Master of Logic Project Report: Lattice Based Cryptography and Fully Homomorphic Encryption

Maximilian Fillinger

August 18, 2012

1 Preliminaries

1.1 Notation

Vectors and matrices are denoted by bold lowercase and uppercase letters. For purposes of matrix multiplication, all vectors are considered as column vectors. We denote the floor and ceiling functions by $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ respectively. Rounding to the closest integer (to the smaller one if there are two) is denoted by $\lfloor\cdot\rceil$. For $q \in \mathbb{Z}$, let $\mathbb{Z}_q = (\lfloor q/2\rfloor, \lfloor q/2\rfloor] \cap \mathbb{Z}$ and for $x \in \mathbb{Z}$, let $[x]_q$ be the unique $y \in \mathbb{Z}_q$ such that $x = y + kq$ for some integer k. For vectors $\mathbf{x} \in \mathbb{Z}^n$, $[\mathbf{x}]_q$ denotes the component-wise application of this operation. Note that here, $\mathbb{Z}_q$ does not denote the ring $\mathbb{Z}/\mathbb{Z}_q$. In particular, for $x \in \mathbb{R}$ and $y \in \mathbb{Z}_q$, $x \cdot y$ denotes the multiplication of x and y in $\mathbb{R}$. The standard scalar product is denoted by $\langle\cdot,\cdot\rangle$ and the euclidean norm by $||\cdot||$.

Let $\sim \subseteq \mathbb{R} \times \mathbb{R}$ be the relation defined by $x \sim y \Leftrightarrow \exists k \in \mathbb{Z} : x + k = y$. Let $\mathbb{T} = \mathbb{R}/\sim$. For $x \in \mathbb{R}$, we denote $x/\sim \in \mathbb{T}$ by $x \mod \mathbb{T}$.

A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is called negligible if for every polynomial function p, there is some $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $\epsilon(n) < p(n)^{-1}$. A function $\delta : \mathbb{N} \to \mathbb{R}$ is called overwhelming if there is a negligible function ϵ such that $\delta(n) \geq 1 - \epsilon(n)$ for all n.

For $s > 0$ and $\mathbf{c} \in \mathbb{R}^n$,let $\rho^{(n)}_{s,\mathbf{c}}(\mathbf{x}) = \exp(\pi||(\mathbf{x} - \mathbf{c})/s||^2)$. The total measure is $\int_{\mathbf{x}\in\mathbb{R}^n} \rho_{s,\mathbf{c}}(\mathbf{x})d\mathbf{x} = s^n$ and the density function of the continuous Gaussian distribution centered on $\mathbf{c}$ with parameter s is $D^{(n)}_{s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x})/s^n$. The discrete Gaussian distribution on a discrete subset $S \subseteq \mathbb{R}^n$ is defined by

$$D^{(n)}_{S,s,\mathbf{c}}(\mathbf{x}) = \begin{cases} D_{s,\mathbf{c}}(\mathbf{x})/D_{s,\mathbf{c}}(S) & \text{if } \mathbf{x} \in S \\ 0 & \text{otherwise} \end{cases}$$

When n is clear from context, it is left out.

We can approximate sampling from a continuous Gaussian distribution by selecting a sufficiently fine discrete grid on $\mathbb{R}^n$ and sampling from the discrete

Gaussian distribution on this grid. For simplicity, we assume that we can sample from this distribution directly.

For $\alpha \in (0,1)$, let $\Psi_\alpha = D^{(1)}_{\alpha,0} \mod \mathbb{T}$. For χ a continuous probability density function on $[0,1)$, let $\bar{\chi}^{(q)}$ be its discretization to $\mathbb{Z}_q$. Usually, q will be clear from context and not written. For Ψ a density function on $\mathbb{T}$, we identify the elements of $\mathbb{T}$ with their unique representatives in $[0,1)$ and define $\bar{\Psi}$ as the discretization.

1.2 Homomorphic Encryption

A homomorphic encryption scheme is a cryptographic scheme that allows to outsource computation to another party without disclosing the input and output of the computed function. It differs from a regular (private-key or public-key) cryptographic scheme by having one additional type of key, the evaluation key, and one additional (probabilistic polynomial time) algorithm **Eval**. For security purposes, we assume that the evaluation key is known to the adversary. **Eval** takes as input an evaluation key evk, a circuit C and for each input wire of C an encryption of 0 or 1 under the corresponding public key pk. We call such a scheme L-homomorphic if for any circuit C of depth $\leq L$ the following holds: Let n be the number of input wires, (pk, sk, evk) a possible output of **Gen** and $m_0, ..., m_n \in \{0,1\}^n$. If $c_i \leftarrow \mathbf{Enc}_{pk}(m_i)$ then $\mathbf{Eval}_{evk}(C, c_1, ..., c_n)$ decrypts to $C(m_1, ..., m_k)$ under sk. It is called a fully homomorphic encryption (FHE) scheme if it is L-homomorphic for every L and it is called leveled fully homomorphic if it is L-homomorphic for an arbitrary value of L supplied at key generation.

To be useful, a homomorphic encryption scheme must also have a property called compactness. Compactness requires that the length of results of **Eval** is bounded by a polynomial in the security parameter n (assuming that C has one output wire). A scheme is compact iff its decryption algorithm can be described by a family of circuits $(C_n)_{n\in\mathbb{N}}$ with size polynomial in n where C_n computes the decryption algorithm for security parameter n. Without compactness, homomorphic evaluation of a circuit C could lead to a ciphertext of size polynomial in C. Thus, its size could grow independently of n which could result in the decryption taking time independent of n. In fact, disregarding compactness makes fully homomorphic encryption trivial: Let Π be any encryption scheme. We could create a non-compact fully homomorphic scheme (without an evaluation key) by taking the identity function as **Eval** and modifying **Dec** so that when it is given a circuit and ciphertexts as input, it decrypts the ciphertexts and evaluates the circuit on the plain texts.

In [7], Craig Gentry presents a process called bootstrapping to generate (leveled) fully homomorphic encryption schemes from a homomorphic encryption scheme that is able to correctly evaluate its own decryption circuits. He also described a bootstrappable encryption scheme. In the leveled fully homomorphic variant, the bootstrapped scheme depends on the same hardness assumptions as the original scheme. In the fully homomorphic variant, the bootstrapped

scheme requires as an additional assumption that the original scheme is circular secure. That is, it must be secure even if the adversary is given an encryption of the secret key under the corresponding public key. This was the first fully homomorphic encryption scheme.

Homomorphic schemes often operate in the following way: Encrypted messages are usually single bits which are encrypted by hiding them inside "noise" generated from a public key and random noise. The secret key-public key pairs are such that the holder of the secret key can remove the noise from the public key and retrieve the encrypted bit by rounding. In schemes that are L-homomorphic but not fully homomorphic, evaluations result in ciphertexts with increased random noise. When it gets too large, decryption may fail. Turning such a scheme into a (leveled) FHE scheme requires some way of controlling the growth of the noise. Gentry's bootstrapping method achieves this by evaluating the decryption circuit homomorphically (details follow in section 2.5).

Later schemes are based on weaker security assumptions and could avoid bootstrapping to some extent. In schemes with $\mathbb{Z}_q^n$ as ciphertext space, the growth of the noise could be controlled by switching to smaller moduli q during evaluation, for example [3]. In [2], a scale-invariant scheme was proposed which can control the growth of the noise without modulus switching. It can be made leveled fully homomorphic without bootstrapping, but it can also be bootstrapped into a fully homomorphic scheme. In the following, this scheme will be described and it will be shown how its security follows from the hardness of the LWE (learning with errors) problem. We will also see how approximating the shortest vector of a lattice, a problem which is conjectured to be hard both in classical and quantum computation, reduces to LWE.

1.3 The LWE-problem

The "Learning With Errors" (LWE)-problem is a problem introduced by Oded Regev [13]. For a function $q : \mathbb{N} \to \mathbb{N}$ and a family of distributions $\chi = (\chi_n)_{n\in\mathbb{N}}$ on $\mathbb{Z}_{q(n)}$ and $\mathbf{s} \in \mathbb{Z}_q^n$, define the distribution $A_{\mathbf{s},q,\chi}$ as the distribution that is sampled by the following process:

1. Let n be the length of $\mathbf{s}$. Sample $\mathbf{a}$ uniformly from $\mathbb{Z}_{q(n)}^n$.
2. Sample e from χ. This is called the error or noise of the sample.
3. Output the pair $(\mathbf{a}, [\langle \mathbf{a}, \mathbf{s}\rangle + e]_{q(n)})$.

For a function $N : \mathbb{N} \to \mathbb{N}$ and the other parameters as before, the search problem $\mathrm{LWE}_{N,q,\chi}$ is to output $\mathbf{s}$ on input of N samples from $A_{\mathbf{s},q,\chi}$. We can represent these samples by a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times N}$ sampled uniformly at random and a vector $\mathbf{b} = [\mathbf{A}^\top \mathbf{s} + \mathbf{e}]_q$ for $\mathbf{e}$ sampled from χ^N.

The decisional variant of LWE is called DLWE. For N, q, χ and $A_{\mathbf{s},q,\chi}$ as before, the problem $\mathrm{DLWE}_{N,q,\chi}$ is to distinguish N samples drawn from $A_{\mathbf{s},q,\chi}$ from N samples drawn uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We say that an algorithm $\mathscr{A}$ solves $\mathrm{DLWE}_{N,q,\chi}$ *in the average case* if there is a non-negligible function f such

that for each n, there is a set $S_n \subseteq \mathbb{Z}_q^n$ with the properties that $|S_n| \geq f(n)|\mathbb{Z}_q^n|$ and $\mathscr{A}$ distinguishes with non-negligible advantage between the distributions $A^N_{\mathbf{s},q,\chi}$ and $U\left(\mathbb{Z}_q^n \times \mathbb{Z}_q\right)$ for all $\mathbf{s} \in S_n$. Stating this less formal, we require that $\mathscr{A}$ distinguishes $A^N_{\mathbf{s},q,\chi}$ from uniform with non-negligible advantage for a non-negligible fraction of the $\mathbf{s}$ in $\mathbb{Z}_q^n$.

An algorithm is said to solve this problem *in the worst case* if it distinguishes $A_{\mathbf{s},q,\chi}$ from uniform with overwhelming advantage for all $\mathbf{s} \in \mathbb{Z}_q^n$. Clearly, the average case reduces to the worst case of DLWE, which in turn reduces to solving LWE with overwhelming probability. Also, $\mathrm{DLWE}_{N,q,\chi}$ is hard in the average case if and only if the distribution that is sampled by selecting $\mathbf{s} \in \mathbb{Z}_q^n$ uniformly at random and outputting N samples of $A_{\mathbf{s},q,\chi}$ is computationally indistinguishable from random.

There also is a continuous version of $\mathrm{(D)LWE}_{N,q,\chi}$ where χ is a distribution on $\mathbb{T}$ and a sample of $A_{\mathbf{s},q,\chi}$ is of the form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle / q + e \mod \mathbb{T})$.

We write $\mathrm{LWE}_{q,\chi}$ and $\mathrm{DLWE}_{q,\chi}$ to denote the LWE problem when the adversary is given an oracle that samples $A_{\mathbf{s},q,\chi}$ instead of inputs sampled from this distribution and the problem of distinguishing an oracle that outputs $A_{\mathbf{s},q,\chi}$ from an oracle that outputs uniform samples. Obviously, there is an algorithm that solves $\mathrm{(D)LWE}_{q,\chi}$ in polynomial time if and only if there is a polynomial time algorithm that solves $\mathrm{(D)LWE}_{N,q,\chi}$ for some function $N = \mathrm{poly}(n)$.

The average-case DLWE problem serves as a good basis for cryptographic schemes because it is equivalent to a statement about computational indistinguishability and because, for certain choices of parameters, worst-case DLWE reduces to average-case DLWE and LWE reduces to worst-case DLWE. Moreover, the lattice problem GapSVP_γ reduces to LWE for an appropriate choice of the distribution χ for γ subexponential in n. A lattice is a discrete additive subgroup of $\mathbb{R}^n$ and GapSVP_γ is the problem to approximate the length of the shortest vector in a given lattice within a factor of γ. Lattice problems have been studied since the 1980s and so far, the best known polynomial time algorithm, the LLL-algorithm found by Lenstra, Lenstra and Lovász in 1982, solves GapSVP_γ only for mildly exponential γ. Moreover, for $\gamma < \sqrt{2}$, the problem GapSVP_γ is known to be NP-hard under probabilistic reductions (see [8], Chapter 4). For these reasons, the problem GapSVP_γ is conjectured to be hard for subexponential γ.

We will have a closer look at lattices in section 3. For now, let us take a look at the worst-case to average case and search to decision reductions. The worst-case to average-case reduction for $\mathrm{DLWE}_{q,\chi}$ is proved in [13] and only requires that $\log q \in \mathrm{poly}(n)$ so that a probabilistic polynomial time algorithm can sample from $\mathbb{Z}_q^n$.

Lemma 1.1 (Worst-case to average-case)**.** *Let q be an integer function with $\log q \in O(n)$. Suppose that there is an algorithm $\mathscr{A}$ that solves $DLWE_{q,\chi}$. That is, for every n, there is a non-negligible fraction of the $\boldsymbol{s} \in \mathbb{Z}_q^n$ such that $\mathscr{A}$ given input 1^n distinguishes an oracle sampling $A_{\boldsymbol{s},q,\chi}$ from an oracle sampling the uniform distribution with non-negligible probability. Then there also is a polynomial time algorithm that for any n and any $\boldsymbol{s} \in \mathbb{Z}_q^n$ outputs 1 with proba-*

bility exponentially close to 1 on input 1^n *and given oracle acces to* $A_{s,q,\chi}$ *and outputs 0 with probability exponentially close to 1 on input* 1^n *and given access to an uniform oracle.*

Proof Sketch: Define $f_{\mathbf{t}}(\mathbf{a}, \mathbf{b}) = (\mathbf{a}, [\mathbf{b} + \langle \mathbf{a}, \mathbf{t} \rangle]_q)$. On input sampled from $A_{\mathbf{s},q,\chi}$, the output of $f_{\mathbf{t}}$ is distributed identically to $A_{\mathbf{s}+\mathbf{t},q,\chi}$ and on input sampled uniformly, the output will be distributed uniformly. Note that for a fixed $\mathbf{s}$ and for $\mathbf{t}$ sampled uniformly at random, $\mathbf{s} + \mathbf{t}$ is distributed uniformly. There is a polynomial p such that if we sample $p(n)$ elements $\mathbf{s} \in \mathbb{Z}_q^n$ uniformly at random, we will obtain with probability exponentially close to 1 at least one $\mathbf{s}$ on which $\mathscr{A}$ has non-negligible advantage. With an oracle for average-case $\mathrm{DLWE}_{q,\chi}$, we can solve worst-case $\mathrm{DLWE}_{q,\chi}$ by the following procedure: Given input 1^n and access to an oracle O, let R be the unknown distribution sampled by O. Do the following $p(n)$ times:

1. Sample $\mathbf{t}$ uniformly at random from $\mathbb{Z}_q^n$.
2. Estimate the acceptance probability of $\mathscr{A}$ on input 1^n with access to an oracle that samples the uniform distribution. We can do this by running $\mathscr{A}$ several times and returning random results when it queries its oracle. By the Chernoff bound, polynomially many repetitions suffice to produce a close enough estimate with probability exponentially close to one.
3. Estimate the acceptance probability of $\mathscr{A}$ on input 1^n and with access to an oracle that samples $(\mathbf{a}, \mathbf{b}) \leftarrow R$ and returns $f_{\mathbf{t}}(\mathbf{a}, \mathbf{b})$. If R is the uniform distribution, this is again uniform; if it is $A_{\mathbf{s},q,\chi}$ then this procedure samples $A_{\mathbf{s}+\mathbf{t},q,\chi}$.
4. Compare the two estimates. If they differ sufficiently, output 1 and halt. Otherwise, start the next repetition; if this was the last repetition, output 0 and halt.

