the evaluation key as a separate entity from the public key.

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 2.12 (compactness and full homomorphism)**.** *A homomorphic encryption scheme is compact if its decryption circuit is independent of the evaluated function. A compact scheme is (pure) fully homomorphic if it can evaluate any efficiently computable function. The scheme is leveled fully homomorphic if it takes 1^L as additional input in key generation, and can only evaluate depth L Boolean circuits.*

Gentry's bootstrapping theorem shows how to go from limited amount of homomorphism to full homomorphism. This method has to do with the *augmented decryption circuit.*

Definition 2.13. *Consider a homomorphic encryption scheme* HE*. Let (sk, pk, evk) be properly generated keys and let $\mathcal{C}$ be the set of properly decryptable ciphertexts. Then the set of augmented decryption functions, $\{f_{c_1,c_2}\}_{c_1,c_2\in\mathcal{C}}$ is defined by $f_{c_1,c_2}(x) = \overline{\mathsf{HE.Dec}_x(c_1) \wedge \mathsf{HE.Dec}_x(c_2)}$. Namely, the function that uses its input as secret key, decrypts c_1, c_2 and returns the NAND of the results.*

The bootstrapping theorem is thus as follows.

Theorem 2.14 (bootstrapping [Gen09b, Gen09a])**.** *A scheme that can homomorphically evaluate its family of augmented decryption circuits can be transformed into a leveled fully homomorphic encryption scheme with the same decryption circuit, ciphertext space and public key.*

Furthermore, if the aforementioned scheme is also weak circular secure (remains secure even against an adversary who gets encryptions of the bits of the secret key), then it can be made into a pure fully homomorphic encryption scheme.

3 Our FHE Scheme

In this section, we describe an FHE scheme secure under a polynomial LWE assumption which, using known reductions [Reg05, Pei09], translates to the worst-case hardness of solving various lattice problems to within polynomial approximation factors. We start with the basic encryption scheme in Section 3.1, and describe "proto-homomorphic" addition and multiplication subroutines in Section 3.2. Departing from the "conventional wisdom" in FHE, our circuit evaluation procedure in Section 3.3 will *not* be a naive combination of these proto-homomorphic operations, but rather a carefully designed procedure that manages the noise growth effectively.

Finally, in Section 3.4, we put this all together to get our FHE scheme under the decisional LWE assumption $\mathsf{DLWE}_{n,q,\alpha}$ with $\alpha = n^{-c}$ for some constant $c > 0$. This polynomial factor is rather large: thus, in Section 4, we apply a carefully designed variant of the dimension-modulus reduction procedure of [BV11] to obtain our final FHE scheme that is secure under the hardness of $\mathsf{DLWE}_{n,q,\alpha}$ with $\alpha \leq 1/\tilde{O}_\kappa(n^\epsilon \cdot \sqrt{n \log(q)})$ which is weakest LWE hardness assumption that underlies the (non-homomorphic) lattice-based PKE schemes [AD97, Reg04, Reg05, Pei09, BLP+13].

3.1 The Basic Encryption Scheme

Our basic encryption scheme closely follows the Gentry-Sahai-Waters FHE scheme [GSW13]. We refer the reader to Section 2.3 for the description of the vector decomposition routines $\mathsf{PowersOfTwo}$, $\mathsf{BitDecomp}$ and $\mathsf{Flatten}$ used in the scheme below.

System Parameters. Let n be the LWE dimension and q be an LWE modulus. Define $N := (n+1) \cdot \lceil \log q \rceil$. Let d denote the maximal homomorphism depth that is allowed by the scheme. Let χ be an error distribution over $\mathbb{Z}$. Typically χ will be a discrete Gaussian $D_{\mathbb{Z},\alpha q}$ for $\alpha = 1/\widetilde{O}_\kappa(\sqrt{n \log(q)} \cdot 4^d)$. Recall that we identify $\mathbb{Z}_q$ with the set $(-q/2, q/2] \cap \mathbb{Z}$.

- $\mathsf{NCCrypt.Keygen}(1^n, q, 4^d)$: Sample a vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. Let $m \triangleq (n+1) \cdot (\log q + O(1))$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and $\mathbf{e} \xleftarrow{\$} \chi^m$. Compute $\mathbf{b} := [\mathbf{A} \cdot \mathbf{s} + \mathbf{e}]_q$, and define

$$\mathbf{P} := [\mathbf{b} \| -\mathbf{A}] \in \mathbb{Z}_q^{m \times (n+1)} \ .$$

  Output $sk = \mathbf{s}$ and $pk = evk = \mathbf{P}$.

  We describe public-key as well as secret-key encryption algorithms. Looking ahead, we remark that a secret-key encryption of μ is somewhat less "noisy" than a public-key encryption of μ.

- $\mathsf{NCCrypt.PubEnc}(pk, \mu)$: To encrypt a bit $\mu \in \{0,1\}$, using the public key $pk = \mathbf{P}$, we let $\mathbf{R} \xleftarrow{\$} \{0,1\}^{N \times m}$, and output the ciphertext

$$\mathbf{C} = \mathsf{Flatten}\left(\mathsf{BitDecomp}(\mathbf{R} \cdot \mathbf{P}) + \mu \cdot \mathbf{I}\right) \in \{0,1\}^{N \times N} \ .$$

- $\mathsf{NCCrypt.SecEnc}(sk, \mu)$: A symmetric encryption of a bit $\mu \in \{0,1\}$, using the secret key $sk = \mathbf{s}$, is performed by sampling $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{N \times n}$ and $\mathbf{e} \xleftarrow{\$} \chi^N$, computing $\mathbf{b} := [\mathbf{A} \cdot \mathbf{s} + \mathbf{e}]_q$, and defining

$$\mathbf{C} = \mathsf{Flatten}\left(\mathsf{BitDecomp}\left([\mathbf{b} \| -\mathbf{A}]\right) + \mu \cdot \mathbf{I}\right) \in \{0,1\}^{N \times N} \ .$$

- $\mathsf{NCCrypt.Dec}(sk, \mathbf{C})$: Let $\mathbf{c}$ be the second row of $\mathbf{C}$. We use standard Regev decryption on $\mathbf{c}$. Namely, we output $\mu^* = 0$ if $\left| [\langle \mathbf{c}, \mathsf{PowersOfTwo}(1, \mathbf{s}) \rangle]_q \right| < q/8$, and $\mu^* = 1$ otherwise.

Correctness. In order to show correctness of this scheme, we analyze the noise magnitude of ciphertexts produced by both the public-key and secret-key encryption algorithms. As we will show shortly, for ciphertexts $\mathbf{C}$ produced by either encryption algorithm, we have

$$\mathbf{C} \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) = \mu \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) + \mathbf{e} \pmod{q}$$

for a noise vector $\mathbf{e}$ of "small magnitude". This motivates our definition of the noise in the ciphertext $\mathbf{C}$ with respect to a secret key vector $\mathbf{s}$ and a message μ as follows.

