Efficient Fully Homomorphic Encryption from (Standard) LWE

Zvika Brakerski[*] Vinod Vaikuntanathan[†]

Abstract

We present a fully homomorphic encryption scheme that is based solely on the (standard) learning with errors (LWE) assumption. Applying known results on LWE, the security of our scheme is based on the worst-case hardness of short vector problems on arbitrary lattices. Additionally, relying on LWE makes our scheme very natural to understand (and implement).

Our construction improves on previous works in two aspects:

1. We show that "somewhat homomorphic" encryption can be based on LWE, using a new *relinearization* technique. In contrast, all previous schemes relied on complexity assumptions related to ideals in various rings.
2. We deviate from the "squashing paradigm" used in all previous works. We introduce a new *dimension reduction* technique, which shortens the ciphertexts and reduces the decryption complexity of our scheme, *without introducing additional assumptions*. In contrast, all previous works required an additional, very strong assumption (namely, the sparse subset sum assumption).

Our scheme has very short ciphertexts and we therefore use it to construct an asymptotically-efficient LWE-based single-server private information retrieval (PIR) protocol. The communication complexity of our protocol (in the public-key model) is $k \cdot \text{polylog}\, k + \log |\mathtt{DB}|$ bits per single-bit query, which is better than any known scheme (here, k is a security parameter).

[*]Weizmann Institute of Science. Email: `zvika.brakerski@weizmann.ac.il`.

[†]University of Toronto. Email: `vinodv@cs.toronto.edu`.

1 Introduction

Fully-homomorphic encryption is one of the most sought after goals – a holy grail – of modern cryptography. In a nutshell, a fully homomorphic encryption scheme is an encryption scheme that allows evaluation of arbitrarily complex programs on encrypted data. The problem was suggested by Rivest, Adleman and Dertouzos [RAD78] back in 1978, yet the first plausible candidate came thirty years later with Gentry's breakthrough work in 2009 [Gen09b, Gen10] (although, there has been partial progress in the meanwhile [GM82, Pai99, BGN05, IP07]).

Gentry's work was phenomenal in that it showed us a first glimpse of the holy grail, when many cryptographers believed it didn't exist! However, his solution involved a slew of new and relatively untested cryptographic assumptions. If these assumptions turn out to be false (god forbid), then we are back in square one, and the appearance of the holy grail would indeed have been a mirage. Our work aims to put fully homomorphic encryption on a firm theoretical footing, by basing it on standard, well-studied cryptographic assumptions.

First, the main building block in Gentry's construction (a so-called "somewhat" homomorphic encryption scheme) was based on the (worst-case, quantum) hardness of problems in *ideal lattices*. [1] Although lattices have become standard fare in cryptography and lattice problems have been relatively well-studied, ideal lattices are a special breed that we know relatively little about. Ideals are a natural mathematical object to build fully homomorphic encryption in that they natively support both addition and multiplication (whereas lattices are closed under addition only). In fact, all subsequent constructions of fully homomorphic encryption [SV10, DGHV10, BV11] relied on ideals in various rings in an explicit way. Our first contribution is the construction of a "somewhat" homomorphic encryption scheme whose security relies solely on the (worst-case, classical) hardness of standard problems in *arbitrary* (not necessarily ideal) *lattices*.

Secondly, in order to achieve full homomorphism, Gentry had to go through a so-called "squashing step" which forced him to make an additional very strong hardness assumption – namely, the hardness of the (average-case) sparse subset-sum problem. As if by a strange law of nature, all the subsequent solutions encountered the same difficulty as Gentry did in going from a "somewhat" to a fully homomorphic encryption, and they all countered this difficulty by relying on the same sparse subset-sum assumption. This additional assumption was considered to be the main caveat of Gentry's solution and removing it has been, perhaps, the main open problem in the design of fully homomorphic encryption schemes. Our second contribution is to remove the necessity of this additional assumption.

Thus, in a nutshell, we construct a fully homomorphic encryption scheme whose security is based solely on the classical hardness of solving standard lattice problems in the worst-case, thus placing the holy grail of fully homomorphic encryption firmly within reach.[2] As icing on the cake, we will see further along in the paper that our solution is quite efficient, and holds the promise of a *practical* fully homomorphic encryption scheme.

To achieve our goals, we deviate from two paradigms that ruled the design of (a handful of) candidate fully homomorphic encryption schemes [Gen09b, SV10, DGHV10, BV11]:

1. We introduce the *re-linearization* technique, and show how to use it to obtain a *somewhat* homomorphic encryption that does not require hardness assumptions on *ideals*.

[1] Roughly speaking, ideal lattices correspond to a geometric embedding of an ideal in a number field.

[2] Strictly speaking, under only the LWE assumption, our scheme can evaluate polynomial-size circuits with a-priori bounded (but arbitrary) depth. A fully homomorphic encryption scheme independent of the circuit depth can be obtained by making an additional "circular security" assumption. See Section 4.3.

2. We present a *dimension reduction* technique, that turns our somewhat homomorphic scheme into a fully homomorphic one, without the need for the artificial *squashing* step and the sparse subset-sum assumption.

Let us explain these techniques in more detail.