An analogous procedure works for the continuous version. □

The search to decision reduction was proved in the same paper for any prime q with $q = \mathrm{poly}(n)$. For the continuous variant with $\chi = \Psi_\alpha$ for some α with $0 < \alpha \leq 1/\omega(\sqrt{\log n})$ ($\omega(\sqrt{\log n})$ denotes an arbitrary function growing faster than $\log n$.) this result was further extended, to $q = q_1^{e_1} \cdot ... \cdot q_m^{e_m}$ where the q_i are polynomially bounded distinct primes with $q_i \geq \omega(\log n)/\alpha$ in [Micciancio, Peikert 2011], Theorem 3.1. In [Regev 2005,2009], it is also proved that if χ is a distribution on $\mathbb{T}$ and $\bar{\chi}$ its discretization to $\mathbb{Z}_q$ and if there is a polynomial time algorithm $\mathscr{A}$ for the discrete version $\mathrm{DLWE}_{N,q,\bar{\chi}}$ then there is a polynomial time algorithm $\mathscr{B}$ for continuous average-case $\mathrm{DLWE}_{N,q,\chi}$. The algorithm $\mathscr{B}$ proceeds by discretizing the samples it receives to $\mathbb{Z}_q$ and running $\mathscr{A}$ on the results. Taking all this together gives us:

Theorem 1.2 (LWE reductions)**.** *Let* $\chi = \Psi_\alpha$ *for* $0 < \alpha \leq 1/\omega(\sqrt{\log n})$ *(where* $\omega(\sqrt{\log n})$ *stands for an arbitrary but fixed function growing faster than* $\sqrt{\log n}$*) and* $\bar{\chi}$ *its discretization to* $\mathbb{Z}_q$*. Let* $q = q_1^{e_1} ... q_k^{e_k}$ *for* q_i *distinct primes with*

$\omega(\log n)/\alpha \leq q_i \leq O(n)$. If there is a polynomial time algorithm that solves $LWE_{N,q,\bar{\chi}}$ for N polynomial in n then there is a polynomial time algorithm that solves $LWE_{q,\chi}$ with probability exponentially close to 1.

2 Brakerski's FHE Scheme without Modulus Switching

2.1 Regev's Encryption Scheme

Brakerski's fully homomorphic encryption scheme relies on the following (non-homomorphic) scheme by Oded Regev [13]. The scheme will be called **Regev**. Let q be a function on $\mathbb{N}$ with integer values and $\log q \in O(n)$, let $\chi = (\chi_n)_{n\in\mathbb{N}}$ be an ensemble of probability distributions on $\mathbb{Z}_q$ for which there is B with $\Pr_\chi[b] = 0$ for $|b| > B$ and let N be a polynomially bounded function with values in $\mathbb{N}$. When the security parameter n is clear from context, q is written for $q(n)$ etc. In order to base the security of the scheme on $\mathrm{GapSVP}_{\tilde{O}(n)}$, we need $\chi = \bar{\Psi}_\alpha$ and q satisfying the conditions in theorem 1.2, $q \geq 2^{n/2} \cdot \log n/\sqrt{n}$ and $N \geq (n+1)\log q$. To have the scheme always decrypt correctly we need $q > N \cdot B$ and to make Brakerski's final scheme (leveled) fully homomorphic using bootstrapping, we require that $q/B \geq O(n \log q)^{L(n)+O(1)}$ where $L(n)$ is the depth of the decryption circuit $+1$. Note that $\bar{\Psi}_\alpha$ is not bounded by a $B < q/2$, but we can find an appropriate B such that $\bar{\Psi}_\alpha$ has negligible statistical distance to a B-bounded distribution. The scheme **Regev** consists of the following algorithms:

- **Regev.SecretKeygen**(1^n): sample a vector $\mathbf{s}$ from $\mathbb{Z}_q^n$ uniformly at random and output it as the secret key sk.
- **Regev.PublicKeygen**$(\mathbf{s})$: Select a matrix $\mathbf{A} \in \mathbb{Z}_q^{N\times n}$ uniformly at random and sample $\mathbf{e}$ from χ^N. Then compute $b = [\mathbf{As} + \mathbf{e}]_q$ and output

  $$P = [b| - A] \in \mathbb{Z}^{N\times(n+1)}$$

  as the public key pk.
- **Regev.Enc**$_{pk}(m)$ for $m \in \{0,1\}$: Let $\mathbf{P} = pk$, sample a vector $\mathbf{r}$ uniformly from $\{0,1\}^N$ and define $\mathbf{m} = (m, 0, ..., 0) \in \mathbb{Z}^{n+1}$. Output

  $$\mathbf{c} = \left[\mathbf{P}^\top \mathbf{r} + \left\lfloor \frac{q}{2} \right\rfloor \mathbf{m}\right]_q \in \mathbb{Z}_q^{n+1}$$

  as the ciphertext.
- **Regev.Dec**$_{sk}(\mathbf{c})$: Let $\mathbf{s} = sk$ and output

  $$\left[\left\lfloor 2 \cdot \frac{[\langle \mathbf{c}, (1,\mathbf{s})\rangle]_q}{q} \right\rceil\right]_2$$

In order to prove correctness, we need the following lemma.

Lemma 2.1. *Let n, N, q, χ, B be parameters for the **Regev** scheme, let $\boldsymbol{s} \in \mathbb{Z}_q^n$ and $m \in \{0,1\}$. For $\boldsymbol{P} \leftarrow \textbf{\textit{Regev.PublicKeygen}}(\boldsymbol{s})$ and $\boldsymbol{c} \leftarrow \textbf{\textit{Regev.Enc}}_{\boldsymbol{P}}(m)$, there exists $e \in \mathbb{Z}_q$ with $|e| \leq N \cdot B$ such that*

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

Proof. Modulo q, the following equivalencies hold:

$$\begin{aligned}
\langle \mathbf{c}, (1, \mathbf{s}) \rangle &\equiv \left\langle \mathbf{P}^\top \mathbf{r}, (1, \mathbf{s}) \right\rangle + \left\lfloor \frac{q}{2} \right\rfloor \langle (m, 0, ..., 0), (1, \mathbf{s}) \rangle \\
&\equiv \mathbf{r}^\top \mathbf{P} (1, \mathbf{s}) + \left\lfloor \frac{q}{2} \right\rfloor \cdot m \\
&\equiv \left\lfloor \frac{q}{2} \right\rfloor \cdot m + \mathbf{r}^\top (\mathbf{b} - \mathbf{A}\mathbf{s}) \\
&\equiv \left\lfloor \frac{q}{2} \right\rfloor \cdot m + \langle \mathbf{r}, \mathbf{e} \rangle
\end{aligned}$$

where $\mathbf{r} \in \{0,1\}^N$ is the random vector sampled in **Regev.Enc** and $\mathbf{e}$ is sampled from χ^N. Since χ is bounded by B, we have:

$$|\langle \mathbf{r}, \mathbf{e} \rangle| = \left| \sum_{i=1}^{N} \mathbf{r}_i \mathbf{e}_i \right| \leq \sum_{i=1}^{N} \mathbf{r}_i |\mathbf{e}_i| \leq \sum_{i=1}^{N} B = N \cdot B$$

so for $e = \langle \mathbf{r}, \mathbf{e} \rangle$, the lemma holds. □

The next lemma implies that if $|e| < \lfloor q/2 \rfloor / 2$, the algorithm **Regev.Dec** will decrypt correctly. Thus, for $q > 4NB$, the decryptions in **Regev** are always correct.

Lemma 2.2 (Correctness)**.** *If $|e| < \lfloor q/2 \rfloor / 2$, then*

$$\left\lfloor 2 \cdot \frac{\left\lfloor \frac{q}{2} \right\rfloor m + e}{q} \right\rceil = m$$

Proof. For $m = 0$, $\lfloor q/2 \rfloor m + e \in (-(q-1)/4, q/4) \cap \mathbb{Z}$ and therefore

$$2 \cdot \frac{\left\lfloor \frac{q}{2} \right\rfloor m + e}{q} \in \left(-\frac{1}{2} + \frac{1}{2q}, \frac{1}{2} \right).$$

If $m = 1$ then $\lfloor q/2 \rfloor m + e \in ((q-1)/4, 3q/4) \cap \mathbb{Z}$ and therefore

$$2 \cdot \frac{\left\lfloor \frac{q}{2} \right\rfloor m + e}{q} \in \left(\frac{1}{2} - \frac{1}{2q}, \frac{3}{2} \right) \cap \frac{1}{q}\mathbb{Z}.$$

Since $\frac{1}{q}\mathbb{Z}$ contains no element both larger than $\frac{1}{2} - \frac{1}{2q}$ and smaller than $\frac{1}{2}$, it follows that

$$2 \cdot \frac{\left\lfloor \frac{q}{2} \right\rfloor m + e}{q} \in \left[\frac{1}{2}, \frac{3}{2} \right)$$

and thus, for every m,

$$\left\lfloor 2 \cdot \frac{\left\lfloor \frac{q}{2} \right\rfloor m + e}{q} \right\rceil = m.$$

□

To reduce the security of this scheme to the DLWE problem, we are going to need the following special case of the leftover hash lemma (see [13] and [9]). Informally, it states that if we select a $n \times N$-matrix over $\mathbb{Z}_q$ uniformly at random for N large enough, then with high probability, adding a randomly selected subset of columns from this matrix generates pseudorandom elements of $\mathbb{Z}_q^n$.

Lemma 2.3. *Let $q : \mathbb{N} \to \mathbb{N}$ be such that for every constant c and sufficiently large n, $q(n) > n^c$. Let $N \geq (n+1)\log q$. For a matrix $\boldsymbol{M} \in \mathbb{Z}_q^{n \times N}$, let $\phi_{\boldsymbol{M}}$ on $\mathbb{Z}_q^n$ be the distribution which is sampled by selecting $\boldsymbol{r} \in \{0,1\}^N$ uniformly at random and outputting $[\boldsymbol{Mr}]_q$. If we select $\boldsymbol{M}$ from $\mathbb{Z}_q^{n \times N}$ uniformly at random then except with negligible probability, the distribution $\phi_{\boldsymbol{M}}$ has negligible statistical distance from the uniform distribution on $\mathbb{Z}_q^n$. It follows that it is pseudorandom.*

Proof. It holds that

$$\phi_{\mathbf{M}}(\mathbf{x}) = \Pr_{\mathbf{r}}\left[[\mathbf{Mr}]_q = \mathbf{x}\right] = \frac{1}{2^N} \cdot \left|\left\{\mathbf{r} \in \{0,1\}^N \,|[\mathbf{Mr}]_q = \mathbf{x}\right\}\right|$$

and thus

$$\begin{aligned}
\sum_{\mathbf{x} \in \mathbb{Z}_q^n} \phi_{\mathbf{M}}(\mathbf{x})^2 &= \sum_{\mathbf{x} \in \mathbb{Z}_q^n} \Pr_{\mathbf{r},\mathbf{r}'}\left[[\mathbf{Mr}]_q = \left[\mathbf{Mr}'\right]_q = \mathbf{x}\right] \\
&= \Pr_{\mathbf{r},\mathbf{r}'}\left[[\mathbf{Mr}]_q = \left[\mathbf{Mr}'\right]_q\right] \\
&\leq \Pr_{\mathbf{r},\mathbf{r}'}\left[\mathbf{r} = \mathbf{r}'\right] + \Pr_{\mathbf{r},\mathbf{r}'}\left[[\mathbf{Mr}]_q = \left[\mathbf{Mr}'\right]_q | \mathbf{r} \neq \mathbf{r}'\right] \\
&= \frac{1}{2^N} + \Pr_{\mathbf{r},\mathbf{r}'}\left[[\mathbf{Mr}]_q = \left[\mathbf{Mr}'\right]_q | \mathbf{r} \neq \mathbf{r}'\right]
\end{aligned}$$

Let $\mathbf{r}, \mathbf{r}'$ be any two distinct vectors in $\{0,1\}^N$ and i the least index such that $\mathbf{r}_i \neq \mathbf{r}'_i$. Let $\mathbf{M}_j$ be the jth column of $\mathbf{M}$. If $\mathbf{r}_i = 1$, then

$$\Pr_{\mathbf{M}}\left[[\mathbf{Mr}]_q = [\mathbf{Mr}']_q\right] = \Pr_{\mathbf{M}}\left[\mathbf{M}_i = \left[\sum_{j=i+1}^{N} \mathbf{M}_j \cdot (\mathbf{r}_j - \mathbf{r}'_j)\right]_q\right] = \frac{1}{q^n}$$

and if $\mathbf{r}_i = 0$, the same probability can be calculated analogously. Taking the

expecation over $\mathbf{M}$, it follows that

$$\begin{aligned}
E_{\mathbf{M}}\left[\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\phi_{\mathbf{M}}(\mathbf{x})^2\right] &\leq \frac{1}{2^N} + E_{\mathbf{M}}\left[\Pr_{\mathbf{r},\mathbf{r}'}\left[[\mathbf{Mr}]_q = \left[\mathbf{Mr}'\right]_q | \mathbf{r}\neq\mathbf{r}'\right]\right] \\
&= \frac{1}{2^N} + \sum_{\mathbf{M}}\left(\frac{1}{q^{n\cdot N}}\sum_{\mathbf{r}\neq\mathbf{r}'}\frac{\left[[\mathbf{Mr}]_q = \left[\mathbf{Mr}'\right]_q\right]}{2^N\cdot(2^N-1)}\right) \\
&= \frac{1}{2^N} + \sum_{\mathbf{r}\neq\mathbf{r}'}\left(\frac{1}{2^N(2^N-1)}\sum_{\mathbf{M}}\frac{\left[[\mathbf{Mr}]_q = \left[\mathbf{Mr}'\right]_q\right]}{q^{n\cdot N}}\right) \\
&= \frac{1}{2^N} + \sum_{\mathbf{r}\neq\mathbf{r}'}\frac{\Pr_{\mathbf{M}}\left[[\mathbf{Mr}]_q = \left[\mathbf{Mr}'\right]_q\right]}{2^N(2^N-1)} \\
&= \frac{1}{2^N} + \frac{1}{q^n}
\end{aligned}$$

where $[A=B]=1$ if $A=B$ and is 0 otherwise. Now consider, for a fixed $\mathbf{M}$, $\mathbf{v} = (\phi_{\mathbf{M}}(\mathbf{x}) - q^{-n})_{\mathbf{x}\in\mathbb{Z}_q^n}$ as a vector in $\mathbf{R}^{q^n}$. Then we have

$$\begin{aligned}
\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\left|\phi_{\mathbf{M}}(\mathbf{x}) - \frac{1}{q^n}\right| = ||\mathbf{v}||_1 &\leq \sqrt{q^n}||\mathbf{v}||_2 \\
&= \sqrt{q^n\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\left(\phi_{\mathbf{M}}(\mathbf{x}) - \frac{1}{q^n}\right)^2} \\
&= \sqrt{q^n\left(\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\phi_{\mathbf{M}}(\mathbf{x})^2 - \frac{2}{q^n}\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\phi_{\mathbf{M}}(\mathbf{x}) + \sum_{\mathbf{x}\in\mathbb{Z}_q^n}\frac{1}{q^{2n}}\right)} \\
&= \sqrt{q^n\left(\left(\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\phi_{\mathbf{M}}(\mathbf{x})^2\right) - \frac{2}{q^n} + \frac{1}{q^n}\right)} \\
&= \sqrt{q^n\left(\left(\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\phi_{\mathbf{M}}(\mathbf{x})^2\right) - \frac{1}{q^n}\right)}
\end{aligned}$$

It follows that

$$E_{\mathbf{M}}\left[\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\left|P_{\mathbf{M}}(\mathbf{x})-\frac{1}{q^n}\right|\right] \le \sqrt{q^n}\cdot E_{\mathbf{M}}\left[\sqrt{\left(\sum_{\mathbf{x}\in\mathbb{Z}_q^n}P_{\mathbf{M}}(\mathbf{x})^2\right)-\frac{1}{q^n}}\right]$$
$$\le \sqrt{q^n}\sqrt{E_{\mathbf{M}}\left[\sum_{\mathbf{x}\in\mathbb{Z}_q^n}P_{\mathbf{M}}(\mathbf{x})\right]-\frac{1}{q^n}}$$
$$\le \sqrt{\frac{q^n}{2^N}}$$

where the second inequality follows since $x \mapsto \sqrt{x}$ is a concave function and by Jensen's inequality, for any concave function ψ, $E(\psi(X)) \le \psi(E(X))$. By Markov's inequality,

$$\Pr_{\mathbf{M}}\left[\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\left|\phi_{\mathbf{M}}(\mathbf{x})-\frac{1}{q^n}\right| > \left(\frac{q^n}{2^N}\right)^{1/4}\right] < \left(\frac{q^n}{2^N}\right)^{-1/4} E_{\mathbf{M}}\left[\sum_{\mathbf{x}\in\mathbb{Z}_q^n}\left|\phi_{\mathbf{M}}(\mathbf{x})-\frac{1}{q^n}\right|\right]$$
$$= \left(\frac{q^n}{2^N}\right)^{1/4}$$

For $N \ge (n+1)\log q$ and q superpolynomial, $\left(q^n/2^N\right)^{1/4} \le q^{-1/4}$ is negligible. Therefore, except with negligible probability (taken over $\mathbf{M}$), $\sum_{\mathbf{x}} |P_{\mathbf{M}}(\mathbf{x}) - q^{-n}|$ is negligible. Now let $\mathbf{M}$ be a matrix such that this sum is negligible and let $\mathscr{A}$ be a (possibly non-efficient) Turing machine. Without loss of generality, $\mathscr{A}$ is deterministic. Let $A \subseteq \mathbb{Z}_q^n$ be the set for which $\mathscr{A}$ outputs 1. The probability that $\mathbf{x} \in \mathbb{Z}_q^n$ sampled from the uniform distribution is in A is $|A|/q^n$ and the probability $\mathbf{Mx}$ sampled from $\phi_{\mathbf{M}}$ is in A is $\sum_{\mathbf{x}\in A}\phi_{\mathbf{M}}(\mathbf{x})$. The difference between those two probabilities is

$$\left|\sum_{\mathbf{x}\in A}\phi_{\mathbf{M}}(\mathbf{x})-\frac{|A|}{q^n}\right| \le \sum_{\mathbf{x}\in\mathbb{Z}_q^n}\left|\phi_{\mathbf{M}}(\mathbf{x})-\frac{1}{q^n}\right|$$

and thus, this difference is negligible. So $\mathscr{A}$ on input sampled from $\phi_{\mathbf{M}}$ outputs 1 with almost the same probability as on input chosen uniformly at random from $\mathbb{Z}_q^n$. Thus, $\phi_{\mathbf{M}}$ is pseudorandom on $\mathbb{Z}_q^n$. □

Now we can reduce the security of **Regev** to the DLWE problem.