Definition 3.1. *For every $\mathbf{C} \in \{0,1\}^{N \times N}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mu \in \mathbb{Z}$, we define*

$$\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C}) \triangleq \left\| (\mathbf{C} - \mu \mathbf{I}) \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \pmod{q} \right\|_\infty$$

The significance of this definition is captured by the following claims. The first claim shows that any ciphertext with small noise is decrypted correctly.

Lemma 3.2. *For every $\mathbf{C} \in \{0,1\}^{N \times N}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mu \in \mathbb{Z}$ such that $\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C}) < q/8$,*

$$\mathsf{NCCrypt.Dec}(\mathbf{s}, \mathbf{C}) = \mu$$

Proof. Since $\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C}) < q/8$, we have

$$\mathbf{C} \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) = \boldsymbol{\eta} + \mu \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \pmod q$$

where $\|\boldsymbol{\eta}\|_\infty < q/8$. Thus, for the second row of $\mathbf{C}$ it holds that

$$\langle \mathbf{c}, \mathsf{PowersOfTwo}(1, \mathbf{s})\rangle = \eta + 2^{\lceil \log(q) \rceil - 2} \cdot \mu \pmod q$$

where $|\eta| < q/8$. Thus, when $\mu = 0$,

$$\left| [\langle \mathbf{c}, \mathsf{PowersOfTwo}(1, \mathbf{s})\rangle]_q \right| = |\eta| < q/8$$

When $\mu = 1$,

$$\left| [\langle \mathbf{c}, \mathsf{PowersOfTwo}(1, \mathbf{s})\rangle]_q \right| \geq q/4 - |\eta| \geq q/8$$

since $q/4 \leq 2^{\lceil \log(q) \rceil - 2} < q/2$. This shows correctness of decryption for ciphertexts with small noise. $\square$

The next claim demonstrates parameter settings for which the (public key and secret key) encryption algorithms produce ciphertexts with small noise.

Lemma 3.3. *Let n be the LWE dimension, q be the LWE modulus and $\chi = D_{\mathbb{Z},\alpha q}$ be the discrete Gaussian distribution. Then, for every $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mu \in \{0, 1\}$,*

- *for $\mathbf{C}_{\mathsf{pub}} \leftarrow \mathsf{NCCrypt.PubEnc}(pk, \mu)$, we have $\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C}_{\mathsf{pub}}) = \widetilde{O}_\kappa(\alpha q \cdot \sqrt{m})$.*
- *for $\mathbf{C}_{\mathsf{sec}} \leftarrow \mathsf{NCCrypt.SecEnc}(sk, \mu)$, we have $\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C}_{\mathsf{sec}}) = \widetilde{O}_\kappa(\alpha q)$.*

with all but negligible (in κ) probability over the coins of $\mathsf{NCCrypt.Keygen}$.

In particular, we have correctness of decryption for the public key encryption for $\alpha < 1/\widetilde{\Omega}_\kappa(\sqrt{m})$, and for the secret key encryption for $\alpha < 1/\widetilde{\Omega}_\kappa(1)$.

Proof. We first show the analysis for the public-key encryption.

$$\begin{aligned}
\mathbf{C}_{\mathsf{pub}} \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) &= \mathsf{Flatten}\Big(\mathsf{BitDecomp}(\mathbf{R} \cdot \mathbf{P}) + \mu \cdot \mathbf{I}\Big) \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \\
&= \Big(\mathsf{BitDecomp}(\mathbf{R} \cdot \mathbf{P}) + \mu \cdot \mathbf{I}\Big) \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \\
&= \mathbf{R} \cdot \mathbf{P} \cdot (1, \mathbf{s})^T + \mu \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \\
&= \mathbf{R} \cdot \mathbf{e} + \mu \cdot \mathsf{PowersOfTwo}(1, \mathbf{s})
\end{aligned}$$

Thus, by Proposition 2.3,

$$\mathsf{noise}(\mathbf{C}_{\mathsf{pub}}) = \|\mathbf{R} \cdot \mathbf{e}\|_\infty = \widetilde{O}_\kappa(\sqrt{m} \cdot \alpha q)$$

This is less than $q/8$ by the choice of $\alpha < 1/\widetilde{\Omega}_\kappa(\sqrt{m})$.

The analysis for the secret key encryption follows analogously, except that

$$\mathbf{C}_{\mathsf{sec}} \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) = \mathbf{e} + \mu \cdot \mathsf{PowersOfTwo}(1, \mathbf{s})$$

Thus,

$$\mathsf{noise}(\mathbf{C}_{\mathsf{sec}}) = \|\mathbf{e}\|_\infty = \widetilde{O}_\kappa(\alpha q)$$

which is less than $q/8$ by the choice of $\alpha < 1/\widetilde{\Omega}_\kappa(1)$. $\square$

Security. Semantic security of the scheme follows from the decisional LWE assumption $\mathsf{DLWE}_{n,q,\alpha}$, similarly to Regev's encryption scheme (see [Reg05, BV11, GSW13] for similar arguments).

Complexity of Decryption. Our decryption algorithm is essentially the same as the decryption algorithm in Regev's encryption scheme, the complexity of which has been thoroughly studied in the context of FHE. The following is an immediate corollary from [BV11, Lemma 4.1].

Proposition 3.4. *There exists a constant* c_{dec} *such that the decryption circuit of the scheme* $\mathsf{NCCrypt}$*, with parameters* n, q*, has depth at most* $c_{\mathsf{dec}} \cdot \log(n \log q)$*.*

3.2 Proto-Homomorphic Operations

We now describe proto-homomorphic addition and multiplication algorithms which will be used in Section 3.3 for homomorphic circuit evaluation. Departing from the "conventional wisdom" in FHE, our circuit evaluation procedure will *not* be a naive combination of homomorphic addition and multiplication, but a carefully designed procedure that manages the noise growth effectively. To further stress the fact that we do not intend for these procedures to be used independently, we call them proto-homomorphic operations.

Proto-Homomorphic Addition. This is a simple addition of the ciphertext matrices.

- $\mathsf{NCCrypt.ProtoAdd}(\mathbf{C}_1, \mathbf{C}_2)$: Output $\mathbf{C}_+ := \mathsf{Flatten}(\mathbf{C}_1 + \mathbf{C}_2)$.

Jumping ahead, we note that in our use of $\mathsf{NCCrypt.ProtoAdd}$ in Section 3.3, both $\mathbf{C}_1$ and $\mathbf{C}_2$ will be encryptions of bits, and at most one of them will be an encryption of 1. The following claim analyzes the noise growth in homomorphic addition.