1.1 Re-Linearization: Somewhat Homomorphic Encryption without Ideals

The starting point of Gentry's construction is a "somewhat" homomorphic encryption scheme. For a class of circuits $\mathcal{C}$, a $\mathcal{C}$-somewhat homomorphic scheme is one that allows evaluation of any circuit in the class $\mathcal{C}$. The simple, yet striking, observation in Gentry's work is that if a (slightly augmented) decryption circuit for a $\mathcal{C}$-somewhat homomorphic scheme resides in $\mathcal{C}$, then the scheme can be converted (or "bootstrapped") into a fully homomorphic encryption scheme.

It turns out that encryption schemes that can evaluate a non-trivial number of addition and multiplication operations[3] are already quite hard to come by (even without requiring that that they are bootstrappable).[4] Gentry's solution to this was based on the algebraic notion of *ideals* in rings. In a very high level, the message is considered to be a ring element, and the ciphertext is the message masked with some "noise". The novelty of this idea is that the noise itself belonged to an ideal I. Thus, the ciphertext is of the form $m + xI$ (for some x in the ring). Observe right off the bat that the scheme is born additively homomorphic; in fact, that will be the case with all the schemes we consider in this paper. The ideal I has two main properties: first, a random element in the ideal is assumed to "mask" the message; and second, it is possible to generate a secret trapdoor that "annihilates" the ideal, i.e., implementing the transformation $m + xI \to m$. The first property guarantees security, while the second enables multiplying ciphertexts. Letting c_1 and c_2 be encryptions of m_1 and m_2 respectively,

$$c_1c_2 = (m_1 + xI)(m_2 + yI) = m_1m_2 + (m_1y + m_2x + xyI)I = m_1m_2 + zI$$

When decrypting, the ideal is annihilated and the product m_1m_2 survives. Thus, c_1c_2 is indeed an encryption of m_1m_2, as required. This nifty solution required, as per the first property, a hardness assumption on ideals in certain rings. Gentry's original work relied on hardness assumptions on *ideal lattices*, while van Dijk, Gentry, Halevi and Vaikuntanathan [DGHV10] presented a different instantiation that considered ideals over the integers.

Our somewhat homomorphic scheme is based on the hardness of the "learning with errors" (LWE) problem, first presented by Regev [Reg05]. The LWE assumption states that if $\mathbf{s} \in \mathbb{Z}_q^n$ is an n dimensional "secret" vector, any polynomial number of "noisy" random linear combinations of the coefficients of $\mathbf{s}$ are computationally indistinguishable from uniformly random elements in $\mathbb{Z}_q$. Mathematically,

$$\left\{\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i\right\}_{i=1}^{\mathrm{poly}(n)} \quad \overset{c}{\approx} \quad \left\{\mathbf{a}_i, u_i\right\}_{i=1}^{\mathrm{poly}(n)} ,$$

where $\mathbf{a}_i \in \mathbb{Z}_q^n$ and $u_i \in \mathbb{Z}_q$ are uniformly random, and the "noise" e_i is sampled from a noise distribution that outputs numbers much smaller than q (an example is a discrete Gaussian distribution over $\mathbb{Z}_q$ with small standard deviation).

[3]All known scheme, including ours, treat evaluated functions as arithmetic circuits. Hence we use the terminology of "addition and multiplication" gates. The conversion to the boolean model (AND, OR, NOT gates) is immediate.

[4]We must mention here that we are interested only in *compact* fully homomorphic encryption schemes, namely ones where the ciphertexts do not grow in size with each homomorphic operation. If we do allow such growth in size, a number of solutions are possible. See, e.g., [SYY99, GHV10a, MGH10].

The LWE assumption does not refer to ideals, and indeed, the LWE problem is at least as hard as finding short vectors in *any lattice*, as follows from the worst-case to average-case reductions of Regev [Reg05] and Peikert [Pei09]. As mentioned earlier, we have a much better understanding of the complexity of lattice problems (thanks to [LLL82, Ajt98, Mic00] and many others), compared to the corresponding problems on ideal lattices. In particular, despite considerable effort, the best known algorithms to solve the LWE problem run in time nearly exponential in the dimension n.[5] The LWE assumption also turns out to be particularly amenable to the construction of simple, efficient and highly expressive cryptographic schemes (e.g., [Reg05, GPV08, AGV09, ACPS09, CHKP10, ABB10] and many others). Our construction of a fully homomorphic encryption scheme from LWE is perhaps an even stronger testament to its power and elegance.

Constructing a (secret-key) encryption scheme whose security is based on the LWE assumption is rather straightforward. To encrypt a bit $m \in \{0,1\}$ using secret key $\mathbf{s} \in \mathbb{Z}_q^n$, we choose a random vector $\mathbf{a} \in \mathbb{Z}_q^n$ and a "noise" e and output the ciphertext

$$c \quad = \quad (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m) \qquad \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

The key observation in decryption is that the two "masks" – namely, the secret mask $\langle \mathbf{a}, \mathbf{s} \rangle$ and the "even mask" $2e$ – do not interfere with each other.[6] That is, one can decrypt this ciphertext by annihilating the two masks, one after the other: The decryption algorithm first re-computes the mask $\langle \mathbf{a}, \mathbf{s} \rangle$ and subtracts it from b, resulting in $2e + m \pmod q$. Since $e \ll q$, then $2e + m \pmod q = 2e + m$. Removing the even mask is now easy – simply compute $2e + m$ modulo 2.[7]

As we will see below, the scheme is naturally additive homomorphic, yet multiplication presents a thorny problem. In fact, a recent work of Gentry, Halevi and Vaikuntanathan [GHV10b] showed that (a slight variant of) this scheme supports *just a single* homomorphic multiplication, but at the expense of a huge blowup to the ciphertext which made additional advance impossible.