Theorem 2.4 (Security)**.** *Let q, N, χ, B be parameters such that average-case $DLWE_{N,q,\chi}$ is hard. Then the ensembles of distributions $D_0(n)$ and $D_1(n)$ generated by running $sk \leftarrow$ **Regev.SecretKeygen**(1^n), $\boldsymbol{P} \leftarrow$ **Regev.PublicKeygen**(sk), $\boldsymbol{c} \leftarrow$ **Regev.Enc**$_{\boldsymbol{P}}(m)$ for $m = 0, 1$ respectively and outputting $(\boldsymbol{P}, \boldsymbol{c})$ are computationally indistinguishable. Therefore, if $DLWE_{N,q,\chi}$ is hard,* ***Regev*** *is semantically secure.*

Proof. We can represent N samples $(\mathbf{a}_i, b_i)$ of the $A_{\mathbf{s},q,\chi}(n)$ distribution as a matrix $\mathbf{P} = [\mathbf{b}|-\mathbf{A}]$ where $\mathbf{b}$ is a vector with ith entry b_i and $\mathbf{A}$ is a matrix with ith column $\mathbf{a}_i$. If $\mathrm{DLWE}_{N,q,\chi}$ is hard, the distribution generated by sampling $\mathbf{s} \in \mathbb{Z}_q^n$ uniformly at random and then sampling $A_{\mathbf{s},q,\chi}(n)^N$ is computationally indistinguishable from uniform. It follows that the distributions D_A and D_B generated by sampling $\mathbf{s} \in \mathbb{Z}_q^n$ uniformly at random and $\mathbf{M}$ from $A_{\mathbf{s},q,\chi}(n)^N$ or from the uniform distribution on $\mathbb{Z}_q^{N\times(n+1)}$, respectively, and $\mathbf{r}$ from $\{0,1\}^N$ and outputting $\left(\mathbf{M}, \left[\mathbf{M}^\top \mathbf{r}\right]_q\right)$ are computationally indistinguishable. Note that a pair sampled from D_A is a pair of a public key (corresponding to some n-bit secret key $\mathbf{s}$ sampled uniformly at random) and an encryption of 0, that is, $D_A = D_0$.

By lemma 2.3, D_B is computationally indistinguishable from uniform except with negligible probability over the choice of $\mathbf{M}$. It follows that the distribution D_0 is computationally indistinguishable from uniform. Also, D_1 is indistinguishable from uniform, for suppose we had a PPT algorithm $\mathscr{A}$ that distinguishes D_1 from uniform with non-negligible probability. Then we can construct an algorithm $\mathscr{A}'$ that on input $(\mathbf{M}, \mathbf{x})$ computes $(\mathbf{M}, \mathbf{x} + (\lfloor q/2 \rfloor, 0, ..., 0))$ and runs $\mathscr{A}$ on it and outputs its result. The operation $\mathbf{x} \mapsto [\mathbf{x} + (\lfloor q/2 \rfloor, 0, ..., 0)]_q$ takes encryptions of 0 to encryptions of 1 and takes input sampled from the uniform distribution to uniformly distributed output. Thus, $\mathscr{A}'$ would be able to distinguish D_0 from uniform with non-negligible probability which is impossible.

Both D_0 and D_1 are computationally indistinguishable from uniform, so they are computationally indistinguishable from each other and **Regev** is semantically secure. □

2.2 Switching Keys

Regev's scheme is already somewhat homomorphic with respect to addition. We could simply add two ciphertexts modulo q. However, the error terms e of those two ciphertexts would add up and when they grow larger than the upper bound established in lemma 2.2, correct decryption is no longer guaranteed. Multiplication is more difficult: If we have ciphertexts $\mathbf{c}_1, \mathbf{c}_2$ encrypting m_1, m_2 respectively under a private key belonging to a secret key $\mathbf{s}$, then $(2/q) \cdot \mathbf{c}_1 \otimes \mathbf{c}_2$ decrypts to the product of the encrypted plaintexts in the following way: It holds that

$$\begin{aligned}
&\langle \frac{2}{q} \cdot \mathbf{c}_1 \otimes \mathbf{c}_2, (1, \mathbf{s}) \otimes (1, \mathbf{s}) \rangle \\
&= \frac{2}{q} \cdot \langle \mathbf{c}_1, (1, \mathbf{s}) \rangle \cdot \langle \mathbf{c}_2, (1, \mathbf{s}) \rangle \\
&= \frac{2}{q} \cdot \left(\left\lfloor \frac{q}{2} \right\rfloor m_1 + e_1 \right) \cdot \left(\left\lfloor \frac{q}{2} \right\rfloor m_2 + e_2 \right) \mod q \\
&= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 m_2 + e_{mult} \mod q
\end{aligned}$$

for some error term e_{mult}. The problem with this approach is that the length of the ciphertext vector is squared by this operation which violates the compactness requirement for FHE schemes. But by key switching, a technique from [4] and [3], we can transform this into a normal-sized ciphertext. A switch key $\mathbf{P_{s:t}}$ from a source secret key $\mathbf{s}$ to a target secret key $\mathbf{t}$ allows to compute an encryption of a message m under $(1, \mathbf{t})$ from an encryption of m under $\mathbf{s}$ without revealing the secret keys or helping to decrypt a ciphertext.

The key switching process will also make use of the following two operations to control the growth of the error term: The first is decomposing vectors in $\mathbb{Z}_q^n$ into their bit representation. If $\mathbf{x} \in \mathbb{Z}_q^n$, there are unique vectors $\mathbf{w}_i \in \{0,1\}^n$ for $0 \leq i \leq \lceil \log q \rceil - 1$ such that

$$\mathbf{x} = \sum_{i=0}^{\lceil \log q \rceil - 1} 2^i \mathbf{w}_i.$$

Let

$$\mathbf{BitDecomp}_q(\mathbf{x}) = \left(\mathbf{w}_0, ..., \mathbf{w}_{\lceil \log q \rceil - 1}\right) \in \{0,1\}^{n \cdot \lceil \log q \rceil}.$$

The second function is

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

These functions can be computed efficiently and for any $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$,

$$\langle \mathbf{x}, \mathbf{y} \rangle = \left\langle \mathbf{BitDecomp}_q(\mathbf{x}), \mathbf{PowersOfTwo}_q(\mathbf{y}) \right\rangle$$

because modulo q,

$$\begin{aligned} & \left\langle \mathbf{BitDecomp}_q(\mathbf{x}), \mathbf{PowersOfTwo}_q(\mathbf{y}) \right\rangle \\ \equiv & \sum_{i=0}^{\lceil \log q \rceil - 1} \left\langle \mathbf{w}_i, 2^i \mathbf{y} \right\rangle \\ \equiv & \left\langle \sum_{i=0}^{\lceil \log q \rceil - 1} 2^i \mathbf{w}_i, \mathbf{y} \right\rangle \\ \equiv & \langle \mathbf{x}, \mathbf{y} \rangle \end{aligned}$$

For the key switching process the following algorithms are used, where q and χ are parameters as for **Regev**:

- $\mathbf{SwitchKeyGen}_{q,\chi}(\mathbf{s}, \mathbf{t})$: Let n_s be the dimension of $\mathbf{s}$, n_t the dimension of $\mathbf{t}$ and $\hat{n}_s = n_s \cdot (\lceil \log q \rceil)$ the dimension of $\mathbf{PowersOfTwo}_q(\mathbf{s})$. Generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{\hat{n}_s \times n_t}$ uniformly at random, sample $\mathbf{e}$ from $\chi^{\hat{n}_s}$ and compute

  $$\mathbf{b} = \left[\mathbf{At} + \mathbf{e} + \mathbf{PowersOfTwo}_q(\mathbf{s})\right]_q.$$

  Then output the matrix

  $$\mathbf{P_{s:t}} = [\mathbf{b} | -\mathbf{A}]$$

  as the switch key.

- $\mathbf{SwitchKey}_q(\mathbf{P_{s:t}}, \mathbf{c_s})$: Compute and output

$$\mathbf{c_t} = \left[\mathbf{P}_{\mathbf{s:t}}^{\top} \cdot \mathbf{BitDecomp}_q(\mathbf{c_s})\right]_q .$$

The next lemma gives an upper bound on the error introduced by key-switching.

Lemma 2.5. *Let $\boldsymbol{s} \in \mathbb{Z}_q^{n_s}$ and $\boldsymbol{t} \in \mathbb{Z}_q^{n_t}$ and $\boldsymbol{c_s} \in \mathbb{Z}_q^{n_s}$. For $\boldsymbol{P_{s:t}} \leftarrow \boldsymbol{SwitchKeyGen}_{q,\chi}(\boldsymbol{s}, \boldsymbol{t})$ and $\boldsymbol{c_t} \leftarrow \boldsymbol{SwitchKey}_q(\boldsymbol{P_{s:t}}, \boldsymbol{c_s})$,*

$$\langle \boldsymbol{c_s}, \boldsymbol{s} \rangle = \langle \boldsymbol{c_t}, (1, \boldsymbol{t}) \rangle - \langle \boldsymbol{BitDecomp}_q(\boldsymbol{c_s}), \boldsymbol{e} \rangle \mod q$$

Proof. Modulo q, it holds that

$$\begin{aligned}
\langle \mathbf{c_t}, (1, \mathbf{t}) \rangle &\equiv \left\langle \left[\mathbf{P}_{\mathbf{s:t}}^{\top} \cdot \mathbf{BitDecomp}_q(\mathbf{c_s})\right]_q , (1, \mathbf{t}) \right\rangle \\
&\equiv \left\langle \mathbf{BitDecomp}_q(\mathbf{c_s}), \mathbf{b} \right\rangle - \left\langle \mathbf{BitDecomp}_q(\mathbf{c_s}), \mathbf{At} \right\rangle \\
&\equiv \left\langle \mathbf{BitDecomp}_q(\mathbf{c_s}), \mathbf{PowersOfTwo}_q(\mathbf{s}) \right\rangle + \langle \mathbf{BitDecomp}_q(\mathbf{c_s}), \mathbf{e} \rangle \\
&\equiv \langle \mathbf{c_s}, \mathbf{s} \rangle + \langle \mathbf{BitDecomp}_q(\mathbf{c_s}), \mathbf{e} \rangle
\end{aligned}$$

□

The security of the key switching algorithms reduces to the hardness of the DLWE problem.

Lemma 2.6 (Security)**.** *Suppose that average-case $DLWE_{\hat{n}_s,n,q,\chi}$ is hard. Let $\boldsymbol{s}$ be an arbitrary vector in $\mathbb{Z}_q^{n_s}$ and let $\boldsymbol{t} \leftarrow \boldsymbol{Regev.SecretKeygen}(1^n)$. Then $\boldsymbol{P} \leftarrow \boldsymbol{SwitchKeyGen}(\boldsymbol{s}, \boldsymbol{t})$ is computationally indistinguishable from a matrix selected uniformly at random from $\mathbb{Z}_q^{\hat{n}_s \times n+1}$.*

Proof. Since $\mathbf{s}$ is arbitrary but fixed, the switch keys are, essentially, samples of $A_{\mathbf{t},q,\chi}^{\hat{n}_s}$ with some constants added. Therefore, the hardness of DLWE implies that the switch keys are indistinguishable from random. In more detail, if average-case $\mathrm{DLWE}_{\hat{n}_s,n,q,\chi}$ is hard, then a matrix generated by sampling $\mathbf{A} \in \mathbb{Z}_q^{\hat{n}_s \times n}$ uniformly at random and $\mathbf{e}$ from χ^{n_s} and outputting $\mathbf{P'} = [\mathbf{At} + \mathbf{e}| - \mathbf{A}]_q$ is indistinguishable from random. Since $\mathbf{s}$ is an arbitrary but fixed vector, $\mathbf{v} = \mathbf{PowersOfTwo}_q(\mathbf{s})$ is a constant vector. If there was a PPT algorithm $\mathscr{A}$ that could distinguish $\mathbf{P}$ from uniform with non-negligible advantage, then the algorithm $\mathscr{A}'$ that on input $[\mathbf{At} + \mathbf{e}| - \mathbf{A}]_q$ runs $\mathscr{A}$ on $[\mathbf{At} + \mathbf{e} + \mathbf{v}| - \mathbf{A}]_q$ and outputs its result would distinguish $\mathbf{P'}$ from uniform with non-negligible advantage. But this is impossible, so $\mathbf{P}$ is computationally indistinguishable from uniform. □

2.3 Scale-Invariant Homomorphic Encryption

Braskerski's scale invariant leveled FHE scheme consists of the following algorithms for parameters N, χ, B as in the **Regev** scheme, where n is a security parameter and L the number of homomorphic operations which can be performed on the ciphertexts.

- **SI-HE.KeyGen** $\left(1^L, 1^n\right)$: Generate $L+1$ secret keys $\mathbf{s}_0, ..., \mathbf{s}_L$ by running **Regev.SecretKeygen** (1^n), let $\mathbf{P}_0 \leftarrow$ **Regev.PublicKeygen**$(\mathbf{s}_0)$ and for all integers i with $0 < i \leq L$, let

$$\tilde{\mathbf{s}}_{i-1} = \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \otimes \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1}))$$

and

$$\mathbf{P}_{i-1:i} \leftarrow \mathbf{SwitchKeyGen}_q\left(\tilde{\mathbf{s}}_{i-1}, \mathbf{s}_i\right).$$

Output $pk = \mathbf{P}_0$ as the public key, $sk = \mathbf{s}_L$ as the secret key and $evk = (\mathbf{P}_{0:1}, \mathbf{P}_{1:2}, ..., \mathbf{P}_{L-1:L})$ as the evaluation key.

- **SI-HE.Enc**$_{pk}(m)$: The same as **Regev.Enc**.
- **SI-HE.Eval**$_{evk}(C, \mathbf{c}_1, ..., \mathbf{c}_n)$ where C is a depth-L circuit consisting of bitwise '+'- and '$\cdot$'-gates with n input bits: The circuit is evaluated using the algorithms **SI-HE.Add** and **SI-HE.Mult** described below.
- **SI-HE.Dec**$_{sk}(\mathbf{c})$: The same as **Regev.Dec**. It is assumed that $\mathbf{c}$ is the result of L homomorphic evaluations. If it is not, some trivial homomorphic evaluations need to be performed before decryption, or all the keys $\mathbf{s}_1, ..., \mathbf{s}_L$ need to be included in sk.

We will now describe the algorithms for homomorphic addition and multiplication of ciphertexts. Both algorithms take input ciphertexts $\mathbf{c}_1, \mathbf{c}_2$ belonging to the same key $\mathbf{s}_{i-1}$ and compute an intermediate ciphertext $\tilde{\mathbf{c}}$ under $\tilde{\mathbf{s}}_{i-1}$. Then, the switch keys are used to turn it into a ciphertext belonging to the key $\mathbf{s}_i$.

Let us now consider those algorithms in detail.