Claim 3.4.1 (Noise Growth in $\mathsf{NCCrypt.ProtoAdd.}$)**.** *For every* $\mathbf{s} \in \mathbb{Z}_q^n$*,* $\mu_1, \mu_2 \in \mathbb{Z}$ *and* $\mathbf{C}_1, \mathbf{C}_2 \in \{0,1\}^{N \times N}$*, we have*

$$\mathsf{noise}_{\mathbf{s},\mu_1+\mu_2}(\mathsf{NCCrypt.ProtoAdd}(\mathbf{C}_1, \mathbf{C}_2)) \leq \mathsf{noise}_{\mathbf{s},\mu_1}(\mathbf{C}_1) + \mathsf{noise}_{\mathbf{s},\mu_2}(\mathbf{C}_2)$$

Proof. Let $\mathbf{C}_+ \leftarrow \mathsf{NCCrypt.ProtoAdd}(\mathbf{C}_1, \mathbf{C}_2)$. We note that

$$\begin{aligned} \mathbf{C}_+ \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) &= \mathsf{Flatten}(\mathbf{C}_1 + \mathbf{C}_2) \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \\ &= (\mathbf{C}_1 + \mathbf{C}_2) \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \\ &= \mathbf{C}_1 \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) + \mathbf{C}_2 \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \\ &= (\mathbf{e}_1 + \mathbf{e}_2) + (\mu_1 + \mu_2) \cdot \mathbf{I} \end{aligned}$$

Thus, by the definition of $\mathsf{noise}_{\mathbf{s},\mu_1+\mu_2}$, we have

$$\mathsf{noise}_{\mathbf{s},\mu_1+\mu_2}(\mathbf{C}_+) \leq \mathsf{noise}_{\mathbf{s},\mu_1}(\mathbf{C}_1) + \mathsf{noise}_{\mathbf{s},\mu_2}(\mathbf{C}_2)$$

□

Homomorphic Multiplication. This is essentially a multiplication of the ciphertext matrices, except that we randomize the first ciphertext.

- $\mathsf{NCCrypt.ProtoMult}(evk, \mathbf{C}_1, \mathbf{C}_2)$:
  - Randomize $\mathbf{C}_1 \in \{0,1\}^{N \times N}$ into a matrix $\widetilde{\mathbf{C}}_1 \in \{0,1\}^{N \times N}$ by replacing each row $\mathbf{c}$ in $\mathbf{C}_1$ by the row
  $$\widetilde{\mathbf{c}} \leftarrow \mathsf{BitDecomp}(\mathsf{Rand}(pk, \mathsf{Combine}(\mathbf{c})))$$
  where Rand is the LWE randomization procedure from Section 2.4.
  - Output $\mathbf{C}_\times \leftarrow \mathsf{Flatten}\big(\widetilde{\mathbf{C}}_1 \cdot \mathbf{C}_2\big)$.

Jumping ahead, we remark that when we use $\mathsf{NCCrypt.ProtoMult}$ in our homomorphic circuit evaluation in Section 3.3, the first ciphertext will be an "evaluated ciphertext" (namely, a result of previous homomorphic evaluations), whereas the second ciphertext will be a "fresh ciphertext" (namely an output of the *secret key* encryption algorithm).

The first new idea in this work is that while the order of the arguments does not matter in homomorphic addition, the homomorphic multiplication algorithm $\mathsf{NCCrypt.ProtoMult}$ is inherently asymmetric, since it is essentially the (non-commutative) matrix multiplication operation. This asymmetry turns out to be the key to achieving improved noise growth, as Claim 3.4.2 below will demonstrate.

Claim 3.4.2 (Noise Growth in $\mathsf{NCCrypt.ProtoMult}$.)**.** *For every $\mathbf{s} \in \mathbb{Z}_q^n$, $\mu_1, \mu_2 \in \{0,1\}$ and $\mathbf{C}_1 \in \{0,1\}^{N \times N}$ and $\mathbf{C}_2 \leftarrow \mathsf{NCCrypt.SecEnc}(sk, \mu_2)$, we have*

$$\mathsf{noise}_{\mathbf{s},\mu_1\mu_2}(\mathsf{NCCrypt.ProtoMult}(\mathbf{C}_1, \mathbf{C}_2)) \leq |\mu_2| \cdot \mathsf{noise}_{\mathbf{s},\mu_1}(\mathbf{C}_1) + \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q})$$

with all but negligible probability over the randomness of $\mathsf{NCCrypt.Keygen}$*,* $\mathsf{NCCrypt.SecEnc}$ *and* $\mathsf{NCCrypt.ProtoMult}$*.*

Remark. In words, Claim 3.4.2 says that if $\mu_2 \in \{0,1\}$ (as will be the case in our homomorphic circuit evaluation in Section 3.3), the noise in $\mathbf{C}_\times$ is at most the noise in $\mathbf{C}_1$, plus a fixed additive term. What's more, if $\mu_2 = 0$, then the noise in $\mathbf{C}_\times$ is independent of that in $\mathbf{C}_1$! *These two facts are the key new ideas that enable our main result.*

Proof. (of Claim 3.4.2.) Let $\mathbf{C}_\times \leftarrow \mathsf{NCCrypt.ProtoMult}(evk, \mathbf{C}_1, \mathbf{C}_2)$. Note that

$$\begin{aligned}
\mathbf{C}_\times \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) &= \mathsf{Flatten}(\widetilde{\mathbf{C}}_1 \cdot \mathbf{C}_2) \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \\
&= \widetilde{\mathbf{C}}_1 \cdot (\mathbf{C}_2 \cdot \mathsf{PowersOfTwo}(1, \mathbf{s})) \\
&= \widetilde{\mathbf{C}}_1 \cdot (\mathbf{e}_2 + \mu_2 \cdot \mathsf{PowersOfTwo}(1, \mathbf{s})) \\
&= \widetilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 + \mu_2 \cdot \widetilde{\mathbf{C}}_1 \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \\
&= \widetilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 + \mu_2 \cdot (\widetilde{\mathbf{e}}_1 + \mu_1 \cdot \mathsf{PowersOfTwo}(1, \mathbf{s})) \\
&= (\widetilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 + \mu_2 \cdot \widetilde{\mathbf{e}}_1) + \mu_1\mu_2 \cdot \mathsf{PowersOfTwo}(1, \mathbf{s}) \qquad (1)
\end{aligned}$$

Since $\mathbf{e}_2 \leftarrow D_{\mathbb{Z},\alpha q}^N$ and each row of $\widetilde{\mathbf{C}}_1 \in \{0,1\}^{N \times N}$ is the result of Rand, by Lemma 2.11, we have

$$\left\| \widetilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 \right\|_\infty \leq \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) \qquad (2)$$

with all but negligible probability.
Also, by lemma 2.10, we have

$$\|\widetilde{\mathbf{e}}_1\|_\infty \le \|\mathbf{e}_1\|_\infty + \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) \tag{3}$$

with all but negligible probability.