To better understand the homomorphic properties of this scheme, let us shift our focus away from the encryption algorithm, on to the decryption algorithm. Given a ciphertext $(\mathbf{a}, b)$, consider the symbolic linear equation

$$f_{\mathbf{a},b}(\mathbf{x}) = b - \langle \mathbf{a}, \mathbf{x} \rangle \pmod q = b - \sum_{i=1}^{n} \mathbf{a}[i] \cdot \mathbf{x}[i] \qquad \in \mathbb{Z}_q$$

where $\mathbf{x} = (\mathbf{x}[1], \ldots, \mathbf{x}[n])$ denotes the variables, and $(\mathbf{a}, b)$ forms the public coefficients of the linear equation. Clearly, decryption of the ciphertext $(\mathbf{a}, b)$ is nothing but evaluating this function on the secret key $\mathbf{s}$ (and then taking the result modulo 2).

Homomorphic addition and multiplication can now be described in terms of this function f. Adding two ciphertexts corresponds to the addition of two linear functions, which is again another linear function. In particular, $f_{(\mathbf{a}+\mathbf{a}',b+b')}(\mathbf{x}) = f_{\mathbf{a},b}(\mathbf{x}) + f_{(\mathbf{a}',b')}(\mathbf{x})$ is the linear function corresponding to the "homomorphically added" ciphertext $(\mathbf{a} + \mathbf{a}', b + b')$. Similarly, multiplying two such

[5]The nearly exponential time is for a large enough error (i.e., one that is a $1/\text{poly}(n)$ fraction of the modulus q). For smaller errors, as we will encounter in our scheme, there are better – but not significantly better – algorithms. In particular, if the error is a $1/2^{n^\epsilon}$ fraction of the modulus q, the best known algorithm runs in time approx. $2^{n^{1-\epsilon}}$.

[6]We remark that using $2e$ instead of e as in the original formulation of LWE is allowed so long as q is odd (since in that case 2 is a unit in $\mathbb{Z}_q$).

[7]Although the simplified presentation of Gentry's scheme above seems to deal with just one mask (the "secret mask"), in reality, the additional "even mask" existed in the schemes of [Gen09b, DGHV10] as well. Roughly speaking, they needed this to ensure semantic security, as we do.

ciphertexts corresponds to a symbolic multiplication of these linear equations

$$\begin{aligned} f_{(\mathbf{a},b)}(\mathbf{x}) \cdot f_{(\mathbf{a}',b)}(\mathbf{x}) &= (b - \sum \mathbf{a}[i]\mathbf{x}[i]) \cdot (b' - \sum \mathbf{a}'[i]\mathbf{x}[i]) \\ &= h_0 + \sum h_i \cdot \mathbf{x}[i] + \sum h_{i,j} \cdot \mathbf{x}[i]\mathbf{x}[j] \ , \end{aligned}$$

which results in a degree-2 polynomial in the variables $\mathbf{x} = (\mathbf{x}[1], \ldots, \mathbf{x}[n])$, with coefficients $h_{i,j}$ that can be computed from $(\mathbf{a}, b)$ and $(\mathbf{a}', b')$ by opening parenthesis of the expression above. Decryption, as before, involves evaluating this quadratic expression on the secret key $\mathbf{s}$ (and then reducing modulo 2). We now run into a serious problem – the decryption algorithm has to know all the coefficients of this quadratic polynomial, which means that that size of the ciphertext just went up from $n+1$ elements to (roughly) $n^2/2$.

This is where our re-linearization technique comes into play. Re-linearization is a way to reduce the size of the ciphertext back down to $n+1$. The main idea is the following: imagine that we publish the encryptions of all the linear and quadratic terms in the secret key $\mathbf{s}$, namely all the numbers $\mathbf{s}[i]$ as well as $\mathbf{s}[i]\mathbf{s}[j]$, under a new secret key $\mathbf{t}$. Thus, these ciphertexts (for the quadratic terms) look like $(\mathbf{a}_{i,j}, b_{i,j})$ where

$$b_{i,j} = \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle + 2e_{i,j} + \mathbf{s}[i] \cdot \mathbf{s}[j] \approx \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle + \mathbf{s}[i] \cdot \mathbf{s}[j] \ .$$

Now, the sum $h_0 + \sum h_i \cdot \mathbf{s}[i] + \sum h_{i,j} \cdot \mathbf{s}[i]\mathbf{s}[j]$ can be written (approximately) as

$$h_0 + \sum h_i (b_i - \langle \mathbf{a}_i, \mathbf{t} \rangle) + \sum_{i,j} h_{i,j} \cdot (b_{i,j} - \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle) \ ,$$

which, lo and behold, is a linear function in $\mathbf{t}$! The bottom-line is that multiplying the two linear functions $f_{(\mathbf{a},b)}$ and $f_{(\mathbf{a}',b')}$ and then re-linearizing the resulting expression results in a linear function (with $n+1$ coefficients), whose evaluation on the new secret key $\mathbf{t}$ results in the product of the two original messages (upon reducing modulo 2). The resulting ciphertext is simply the coefficients of this linear function, of which there are at most $n+1$. This ciphertext will decrypt to $m \cdot m'$ using the secret key $\mathbf{t}$.