- **SI-HE.Add**$_{evk,i}(\mathbf{c}_1, \mathbf{c}_2)$: Let

$$\tilde{\mathbf{c}} = \mathbf{PowersOfTwo}_q(\mathbf{c}_1 + \mathbf{c}_2) \otimes \mathbf{PowersOfTwo}_q((1, 0, ..., 0))$$

where the length of the $(1, 0, ..., 0)$-vector is the same as $\mathbf{c}_1 + \mathbf{c}_2$. Output

$$\mathbf{c} = \mathbf{SwitchKey}_q(\mathbf{P}_{i:i+1}, \tilde{\mathbf{c}}).$$

In this algorithm, we add both ciphertexts as described in the beginning of section 2.2 and transform the result into a ciphertext under key $\tilde{\mathbf{s}}_i$ by computing **PowersOfTwo**$_q$ and tensoring the result with a vector that will not affect the outcome of decryption. If we take the scalar product of $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{s}}_i$, we get

$$\begin{aligned}
&\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}_i \rangle \\
=\ &\left\langle \mathbf{PowersOfTwo}_q(\mathbf{c}_1 + \mathbf{c}_2), \mathbf{BitDecomp}_q((1, \mathbf{s}_i)) \right\rangle \\
&\quad \cdot \left\langle \mathbf{PowersOfTwo}_q((1, 0, ..., 0)), \mathbf{BitDecomp}_q((1, \mathbf{s}_i)) \right\rangle \mod q \\
=\ &\langle \mathbf{c}_1 + \mathbf{c}_2, (1, \mathbf{s}_i) \rangle \cdot \langle (1, 0, ..., 0), (1, \mathbf{s}_i) \rangle \mod q \\
=\ &\langle \mathbf{c}_1 + \mathbf{c}_2, (1, \mathbf{s}_i) \rangle
\end{aligned}$$

as required. For a bound on the total error introduced by addition and key-switching, see lemma 2.8.

- **SI-HE.Mult**$_{evk,i}(\mathbf{c}_1, \mathbf{c}_2)$: Let

$$\tilde{\mathbf{c}} = \left\lfloor \frac{2}{q}(\mathbf{PowersOfTwo}_q(\mathbf{c}_1) \otimes \mathbf{PowersOfTwo}_q(\mathbf{c}_2)) \right\rceil$$

and output

$$\mathbf{c} = \mathbf{SwitchKey}_q(\mathbf{P}_{i:i+1}, \tilde{\mathbf{c}}).$$

Here, we have not only errors introduced by multiplication and addition, but also by rounding. Taking the scalar product of $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{s}}_i$ results in

$$\begin{aligned}
\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}_i \rangle &= \left\langle \left\lfloor \frac{2}{q}(\mathbf{PowersOfTwo}_q(\mathbf{c}_1) \otimes \mathbf{PowersOfTwo}_q(\mathbf{c}_2)) \right\rceil, \tilde{\mathbf{s}}_i \right\rangle \\
&\approx \frac{2}{q} \left\langle \mathbf{PowersOfTwo}_q(\mathbf{c}_1), \mathbf{BitDecomp}_q(1, \mathbf{s}_i) \right\rangle \\
&\quad \cdot \left\langle \mathbf{PowersOfTwo}_q(\mathbf{c}_2), \mathbf{BitDecomp}_q(1, \mathbf{s}_i) \right\rangle \\
&= \frac{2}{q} \langle \mathbf{c}_1, (1, \mathbf{s}_i) \rangle \cdot \langle \mathbf{c}_2, (1, \mathbf{s}_i) \rangle
\end{aligned}$$

which is the same formula for multiplication as in section 2.2. For an upper bound on the total error, see again lemma 2.8.

The reduction of the security of this scheme to DLWE follows from a hybrid argument using the security of **Regev** and the fact that switch keys are computationally indistinguishable from random provided that DLWE is hard. We need to show that the distributions D_m $(m = 0, 1)$ given by $(pk, evk, \mathbf{SI\text{-}HE.Enc}_{pk}(m))$ where pk and evk are sampled according to $\mathbf{SI\text{-}HE.KeyGen}(1^n)$ are computationally indistinguishable. It holds that $evk = (\mathbf{P}_{0:1}, \mathbf{p}_{1:2}, ..., \mathbf{P}_{L-1:L})$ where $P_{i:i+1} \leftarrow \mathbf{SwitchKeyGen}_q(\tilde{\mathbf{s}}_i, \mathbf{s}_{i+1})$. Define $evk^{(L)} = evk$ and

$$evk^{(i)} = (\mathbf{P}_{0:1}, ..., \mathbf{P}_{i-1:i}, \mathbf{R}_{i+1}, ..., \mathbf{R}_L)$$

for $0 \leq i < L$ where the $\mathbf{R}_j$ are matrices of the same size as the switch keys but selected uniformly at random. Let D_m^i be the distribution given by $\left(pk, evk^{(i)}, \mathbf{SI\text{-}HE.Enc}_{pk}(m)\right)$.

We now show that $D_m^{(i)}$ is computationally indistinguishable from $D_m^{(i-1)}$. A sample $\left(pk, evk^{(i)}, \mathbf{SI\text{-}HE.Enc}_{pk}(m)\right)$ of $D_m^{(i)}$ does not contain any information which depends on $\mathbf{s}_i$ except for the switch key $\mathbf{P}_{i-1:i}$. Because the secret key $\mathbf{s}_i$ is sampled uniformly at random, the switch key $\mathbf{P}_{i-1:i} \leftarrow \mathbf{SwitchKeyGen}_q(\tilde{\mathbf{s}}_{i-1}, \mathbf{s}_i)$ is computationally indistinguishable from random if average-case $\mathrm{DLWE}_{N,q,\chi}$ is hard. Therefore $D_m^{(i)}$ and $D_m^{(i-1)}$ are computationally indistinguishable.

By the security proof for Regev's scheme, theorem 2.4, we know that, assuming the hardness of average-case $\mathrm{DLWE}_{N,q,\chi}$, the distributions D'_m given by $(pk, \mathbf{SI\text{-}HE.Enc}_{pk}(m))$ for $m = 0, 1$ are computationally indistinguishable. Then also the distributions $D_0^{(0)}$ and $D_1^{(0)}$ must be computationally indistinguishable because they are just D'_0 or D'_1 respectively, together with a tuple of

random matrices. This implies that D_0 and D_1 are computationally indistinguishable which completes the proof of semantic security.

Theorem 2.7 (Security)**.** *If average-case $DLWE_{N,q,\chi}$ is hard, then the scheme **SI-HE** with parameters N, q, χ is semantically secure.*

2.4 Homomorphic Properties of SI-HE

Let us now look at the homomorphic properties of Brakerski's scheme. We show that if

$$\frac{q}{B} \geq O(n \log q)^{L+4}$$

then for any depth-L circuit C and $(pk, sk, evk) \leftarrow \textbf{SI-HE.KeyGen}\left(1^L, 1^n\right)$, the result of

$$\textbf{SI-HE.Eval}_{evk}(C, \textbf{SI-HE.Enc}_{pk}(m_1), ..., \textbf{SI-HE.Enc}_{pk}(m_n))$$

decrypts to $C(m_1, ..., m_n)$ under sk. This result is based on the following lemma:

Lemma 2.8. *Let q, N, χ, B be parameters for **SI-HE** with $N = O(n \log q)$ and $(pk, sk, evk) \leftarrow \boldsymbol{SI\text{-}HE.KeyGen}\left(1^L, 1^n\right)$. For $i \in \{1, ..., L\}$, suppose that $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathbb{Z}_q^{n+1}$ are such that*

$$\langle \boldsymbol{c}_1, (1, \boldsymbol{s}_{i-1}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor m_1 + e_1$$
$$\langle \boldsymbol{c}_2, (1, \boldsymbol{s}_{i-1}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor m_2 + e_2$$

where $|e_1|, |e_2| \leq E$ for some $E < \lfloor q/2 \rfloor / 2$. Then for

$$\boldsymbol{c}_{add} \leftarrow \boldsymbol{SI\text{-}HE.Add}_{evk,i}(\boldsymbol{c}_1, \boldsymbol{c}_2)$$
$$\boldsymbol{c}_{mult} \leftarrow \boldsymbol{SI\text{-}HE.Mult}_{evk,i}(\boldsymbol{c}_1, \boldsymbol{c}_2)$$

it holds that

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

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

The constants hidden in the O-notation will be made explicit in the proof.

Proof. Let us first look at the simpler case, addition. Here, we need to find an upper bound for two types of errors: The error introduced by key-switching, δ_1 and the error term in the sum of the ciphertexts, δ_2. By lemma 2.5, we have

$$\begin{aligned} \langle \mathbf{c}_{add}, (1, \mathbf{s}_i) \rangle &= \left\langle \textbf{SwitchKey}_q(\mathbf{P}_{i-1:i}, \tilde{\mathbf{c}}_{add}), (1, \mathbf{s}_i) \right\rangle \mod q \\ &= \langle \tilde{\mathbf{c}}_{add}, \tilde{\mathbf{s}}_{i-1} \rangle + \langle \textbf{BitDecomp}_q(\tilde{\mathbf{c}}_{add}), \tilde{\mathbf{e}} \rangle \mod q. \end{aligned}$$

where

$$\begin{aligned}\tilde{\mathbf{c}}_{add} =& \mathbf{PowersOfTwo}_q([\mathbf{c}_1 + \mathbf{c}_2]_q) \\ & \otimes \mathbf{PowersOfTwo}_q((1, 0, ..., 0)) \in \mathbb{Z}_q^{(n+1)^2\lceil \log q \rceil^2}\end{aligned}$$

and thus $\mathbf{BitDecomp}_q(\tilde{\mathbf{c}}_{add}) \in \{0,1\}^{(n+1)^2\lceil \log q \rceil^3}$ and $\tilde{\mathbf{e}}$ is sampled from $\chi^{(n+1)^2\lceil \log q \rceil^3}$. Let $\delta_1 = \langle \mathbf{BitDecomp}_q(\tilde{\mathbf{c}}_{add}), \tilde{\mathbf{e}} \rangle$. This is the error introduced by key switching. We can find an upper bound for $|\delta_1|$ the following way: Since $|\chi| \leq B$ and $\mathbf{BitDecomp}_q(\tilde{\mathbf{c}}_{add})$ is a binary vector of length $(n+1)^2\lceil \log q \rceil^3$, it follows that

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

Now we find an upper bound for the error δ_2 in $\langle \tilde{\mathbf{c}}_{add}, \tilde{\mathbf{s}}_{i-1} \rangle$:

$$\begin{aligned}\langle \tilde{\mathbf{c}}_{add}, \tilde{\mathbf{s}}_{i-1} \rangle &= \langle \mathbf{PowersOfTwo}_q([\mathbf{c}_1 + \mathbf{c}_2]_q), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \rangle \\ &\quad \cdot \langle \mathbf{PowersOfTwo}_q((1, 0, ..., 0)), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \\ &= (\langle \mathbf{c}_1, (1, \mathbf{s}_{i-1}) \rangle + \langle \mathbf{c}_2, (1, \mathbf{s}_{i-1}) \rangle) \cdot \langle (1, 0, ..., 0), (1, \mathbf{s}_{i-1}) \rangle \mod q \\ &= \left\lfloor \frac{q}{2} \right\rfloor m_1 + e_1 + \left\lfloor \frac{q}{2} \right\rfloor m_2 + e_2 \mod q \\ &= \left\lfloor \frac{q}{2} \right\rfloor [m_1 + m_2]_2 - \left(- \left\lfloor \frac{q}{2} \right\rfloor (m_1 + m_2 - [m_1 + m_2]_2) \right) + e_1 + e_2 \mod q\end{aligned}$$

Let $\tilde{m} = [-\lfloor q/2 \rfloor (m_1 + m_2 - [m_1 + m_2]_2)]_q$ so that

$$\langle \tilde{\mathbf{c}}_{add}, (1, \tilde{\mathbf{s}}_{i-1}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor [m_1 + m_2]_2 - \tilde{m} + e_1 + e_2 \mod q.$$

Note that $m_1 + m_2 - [m_1 + m_2]_2 = 2$ if and only if $m_1 = m_2 = 1$. Otherwise, it equals 0. Therefore, if q is even, $\tilde{m} \in \{0, [-q]_q\} = \{0\}$. If q is odd and $m_1 = m_2 = 1$, then $\tilde{m} = [-(q-1)]_q = 1 = (1/2)(m_1 + m_2 - [m_1 + m_2]_2)$. This gives us

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

therefore $\tilde{m} \in \{0, 1\}$ and we get the following bound for the error δ_2:

$$|\delta_2| = | - \tilde{m} + e_1 + e_2| \leq 1 + 2E.$$

Setting $e_{add} = \delta_1 + \delta_2$, we have

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

and

$$\begin{aligned}|e_{add}| = |\delta_1 + \delta_2| &\leq (n+1)^2\lceil \log q \rceil^3 B + 1 + 2E \\ &= O(n \log q)(n+1) \log^2 q \cdot B + O(1) \cdot E \\ &\leq O(n \log q) \cdot \max((n+1) \log^2 q \cdot B, E) \\ &= O(n \log q) \cdot \max(O(n \log^2 q) B, E)\end{aligned}$$

This shows that the lemma holds for addition. Now for multiplication: Here, we have an error term from key-switching, δ_1, from rounding, δ_2 and from multiplying the ciphertexts, δ_3. The key-switching error is the same as in the case of addition.

$$\langle \mathbf{c}_{mult}, (1, \mathbf{s}_i) \rangle = \langle \tilde{\mathbf{c}}_{mult}, \tilde{\mathbf{s}}_{i-1} \rangle + \delta_1 \mod q$$

where $\delta_1 = \langle \mathbf{BitDecomp}_q(\tilde{\mathbf{c}}_{mult}), \tilde{\mathbf{e}} \rangle$ is the error introduced by key switching. As before, $|\delta_1| \leq (n+1)^2 \lceil \log q \rceil^3 B$. Now consider

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

We need to make use of the properties of the tensor to proceed, but since the tensor product is rounded, we first need to take the error introduced by rounding into account. For some vector $\mathbf{c}' \in (-1/2, 1/2]^{(n+1)^2 \lceil \log q \rceil^2}$, we have

$$\begin{aligned} \langle \tilde{\mathbf{c}}_{mult}, \tilde{\mathbf{s}}_{i-1} \rangle = &\left\langle \frac{2}{q} (\mathbf{PowersOfTwo}_q(\mathbf{c}_1) \otimes \mathbf{PowersOfTwo}_q(\mathbf{c}_2)), \tilde{\mathbf{s}}_{i-1} \right\rangle \\ &+ \langle \mathbf{c}', \tilde{\mathbf{s}}_{i-1} \rangle \end{aligned}$$

Let $\delta_2 = \langle \mathbf{c}', \tilde{\mathbf{s}}_{i-1} \rangle$. Since $\tilde{\mathbf{s}}_{i-1} = \mathbf{BitDecomp}_q(\mathbf{s}_{i-1}) \otimes \mathbf{BitDecomp}_q(\mathbf{s}_{i-1})$ is a binary vector of length $(n+1)^2 \lceil \log q \rceil^2$, it follows that

$$|\delta_2| \leq \frac{(n+1)^2 \lceil \log q \rceil^2}{2}$$

Splitting up the tensor product, we get

$$\begin{aligned} \langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}_{i-1} \rangle - \delta_2 = &\frac{2}{q} \langle \mathbf{PowersOfTwo}_q(\mathbf{c}_1), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \rangle \\ &\cdot \langle \mathbf{PowersOfTwo}_q(\mathbf{c}_2), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \rangle \end{aligned}$$

Using the fact that $\langle \mathbf{PowersOfTwo}_q(\mathbf{c}_b), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \rangle = \langle \mathbf{c}_b, (1, \mathbf{s}_{i-1}) \rangle \mod q$ and our assumption that $\langle \mathbf{c}_b, (1, \mathbf{s}_{i-1}) \rangle = \lfloor q/2 \rfloor m_b + e_b \mod q$, we can conclude that there are integers I_1, I_2 such that

$$\begin{aligned} \langle \mathbf{PowersOfTwo}_q(\mathbf{c}_1), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor m_1 + e_1 + qI_1 \\ \langle \mathbf{PowersOfTwo}_q(\mathbf{c}_2), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor m_2 + e_2 + qI_2 \end{aligned}$$

and hence,

$$
\begin{aligned}
&\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}_{i-1} \rangle - \delta_2 \\
&= \frac{2}{q} \left(\left\lfloor \frac{q}{2} \right\rfloor m_1 + e_1 + qI_1 \right) \left(\left\lfloor \frac{q}{2} \right\rfloor m_2 + e_2 + qI_2 \right) \\
&= \frac{2}{q} \left(\left\lfloor \frac{q}{2} \right\rfloor^2 m_1 m_2 + \left\lfloor \frac{q}{2} \right\rfloor m_1 e_2 + \left\lfloor \frac{q}{2} \right\rfloor m_1 q I_2 + \left\lfloor \frac{q}{2} \right\rfloor e_1 m_2 + e_1 e_2 + q e_1 I_2 \right. \\
&\qquad \left. + \left\lfloor \frac{q}{2} \right\rfloor q I_1 m_2 + q I_1 e_2 + q^2 I_1 I_2 \right) \\
&= \left\lfloor \frac{q}{2} \right\rfloor m_1 m_2 - \frac{2}{q} \left(\frac{q}{2} - \left\lfloor \frac{q}{2} \right\rfloor \right) \left\lfloor \frac{q}{2} \right\rfloor m_1 m_2 + m_1 e_2 - \frac{2}{q} \left(\frac{q}{2} - \left\lfloor \frac{q}{2} \right\rfloor \right) m_1 e_2 \\
&\qquad + q m_1 I_2 - 2 \left(\frac{q}{2} - \left\lfloor \frac{q}{2} \right\rfloor \right) m_1 I_2 + e_1 m_2 - \frac{2}{q} \left(\frac{q}{2} - \left\lfloor \frac{q}{2} \right\rfloor \right) e_1 m_2 \\
&\qquad + \frac{2 e_1 e_2}{q} + 2 e_1 I_2 + q I_1 m_2 - 2 \left(\frac{q}{2} - \left\lfloor \frac{q}{2} \right\rfloor \right) I_1 m_2 + 2 I_1 e_2 + 2 q I_1 I_2 \\
&= \left\lfloor \frac{q}{2} \right\rfloor m_1 m_2 + q \cdot (I_1 m_2 + I_2 m_q + 2 I_1 I_2) + \delta_3 \\
&= \left\lfloor \frac{q}{2} \right\rfloor m_1 m_2 + \delta_3 \mod q
\end{aligned}
$$

where

$$
\begin{aligned}
\delta_3 = &\left(\frac{q}{2} - \left\lfloor \frac{q}{2} \right\rfloor \right) \cdot \left(-\frac{2}{q} \left\lfloor \frac{q}{2} \right\rfloor m_1 m_2 - \frac{2}{q} m_1 e_2 - \frac{2}{q} e_1 m_2 - 2 m_1 I_2 - 2 m_2 I_1 \right) \\
&+ m_1 e_2 + e_1 m_2 + \frac{2 e_1 e_2}{q} + 2 e_1 I_2 + 2 I_1 e_2
\end{aligned}
$$