Putting together Eq. (1),(2) and (3), we have

$$\mathsf{noise}_{\mathbf{s},\mu_1\mu_2}(\mathbf{C}_\times) \le \left\|\widetilde{\mathbf{C}}_1 \cdot \mathbf{e}_2\right\|_\infty + |\mu_2| \cdot \|\widetilde{\mathbf{e}}_1\|_\infty \le |\mu_2| \cdot \mathsf{noise}_{\mathbf{s},\mu_1}(\mathbf{C}_1) + \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q})$$

which finishes the proof. □

3.3 Homomorphic Evaluation of Circuits

We now describe how to homomorphically evaluate a Boolean circuit Ψ with two-input NAND gates that takes ℓ inputs, and has depth d. In particular, our scheme will be able to evaluate circuits of depth $c \cdot \log n$ (for any constant c) under a polynomial LWE assumption, namely $\mathsf{DLWE}_{n,q,\chi}$ where $\chi = D_{\mathbb{Z},\alpha}$ and $\alpha = 1/n^{\Theta(c)}$. Since the depth of the decryption circuit is $c_{\mathsf{dec}} \cdot \log(n \log(q)) \le 2c_{\mathsf{dec}} \cdot \log(n)$ (for some constant $c_{\mathsf{dec}} > 0$), the scheme is bootstrappable, and by the bootstrapping theorem (Theorem 2.14), we get a leveled FHE scheme under the same assumption.

To evaluate a circuit, our scheme first turns it into a width-5 permutation branching program [BDFP86, Bar89], a model of computation that we describe below.

Width-5 Permutation Branching Programs. A permutation branching program Π of length L with input space $\{0,1\}^\ell$ is a sequence of L tuples of the form $\big(\mathsf{var}(t), \sigma_{t,0}, \sigma_{t,1}\big)$ where

- $\mathsf{var} : [L] \to [\ell]$ is a function that associates the t-th tuple with an input bit $x_{\mathsf{var}(t)}$.
- $\sigma_{j,0}$ and $\sigma_{j,1}$ are permutations on 5 elements. We will think of $\sigma_{j,0}$ and $\sigma_{j,1}$ as bijective functions from the set $\{1,2,3,4,5\}$ to itself.

The computation of the program Π on input $\mathbf{x} = (x_1, \ldots, x_\ell)$ proceeds as follows. The state of the computation at any point in time t is a number $\zeta_t \in \{1,2,3,4,5\}$. Computation starts with the initial state $\zeta_0 = 1$. The state ζ_t is computed recursively as

$$\zeta_t = \sigma_{t,\mathsf{var}(t)}(\zeta_{t-1})$$

Finally, after L steps, our state is ζ_L. The output of the computation is 1 if $\zeta_L = 1$, and 0 otherwise.

To manage the growth of noise in homomorphic evaluation, we need to work with bits rather than numbers. Thus, we prefer to represent the state $\zeta_t \in \{1,2,3,4,5\}$ by a 0-1 vector $\mathbf{v}_t$ which is the unit vector $\mathbf{u}_{\zeta_t}$ in 5 dimensions.

The computation then proceeds as follows. The idea is that $\mathbf{v}_t[i] = 1$ if and only if $\sigma_{t,\mathsf{var}(t)}(\zeta_{t-1}) = i$. Turning this around, $\mathbf{v}_t[i] = 1$ if and only if either:

- $\mathbf{v}_{t-1}[\sigma_{t,0}^{-1}(i)] = 1$ and $x_{\mathsf{var}(t)} = 0$; or
- $\mathbf{v}_{t-1}[\sigma_{t,1}^{-1}(i)] = 1$ and $x_{\mathsf{var}(t)} = 1$.

The following formula captures this condition. For $t = 1, \ldots, L$, and $i \in \{1, 2, 3, 4, 5\}$, we have:

$$\begin{aligned} \mathbf{v}_t[i] &:= \mathbf{v}_{t-1}[\sigma_{t,0}^{-1}(i)] \cdot (1 - x_{\mathsf{var}(t)}) + \mathbf{v}_{t-1}[\sigma_{t,1}^{-1}(i)] \cdot x_{\mathsf{var}(t)} \\ &= \mathbf{v}_{t-1}[\gamma_{t,i,0}] \cdot (1 - x_{\mathsf{var}(t)}) + \mathbf{v}_{t-1}[\gamma_{t,i,1}] \cdot x_{\mathsf{var}(t)} \end{aligned} \tag{4}$$

where $\gamma_{t,i,0} \triangleq \sigma_{t,0}^{-1}(i)$ and $\gamma_{t,i,1} \triangleq \sigma_{t,1}^{-1}(i)$ are constants that are publicly computable given the description of the branching program. It is this form that we will work with in our homomorphic evaluation.

The important property that we will use is that circuits of depth d can be simulated by branching programs of depth $L = 4^d$.

Theorem 3.5 (Barrington's Theorem [Bar89])**.** *Every Boolean NAND circuit Ψ that acts on ℓ inputs and has depth d can be computed by a width-5 permutation branching program Π of length 4^d. Given the description of the circuit Ψ, the description of the branching program Π can be computed in* $\mathrm{poly}(\ell, 4^d)$ *time.*

Homomorphic Evaluation $\mathsf{NCCrypt.Eval}(\Psi, \mathbf{C}_1, \ldots, \mathbf{C}_\ell)$**.** The homomorphic evaluation procedure will first convert the depth-d circuit Ψ into a width-5 permutation branching program Π of length $L = 4^d$.

- **[Initialization]** We will maintain the encrypted state of the computation of the branching program for every step t. We denote this by $\mathbf{V}_t = (\mathbf{V}_{t,1}, \mathbf{V}_{t,2}, \mathbf{V}_{t,3}, \mathbf{V}_{t,4}, \mathbf{V}_{t,5})$, where each $\mathbf{V}_t[i] \in \{0,1\}^{N \times N}$ will be an encryption of $\mathbf{v}_t[i]$.
  - We initialize the state as follows. Compute $\mathbf{V}_{0,i} := \mathbf{v}_0[i] \cdot \mathbf{I}$.
    Note that $\mathbf{V}_{0,i}$ is in fact a valid encryption of the bit $\mathbf{v}_0[i]$ with zero noise.
  - We also compute encryptions of the complements of the input bits, for convenience. That is, set $\bar{\mathbf{C}}_k := \mathbf{I} - \mathbf{C}_k$. Note that $\bar{\mathbf{C}}_k$ is an encryption of $\bar{x}_k = 1 - x_k$ with the same noise as $\mathbf{C}_k$.
- **[Evaluation]** The evaluation proceeds iteratively for $t = 1, \ldots, L$, where L is the length of the branching program Π. Assuming that we have $\mathbf{V}_{t-1} := (\mathbf{V}_{t-1,1}, \mathbf{V}_{t-1,2}, \ldots, \mathbf{V}_{t-1,5})$, the encryption of the state of the branching program computation at time $t-1$, we compute $\mathbf{V}_t := (\mathbf{V}_{t,1}, \mathbf{V}_{t,2}, \ldots, \mathbf{V}_{t,5})$ by homomorphically evaluating Eq. (4) above.

  That is, for $i \in \{1, 2, 3, 4, 5\}$, we compute

$$\mathbf{V}_{t,i} := \mathsf{NCCrypt.ProtoAdd}\Big(\mathsf{NCCrypt.ProtoMult}\big(\mathbf{V}_{t-1,\gamma_0}, \bar{\mathbf{C}}_{\mathsf{var}(t)}\big), \tag{5}$$

$$\mathsf{NCCrypt.ProtoMult}\big(\mathbf{V}_{t-1,\gamma_1}, \mathbf{C}_{\mathsf{var}(t)}\big)\Big) \tag{6}$$

- **[Output]** Upon finishing the evaluation stage, we have $\mathbf{V}_L := (\mathbf{V}_{L,1}, \mathbf{V}_{L,2}, \ldots, \mathbf{V}_{L,5})$. Output $\mathbf{V}_{L,1}$ as the result of the homomorphic evaluation.