In this semi-formal description, we ignored an important detail which has to do with the fact that the coefficients $h_{i,j}$ are potentially large. Thus, even though $(b_{i,j} - \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle) \approx \mathbf{s}[i]\mathbf{s}[j]$, it may be the case that $h_{i,j} \cdot (b_{i,j} - \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle) \not\approx h_{i,j} \cdot \mathbf{s}[i]\mathbf{s}[j]$. This is handled by considering the binary representation of $h_{i,j}$, namely $h_{i,j} = \sum_{\tau=0}^{\lfloor \log q \rfloor} 2^\tau \cdot h_{i,j,\tau}$. If, for each value of τ, we had a pair $(\mathbf{a}_{i,j,\tau}, b_{i,j,\tau})$ such that

$$b_{i,j,\tau} = \langle \mathbf{a}_{i,j,\tau}, \mathbf{t} \rangle + 2e_{i,j,\tau} + 2^\tau \mathbf{s}[i] \cdot \mathbf{s}[j] \approx \langle \mathbf{a}_{i,j,\tau}, \mathbf{t} \rangle + 2^\tau \mathbf{s}[i] \cdot \mathbf{s}[j] \ ,$$

then indeed

$$h_{i,j} \cdot \mathbf{s}[i]\mathbf{s}[j] = \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,j,\tau} 2^\tau \mathbf{s}[i]\mathbf{s}[j] \approx \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,j,\tau} (b_{i,j,\tau} - \langle \mathbf{a}_{i,j,\tau}, \mathbf{t} \rangle) \ ,$$

since $h_{i,j,\tau} \in \{0, 1\}$. This increases the number of pairs we need to post by a factor of $(\lfloor \log q \rfloor + 1)$, which is polynomial.

This process allows us to do one multiplication without increasing the size of the ciphertext, and obtain an encryption of the product under a new secret key. *But why stop at two keys* $\mathbf{s}$ *and* $\mathbf{t}$? Posting a "chain" of D secret keys (together with encryptions of quadratic terms of one secret key using the next secret key) allows us to perform up to D multiplications without blowing up the ciphertext size. It is possible to achieve $D = n^\epsilon$ for an arbitrary constant $\epsilon < 1$ under reasonable assumptions, but beyond that, the growth of the error in the ciphertext kicks in, and destroys the ciphertext. Handling this requires us to use the machinery of bootstrapping, which we explain in the next section.

In conclusion, the above technique allows us to remove the need for "ideal assumptions" and obtain *somewhat* homomorphic encryption from LWE. We refer the reader to Section 3 for the full presentation and formal analysis.

1.2 Dimension Reduction: Fully Homomorphic Encryption Without Squashing

As explained above, the "bootstrapping" method for achieving full homomorphism requires a $\mathcal{C}$-somewhat homomorphic scheme whose decryption circuit resides in $\mathcal{C}$. All prior somewhat homomorphic schemes fell short in this category and failed to achieve this requirement in a natural way. Thus Gentry, followed by all other previous schemes, resorted to "squashing": a method for reducing the decryption complexity at the expense of making an additional and fairly strong assumption, namely the sparse subset sum assumption. We refrain from elaborating on this prior solution as it has no direct relation to this work.

We "upgrade" our somewhat homomorphic scheme, explained in Section 1.1, into a scheme that enjoys the same amount of homomorphism but has smaller decryption circuit. All of this, without making stronger assumptions!

Our starting point is the somewhat homomorphic scheme from Section 1.1. Recall that a ciphertext in that scheme is of the form $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, and decryption is done by computing $(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q) \pmod 2$. One can verify that this computation, presented as a polynomial in the bits of $\mathbf{s}$, has degree at least $\max(n, \log q)$, which is more than the maximal degree D that our scheme can homomorphically evaluate. The bottom line is that decryption complexity is governed by $(n, \log q)$ which are too big for our homomorphism capabilities.

Our *dimension reduction* idea enbales us to take a ciphertext with parameters $(n, \log q)$ as above, and convert it into a ciphertext of the same message, but with parameters $(k, \log p)$ which are much smaller than $(n, \log q)$. To give a hint as to the magnitude of improvement, we typically set k to be of size security parameter and $p = \text{poly}(k)$. We can then set $n = k^c$ for essentially *any constant* c, and $q = 2^{n^\epsilon}$. We will thus be able to homomorphically evaluate functions of degree roughly $D = n^\epsilon = k^{c \cdot \epsilon}$ and we can choose c to be large enough so that this is sufficient to evaluate the $(k, \log p)$ decryption circuit.

To understand dimension reduction technically, we go back to re-linearization. We showed above that, posting proper public parameters, one can convert a ciphertext $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m)$, that corresponds to a secret key $\mathbf{s}$, into a ciphertext $(\mathbf{a}', b' = \langle \mathbf{a}', \mathbf{t} \rangle + 2e' + m)$ that corresponds to a secret key $\mathbf{t}$.[8] The crucial observation is that $\mathbf{s}$ and $\mathbf{t}$ need not have the same dimension n. Specifically, if we chose $\mathbf{t}$ to be of dimension k, the procedure still works. This brings us down from $(n, \log q)$ to $(k, \log q)$, which is a big step but still not sufficient.