To find an upper bound for δ_3, we need to find one for I_1, I_2. For $b = 0, 1$, it holds that

$$
\left\langle \mathbf{PowersOfTwo}_q(\mathbf{c}_b), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \right\rangle \leq \frac{q}{2}(n+1)\lceil \log q \rceil
$$

since the entries of the first vector in the scalar product are in $\mathbb{Z}_q$ and the second one is a binary vector of length $(n+1)\lceil \log q \rceil$. Therefore, we get

$$
\begin{aligned}
|I_b| &= \frac{|\left\langle \mathbf{PowersOfTwo}_q(\mathbf{c}_b), \mathbf{BitDecomp}_q((1, \mathbf{s}_{i-1})) \right\rangle - \left\lfloor \frac{q}{2} \right\rfloor m_b - e_b|}{q} \\
&\leq \frac{1}{2}(n+1)\lceil \log q \rceil + \frac{1}{q} \left\lfloor \frac{q}{2} \right\rfloor m_b + \frac{E}{q} \\
&\leq \frac{1}{2}(n+1)\lceil \log q \rceil + \frac{1}{2} + \frac{1}{4} \\
&= \frac{1}{2}(n+1)\lceil \log q \rceil + \frac{3}{4}
\end{aligned}
$$

Since $(q/2) - \lfloor q/2 \rfloor = 0$ or $= 1/2$,

$$\begin{aligned}
|\delta_3| &\leq \frac{1}{2} \cdot \left(1 + \frac{4E}{q} + 2I_1 + 2I_2\right) + 2E + \frac{2E^2}{q} + 2EI_1 + 2EI_2 \\
&\leq \frac{1}{2} + \frac{2E + 2E^2}{q} + (2 + 4E)\left(\frac{1}{2}(n+1)\lceil \log q \rceil + \frac{3}{4}\right) \\
&< 1 + \frac{E}{2} + 2(2E+1)\left(\frac{1}{2}(n+1)\lceil \log q \rceil + \frac{3}{4}\right)
\end{aligned}$$

using $E < \lfloor q/2 \rfloor/2 \leq q/4$ for the last inequality. Taking all three error terms together, we have

$$\begin{aligned}
|e_{mult}| &= |\delta_1 + \delta_2 + \delta_3| \\
&< (n+1)^2 \lceil \log q \rceil^3 B + \frac{(n+1)^2 \lceil \log q \rceil^2}{2} \\
&\quad + 1 + \frac{E}{2} + 2(2E+1)\left(\frac{1}{2}(n+1)\lceil \log q \rceil + \frac{3}{4}\right) \\
&= O(n \log q) O(n \log^2 q) B + O(n^2 \log^2 q) + O(n \log q) E \\
&\leq O(n \log q) \max(E, O(n \log^2 q) B)
\end{aligned}$$

which completes the proof. □

From this lemma, the following theorem follows by induction:

Theorem 2.9. *Let N, q, χ, B be parameters for **SI-HE** with $N = O(n \log q)$ and $|\chi| \leq B < \lfloor q/2 \rfloor/2$ and*

$$\frac{q}{B} \geq O(n \log q)^{L+4}.$$

*Then for n large enough, the scheme **SI-HE** is L-homomorphic when the keys are generated by **SI-HE.KeyGen**$\left(1^L, 1^n\right)$.*

Proof. Consider the homomorphic evaluation of a depth-L circuit C. Let E_i be the maximum of the absolute values of the error terms after evaluating all gates at depth i. It holds that

$$E_0 \leq NB = O(n \log q) B \leq q \cdot O(n \log q)^{-L-3} < \left\lfloor \frac{q}{2} \right\rfloor /2$$

for n large enough. Using the lemma, we show that if $E_i < \lfloor q/2 \rfloor/2$ and $E_i \leq O(n \log q)^{i+3} B$ then $E_{i+1} < \lfloor q/2 \rfloor/2$ and $E_{i+1} \leq O(n \log q)^{i+4} B$ as long as $i < L$. First, let us consider the case that $(n \log^2 q) B \geq E_i$. Then, it follows from lemma 2.8 that

$$E_{i+1} \leq O(n \log q)^3 B \leq q \cdot O(n \log q)^{-L-1} < \left\lfloor \frac{q}{2} \right\rfloor /2$$

If $E_i > (n \log^2 q)B$, then

$$\begin{aligned} E_{i+1} \leq O(n \log q) O(n \log q)^{i+3} B &= O(n \log q)^{i+4} B \\ &\leq q \cdot O(n \log q)^{-L-i} \\ &< \left\lfloor \frac{q}{2} \right\rfloor /2 \end{aligned}$$

which proves that $E_L < \lfloor q/2 \rfloor /2$. Since the decryption algorithm in **SI-HE** is the same as in **Regev** and this algorithm decrypts correctly if the error is smaller than $\lfloor q/2 \rfloor /2$, it follows that the decryption of the result of the homomorphic evaluation will be correct. □

2.5 Applications of the Homomorphic Properties

If we allow q to depend on both n and L, we can make **SI-HE** a leveled fully homomorphic encryption scheme. However, it can not be made fully homomorphic, because we have no guarantee that decryption will succeed for an unlimited number of homomorphic evaluations. Craig Gentry's bootstrapping theorem allows to modify the evaluation algorithm so that it becomes a leveled fully homomorphic encryption scheme with q independent of L or a fully homomorphic encryption scheme. However, proving the latter scheme secure requires an additional assumption, circular security.

Theorem 2.10 (Bootstrapping)**.** *Let Π be a homomorphic encryption scheme whose decryption algorithm can be computed by a family of circuits $(C_n)_{n \in \mathbb{N}}$ where n is the security parameter supplied at key generation. Let L_n be the depth of C_n. If the results of its evaluation algorithm decrypt correctly for circuits with addition- and multiplication-gates of depth $L_n + 1$ for all n, then the evaluation algorithm can be modified to a leveled fully homomorphic scheme Π' which is cpa-secure if Π is. It can also be modified to a fully homomorphic scheme Π'' which is cpa-secure if Π is cpa-secure and circular secure. Informally, circular security means that knowing an encryption of the secret key does not help an adversary.*

Proof. To create Π' from $\Pi = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}, \mathbf{Eval})$, we let $\mathbf{Gen}'(1^L, 1^n)$ be an algorithm that generates L public key - secret key - evaluation key triples

$$(pk_0, sk_0, evk_0), ..., (pk_{L-1}, sk_{L-1}, evk_{L-1})$$

using $\mathbf{Gen}(1^n)$ and outputs $sk = (sk_0, ..., sk_{L-1})$, $pk = pk_0$ and

$$\begin{aligned} evk = &(pk_0, evk_0, pk_1, evk_1, \mathbf{Enc}_{pk_1}(sk_0), pk_2, evk_2, \mathbf{Enc}_{pk_2}(sk_1), ..., \\ &pk_{L-1}, evk_{L-1}, \mathbf{Enc}_{pk_{L-1}}(sk_{L-2})) \end{aligned}$$

where $\mathbf{Enc}_{pk_{l+1}}(sk_l)$ denotes the bitwise encryption of sk_l (preserving their order). The encryption algorithm $\mathbf{Enc}'$ remains the same as $\mathbf{Enc}$. Decryption is accomplished by using $\mathbf{Dec}$ with the appropriate secret key from the tuple sk.

This requires some indication about how many homomorphic evaluations have been performed on the ciphertext. We now show how $\mathbf{Eval}'$ homomorphically evaluates a gate G with inputs c_0 and c_1 such that for some $m_0, m_1 \in \{0,1\}$, $\mathbf{Dec}_{sk_l}(c_b) = m_b$ for $b = 0, 1$. Our goal is to compute c_G which decrypts to $G(m_0, m_1)$ under sk_{l+1}. We can then conclude by induction that **Eval** correctly evaluates depth-L circuits.

First, compute $c'_b \leftarrow \mathbf{Enc}_{pk_{l+1}}(c_b)$ for $b = 0, 1$. Now, c'_0, c'_1 have an outer encryption (by pk_{l+1}) and an inner encryption (which can be removed by secret key sk_l). Since we have an encryption of sk_l under pk_{l+1} in evk, we can use the algorithm $\mathbf{Eval}_{evk_{l+1}}$ to homomorphically decrypt the inner encryption. Removing the inner encryption removes the noise that was generated by the inner encryption and subsequent homomorphic evaluations and "refreshes" the ciphertext. However, the homomorphic evaluation of the decryption circuit adds noise of its own; if that noise is not too large the result will be a ciphertext with reduced noise. If **Eval** is able to correctly evaluate two copies of the decryption circuit joined by G, we can do the following:

Let G' be the circuit that results from placing a copy of the decryption circuit at each input wire of G. The result of the gate evaluation is

$$c_G \leftarrow \mathbf{Eval}_{evk_{l+1}}\left(G', \mathbf{Enc}_{pk_{l+1}}(sk_l), c'_0, \mathbf{Enc}_{pk_{l+1}}(sk_l), c'_1\right)$$

and since we supposed that G' is correctly evaluated, it will decrypt to $G(m_0, m_1)$ under sk_{l+1}.

Since circuits of depth 0 can trivially be evaluated, we have a base for the induction. Now suppose that we can correctly evaluate circuits of depth $k < L$, receiving ciphertexts which can be correctly decrypted by sk_k. We need to show that we can evaluate circuits of depth $k + 1$. Let C be a circuit of this depth. Without loss of generality, C consists of two circuits C_0 and C_1 of depth k joined by a gate G, because if one circuit is more shallow, we can augment its depth with gates performing trivial operations. Let c_0 and c_1 be the results of the homomorphic evaluation of C_0 and C_1. The gate-evaluation procedure described above requires only that the input ciphertexts decrypt correctly under key sk_k, which holds by induction hypothesis. If we apply this procedure, we receive a ciphertext c that decrypts under sk_{k+1} to the correct result.

Now that we have proved that the scheme is leveled fully homomorphic, let us focus on its security. Let $\mathscr{A}$ be an adversary for Π'. There is no difference between ciphertexts under Π and Π'. Therefore, the only difference between adversaries for Π' and Π is that an adversary for Π' is given a longer evaluation key containing several public keys, evaluation keys and encrypted secret keys of Π. If Π is cpa-secure, then from the point of view of $\mathscr{A}$, $\mathbf{Enc}_{pk_{L-1}}(sk_{L-2})$ is indistinguishable from an encryption of an all-zero string of the same length as sk_{L-2} since $\mathscr{A}$ is given no input that depends on sk_{L-1}. With a hybrid argument we can conclude that replacing the encrypted secret keys with encryptions of all-zero strings can at most reduce $\mathscr{A}$'s advantage by a negligible amount. Thus, an adversary for Π has all it needs to simulate the "world" of an adversary for Π' and use it to break Π. So cpa-security of Π implies cpa-security of Π'.

The scheme Π'' changes the following: $\mathbf{Gen}''(1^n)$ computes $(pk, sk, evk) \leftarrow \mathbf{Gen}(1^n)$ and outputs $(pk, sk, (evk, pk, \mathbf{Enc}_{pk}(sk)))$. The evaluation of a gate in $\mathbf{Eval}''$ works in the same way as in $\mathbf{Eval}'$ except that pk replaces pk_l and pk_{l+1} and sk replaces sk_l and sk_{l+1}. This allows us to evaluate circuits of unlimited depth if Π is able to evaluate circuits of depth $L_n + 1$. However, we now need to assume circular security of Π to prove that Π'' is secure. □

Cpa-security of Π does not guarantee that $\mathbf{Enc}_{pk}(sk)$ is indistinguishable from $\mathbf{Enc}_{pk}(0...0)$: Consider a cpa-secure scheme where key- and message-length are the same. If $\mathbf{Enc}_{pk}(sk)$ is not given to the adversary, the only way to obtain it would be to guess sk and encrypt it which can happen only with negligible probability since the scheme was cpa-secure. But if the encryption algorithm is such that $\mathbf{Enc}_{pk}(sk) = sk$ for any key-pair (pk, sk), giving an adversary $\mathbf{Enc}_{pk}(sk)$ breaks the scheme completely. It has been conjectured that bit encryption schemes are circular secure, however, see [14] for a bit encryption scheme whose secret key can be completely recovered when given an encryption of it even though it is cpa-secure given an extension of the Symmetric External Diffie-Hellman (SXDH) assumption[1] to l-multilinear groups for all l. This is not an assumption which is usually believed to be true but it is not known to be false either. To prove the conjecture about bit encryption, this assumption must be proved false.

To select q and L in **SI-HE** such that it can be bootstrapped, we need to know the depth of the decryption circuit. In [4], Lemma 4.5 shows that the decryption algorithm used in **SI-HE** can be computed by a depth-$O(\log n \log\log q)$ circuit. This shows that for $q/B \geq O(n \log q)^{O(\log n \log\log q)}$ we can make **SI-HE** bootstrappable if we fix L accordingly.

3 Reduction of GapSVP to LWE

3.1 Lattices

Let $\mathbf{B} \in \mathbb{R}^{n \times m}$ be a matrix whose columns are linear independent vectors in $\mathbb{R}^n$. The lattice generated by $\mathbf{B}$ is

$$\mathscr{L}(\mathbf{B}) = \{\mathbf{Bz} \,|\mathbf{z} \in \mathbb{Z}^m\}$$

and we say that $\mathbf{B}$ is a (lattice) basis for $\mathscr{L}(\mathbf{B})$. A set $L \subseteq \mathbb{R}^n$ is a lattice iff it is a lattice generated by some basis or equivalently, iff it is a discrete additive subgroup of $\mathbb{R}^n$. If L is generated by $\mathbf{B} \in \mathbb{R}^{n \times n}$ it is called a lattice of full rank. Lattices in this sense have no relation to lattices in Order Theory.

[1] Let G_0, G_1, G be circular groups of prime order p, q, r respectively. A bilinear map from $G_0 \times G_1$ to G is a non-degenerate map f such that for $g_0 \in G_0$ and $g_1 \in G_1$, we have $f(g_0^a, g_1) = f(g_0, g_1)^a$ and $f(g_0, g_1^a) = f(g_0, g_1)^a$. The SXDH assumption states that there are (families of) groups G_0, G_1, G such that there is a bilinear map from $G_0 \times G_1$ to G where the Decisional Diffie-Hellman assumption holds for both G_0 and G_1. The definition of l-multilinear maps and of the extended SXDH assumption is immediate.

The dual of a lattice L is defined as

$$L^* = \{\mathbf{y} \in span(\mathbf{B}) | \forall \mathbf{x} \in L : \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$$

where $span(\mathbf{B})$ is the subspace of $\mathbb{R}^n$ spanned by the column vectors of $\mathbf{B}$. For square $\mathbf{B}$, this is $\mathbb{R}^n$ itself. If $\mathbf{B}$ is a basis of L then $\mathbf{B}\left(\mathbf{B}^\top\mathbf{B}\right)^{-1}$ is a basis of L^*. For square $\mathbf{B}$, this basis for L^* equals $\left(\mathbf{B}^\top\right)^{-1}$. For any lattice L, it holds that $L = L^{**}$. From now on, we only consider lattices of full rank.