We now show that the scheme correctly evaluates circuits.

Lemma 3.6 (Correctness of Homomorphic Evaluation). *Let n be the LWE dimension, q the LWE modulus, Ψ be any Boolean circuit of depth d, and*

$$\alpha \leq 1/\widetilde{\Theta}_\kappa(4^d \cdot \sqrt{n \log q})$$

For every $x_1, \ldots, x_\ell \in \{0,1\}$, every Boolean circuit Ψ of depth at most d, and every secret key sk, letting $\mathbf{C}_k \leftarrow \mathsf{NCCrypt.SecEnc}(sk, x_k)$ be the secret key *encryptions of the inputs, and $\mathbf{C}_\Psi \leftarrow \mathsf{NCCrypt.Eval}(evk, \Psi, \mathbf{C}_1, \ldots, \mathbf{C}_k)$ be the evaluated ciphertext, we have:*

$$\mathsf{NCCrypt.Dec}(sk, \mathbf{C}_\Psi) = \Psi(x_1, \ldots, x_\ell)$$

with overwhelming probability over the coin tosses of all the algorithms. $\mathsf{NCCrypt.Eval}$ *runs in time* $\mathrm{poly}(4^d, \ell, n, \log q)$.

Note that we stated the correctness of homomorphic evaluation on ciphertexts produced by the secret-key encryption algorithm $\mathsf{NCCrypt.SecEnc}$. A similar lemma can be shown in the case of public-key encryption, if α is smaller by a factor of $\sqrt{n \log q}$. However, in our "optimal FHE" scheme in Section 4, we will only need to invoke this lemma with secret-key encryption.

Proof. It is easy to see that each step of the homomorphic evaluation algorithm, given by Eq. (5), simulates the execution of the branching program, given by Eq. (4). It remains to bound the noise growth during $\mathsf{NCCrypt.Eval}$. We show this by induction.

In the sequel, we will abbreviate the noise function $\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C})$ to $\mathsf{noise}(\mathbf{C})$ since the secret key is fixed throughout the evaluation, and the message μ is clear from the context.

Clearly, $\mathsf{noise}(\mathbf{V}_{0,i}) = 0$, since they $\mathbf{V}_{0,i}$ are just the messages, with no noise. Assume, as the inductive hypothesis, that for all $i \in \{1,2,3,4,5\}$,

$$\mathsf{noise}(\mathbf{V}_{t-1,i}) = (t-1) \cdot \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q})$$

We will now bound $\mathsf{noise}(\mathbf{V}_{t,i})$ for all $i \in \{1,2,3,4,5\}$. Note that

$$\begin{aligned}\mathsf{noise}\big(\mathbf{V}_{t,i}\big) &\leq \mathsf{noise}\Big(\mathsf{NCCrypt.ProtoMult}\big(\mathbf{V}_{t-1,\gamma_0}, \bar{\mathbf{C}}_{\mathsf{var}(t)}\big)\Big) + \mathsf{noise}\Big(\mathsf{NCCrypt.ProtoMult}\big(\mathbf{V}_{t-1,\gamma_1}, \mathbf{C}_{\mathsf{var}(t)}\big)\Big) \\ &\leq |1 - x_{\mathsf{var}(t)}| \cdot \mathsf{noise}(\mathbf{V}_{t-1,\gamma_0}) + |x_{\mathsf{var}(t)}| \cdot \mathsf{noise}(\mathbf{V}_{t-1,\gamma_1}) + \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q})\end{aligned}$$

where the second inequality holds by Claim 3.4.2 since all the ciphertexts encrypt bits, $\mathbf{C}_{\mathsf{var}(t)}$ is a fresh secret-key encryption, and $\bar{\mathbf{C}}_{\mathsf{var}(t)}$ contains exactly the same noise as $\mathbf{C}_{\mathsf{var}(t)}$.

Since exactly one of $x_{\mathsf{var}(t)}$ and $1 - x_{\mathsf{var}(t)}$ is non-zero, we have

$$\begin{aligned}\mathsf{noise}\big(\mathbf{V}_{t,i}\big) &\leq \max(\mathsf{noise}(\mathbf{V}_{t-1,\gamma_0}), \mathsf{noise}(\mathbf{V}_{t-1,\gamma_1})) + \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) \\ &\leq t \cdot \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q})\end{aligned}$$

by the inductive hypothesis.

Thus, in particular,

$$\mathsf{noise}(\mathbf{V}_\Psi) = \mathsf{noise}\big(\mathbf{V}_{L,1}\big) \leq 4^d \cdot \widetilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) < q/8$$

by our setting of the parameter α. □

3.4 Achieving Fully Homomorphic Encryption

We know by Lemma 3.4 that the depth of the decryption circuit of $\mathsf{NCCrypt}$ is $c_{\mathsf{dec}} \cdot \log(n \log q) = c_{\mathsf{dec}} \log N$ for some constant $c_{\mathsf{dec}} > 0$. Setting the depth $d = c \log N$ for some constant $c > c_{\mathsf{dec}}$ in Lemma 3.6 and $\alpha \leq 1/\widetilde{\Theta}_\kappa(4^d \cdot \sqrt{n \log q})$ gives us a bootstrappable encryption scheme. By the bootstrapping theorem (Theorem 2.14), this can be turned into a leveled FHE scheme, without additional assumptions. We state this theorem below:

Theorem 3.7. *Let n be the LWE dimension, q be the LWE modulus, $N := (n+1) \cdot \lceil \log q \rceil$, and let $c > c_{\mathsf{dec}}$ be a large enough constant (where c_{dec} is the decryption depth constant from Proposition 3.4), and*

$$\alpha \leq 1/\widetilde{\Theta}_\kappa((n \log q)^{2c+1/2})$$

Then, there is a leveled FHE scheme that is secure under the decisional LWE assumption $\mathsf{DLWE}_{n,q,\alpha}$.

In the next section, we will use a variant of the dimension-modulus reduction of [BV11], the effect of which will be to reduce the constant c above to a very small $\epsilon \to 0$, thus achieving a value of α that matches the best known lattice-based PKE schemes.

4 Successive Dimension-Modulus Reduction

In this section we revisit the dimension-modulus reduction technique from [BV11] and show that by successive application of this technique, we can achieve comparable lattice approximation factor to the best known factor for public key encryption.

We start by revisiting [BV11]'s dimension-modulus reduction in Section 4.1, and then proceed in Section 4.2 to present a bootstrappable homomorphic encryption scheme that is based on $\widetilde{O}(n^{1+\epsilon} \cdot \sqrt{n \log(q)})$-approximate GapSVP.