Having the above observation in mind, we wonder if we can take $\mathbf{t}$ to have not only low dimension but also small modulus p, thus completing the transition from $(n, \log q)$ to $(k, \log p)$. This is indeed possible using some additional ideas, where the underlying intuition is that $\mathbb{Z}_p$ can "approximate"

[8]In the previous section, we applied re-linearization to a quadratic function of $\mathbf{s}$, while here we apply it to the ciphertext $(\mathbf{a}, b)$ that corresponds to a linear functions of $\mathbf{s}$. This only makes things easier.

$\mathbb{Z}_q$ by simple scaling, up to a small error.

The public parameters for the transition from $\mathbf{s}$ to $\mathbf{t}$ will be $(\mathbf{a}_{i,\tau}, b_{i,\tau}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$, where

$$b_{i,\tau} = \langle \mathbf{a}_{i,\tau}, \mathbf{t} \rangle + e + \left\lfloor \frac{p}{q} \cdot 2^\tau \cdot \mathbf{s}[i] \right\rceil .^{9}$$

Namely, we scale $2^\tau \cdot \mathbf{s}[i] \in \mathbb{Z}_q$ into an element in $\mathbb{Z}_p$ by multiplying by p/q and rounding. The rounding incurs an additional error of magnitude at most $1/2$. It follows that

$$2^\tau \cdot \mathbf{s}[i] \approx \frac{q}{p} \cdot (b_{i,\tau} - \langle \mathbf{a}_{i,\tau}, \mathbf{t} \rangle) ,$$

which enables converting a linear equation in $\mathbf{s}$ into a linear equation in $\mathbf{t}$. The result of dimension reduction, therefore, is a ciphertext $(\hat{\mathbf{a}}, \hat{b}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ such that $\hat{b} - \langle \hat{\mathbf{a}}, \mathbf{t} \rangle = m + 2\hat{e}$. For security, we need to assume the hardness of LWE with parameters k, p. We can show that in the parameter range we use, this assumption is as hard as the one used for the somewhat homomorphic scheme.[10]

In conclusion, dimension reduction allows us to achieve a bootstrappable scheme, based on the LWE assumption alone. We refer the reader to Section 4 for the full presentation and formal analysis.

As a nice byproduct of this technique, the ciphertexts of the resulting fully homomorphic scheme become very short! They now consist of $(k+1)\log p = O(k \log k)$ bits. This is a desirable property which is also helpful in achieving efficient private information retrieval protocols (see below).

1.3 Other Results and Applications

Near-Optimal Private Information Retrieval. In (single-server) private information retrieval (PIR) protocols, a very large *database* is maintained by a *sender* (the sender is also sometimes called the server, or the database). A *receiver* wishes to obtain a specific entry in the database, without revealing any information about the entry to the server. Typically, we consider databases that are exponential in the security parameter and hence we wish that the receiver's running time and communication complexity are polylogarithmic in the size of the database N (at least $\log N$ bits are required to specify an entry in the database). The first polylogarithmic candidate protocol was presented by Cachin, Micali and Stadler [CMS99] and additional polylograithmic protocols were introduced by Lipmaa [Lip05] and by Gentry and Razman [GR05]. Of which, the latter achieves the best communication complexity of $O(\log^{3-o(1)}(N))$.[11] The latter two protocols achieve constant amortized communication complexity when retrieving large consecutive blocks of data. See a survey in [OS07] for more details on these schemes.

[9] A subtle technical point refers to the use of an error term e, instead of $2e$ as we did for re-linearization. The reason is roughly that $\frac{q}{p} \cdot 2$ is non-integer. Therfore we "divide by 2" before performing the dimension-reduction and "multiply back" by 2 after.

[10] For the informed reader we mention that while k, p are smaller than n, q and therefore seem to imply lesser security, we are able to use much higher relative noise in our k, p scheme since it needs not support homomorphism. Hence the two assumptions are of roughly the same hardness.

[11] It is hard to compare the performance of different PIR protocols due to the multitude of parameters. To make things easier to grasp, we compare the protocols on equal grounds: We assume that the database size and the adversary's running time are exponential in the security parameter and assume the maximal possible hardness of the underlying assumption against known attacks. We also assume that each query retrieves a single bit. We will explicitly mention special properties of individual protocols that are not captured by this comparison.

Fully homomorphic, or even somewhat homomorphic, encryption is known to imply polylogarithmic PIR protocols. Most trivially, the receiver can encrypt the index it wants to query, and the database will use that to homomorphically evaluate the database access function, thus retrieving an encryption of the answer and sending it to the receiver. The total communication complexity of this protocol is the sum of lengths of the public key, encryption of the index and output ciphertext. However, the public key is sent only once, it is independent of the database and the query, and it can be used for many queries. Therefore it is customary to analyze such schemes in the *public key model* where sending the public key does not count towards the communication complexity. Gentry [Gen09a] proposes to use his somewhat homomorphic scheme towards this end, which requires $O(\log^3 N)$ bit communication.[12] We show how, using our somewhat homomorphic scheme, in addition to new ideas, we can bring down communication complexity to a near optimal $\log N \cdot \text{polyloglog}\, N$ (one cannot do better than $\log N$). Details follow.