If we let $\mathbf{D} = \left(\mathbf{B}^{-1}\right)^\top$ and $\bar{\mathbf{b}}_1, ..., \bar{\mathbf{b}}_n$ the Gram-Schmid orthogonalization of $\mathbf{B} = (\mathbf{b}_1, ..., \mathbf{b}_n)$, we have $\bar{\mathbf{d}}_i = \bar{\mathbf{b}}_i / ||\bar{\mathbf{b}}_i||^2$. It follows that $||\bar{\mathbf{d}}_i|| = 1/||\bar{\mathbf{b}}_i||$. This holds because $\mathbf{d}_i$ is the ith row of $\mathbf{B}^{-1}$ and thus $\langle \mathbf{d}_i, \mathbf{b}_i \rangle = 1$.

Given a lattice basis $\mathbf{B}$, the set

$$\mathscr{P}(\mathbf{B}) = \{\mathbf{B}\mathbf{v} \,|\mathbf{v} \in [0,1)^n\}$$

is called the fundamental paralleliped of $\mathbf{B}$. For a vector $\mathbf{v} \in \mathbb{R}^n$, we define $\mathbf{v}$ mod $\mathscr{P}(\mathbf{B})$ to be the unique point $\mathbf{w}$ in $\mathscr{P}(\mathbf{B})$ such that $\mathbf{v} = \mathbf{w} + \mathbf{x}$ for some $\mathbf{x} \in \mathscr{L}(\mathbf{B})$.

An important lattice parameter is $\lambda(L) = \min\{||\mathbf{v}|| \ | \mathbf{v} \in L \setminus \{0\}\}$, the length of the shortest vector in L. This parameter is well defined because L is discrete and thus, any sequence $\mathbf{v}_0, \mathbf{v}_1, ...$ converging to 0 must have $\mathbf{v}_i = 0$ for all i larger than some i_0. Also, the distance between two distinct lattice vectors $\mathbf{x}$ and $\mathbf{y}$ is at least $\lambda(L)$ because otherwise, the lattice vector $\mathbf{x} - \mathbf{y}$ would be shorter than $\lambda(L)$. For an arbitrary vector $\mathbf{t} \in \mathbb{R}^n$, the distance of $\mathbf{t}$ from lattice L is defined as $dist(\mathbf{t}, \mathbf{B}) = \min\{||\mathbf{t} - \mathbf{x}|| \ | \mathbf{x} \in L\}$.

Let us now look at the shortest vector problem (SVP). In fact, SVP is a family of problems: Search-SVP is the problem to find a vector of minimal length in $\mathscr{L}(\mathbf{B})$ on input $\mathbf{B}$. Optimization-SVP is the problem to find $\lambda(\mathscr{L}(\mathbf{B}))$ on input $\mathbf{B}$. Finally, decision-SVP is the problem to decide on input $(\mathbf{B}, d)$ whether $\lambda(\mathscr{L}(\mathbf{B})) \leq d$. These problems are NP-hard under randomized and non-uniform reductions. ([8], Chapter 4).

Easier variants of these problems only require approximate solutions within some factor γ. For some function $\gamma : \mathbb{N} \to \mathbb{R}^+$, the approximate version of decision-SVP can be formulated as the promise problem GapSVP$_\gamma$. A probabilistic algorithm $\mathscr{A}$ that outputs 0 or 1 solves this problem if it satisfies the following conditions:

- On input $(\mathbf{B}, d)$ with $\lambda(\mathscr{L}(\mathbf{B})) \leq d$, it outputs 1 with probability $\geq \frac{2}{3}$. Such inputs are called YES-instances of GapSVP$_\gamma$.
- On input $(\mathbf{B}, d)$ with $\mathscr{L}(\mathbf{B})$ of rank n and $\lambda(\mathscr{L}(\mathbf{B})) > \gamma(n) \cdot d$, it outputs 1 with probability $< \frac{1}{3}$. Such inputs are called NO-instances of GapSVP$_\gamma$.

On all other inputs, the output of $\mathscr{A}$ is irrelevant. So far, no polynomial-time classical or quantum algorithm that solves GapSVP$_\gamma$ for γ subexponential in n has been found. Another variant of GapSVP is GapSVP$_{\gamma,\zeta}$ which adds a promise

that the shortest vector has length at most ζ. That is, a valid input is $(\mathbf{B}, d)$ such that $\lambda(\mathscr{L}(\mathbf{B})) \leq \zeta$, $\min_i ||\bar{\mathbf{b}}_i|| \geq 1$ and $1 \leq d \leq \zeta/\gamma$; a YES-instance has $\lambda(\mathscr{L}(\mathbf{B})) \leq d$ in addition and a NO-instance has $\lambda(\mathscr{L}(\mathbf{B})) > \gamma d$. We can achieve the condition $\min_i ||\bar{\mathbf{b}}_i|| \geq 1$ by scaling the lattice and for $d > \zeta/\gamma$ the input is a NO-instance if $\lambda(\mathscr{L}(\mathbf{B})) \leq \zeta$. Therefore, these two additional conditions are no additional restrictions. For $\gamma \geq 2^{n/2}$, this problem is equivalent to GapSVP_γ because the lattice and we can use the LLL-algorithm on a basis $\mathbf{B}$ to obtain a basis $\mathbf{B}'$ whose first vector is shorter than $2^{n/2} \cdot \min_i ||\bar{\mathbf{b}}_i'||$. Scaling the lattice and d by a factor of $1/\min ||\bar{\mathbf{b}}_i||$ yields a valid input for $\text{GapSVP}_{\gamma,\zeta}$.

Another important class of problems are the closest vector problems (CVP). Given a basis $\mathbf{B}$ and a vector $\mathbf{t}$, the goal is to find a vector $\mathbf{x} \in \mathscr{L}(\mathbf{B})$ with minimum distance from $\mathbf{t}$. Similar to SVP, optimization and decision variants, as well as approximate variants of this problem can be formulated.

3.2 Gaussian Distributions on Lattices and the Smoothing Parameter

Given $\epsilon > 0$, the *smoothing parameter* $\eta_\epsilon(L)$ is the least s such that $\rho_{1/s,\mathbf{0}}(L^* \setminus \{\mathbf{0}\}) < \epsilon$. Its name derives from the fact that for any lattice $L = \mathscr{L}(\mathbf{B})$ and $s \geq \eta_\epsilon(L)$, the distribution $D_{s,\mathbf{c}} \mod \mathscr{P}(\mathbf{B})$ and the uniform distribution $U(\mathscr{P}(\mathbf{B}))$ have statistical distance $\epsilon/2$. (see [10], Lemma 4.1)

We have the following upper bound on $\eta_{2^{-n}}(L)$ ([10], Lemma 3.2):

$$\eta_{2^{-n}}(L) \leq \frac{\sqrt{n}}{\lambda(L^*)}$$

3.3 The Reduction

There are two reductions of GapSVP_γ for γ slightly superpolynomial to $\text{LWE}_{q,\Psi_\alpha}$, the first one is a quantum reduction by Oded Regev in [13], the second one is classical and was found by Chris Peikert in [11]. The first step in Regev's reduction was to find for $\epsilon(n)$ negligible in n, $\alpha(n) \in (0,1)$, $q(n) \geq 2$ and $N = O(n)$ a classical probabilistic algorithm $R^{W,D}$ that does the following:

A valid input for R is $(\mathbf{B}, r, \mathbf{x})$ with $\mathbf{B}$ a $n \times n$ lattice basis, $r \geq \sqrt{2}q \cdot \eta_\epsilon(L^*)$ and $\mathbf{x} \in \mathbb{R}^n$ having distance at most $\alpha q/(\sqrt{2}r)$ from $L = \mathscr{L}(\mathbf{B})$. If it has access to an oracle W that solves $\text{LWE}_{q,\Psi_\alpha}$ given a polynomial number of samples and an oracle D that samples from $D_{\mathscr{L}(\mathbf{B})^*,r,\mathbf{0}}$, it outputs a vector $\mathbf{y} \in L$ with minimal distance to $\mathbf{x}$ with overwhelming probability. On other inputs, it may output anything. The algorithm R will be described in the next section. For now, we use it as a "black box".

In [12], an algorithm is described that, given a lattice basis $\mathbf{B}$ and $r \geq \omega(\log n) \cdot \max_i ||\bar{\mathbf{b}}_i||$ samples a distribution with negligible statistical distance to $D_{\mathscr{L}(\mathbf{B}),r,\mathbf{0}}$, so for r large enough, R only needs the LWE-oracle.

It is not immediately obvious how R could be useful. As Regev notes, the only sure way to create valid inputs seems to be to take a lattice vector $\mathbf{v}$ and disturb it by some small amount. But then we already know that R will output $\mathbf{v}$

with overwhelming probability and learn nothing from its output. Nevertheless, this algorithm is useful in the setting of quantum computation, because it allows us to "uncompute" $\mathbf{v}$ from $\mathbf{v}+\mathbf{w}$ for $\mathbf{w}$ small enough. This is what R is used for in Regev's reduction.

The key idea of Peikert's classical reduction is that the size of the disturbance which R can handle correctly contains information on $\lambda(L)$ and that we can use the "useless" procedure described above to test for which disturbance vectors $\mathbf{w}$ the algorithm returns $\mathbf{v}$ on input $\mathbf{v}+\mathbf{w}$.

The reduction from $\mathrm{GapSVP}_{\gamma,\zeta}$ with $\gamma \geq n/(\alpha\cdot\sqrt{\log n})$ to $\mathrm{LWE}_{q,\Psi_\alpha}$ with $q \geq (\zeta/\sqrt{n})\cdot\omega(\log n)$ proceeds as follows: The input is $(\mathbf{B}, d)$ such that $\min_i ||\bar{\mathbf{b}}_i|| \geq 1$, $1 \leq d \leq \zeta/\gamma$ and $\lambda(\mathscr{L}(\mathbf{B})) \leq \zeta$. Repeat the following procedure N times where N is some large polynomial in n.

1. Let $d' = d\cdot\sqrt{n/(4\log n)}$. Sample $\mathbf{w}$ from the uniform distribution on $d'\cdot B_n$ where B_n is the n-dimensional unit ball and define $\mathbf{x} = \mathbf{w} \mod \mathscr{P}(\mathbf{B})$.

2. Run the algorithm R on input $(\mathbf{B}, r, \mathbf{x})$ for

$$r = \frac{\sqrt{2n}q}{\gamma\cdot d}$$

   where the oracle that samples $D_{\mathscr{L}(\mathbf{B})^*,r,\mathbf{0}}$ is implemented by sampling the algorithm from [12] mentioned above. If the result differs from $\mathbf{x}-\mathbf{w}$, accept.

If the algorithm did not accept after N iterations, reject.

We can replace the oracle for $D_{\mathscr{L}(\mathbf{B})^*,r,\mathbf{0}}$ because we have $\max_i ||\bar{\mathbf{d}}_i|| \leq 1/\min_i ||\bar{\mathbf{b}}_i|| \leq 1$ for the dual basis $\mathbf{D} = \left(\mathbf{B}^{-1}\right)^\top$ and therefore

$$r = \frac{\sqrt{2n}q}{\gamma\cdot d} \geq \frac{\sqrt{2n}q}{\zeta} \geq \omega(\log n) \geq \omega(\log n)\max_i ||\bar{\mathbf{d}}_i||$$

as required. Now suppose that $(\mathbf{B}, d)$ is a NO-instance. It suffices to show that the parameters for R are such that it has to output $\mathbf{x}-\mathbf{w}$ with overwhelming probability. Since $d > \gamma\cdot\lambda(\mathscr{L}(\mathbf{B}))$, it follows that

$$\eta_{2^{-n}}(\mathscr{L}(\mathbf{B})^*) \leq \frac{\sqrt{n}}{\gamma\cdot d}$$

so $r \geq \sqrt{2}q\eta_{2^{-n}}$. Since $\mathbf{x} = \mathbf{w} \mod \mathscr{P}(\mathbf{B})$, $\mathbf{x}-\mathbf{w} \in \mathscr{L}(\mathbf{B})$ and the distance between $\mathbf{x}$ and $\mathbf{x}-\mathbf{w}$ is

$$||\mathbf{w}|| \leq d' = d\cdot\sqrt{\frac{n}{\log 4n}} \leq \frac{\alpha\cdot\gamma\cdot d}{\sqrt{4n}} = \frac{\alpha q}{\sqrt{2}r}.$$

Since $\lambda(\mathscr{L}(\mathbf{B})) > \gamma d \geq 2d'$, $\mathbf{x}-\mathbf{w}$ is the unique closest vector to $\mathbf{x}$. This proves that R will output $\mathbf{x}-\mathbf{w}$ with overwhelming probability, so the algorithm described above rejects with overwhelming probability.

Now suppose $(\mathbf{B}, d)$ is a YES-instance, that is, $d \leq \lambda(\mathscr{L}(\mathbf{B}))$. Let $\mathbf{y}$ be a vector in $\mathscr{L}(\mathbf{B})$ be a vector of length $\lambda(\mathscr{L}(\mathbf{B}))$. By lemma 3.6 and the following discussion in [Goldreich, Goldwasser 1998], for any constants $c, d > 0$ and $\mathbf{y} \in \mathscr{L}(\mathbf{B})$ with $||\mathbf{y}|| \leq d$ and $d' = d \cdot \sqrt{n/(c \log n)}$, the statistical distance between the uniform distributions on $d' \cdot B_n$ and $d' \cdot B_n + \mathbf{y}$ is $1 - 1/\text{poly}(n)$. Let $\mathbf{w}' = \mathbf{w} + \mathbf{y}$. Since $\mathbf{y} \in \mathscr{L}(\mathbf{B})$, it holds that $\mathbf{x}' = \mathbf{w}' \mod \mathscr{P}(\mathbf{B}) = \mathbf{x}$. Since statistical distance does not increase under any function, we have

$$\begin{aligned}
\Pr\left[R(\mathbf{x}) = \mathbf{w} - \mathbf{x}\right] &\leq 1 - \frac{1}{\text{poly}(n)} + \Pr\left[R(\mathbf{x}') = \mathbf{w}' - \mathbf{x}'\right] \\
&= 2 - \frac{1}{\text{poly}(n)} + \Pr\left[R(\mathbf{x}) \neq \mathbf{w}' - \mathbf{x}\right] \\
&\leq 2 - \frac{1}{\text{poly}(n)} + \Pr\left[R(\mathbf{x}) = \mathbf{w} - \mathbf{x}\right]
\end{aligned}$$

It follows that $\Pr\left[R(\mathbf{x}) = \mathbf{w} - \mathbf{x}\right] \leq 1 - 1/\text{poly}(n)$. Therefore, for N large enough, R will output something else than $\mathbf{x} - \mathbf{w}$ on one iteration with overwhelming probability.

3.4 Description of R

We now complete the reduction by giving a description of the algorithm R from [13]. First, we need some additional reductions. The first one shows that it suffices to compute the coefficients of the vector returned by R modulo q for any $q \geq 2$.

Lemma 3.1. *There is a polynomial-time algorithm that, on input $q \geq 2$, a lattice basis $\boldsymbol{B}$, $d < \lambda(\mathscr{L}(\boldsymbol{B}))$ and $\boldsymbol{x}$ with $||\boldsymbol{x}|| \leq d$ outputs the coefficient vector $\boldsymbol{a} \in \mathbb{Z}^n$ of the closest lattice vector $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{a}$ to $\boldsymbol{x}$ given access to an oracle that outputs these coefficient vectors modulo q.*

Proof. Define sequences of vectors $\mathbf{x}_1, ..., \mathbf{x}_{n+1}$ and $\mathbf{a}_1, ..., \mathbf{a}_{n+1}$ by $\mathbf{x}_1 = \mathbf{x}$, $\mathbf{a}_i = \mathbf{B}^{-1}\kappa_{\mathscr{L}(\mathbf{B})}(\mathbf{x}_i)$ the coefficients of the closest lattice vector to $\mathbf{x}_i$ and $\mathbf{x}_{i+1} = (\mathbf{x}_i - \mathbf{B}(\mathbf{a}_i \mod q))/q$. Note that we can efficiently compute the $\mathbf{x}_{i+1}$ using the oracle on input $\mathbf{x}_i$. We then apply an algorithm for approximately solving the closest vector problem on $(\mathbf{B}, \mathbf{x}_{n+1})$, see [8], Section 2.3 for example. The distance of $\mathbf{x}_{n+1}$ from the lattice is at most d/q^n. Applying the algorithm will give us a lattice point within distance

$$\frac{2^n \cdot d}{q^n} \leq d < \frac{\lambda(\mathscr{L}(\mathbf{B}))}{2}$$

of $\mathbf{x}_{n+1}$. But there can be only one point within this distance, so the closest lattice vector $\mathbf{x}'$ to $\mathbf{x}_{n+1}$ will be computed exactly. This gives us $\mathbf{a}_{n+1} = \mathbf{B}^{-1}\mathbf{x}'$. Using the fact that $\mathbf{a}_{i+1} = \mathbf{B}\left(\mathbf{a}_i + (\mathbf{a}_i \mod q)\right)/q$, we can work our way backwards to compute $\mathbf{a}_1$. □

The second one shows that it is possible to distinguish samples of $A_{\mathbf{s},q,\Psi_\beta}$ for unknown β from uniform given access to an oracle that solves $\mathrm{LWE}_{q,\Psi_\alpha}$ for some $\alpha \geq \beta$. For this, we need an algorithm to verify whether a vector $\mathbf{s}'$ is really the solution to a round of $\mathrm{LWE}_{q,\Psi_\beta}$ without access to an oracle. This is the subject of the next lemma.