4.1 Dimension-Modulus Reduction (Revisited)

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

- $\mathsf{SwitchKeyGen}_{q:p,\chi}(\mathbf{s}, \mathbf{t})$: 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 $\mathbf{A}_{\mathbf{s:t}} \xleftarrow{\$} \mathbb{Z}_p^{\hat{n}_s \times n_t}$ and a noise vector $\mathbf{e}_{\mathbf{s:t}} \xleftarrow{\$} \chi^{\hat{n}_s}$. The function's output is a matrix

  $$\mathbf{P}_{\mathbf{s:t}} = [\mathbf{b}_{\mathbf{s:t}} \| -\mathbf{A}_{\mathbf{s:t}}] \in \mathbb{Z}_p^{\hat{n}_s \times (n_t+1)} ,$$

  where

  $$\mathbf{b}_{\mathbf{s:t}} := \Big[\mathbf{A}_{\mathbf{s:t}} \cdot \mathbf{t} + \mathbf{e}_{\mathbf{s:t}} + \lfloor (p/q) \cdot \mathsf{PowersOfTwo}_q(\mathbf{s}) \rceil_G \Big]_p \in \mathbb{Z}_p^{\hat{n}_s} .$$

  Here, $\lfloor \cdot \rceil_G$ is the Gaussian rounding procedure from Corollary 2.1.

- $\mathsf{SwitchKey}_q(\mathbf{P}_{\mathbf{s:t}}, \mathbf{c}_s)$: To switch a source ciphertext $\mathbf{c}_s \in \mathbb{Z}_q^{n_s}$ from a secret key $\mathbf{s}$ to $(1, \mathbf{t})$, output

  $$\mathbf{c}_t := \left[\mathbf{P}_{\mathbf{s:t}}^T \cdot \mathsf{BitDecomp}_q(\mathbf{c}_s)\right]_p \in \mathbb{Z}_p^{n_t+1} .$$

Lemma 4.1 (correctness)**.** *Let* $\mathbf{s} \in \mathbb{Z}^n$, $\mathbf{t} \in \mathbb{Z}^k$ *be some vectors. Let* χ *be the discrete Gaussian* $D_{\mathbb{Z},\alpha p}$*, and let* $\mathbf{P}_{\mathsf{s:t}} \leftarrow \mathsf{SwitchKeyGen}_{q:p,\chi}(\mathbf{s}, \mathbf{t})$ *and* $\mathbf{P}_{\mathsf{rand}} \leftarrow \mathsf{RandParam}_{D_{\mathbb{Z},\beta q}}(\mathbf{s})$*. Let* $\mathbf{c}_s \in \mathbb{Z}_q^n$ *and let* $\mathbf{c}'_s \leftarrow \mathsf{Rand}(\mathbf{P}_{\mathsf{rand}}, \mathbf{c}_s)$*. Finally, set* $\mathbf{c}_t \leftarrow \mathsf{SwitchKey}(\mathbf{P}_{\mathsf{s:t}}, \mathbf{c}'_s)$*. Then there exists* δ *such that*

$$(p/q) \cdot \langle \mathbf{c}_s, \mathbf{s} \rangle - \delta = \langle \mathbf{c}_t, (1, \mathbf{t}) \rangle \pmod p ,$$

and $|\delta| < \widetilde{O}_\kappa(\sqrt{n \log(q)}) \cdot \alpha p$ *with all but* $\operatorname{negl}(\kappa)$ *probability (over the coins in the experiment, and regardless of the generation of* $\mathbf{s}, \mathbf{t}, \mathbf{c}_s$*).*

Proof. We expand the expression for $\langle \mathbf{c}_t, (1, \mathbf{t}) \rangle$:

$$\begin{aligned}\langle \mathbf{c}_t, (1, \mathbf{t}) \rangle &= \langle \mathbf{P}_{\mathsf{s:t}}^T \cdot \mathsf{BitDecomp}_q(\mathbf{c}'_s), (1, \mathbf{t}) \rangle \\ &= \langle \mathsf{BitDecomp}_q(\mathbf{c}'_s), \mathbf{P}_{\mathsf{s:t}} \cdot (1, \mathbf{t}) \rangle \\ &= \langle \mathsf{BitDecomp}_q(\mathbf{c}'_s), \mathbf{e}_{\mathsf{s:t}} + \lfloor (p/q) \cdot \mathsf{PowersOfTwo}_q(\mathbf{s}) \rceil_G \rangle .\end{aligned}$$

It follows that $\delta = \delta_1 + \delta_2$ where

$$\delta_1 = \langle \mathsf{BitDecomp}_q(\mathbf{c}'_s), \mathbf{e}_{\mathsf{s:t}} \rangle ,$$

which, by Lemma 2.11, is bounded by $|\delta_1| \le \widetilde{O}_\kappa(\sqrt{n \log(q)})\alpha p$ with all but $\operatorname{negl}(\kappa)$ probability; and

$$\begin{aligned}\delta_2 &= \left\langle \mathsf{BitDecomp}_q(\mathbf{c}'_s), \lfloor (p/q) \cdot \mathsf{PowersOfTwo}_q(\mathbf{s}) \rceil_G \right\rangle - (p/q) \cdot \langle \mathbf{c}'_s, \mathbf{s} \rangle \\ &= \left\langle \mathsf{BitDecomp}_q(\mathbf{c}'_s), \lfloor (p/q) \cdot \mathsf{PowersOfTwo}_q(\mathbf{s}) \rceil_G - (p/q) \cdot \mathsf{PowersOfTwo}_q(\mathbf{s}) \right\rangle .\end{aligned}$$

Applying Lemma 2.11, we get that $|\delta_2| \le \widetilde{O}_\kappa(\sqrt{n \log(q)})$ with all but $\operatorname{negl}(\kappa)$ probability. □

Security follows in a straightforward manner, the proof is omitted.

Lemma 4.2 (security)**.** *Let* $\mathbf{s} \in \mathbb{Z}^{n_s}$ *be any vector. If we generate* $\mathbf{t} \xleftarrow{\$} \mathbb{Z}_p^k$ *and* $\mathbf{P} \leftarrow \mathsf{SwitchKeyGen}_{q:p,\chi}(\mathbf{s}, \mathbf{t})$*, then* $\mathbf{P}$ *is computationally indistinguishable from uniform over* $\mathbb{Z}_p^{\hat{n}_s \times (n_t+1)}$*, assuming the decisional LWE assumption* $\mathsf{DLWE}_{k,p,\chi}$*.*

4.2 A Bootstrappable Scheme

Let $q : \mathbb{N} \to \mathbb{N}$ be a monotone function such that $q(n) \le 2^n$ for all n, and $\alpha : \mathbb{N} \to \mathbb{R}$. Let $\chi(n)$ denote the discrete Gaussian distribution $D_{\mathbb{Z},\alpha(n)q(n)}$. Finally, let $\epsilon > 0$.