A major obstacle in the naive use of somewhat homomorphic encryption for PIR is that homomorphism is obtained with respect to the boolean representation of the evaluated function. Therefore, the receiver needs to encrypt the index to the database in a bit-by-bit manner. The query is then composed of $\log N$ ciphertexts, which necessitate at least $\log^2 N$ bits of communication. As a first improvement, we notice that the index needs not be encrypted under the somewhat homomorphic scheme. Rather, we can encrypt using any *symmetric* encryption scheme. The database will receive, an encrypted symmetric key (under the homomorphic scheme), which will enable it to convert symmetric ciphertexts into homomorphic ciphertexts without additional communication. The encrypted secret key can be sent as a part of the public key as it is independent of the query. This, of course, requires that our somewhat homomorphic scheme can homomorphically evaluate the decryption circuit of the symmetric scheme. Fully homomorphic schemes will certainly be adequate for this purpose, but known somewhat homomorphic schemes are also sufficient (depending on the symmetric scheme to be used). Using the most communication efficient symmetric scheme, we bring down the query complexity to $O(\log N)$. As for the sender's response, our dimension reduction technique guarantees very short ciphertexts. This translates into $\log N \cdot \text{polyloglog}\, N$ bits per ciphertext, thus our performance. We remark that in terms of retrieving large blocks of consecutive data, one can slightly reduce the overhead to $O(\log N)$ bits of communication for every bit of retrieved data. We leave it as an open problem to bring the amortized communication down to a constant. See Section 5 for the full details.

Prior to this work, it was not at all known how to achieve even polylogarithmic PIR under the LWE assumption. We stress that even if the size of the public key does count towards the communication complexity, our protocol still has polylogarithmic communication.

(Limited) Identity Based Fully Homomorphic Encryption. Our methods can also be used to obtain a limited variant of fully homomorphic identity based encryption where encrypting messages for a user requires only his identity and (global) public parameters, but performing homomorphic operations on this ciphertext requires additional (user-specific) "homomorphism parameters". These parameters can be computed using the (individual) secret key. Such a scheme, albeit limited, may still be useful since security is guaranteed so long as the initial encryption was done with the public parameters, even if the homomorphism parameters are fake. Therefore users can encrypt using the public parameters and be assured that even if they are using improper homomorphism parameters (e.g., ones generated by an impersonator), security is still preserved.

[12]Gentry does not provide a detailed analysis of this scheme, the above is based on our analysis of its performance.

A crucial observation is that a scheme with such property can be obtained *generically* by combining any fully homomorphic encryption scheme with any identity based encryption scheme. However, we feel that our specific constructions is more natural and has added value over the generic approach as we describe next.

In this work, we apply re-linearization and dimension reduction to LWE-based encryption scheme a la Regev [Reg05]. However, the same ideas can be applied to the dual scheme due to Gentry, Peikert and Vaikuntanathan [GPV08]. This results in a slightly less efficient scheme, but one that can be made identity based in the aforementioned sense. This can be done by using one of the many known constructions of IBE from LWE [GPV08, CHKP10, ABB10]. In all of them, the ciphertext can be interpreted as a linear function whose decryption is the evaluation on the secret key. Therefore, all we need for homomorphism is to post additional re-linearization and dimension reduction parameters which can be computed from the individual secret key.

The analysis is identical to that of the original schemes and we do not repeat it in this work.

1.4 Other Related Work

Aside from Gentry's scheme (and a variant thereof by Smart and Vercauteren [SV10] and an optimization by Stehle and Steinfeld [SS10]), there are two other fully homomorphic encryption schemes [DGHV10, BV11]. The innovation in both these schemes is the construction of a new *somewhat homomorphic* encryption scheme. Both these works then invoke Gentry's *squashing* and bootstrapping transformation to convert it to a fully homomorphic scheme, and thus the security of both these schemes relies on the sparse subset-sum assumption. The first of these schemes is due to van Dijk, Gentry, Halevi and Vaikuntanathan [DGHV10]. Their scheme works over the integers and relies on a new assumption which, roughly speaking, states that finding the greatest common divisor of many "noisy" multiples of a number is computationally hard. They cannot, however, reduce their assumption to worst-case hardness. The second is a recent work of Brakerski and Vaikuntanathan [BV11], who construct a somewhat homomorphic encryption scheme based on the ring LWE problem [LPR10] whose security can be reduced to the worst-case hardness of problems on *ideal lattices*.

The efficiency of implementing Gentry's scheme also gained much attention. Smart and Vercauteren [SV10], as well as Gentry and Halevi [GH11b] conduct a study on reducing the complexity of implementing the scheme.

In a recent independent work, Gentry and Halevy [GH11a] showed how the sparse subset sum assumption can be replaced by either a (decision) Diffie-Hellman assumption or an ideal lattice assumption, by representing the decryption circuit as an arithmetic circuit with only one level of (high fan-in) multiplications.