Lemma 3.2. *There exists a polynomial-time algorithm that, on input $\boldsymbol{s}'$ and n samples of $A_{\boldsymbol{s},q,\Psi_\beta}$ for some unknown $\beta < 1$ decides whether $\boldsymbol{s} = \boldsymbol{s}'$ with overwhelming probability.*

Proof. Let $(\mathbf{a}_1, x_1), ..., (\mathbf{a}_n, x_n)$ be the samples of $A_{\mathbf{s},\Psi_\beta}$ and let $y_i = x_i - \langle \mathbf{a}_i, \mathbf{s}' \rangle \mod \mathbb{T}$ for $i = 1, ..., n$. Compute

$$\mathbf{z} = \frac{1}{n} \sum_{i=1}^{n} \cos(2\pi y_i)$$

and accept if $z > 0.02$; otherwise reject.

Let ξ be the distribution given by $e + \langle \mathbf{a}, \mathbf{s} - \mathbf{s}' \rangle \mod \mathbb{T}$ for e sampled from Ψ_β and $\mathbf{a}$ sampled uniformly from $\mathbb{Z}_q^n$. The y_i are sampled from this distribution. If $\mathbf{s} = \mathbf{s}'$, $\xi = \Psi_\beta$. If that is the case,

$$E_{y \sim \xi}(\cos(2\pi y)) = \int_0^1 \cos(2\pi y)\xi(y)\mathrm{d}y = \mathrm{Re}\left[\int_0^1 \exp(2\pi i y)\xi(y)\mathrm{d}y\right]$$

from whic it follows that $E(\cos(2\pi y)) = \exp(-\pi\beta^2) \geq 0.04$. By the Chernoff bound, $|z - E(\cos(2\pi y))| \leq 0.01$ with probability exponentially close to 1. So in this case, the algorithm will decide correctly with overwhelming probability.

Now suppose that $\mathbf{s} \neq \mathbf{s}'$. Then, ξ has period $1/k$ for some $k \geq 2$. Let j be an index for which $\mathbf{s}_j \neq \mathbf{s}'_j$. The distribution $\mathbf{a}_j \cdot (\mathbf{s}_j - \mathbf{s}'_j) \mod \mathbb{T}$ for $\mathbf{a}_j$ selected uniformly from $\mathbf{Z}_q$. We have $\gcd(q, \mathbf{s}_j - \mathbf{s}'_j) < q$, so $\mathbf{a}_j \cdot (\mathbf{s}_j - \mathbf{s}'_j)/q \mod \mathbb{T}$ has period $1/k$ for $k \geq 2$. Since we can write $\xi = \mathbf{a}_j \cdot (\mathbf{s}_j - \mathbf{s}'_j)/q \mod \mathbb{T} + \xi'$ for some distribution ξ' sampled independently, it follows that ξ has period $1/k$. From

$$\int_0^1 \exp(2\pi i y)\xi(y)\mathrm{d}y = \int_0^1 \exp\left(2\pi i y + \frac{1}{k}\right)\xi(y)\mathrm{d}y = \exp\left(\frac{2\pi i}{k}\right) \cdot \int_0^1 \exp(2\pi i y)\xi(y)\mathrm{d}y$$

it follows that $E(cos(2\pi y)) = 0$ and by the Chernoff-bound, the algorithm rejects with probability exponentially close to 1. □

Lemma 3.3. *There is an efficient algorithm that on input α and given polynomially many samples from $A_{\boldsymbol{s},q,\Psi_\beta}$ for some unknown $\beta \leq \alpha \leq 1$, given access to an oracle which solves LWE_{q,Ψ_α} with overwhelming probability using polynomially many samples from $A_{\boldsymbol{s},q,\Psi_\alpha}$, outputs $\boldsymbol{s}$ with overwhelming probability.*

Proof. The basic idea is that if we add additional error terms to the samples from $A_{\mathbf{s},q,\Psi_\beta}$, we can transform them so that they are exponentially close to samples of Ψ_α. If we increased them by the right amount, the oracle will output

$\mathbf{s}$. If not, there is no guarantee that its output will be correct, but using the algorithm of lemma 3.2, we can find out. Let c be a constant such that n^c samples suffice for the oracle. Let $Z = \left\{ k \cdot n^{-2c}\alpha^2 \,\middle|\, k \in \mathbb{Z}, 0 \leq k \leq n^{-2c} \right\}$. For every element $\gamma \in Z$, we do the following n times: We obtain n^c samples of $A_{\mathbf{s},\Psi_{\sqrt{\beta^2+\gamma}}}$ by adding an error term sampled from $\Psi_{\sqrt{\gamma}}$. Then we run the oracle on these samples and receive a candidate $\mathbf{s}'$ for the solution. If the algorithm of the previous lemma accepts, output $\mathbf{s}'$ and halt. Otherwise continue.

The algorithm from lemma 3.2 prevents us from outputting a wrong answer, except with neglgible probability. It remains to show that we will obtain the correct answer with overwhelming probability in one of the iterations. Let γ be the smallest element of Z such that $\gamma \geq \alpha^2 - \beta^2$. Then $\gamma < \alpha^2 - \beta^2 + n^{-2c}\alpha^2$. Let $\alpha' = \sqrt{\beta^2 + \gamma}$. Then

$$\alpha \leq \alpha' \leq \sqrt{\alpha^2 + n^{-2c}\alpha^2} \leq \left(1 + n^{-2c}\right)\alpha$$

By [13], Claim 2.2, the distributions Ψ_α and Ψ_β for $0 < \alpha \leq \beta \leq 2\alpha$ have statistical distance at most $9(\beta/\alpha - 1)$. It follows that Ψ_α and $\Psi_{\alpha'}$ have statistical distance at most $9n^{-2c}$. Therefore, $\Psi_\alpha^{n^c}$ has at most statistical distance $9n^{-c}$ from $\Psi_{\alpha'}^{n^c}$. Let O_α and $O_{\alpha'}$ denote the output of the oracle when given n^c samples of Ψ_α and $\Psi_{\alpha'}$ respectively. Since the application of functions does not increase statistical distance, it follows that

$$\begin{aligned} &\left|\Pr\left[O_\alpha \neq \mathbf{s}\right] - \Pr\left[O_{\alpha'} \neq \mathbf{s}\right]\right| \leq 9n^{-c} \\ \Rightarrow\ &\Pr\left[O_{\alpha'} \neq \mathbf{s}\right] \leq 9n^{-c} + \Pr\left[O_\alpha \neq \mathbf{s}\right] \leq 9n^{-c} + 2^{-\Omega(n)} \end{aligned}$$

We can conclude that for n^c samples of $A_{\mathbf{s},\Psi_{\alpha'}}$, the oracle will output $\mathbf{s}$ with probability at least $1/2$ (for n large enough), so with overwhelming probability, in one of the n iterations for γ, the correct result $\mathbf{s}$ will be found. □

The algorithm in the following lemma allows to compute R together with the algorithms of the previous lemmas.

Lemma 3.4. *Let ϵ be a negligible function and $\alpha : \mathbb{N} \to (0,1)$. There is an efficient algorithm that, given access to an oracle solving LWE_{q,Ψ_β} for all $\beta \leq \alpha$ and given input $(\boldsymbol{B}, r, \boldsymbol{x})$ with $\boldsymbol{x}$ having distance $r > \sqrt{2}q\eta_\epsilon(\mathscr{L}(\boldsymbol{B}))$ from $\mathscr{L}(\boldsymbol{B})$ and polynomially many samples of $D_{\mathscr{L}(\boldsymbol{B}),r}$ returns the coefficients (modulo p) of the vector in $\mathscr{L}(\boldsymbol{B})^*$ that is closest to $\boldsymbol{x}$.*

Proof Sketch: To solve this problem, we generate polynomially many samples from a distribution close to $A_{\mathbf{s},\Psi_\beta}$ for $\beta \leq \alpha$ and $\mathbf{s} = \mathbf{B}^\top \kappa_{\mathscr{L}(\mathbf{B})^*}(\mathbf{x}) \mod q$ and then use our oracle to find $\mathbf{s}$.

To generate the samples, we do the following: Let $\mathbf{v}$ be a vector sampled form $D_{\mathscr{L}(\mathbf{B}),r}$ and let $\mathbf{a} = \mathbf{B}^{-1}\mathbf{v} \mod p$. Output $(\mathbf{a}, \langle \mathbf{x}, \mathbf{v} \rangle / q + e \mod \mathbb{T})$ for e sampled from $\Psi_{\alpha q/\sqrt{2}r}$. By claims 3.8 to 3.10 in [13], it follows that this has negligible statistical distance to our target distribution. In more detail, $\mathbf{a}$ is close to uniform on $\mathbb{Z}_q$. Now fix $\mathbf{a}$ and let $\mathbf{x}' = \mathbf{x} - \kappa_{\mathscr{L}(\mathbf{B})*}(\mathbf{x})$. Then we have $\langle \mathbf{x}, \mathbf{v} \rangle / q + e = \langle \mathbf{x}'/q, \mathbf{v} \rangle + e + \langle \kappa_{\mathscr{L}(\mathbf{B})*}(\mathbf{x}), \mathbf{v} \rangle$.

It holds that

$$\langle \kappa_{\mathscr{L}(\mathbf{B})}(\mathbf{x}), \mathbf{v} \rangle = \left\langle \mathbf{B}^\top \kappa_{\mathscr{L}(\mathbf{B})*}(\mathbf{x}), \mathbf{B}^{-1}\mathbf{v} \right\rangle = \langle \mathbf{s}, \mathbf{a} \rangle \mod q$$

and it follows from Corollary 3.10 in [13] that $\langle \mathbf{x}'/q, \mathbf{v} \rangle + e$ is within negligible statistical distance of Ψ_β for $\beta = \sqrt{(r||\mathbf{x}'||^2/q) + \alpha^2/2} \leq \alpha$. □

Together with the previous section, this implies the following theorem:

Theorem 3.5 (Reduction to LWE)**.** *Let $\alpha(n) \in (0,1)$, $q(n) \geq (\zeta/\sqrt{n})\omega(\sqrt{\log n})$ and $\gamma \geq n/(\alpha \cdot \sqrt{\log n})$. There exists a polynomial-time algorithm that solves with overwhelming probability $GapSVP_{\gamma,\zeta}$ given access to an oracle that solves LWE_{q,Ψ_α} using polynomially many samples of $A_{\boldsymbol{s},\Psi_\alpha}$.*

The hardness of Brakerski's scheme can be based on this reduction.

Corollary 3.6 (Security of Brakerski's encryption scheme)**.** *Suppose that there is a polynomial-time adversary that breaks with non-negligible probability the cpa-security of Brakerski's scheme with parameters $\chi = \bar{\Psi}_\alpha$ for $\alpha \leq 1/\log n$ and $q = q_1^{e_1}...q_m^{e_m}$ where the q_i are distinct polynomially bounded primes with $\log n/\alpha \leq q_i$ and N polynomial in n. If $q \geq 2^{n/2} \cdot \log n/\sqrt{n}$, there is a polynomial-time algorithm that solves $GapSVP_\gamma$ for $\gamma \in \tilde{O}(n) = O(n \log n)$.*

Proof. If there is such an adversary, then, by theorem 2.4, there is an efficient algorithm that solves $\mathrm{DLWE}_{N,q,\bar{\Psi}_\alpha}$ in the average case. By theorem 1.2, this implies that there exists a polynomial-time algorithm that solves $\mathrm{LWE}_{q,\Psi_\alpha}$. Since it runs in polynomial time, it requires only polynomially many samples from its oracle. Therefore, we could use it to instantiate the LWE-oracle in the algorithm of the previous theorem. For any γ with

$$\gamma \geq \frac{n}{\alpha \cdot \sqrt{\log n}} = \frac{n \log n}{\sqrt{\log n}} = n\sqrt{\log n}$$

this would give us an algorithm to solve GapSVP_γ in polynomial time with overwhelming probability. □

4 Private Information Retrieval and Information-theoretic FHE

4.1 Introduction

A Private Information Retrieval (PIR) protocol is a protocol that allows an user to access a database stored on one or more server(s) such that the information about which entry was queried by the user remains private. The trivial solution is to simply download the whole database and access it privately. In a paper from 1998, Chor, Goldreich, Kushilevitz and Sudan ([6]) proved that, in the case of a single server, no PIR protocol can achieve perfect privacy with communication complexity less than that of the trivial protocol. It follows that no

fully homomorphic encryption scheme with compactness that allows to encrypt at least n bits on security parameter n can achieve perfect secrecy: If there were such a scheme we could use it to design a PIR protocol which achieves perfect privacy with less communication.

In 2007, Chakrabarti and Shubina ([5]) described lower bounds for almost-private information retrieval where we allow a small amount of information to be leaked and some probability for error. These have implications for FHE-schemes which allow some amount of leakage and/or some probability of decryption error. The leakage and decryption error can not grow arbitrarily small when the security parameter n grows larger. Consider the case with no decryption error as an example: If we let $\delta(n)$ be the maximal statistical distance between the encryptions of two messages m_0, m_1 then $\delta(n)$ must either be ≥ 2 for all but finitely many n or it must *increase* towards 2 as the security parameter grows.

4.2 Private Information Retrieval

The database is modeled as a string $x = x_0...x_{n-1}$ of n bits. The user is interested in some bit x_i in the string. The formal definition for single-server PIR (similar to [6] and [5]) is the following:

An (ϵ, δ)-PIR protocol consists of three (not necessarily efficient) functions

- $Q(i, r)$, a query from the user to the server where i is the index the user is interested in and r is a random string of length $l(n)$ for some fixed function l.
- $A(x, Q)$, the server's answer when the database is x and the query is Q.
- $R(i, r, A)$, the function that allows the user to reconstruct x_i from the server's answer.

such that the following conditions hold:

- *ϵ-correctness*: The probability that the user ends up with a wrong result is at most ϵ, or, more formal, for any x and i,

$$\Pr_r \left[R(i, r, A(x, Q(i, r))) \neq x_i\right] \leq \epsilon.$$

- *δ-privacy*: The queries the user makes do not reveal much information about the index he is interested in. For all pairs (i, j) of indices,

$$\sum_{q \in \mathrm{Ran}(Q)} \left|\Pr_r \left[Q(i, r) = q\right] - \Pr_r \left[Q(j, r) = q\right]\right| \leq \delta.$$

4.3 Lower Bound for $(0, 0)$-PIR and an Impossibility Result for Perfectly Secret FHE

A trivial $(0, 0)$-PIR protocol is that the user makes no query at all (Q(i,r) is always the empty string) and the server sends the whole database to the user who then reads the index he is interested in. For $(0, 0)$-PIR, this is already optimal with respect to communication complexity.

Theorem 4.1. *Let Π be a $(0,0)$-PIR protocol. In the worst case, the answer of the server must have a length of at least n bits where n is the length of the database. It follows that Π has communication complexity n.*

Proof. The idea of the proof is that for every possible database the answer must contain information about all n bits of the database, because otherwise, either the user would not be able to always reconstruct the right bit or the server must have received some amount of information about the desired entry. Let us now make this proof more formal:

Let Π be a $(0,0)$-PIR protocol. Fix some $q = Q(0, r_0)$ for arbitrary randomness r_0 and define $A_q = \mathrm{Ran}(A(\cdot, q))$. Suppose that the answer of the server is always shorter than n bits. Then $|A_0| < 2^n$. Therefore, there exist two distinct databases x and y of size n such that $A(x,q) = A(y,q)$. Now fix some index k where x and y differ. By the privacy requirement for our protocol, it follows that $q = Q(k, r_1)$ for some string r_1. Now let $a_q = A(x,q) = A(y,q)$.

By 0-correctness, we have for every possible randomness r,

$$x_k = R(k, r, A(x, Q(k, r)))$$
$$y_k = R(k, r, A(y, Q(k, r)))$$

And thus, in particular

$$x_k = R(k, r_1, A(x, Q(k, r_1))) = R(k, r_1, A(x, q)) = R(k, r_1, a_q)$$
$$y_k = R(k, r_1, A(y, Q(k, r_1))) = R(k, r_1, A(y, q)) = R(k, r_1, a_q) = x_k$$

but we have chosen k such that $x_k \neq y_k$. This proves that a $(0,0)$-PIR protocol requires answers of length n in the worst case and that no such protocol with communication complexity $< n$ exists. □

This implies the following impossibility result for FHE with perfect secrecy.