The typical value of $\alpha(n)$ will be $1/\widetilde{O}_\kappa(\sqrt{n \log(q)} \cdot n^\epsilon)$, where κ is the security parameter. As to the function $q(n)$, we will be interested in two ranges of parameters: In the first we will set $q(n)$ such that $\alpha(n) \cdot q(n) \approx \sqrt{n}$ (i.e. we set q such that $q(n) = \widetilde{O}_\kappa(n^{1+\epsilon}\sqrt{\log(q)})$. This is the minimal q that allows to apply worst-case to average-case reductions for LWE. The second case is where $q(n) = 2^{n/2}$, which is the minimal q that allows to apply *classical* worst-case to average case reductions to the GapSVP problem.

The scheme $\mathsf{DimReduced}$ is defined as follows.

- $\mathsf{DimReduced.Keygen}(1^k)$: Define $n \triangleq (k \log(q(k)))^{2c_{\mathsf{dec}}/\epsilon}$ (namely, $n^\epsilon = 4^{c_{\mathsf{dec}} \cdot \log(k \log(q(k)))}$), where c_{dec} is as in Proposition 3.4. Assume for convenience that $n = k^{(1+\epsilon)^L}$ for some $L \in \mathbb{N}$ (otherwise round n to the next power up). Further define $k_i = k^{(1+\epsilon)^i}$, so $k_0 = k$ and $k_L = n$, and define $q_i = q(k_i)$. For convenience we denote $p \triangleq q_0$.

  Let $(\mathsf{ncsk}, \mathsf{ncevk}, \mathsf{ncpk}) \leftarrow \mathsf{NCCrypt.Keygen}(1^n, q(n), 4^{c_{\mathsf{dec}} \cdot \log(k \log(q(k)))+1})$. Define $\mathbf{s}_L \triangleq \mathsf{ncsk} \in \mathbb{Z}_{q_L}^{n_L}$. For all $i = 0, \ldots, L-1$, sample $\mathbf{s}_i \xleftarrow{\$} \mathbb{Z}_{q_i}^{n_i}$.

  Next, we generate dimension-modulus switching parameters (see Section 4.1), and randomization parameters (see Section 2.4) for all $i \in [L]$:

$$\mathbf{P}_{i:(i-1)} \leftarrow \mathsf{SwitchKeyGen}_{q_i:q_{i-1}}((1, \mathbf{s}_i), \mathbf{s}_{i-1}) \ ,$$

  and

$$\mathbf{P}_{\mathsf{rand},i} \leftarrow \mathsf{RandParam}_{D_{\mathbb{Z},\alpha(k_i)q_i}}(\mathbf{s}_i) \ .$$

  Finally, output the keys $sk \triangleq (\mathbf{s}_0, \mathbf{s}_L)$, $pk \triangleq \mathsf{ncpk}$, $evk \triangleq (\mathsf{ncevk}, \{\mathbf{P}_{i:(i-1)}\}_{i \in [L]}, \{\mathbf{P}_{\mathsf{rand},i}\}_{i \in [L]})$.

  We note that as ϵ approaches 0, $n = k^{\Theta(1/\epsilon)}$ becomes larger. If k is proportional to the security parameter, then ϵ must be bounded by a constant to keep n polynomially bounded. We further note that $L = \Theta(\log(1/\epsilon)/\epsilon)$.

- $\mathsf{DimReduced.PubEnc}_{pk}(\mu)$ / $\mathsf{DimReduced.SecEnc}_{sk}(\mu)$: The asymmetric and symmetric encryption procedures are identical to $\mathsf{NCCrypt}$. Since $\mathsf{DimReduced}$'s public key and secret key contain those of $\mathsf{NCCrypt}$, this can be done in a straightforward manner.

- $\mathsf{DimReduced.Eval}_{evk}(f, \mathbf{C}_1, \ldots, \mathbf{C}_t)$: To perform homomorphic evaluation, we first compute

$$\mathbf{C}_f \leftarrow \mathsf{NCCrypt.Eval}_{\mathsf{ncevk}}(f, \mathbf{C}_1, \ldots, \mathbf{C}_t) \ .$$

  We then consider $\mathbf{c}_f \in \{0,1\}^{n\lceil \log(q_L) \rceil}$, which is the second row of $\mathbf{C}_f$. We set $\mathbf{c}_L := \mathsf{Combine}_{q_L}(\mathbf{c}_f)$.

  We then compute, in order for $i = L-1, \ldots, 0$, the ciphertexts $\mathbf{c}_{\mathsf{rand},i+1} \leftarrow \mathsf{Rand}(\mathbf{P}_{\mathsf{rand},i+1}, \mathbf{c}_{i+1})$, and then $\mathbf{c}_i \leftarrow \mathsf{SwitchKey}(\mathbf{P}_{(i+1):i}, \mathbf{c}_{\mathsf{rand},i+1})$. Finally, $\mathbf{c}_0 \in \mathbb{Z}_p^k$ is output as the final ciphertext.

- $\mathsf{DimReduced.Dec}_{sk}(\mathbf{c})$: We recall that $\mathbf{c} \in \mathbb{Z}_p^k$. We output $\mu^* = 0$ if $\left| [\langle \mathbf{c}, (1, \mathbf{s}_0) \rangle]_p \right| < p/8$, and $\mu^* = 1$ otherwise.

Security is stated in the next lemma and follows immediately using a hybrid argument and using the security properties of the scheme $\mathsf{NCCrypt}$, the ciphertext randomization procedure (Lemma 2.9) and the dimension-modulus reduction procedure (Lemma 4.2). The formal proof is omitted.

Lemma 4.3. *The scheme* $\mathsf{DimReduced}$ *is secure under the* $\mathsf{DLWE}_{k,q(k),\alpha(k)}$ *assumption.*

Correctness poses a more challenging task. We want to prove that $\mathsf{DimReduced}$ can homomorphically evaluate an augmented decryption circuit. Proving this when $\alpha(n)$ is small (e.g. $\alpha(n) = 1/n^3$) is fairly easy. However, since we wish to achieve optimal parameters, the analysis is more involved and appears in the following lemma.

We define $\tau \triangleq 2^{\lceil \log(q_L) \rceil - 2}/q_L$ and notice that $\tau \in (1/4, 1/2]$.

Lemma 4.4. *Let $\alpha(n) = 1/\widetilde{O}_\kappa(\sqrt{n\log(q)} \cdot n^\epsilon)$, and let $q(n) \geq \widetilde{O}(\sqrt{n}/\alpha(n))$. Consider a set of keys generated by $(sk, pk, evk) \leftarrow \mathsf{DimReduced.Keygen}(1^k)$, and recall that $sk = (\mathbf{s}_0, \mathbf{s}_L)$. Let $\mathbf{K}_i \leftarrow \mathsf{DimReduced.SecEnc}_{sk}(\mathsf{BitDecomp}(\mathbf{s}_0)[i])$. Namely, $\mathbf{K}_i$ is the symmetric encryption of the ith bit of $\mathbf{s}_0$.*

Let f be an augmented decryption circuit as per Definition 2.13. Namely, let $\mathbf{c}, \mathbf{c}' \in \mathbb{Z}_p^k$ and $\mu, \mu' \in \{0,1\}$ be such that

$$[\langle \mathbf{c}, \mathbf{s}_0 \rangle]_p = \mu \cdot \tau p + e \ ,$$

where $|e| < p/8$, and similarly for $\mathbf{c}'$. The function f is the function that on input x, treats x as a secret key and decrypts $\mathbf{c}, \mathbf{c}'$, and outputs the NAND of their decryptions.