1.5 Paper Organization

Some preliminaries and notation are described in Section 2. In Section 3, we present our new re-linearization technique and obtain an LWE-based somewhat homomorphic encryption scheme with long ciphertext and (relatively) high decryption complexity. Then, in Section 4, we present our dimension reduction technique and show how to modify the scheme to significantly reduce the ciphertext sizes and decryption complexity, which implies fully homomorphic encryption. In Section 5 we describe our private information retrieval (PIR) protocol.

2 Preliminaries

Notations. Let $\mathcal{D}$ denote a distribution over some finite set S. Then, $x \xleftarrow{\$} \mathcal{D}$ is used to denote the fact that x is chosen from the distribution $\mathcal{D}$. When we say $x \xleftarrow{\$} S$, we simply mean that x is chosen from the uniform distribution over S.

In this work, we utilize "noise" distributions over integers. The only property of these distributions we use is their magnitude. Hence, we define B-bounded distributions which are ones where the magnitude is bounded with high probability. A definition follows.

Definition 2.1 (B-bounded distributions)**.** *A distribution ensemble $\{\chi_n\}_{n\in\mathbb{N}}$, supported over the integers, is called B-bounded if*

$$\Pr_{e \xleftarrow{\$} \chi_n}[|e| > B] \le 2^{-\Omega(n)} .$$

We denote scalars in plain (e.g. x) and vectors in bold lowercase (e.g. $\mathbf{v}$), and matrices in bold uppercase (e.g. $\mathbf{A}$). The ℓ_i norm of a vector is denoted by $\|\mathbf{v}\|_i$. Inner product is denoted by $\langle \mathbf{v}, \mathbf{u} \rangle$, recall that $\langle \mathbf{v}, \mathbf{u} \rangle = \mathbf{v}^T \cdot \mathbf{u}$. Let $\mathbf{v}$ be an n dimensional vector. For all $i = 1, \ldots, n$, the i^{th} element in $\mathbf{v}$ is denoted $\mathbf{v}[i]$. We use the convention that $\mathbf{v}[0] \triangleq 1$.

We use the following variant of the leftover hash lemma [ILL89].

Lemma 2.1 (matrix-vector leftover hash lemma)**.** *Let $\kappa \in \mathbb{N}$, $n \in \mathbb{N}$, $q \in \mathbb{N}$, and $m \ge n \log q + 2\kappa$. Let $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$ be a uniformly random matrix, let $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$ and let $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^n$. Then,*

$$\Delta\big((\mathbf{A}, \mathbf{A}^T\mathbf{r}), (\mathbf{A}, \mathbf{y})\big) \le 2^{-\kappa}$$

where $\Delta(A, B)$ denotes the statistical distance between the distributions A and B.

2.1 Learning With Errors (LWE)

The LWE problem was introduced by Regev [Reg05] as a generalization of "learning parity with noise". For positive integers n and $q \ge 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on $\mathbb{Z}_q$, let $A_{\mathbf{s},\chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random and a noise term $e \leftarrow \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. A formal definition follows.

Definition 2.2 (LWE)**.** *For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ is defined as follows: Given m independent samples from $A_{\mathbf{s},\chi}$ (for some $\mathbf{s} \in \mathbb{Z}_q^n$), output $\mathbf{s}$ with noticeable probability.*

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

For cryptographic applications we are primarily interested in the average case decision problem DLWE, where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. There are known quantum [Reg05] and classical [Pei09] reductions between $\mathsf{DLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be (discretized versions of) the Gaussian distribution, which is B-bounded for an appropriate B. Since the exact distribution χ does not matter for our results, we state a corollary of the results of [Reg05, Pei09] in terms of the bound on the distribution.

Corollary 2.2 ([Reg05, Pei09]). *Let $q = q(n) \in \mathbb{N}$, and $\sigma = \sigma(n) \in \mathbb{R}^+$ be such that $\sigma > 2\sqrt{n}$. Then there exists an efficiently sampleable $(\sigma \cdot \sqrt{n})$-bounded distribution χ such that if there is an efficient algorithm that solves the (average-case) $\mathsf{DLWE}_{n,q,\chi}$ problem. Then:*

- *There is a quantum algorithm that solves $\mathsf{SIVP}_{\tilde{O}(n\sigma/q)}$ and $\mathsf{gapSVP}_{\tilde{O}(n\sigma/q)}$ on any n-dimensional lattice, and runs in time* $\mathrm{poly}(n, q)$ [Reg05].
- *There is a classical algorithm that solves the ζ-to-γ decisional shortest vector problem $\mathsf{gapSVP}_{\zeta,\gamma}$, where $\gamma = \tilde{O}(n \cdot q/\sigma)$, and $\zeta = \tilde{O}(q\sqrt{n})$, on any n-dimensional lattice, and runs in time* $\mathrm{poly}(n, q)$ [Pei09].

We note that the best known algorithms for these problems run in time nearly exponential in the dimension n [AKS01, MV10]. More generally, the best algorithms that approximate these problems to within a factor of 2^k run in time $2^{\tilde{O}(n/k)}$.

2.2 Bootstrappable Encryption Schemes, and the Bootstrapping Theorem

In this section we formally define the notion of a bootstrappable encryption scheme and present the bootstrapping theorem which implies that a bootstrappable scheme can be converted into a fully homomorphic one.