Corollary 4.2. *Let $\Pi = (Gen, Enc, Dec, Eval)$ be a fully homomorphic encryption scheme with perfect secrecy. Assume further that it is compact, that is, there are polynomials p and q such that the encryption of a message m has length $p(n) \cdot |m|$ for security parameter n and the length of the output of $Eval$ for a circuit with one output wire is $q(n)$. Suppose that the scheme can encrypt messages of length n. (If it could not, then the homomorphic properties would not be very useful as the input size would be very limited.)*[2]

If there were such a scheme, there also would be a $(0,0)$-PIR protocol with less than n bits of communication.

[2] If we allow for schemes that can only encrypt $\log n$ bits, then there is a trivial FHE-scheme with perfect secrecy: As keys, we use bijective functions $k : \{0, ..., \log n - 1\} \times \{0,1\} \to \{0, ..., 2\log n - 1\}$. Key generation consists in selecting such a function uniformly at random. If b is the ith bit to be encrypted, its encryption is $k(i,b)$. An encryption c of a single bit is decrypted by looking up $k^{-1}(c)$. The function $Eval(C, c_1, ..., c_k)$ for a circuit C with one output wire outputs a list $c_{i_1}, ..., c_{i_l}$ that contains all the ciphertexts from $c_1, ..., c_k$ without repetitions and a "truth table" of C for every possible combination of values for $c_{i_1}, ..., c_{i_l}$. If the scheme is executed correctly, there are at most $\log n$ distinct c_is, so the table has size polynomial in n, satisfying compactness. Decryption of such a table is done by decrypting the listed ciphertexts and looking up the result in the table.

Proof. Consider the following protocol where the user encrypts the index of the desired entry with the FHE scheme and the server homomorphically evaluates a circuit that outputs the bit corresponding to the index:

- $Q(i,(r_1,r_2))$: Run $Gen\left(1^{\log n}\right)$ with randomness r_1 and let k be the resulting key. Output the result of $Enc_k(i)$ with randomness r_2.
- $A(x,Q)$: Fix a circuit C that outputs x_j on input j for every possible index j. Then output the result of $Eval(C,Q)$.
- $R(i,(r_1,r_2),A)$: Generate the key k from randomness r_1 and output $Dec_k(A)$.

Correctness follows if the encryption scheme is correct. Because of perfect secrecy, $Q(i,(r_1,r_2))$ and $Q(j,(r_1,r_2))$ are identically distributed. Therefore, this is a $(0,0)$-PIR protocol. The communication is

$$p(\log n)\cdot\log n + q(\log n) \leq (\log n)^{c_1} + (\log n)^{c_2}$$

for some constants c_1, c_2. This is asymptotically smaller than n. □

Therefore, the existence of such an FHE-scheme contradicts the lower bound proved before.

4.4 Lower Bound for $(0,\delta)$-PIR and FHE with Imperfect Information-theoretic Security

The following theorem was proved in [5].

Theorem 4.3. *In a $(0,\delta)$-PIR protocol, the answer of the server needs have a length of at least $(1-\delta/2)\cdot n$ bits in the worst case. Therefore, such a protocol has communication complexity at least $(1-\delta/2)\cdot n$ bits.*

Proof. Let $\Pi = (Q,A,R)$ be a $(0,\delta)$-PIR protocol. We will find $z \in \mathrm{Ran}(Q)$ such that the range of $A(\cdot,z)$ must contain at least $2^{(1-\delta/2)n}$ elements. For $q \in \mathrm{Ran}(Q)$ and indices $i \leq n$, define $p_{iq} = \Pr_r[Q(i,r)=q]$ and $J_q = \{i \in \mathbb{N} | 1 \leq i \leq n, p_{iq} > 0\}$. Then we have

$$|J_q| p_{1q} \geq \sum_{i=1}^{n} \min(p_{1q}, p_{iq}) = \sum_{i=1}^{n} \left(\frac{p_{1q}+p_{iq}}{2} - \frac{|p_{1q}-p_{iq}|}{2} \right)$$

where the inequality holds because for $i \notin J_q$, $p_{iq} = 0$. Summing over the

$q \in \mathrm{Ran}(Q)$ gives us

$$\begin{aligned}\sum_{q\in\mathrm{Ran}(Q)} |J_q| p_{1q} &\geq \sum_{q\in\mathrm{Ran}(Q)} \sum_{i=1}^{n} \left(\frac{p_{1q}+p_{iq}}{2} - \frac{|p_{1q}-p_{iq}|}{2} \right) \\ &= \sum_{i=1}^{n} \left(\sum_{q\in\mathrm{Ran}(Q)} \frac{p_{1q}+p_{iq}}{2} - \sum_{q\in\mathrm{Ran}(Q)} \frac{|p_{1q}-p_{iq}|}{2} \right) \\ &\geq \sum_{i=1}^{n} \left(1 - \frac{\delta}{2}\right)\end{aligned}$$

The last inequality holds because $\sum_q p_{kq} = 1$ for any index k, since we are summing over the range of Q and because of δ-privacy. Choose z such that $|J_z|$ is maximal. Since $\sum_q |J_q| p_{1q} \leq |J_z| \sum_q p_{1q} = |J_z|$, it follows that $|J_z| \geq (1-\delta/2)n$.

Now suppose that the number of possible outputs of $A(\cdot, z)$ is less than $|J_z|$. Let $Y = \{x \in \{0,1\}^n | x_j = 0 \text{ for } j \notin J\}$. Then $|Y| = 2^{|J_z|}$ and thus, there are distinct $x, y \in Y$ such that $A(x,z) = A(y,z)$. Let j be an index such that $x_j \neq y_j$. Then j must be in J_z, so $p_{jz} > 0$. But then it follows that for some r, $Q(j,r) = z$. Therefore, we would have $A(x, Q(j,r)) = A(y, Q(j,r))$ and a contradiction follows as in theorem 4.1. □

A protocol that almost achieves this lower bound for arbitrary δ was described in the same paper. It needs to communicate $\lceil \log n \rceil + \lceil (1-\delta/(2+\delta)) \rceil n$ bits. Let $\delta' = \delta/(2+\delta)$ and for $i \in \{0, ..., n-1\}$, let

$$\begin{aligned} S_i &= \{k \in \{0, ..., n-1\} \,|\, (k-i) \mod n \leq \lceil (1-\delta')\, n \rceil \} \\ T_i &= \{k \in \{0, ..., n-1\} \,|\, (i-k) \mod n \leq \lceil (1-\delta')\, n \rceil \} \end{aligned}$$

The protocol consists of the following functions:

- $Q(i,r)$: For r sampled uniformly at random, the output of $Q(i,r)$ is distributed uniformly on S_i.
- $A(x, Q)$: Let $i_1, ..., i_k$ be the indices in T_Q in ascending order. Output $x_{i_1}, ..., x_{i_k}$.
- $R(i, r, A)$: List the indices in $T_{Q(i,r)}$ in ascending order. As proved below, i will be in this list. Let j be the position of i. Output A_j.

Lemma 4.4 (Correctness)**.** *If $j \in S_i$, then $i \in S_j$. Therefore, the above protocol always returns the correct result.*

Proof. If $j \in S_i$, then $(j-i) \mod n \leq (1-\delta')\, n$. Since $(j-i) = -(i-j)$, we have $(j-i) \mod n = (i-j) \mod n$. □

Lemma 4.5 (Privacy)**.** *It holds that for any pair i, j of indices,*

$$\sum_{q \in Ran(Q)} |Pr_r\left[Q(i,r) = q\right] - Pr_r\left[Q(j,r) = q\right] \leq \delta$$

Therefore, the protocol satisfies δ-privacy.

Proof. For any i, $|S_i| \geq (1-\delta')\,n$. Consider two distinct indices i and j. For $q \in S_i \cap S_j$, $\Pr_r\left[Q(i,r) = q\right] = \Pr_r\left[Q(j,r) = q\right]$. The same holds for $q \notin S_i \cup S_j$. For $q \in (S_i \setminus S_j) \cup (S_j \setminus S_i)$ on the other hand, we have

$$|\Pr_r\left[Q(i,r) = q\right] - \Pr_r\left[Q(j,r) = q\right]| \leq \frac{1}{\min(|S_i|, |S_j|)} \leq \frac{1}{(1-\delta')\,n}$$

Also, $|S_i \setminus S_j| \leq n - (1-\delta')\,n = \delta' n$ and likewise, $|S_i \setminus S_j| \leq \delta' n$. Therefore, the privacy parameter of this scheme has the upper bound

$$\frac{2\delta' n}{(1-\delta')\,n} = \frac{\delta\,(1-\delta')}{1-\delta'} = \delta$$

□

Lemma 4.6 (Communication Complexity)**.** *The protocol requires communication of*

$$\lceil \log n \rceil + \left\lceil \left(1 - \frac{\delta}{2+\delta}\right) n \right\rceil = O(\log n) + \left(1 - \frac{\delta}{2} + O\left(\delta^2\right)\right) n$$

bits.

Proof. The query is an integer between 0 and $n-1$ and therefore requires $\lceil \log n \rceil$ bits. The answer is a string of length $|T_j|$ for some j, that is, it consists of $\lceil (1 - \frac{\delta}{2+\delta}) n \rceil$ bits. □

Taking all this together, we get the following theorem.

Theorem 4.7. *For every δ, there is a $(0, \delta)$-PIR protocol with communication complexity $O(\log n) + (1 - \delta/2 + O(\delta^2))n$.*

We can apply the lower bound to FHE schemes:

Corollary 4.8. *Let $\Pi = (Gen, Enc, Dec)$ be an FHE-scheme with compactness that can encrypt n bits on security parameter n. Let δ be a function such that for any two messages m_0, m_1 of length n, we have*

$$\sum_{c \in Ran(Enc)} |Pr\left[Enc_k(m_0) = c\right] - Pr\left[Enc_k(m_1) = c\right]| \leq \delta(n)$$

where the probabilities are taken over the key generation and the randomness in Enc. Then for every constant $\delta' < 2$, $\delta(n) > \delta'$ for n large enough.

Proof. Suppose there were a scheme Π with $\delta(n) \leq \delta' < 2$ for all n except for finitely many. Using the same protocol as in the $(0,0)$-case, we could get a $(0,\delta')$-PIR protocol that needs to communicate at most $O(\log n)$ bits. But the lower bound on $(0,\delta')$-PIR is $c \cdot n$ for some constant $c > 0$, so such a scheme is impossible. □

It follows that either $\delta(n) \geq 2$ for all n except finitely many or $\delta(n) \to 2$ for $n \to \infty$.

4.5 Allowing Errors

The lower bound for (ϵ,δ)-PIR is based on the following fact which is implied by [1], Theorem 1.

Fact 4.9. *Let $Index_n$ be the following problem: Alice holds a string $x_0...x_{n-1}$ of bits and Bob an index i. The goal is to output x_i. A public-coin one-way communication protocol that solves this problem with error probability at most ϵ requires communication of $(1-H(\epsilon))n$ bits. Here, $H(\epsilon)$ is the Shannon entropy, $H(\epsilon) = -\epsilon \log \epsilon - (1-\epsilon)\log(1-\epsilon)$.*

Proof. We first need to define the following characteristics of a boolean function f in two variables. The deterministic communication complexity of f, $DC(f)$, is the minimal communication complexity of a deterministic one-way communication protocol to compute $f(x,y)$. Let $CM(f)$ be the communication matrix for f and $nrow(f)$ the number of distinct rows in it. We say that Y is a control set for f if for every two x, x' such that there exists a y with $f(x,y) \neq f(x',y)$, there is $y \in Y$ such that $f(x,y) \neq f(x',y)$. Let $ts(f)$ be the cardinality of the smallest control set for f. The value $dcc(f) = ts(f)/\log(nrow(f))$ is called the deterministic communication characteristic of f. The probabilistic communication characteristic for success probability p is defined by $pcc_p(f) = dcc(f) \cdot H(p)$.

By Theorem 1 from [1], if f is a Boolean function and C a probabilistic one-way communication protocol that correctly computes $f(x,y)$ with probability $\geq p$ for $p > 1/2$, then C has communication complexity at least $DC(f)(1 - pcc_p(f)) - 1$. If f is the function for the $Index_n$ problem, that is,

$$f : \{0,1\}^n \times \{0,1\}^{\lceil \log n \rceil} \to \{0,1\}, (x,i) \mapsto x_i$$

then we get $(1 - H(p))n - 1$ as the lower bound because $DC(f) = n$, a control set for f needs to contain every possible index and for every two $x, x' \in \{0,1\}^n$, the rows in $CM(f)$ associated to x and x' differ. Thus, $ts(f) = n$ and $\log(nrow(f)) = n$. Taking $\epsilon = 1 - p$ as the maximal error probability and using the fact that $H(p) = H(1-p)$ gives us the following lower bound:

$$PC(f) \geq (1 - H(\epsilon))n$$

□

From this fact, the following theorem was derived in [5].

Theorem 4.10. *Let ϵ, δ be such that $\epsilon + \delta/2 < 1/2$. Then every single-server (ϵ, δ)-PIR protocol needs to communicate at least $(1 - H(\epsilon + \delta/2))n$ bits.*

Proof. Let Π be such a protocol and let ρ be the length of the random string r. Define $D_{i,q}$ to be the conditional distribution of the random string r given that $Q(i, r) = q$. Let the function

$$D : \{0, ..., n-1\} \times \mathrm{Ran}(Q) \times \{0,1\}^{\rho'} \to \{0,1\}^{\rho}$$

be such that for $r' \in \{0,1\}^{\rho'}$ selected uniformly at random, $D(i, q, r')$ is distributed according to $D_{i,q}$. For

$$f(i, x, q, r') = \begin{cases} 0 & \text{if } R(i, D(i, q, r'), A(x, q)) = x_i \\ 1 & \text{otherwise} \end{cases}$$

we get the expected value

$$\begin{aligned} E_{r,r'}\left[f\left(i, x, Q(i, r), r'\right)\right] &= \Pr_{r,r'}\left[R(i, D(i, Q(i, r), r'), A(x, Q(i, r))) \neq x_i\right] \\ &= \Pr_{r}\left[R(i, r, A(x, Q(i, r)) \neq x_i\right] \\ &\leq \epsilon \end{aligned}$$

because of the bound on the error. By δ-privacy, it follows that

$$E_{r,r'}\left[f\left(i, x, Q(1, r), r'\right)\right] \leq \epsilon + \delta/2$$

Therefore, we can use the following protocol to solve $Index_n$ with error probability at most $\epsilon + \delta/2$. Alice and Bob share random strings (r, r'). Alice sends $a = A(x, Q(1, r))$ to Bob who then computes $R(i, D(i, Q(1, r), r'), a)$ and outputs the result as his guess for x_i. By the fact stated before, the length of $A(x, Q(1, r))$ must be $(1 - H(\epsilon))n$ in the worst case. □

Appyling this to FHE schemes which allow for decryption errors with probability $\epsilon(n)$ and $\delta(n)$ as in the previous section, we can conclude that either $\epsilon(n) + \delta(n)/2 \geq 1/2$ or for every constant $k < 1$ and n large enough, $H(\epsilon(n) + \delta(n)/2) > k$. From this, it follows that either $\epsilon(n) + \delta(2)/2 \geq 1/2$ for all except finitely many n or $\epsilon(n) + \delta(n)/2 \to 1/2$ for $n \to \infty$.

References

[1] Farid Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theoretical Computer Science*, 157:139–159, 1996.

[2] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *CRYPTO*, 2012.

[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully Homomorphic Encryption withoug Bootstrapping. In *ITCS*, 2012.

[4] Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *CRYPTO*, pages 505–524, 2011.

[5] Amit Chakrabarti and Anna Shubina. Nearly Private Information Retrieval. In *MFCS*, pages 383–393, 2007.

[6] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. *J. ACM*, 45:965–982, 1998.

[7] Craig Gentry. *A Fully Homomorphic Encryption Scheme.* PhD thesis, Stanford University, 2009.

[8] Shafi Goldwasser and Daniele Micciancio. *Complexity of Lattice Problems. A Cryptographic Perspective.* Kluwer Academic Publishers, 2002.

[9] Russel Impagliazzo and David Zuckerman. How to Recycle Random Bits. In *FOCS*, pages 248–253, 1989.

[10] Daniele Micciancio and Oded Regev. Worst-case to Average-case Reductions on Gaussian Measures. *SIAM J. on Computing*, 37(1):267–302, 2007.

[11] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem (extended abstract). In *STOC*, pages 333–342, 2009.

[12] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In *CRYPTO*, pages 80–97, 2010.

[13] Oded Regev. On Lattices, Learning With Errors, Linear Codes and Cryptography. *J. ACM*, 56(6):Article 34, 2009.

[14] Ron Rothblum. On the Circular Security of Bit Encryption. Cryptology ePrint Archive, Report 2012/102, 2012. `http://eprint.iacr.org/2012/102`.