Let

$$\mathbf{c}_0 \leftarrow \mathsf{DimReduced.Eval}_{evk}(f, \mathbf{K}_1, \mathbf{K}_2, \ldots) \ ,$$

and note that this is syntactically well defined since f can be represented as a Boolean circuit of depth $c_{\mathsf{dec}} \cdot \log(k\log(q(k))) + 1$.

Then with all but $\mathrm{negl}(\kappa)$ *probability,*

$$[\langle \mathbf{c}_0, \mathbf{s}_0 \rangle]_p = \mu^* \cdot \tau p + e_f \ ,$$

where $\mu^* = \overline{(\mu \wedge \mu')}$, $|e_f| < p/8$ are as above.

Proof. Consider the process of execution of $\mathsf{DimReduced.Eval}_{evk}(f, \mathbf{K}_1, \mathbf{K}_2, \ldots)$. It starts by generating $\mathbf{c}_L$, where we are guaranteed by the correctness of $\mathsf{NCCrypt}$ that

$$[\langle \mathbf{c}_L, \mathbf{s}_L \rangle]_{q_L} = \mu^* \cdot \tau q_L + e_L \ ,$$

where

$$|e_L|/q_L \leq \widetilde{O}_\kappa(\sqrt{n\log(q_L)} \cdot 4^{c_{\mathsf{dec}} \cdot \log(k\log(q(k)))+1})\alpha(n) = \widetilde{O}_\kappa(n^\epsilon \cdot \sqrt{n\log(q_L)}) \cdot \alpha(n) \ ,$$

and we will set $\alpha(n) = 1/(\sqrt{n\log(q)} \cdot n^\epsilon \mathrm{polylog}(\kappa))$ with sufficiently large polylogarithmic factor to offset the one coming from the noise so that

$$|e_L|/q_L \leq 1/\mathrm{polylog}(\kappa) \ .$$

We then commence with $L = O(1)$ levels of randomization followed by modulus-dimension reduction. Let us consider the effect of these operations at level i.

Lemma 2.10 guarantees that in the randomization step, the relative noise grows by an additive factor of at most $\widetilde{O}_\kappa(\sqrt{k_i\log(q_i)}) \cdot \alpha(k_i) = k_i^{-\epsilon} \cdot \widetilde{O}_\kappa(1)/\mathrm{polylog}(\kappa)$. Again, the idea is to define α with sufficiently large polylogarithmic factor to offset those coming from $\widetilde{O}_\kappa(\cdot)$.

Lemma 4.1 guarantees that in the key switching step, the relative noise grows by an additive factor of $\widetilde{O}_\kappa(\sqrt{k_i\log(q_i)}) \cdot \alpha(k_{i-1})$. We recall that $k_i = k_{i-1}^{1+\epsilon}$ and that $q(n) < 2^n$. Therefore

$$\sqrt{k_i\log(q_i)} \leq \sqrt{(k_{i-1}\log(q_{i-1}))^{1+\epsilon}} \leq \sqrt{(k_{i-1}\log(q_{i-1}))} \cdot k_{i-1}^\epsilon \ .$$

Therefore, setting the polylogarithmic factors right, we get an additive relative error of at most $1/\mathrm{polylog}(\kappa)$.

Putting all of these together, we get that

$$|e_f| \leq L/\mathrm{polylog}(\kappa) \ll p/8 \ ,$$

and the result follows. □

Finally, we derive the worst-case lattice approximation factor based on Corollary 2.6. We recall that a bootstrappable homomorphic encryption scheme implies a leveled FHE scheme under the same assumptions, and a pure FHE scheme with an additional circular security assumption.

Corollary 4.5. *For all $\epsilon > 0$, there exist:*

- *A bootstrappable homomorphic encryption scheme based on the worst-case* quantum *hardness of solving* $\mathsf{GapSVP}_{\widetilde{O}(n^{1.5+\epsilon})}$ *and* $\mathsf{SIVP}_{\widetilde{O}(n^{1.5+\epsilon})}$.
- *A bootstrappable homomorphic encryption scheme based on the worst-case* classical *hardness of solving* $\mathsf{GapSVP}_{\widetilde{O}(n^{2+\epsilon})}$.

The first (quantum) case is derived from Lemma 4.4 by setting $q(n) = \sqrt{n}/\alpha(n) = \mathrm{poly}(n)$, and the second (classical) case is derived by setting $q(n) = 2^{n/2}$.

Improving Key and Ciphertext sizes. The scheme DimReduced uses a ladder of LWE instances, ranging from short (k, p) to polynomially larger (n, q). In the description above, the public key of the scheme is derived from that of NCCrypt, and therefore depends on n and not on k. Likewise, the "input ciphertexts" (the ones before homomorphic evaluation) also depend on n.

We note here that this can be fixed in such a way that only the *evk* depends on n, and the rest of the parameters are exactly the same as Regev's scheme with parameters (k, p). This is done in a standard way (used e.g. in [BV11]) as follows.

We will generate the public key as a standard Regev public key with parameters (k, p), and in the evaluation key we will encrypt the bits of the respective secret key using NCCrypt. This will allow to perform homomorphic operations by evaluating the augmented decryption circuit. Namely, the ciphertexts visible to the user of the scheme will always be short, but in the process of homomorphic evaluation, larger ciphertexts are used to accommodate the homomorphic operation, and once it is done dimension-modulus reduction will be used to shrink the output ciphertext back to the original size. Since this is standard practice, we omit the technical description.

References

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In Frank Thomson Leighton and Peter W. Shor, editors, *STOC*, pages 284–293. ACM, 1997.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

[BDFP86] Allan Borodin, Danny Dolev, Faith E. Fich, and Wolfgang J. Paul. Bounds for width two branching programs. *SIAM J. Comput.*, 15(2):549–560, 1986.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS*, pages 309–325. ACM, 2012. Invited to ACM Transactions on Computation Theory.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. *CoRR*, abs/1306.0281, 2013. Preliminary version in STOC 2013.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *FOCS*, pages 97–106. IEEE, 2011. Invited to SIAM Journal on Computing.

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. http://crypto.stanford.edu/craig.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[Gen10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In *CRYPTO*, pages 116–137, 2010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC*, pages 197–206. ACM, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *IACR Cryptology ePrint Archive*, 2013:340, 2013. Preliminary version in CRYPTO 2013.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594. Springer, 2007.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 465–484. Springer, 2011.

[MP11] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. *IACR Cryptology ePrint Archive*, 2011:501, 2011. Extended abstract in Eurocrypt 2012.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *STOC*, pages 333–342. ACM, 2009.

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005. Full version in [Reg09].

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.