Definition 2.3 (bootstrappable encryption scheme). *Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a homomorphic encryption scheme that can evaluate a class of circuits $\mathcal{C}$. Let f_{add} and f_{mult} be the the augmented decryption functions of the scheme defined as*

$$f_{\mathsf{add}}^{c_1,c_2}(s) = \mathsf{Dec}_s(c_1) \textit{ XOR } \mathsf{Dec}_s(c_2) \qquad \textit{and} \qquad f_{\mathsf{mult}}^{c_1,c_2}(s) = \mathsf{Dec}_s(c_1) \textit{ AND } \mathsf{Dec}_s(c_2) \ .$$

Then $\mathcal{E}$ is bootstrappable if

$$\left\{f_{\mathsf{add}}^{c_1,c_2}, f_{\mathsf{mult}}^{c_1,c_2}\right\}_{c_1,c_2} \subseteq \mathcal{C} \ .$$

Namely, if the scheme can homomorphically evaluate f_{add} and f_{mult}.

We describe two variants of Gentry's bootstrapping theorem. The first will imply a scheme that is homomorphic w.r.t. depth d circuits, for all d, but requires no additional assumption; where the second makes an additional *(weak) circular security* assumption, but guarantees a scheme that does not need an upper bound on the depth of the circuits to be evaluated. The first variant follows.

Theorem 2.3 ([Gen09b, Gen09a]). *Let $\mathcal{E}$ be a bootstrappable scheme. Then, for any polynomial d (in the security parameter), there is a fully homomorphic encryption scheme that can evaluate* any Boolean function *of polynomial-size and depth at most d.*

For the second variant, we need to define circular security.

Definition 2.4 (weak circular security). *A public key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is weakly circular secure if it is semantically secure even for an adversary with auxiliary information containing encryptions of all secret key bits: $\{\mathsf{Enc}(pk, sk[i])\}_i$.*

We can now state the second theorem.

Theorem 2.4 ([Gen09b, Gen09a]). *Let $\mathcal{E}$ be a bootstrappable scheme that is also weakly circular secure. Then there is a fully homomorphic encryption scheme that can evaluate* any Boolean function *of polynomial-size.*

3 Somewhat Homomorphic Encryption via Re-Linearization

In this section, we present an LWE-based somewhat homomorphic encryption scheme using our *re-linearization* technique. We present the scheme in Section 3.1, and its analysis in Section 3.2 and Section 3.3. Finally, we instantiate the parameters, and analyze the efficiency of the scheme in Section 3.4.

3.1 The Scheme

We present a somewhat homomorphic public-key encryption scheme whose message space is GF(2).[13] Let $\kappa \in \mathbb{N}$ be the security parameter. The scheme is parameterized by a "dimension" $n \in \mathbb{N}$, a positive integer $m \in \mathbb{N}$, an odd modulus $q \in \mathbb{N}$ and a "noise" distribution χ over $\mathbb{Z}_q$, all of which are inherited from the LWE assumption we use. An additional parameter of the scheme is a number $D \in \mathbb{N}$ that represents the maximum degree of the polynomial that the scheme can homomorphically evaluate.

During the exposition of the scheme, we invite the reader to keep the following range of parameters in mind: the dimension n is polynomial in the security parameter κ, $m \geq 2n \log q$ is a polynomial in n, the modulus $q = 2^{n^{\epsilon}}$ (where $\epsilon \in (0, 1)$ is some constant) is sub-exponential in n, χ is some noise distribution that produces small samples (say, of magnitude at most n) in $\mathbb{Z}_q$, and the maximum degree $D = n^{\epsilon'}$ for some $\epsilon' < \epsilon$.

- $\mathsf{SH.Keygen}(1^{\kappa})$: For key generation, sample $D+1$ vectors $\mathbf{s}_0, \ldots, \mathbf{s}_D \xleftarrow{\$} \mathbb{Z}_q^n$, and compute, for all $d \in [D]$, $0 \leq i \leq j \leq n$, and $\tau \in \{0, \ldots, \lfloor \log q \rfloor\}$, the value

$$\psi_{d,i,j,\tau} := \Big(\mathbf{a}_{d,i,j,\tau}\ ,\ b_{d,i,j,\tau} := \langle \mathbf{a}_{d,i,j,\tau}, \mathbf{s}_d \rangle + 2 \cdot e_{d,i,j,\tau} + 2^{\tau} \cdot \mathbf{s}_{d-1}[i] \cdot \mathbf{s}_{d-1}[j]\Big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \ , \quad (1)$$

  where $\mathbf{a}_{d,i,j,\tau} \xleftarrow{\$} \mathbb{Z}_q^n$, $e_{d,i,j,\tau} \xleftarrow{\$} \chi$. We define $\Psi \triangleq \{\psi_{d,i,j,\tau}\}_{d,i,j,\tau}$ to be the set of all these values.[14]

  We would like to invite the reader's attention to an important detail in this description. Recall that for any vector $\mathbf{s} \in \mathbb{Z}_q^n$, we denote its n components by $\mathbf{s}[1], \ldots, \mathbf{s}[n]$, and additionally, we let $\mathbf{s}[0] \triangleq 1$ purely for notational convenience. The effect of this notational wizardry is that Ψ contains the "encryptions" of not just the quadratic terms $\mathbf{s}[i] \cdot \mathbf{s}[j]$, but also all the linear terms $\mathbf{s}[i]$ by themselves! We also wish to remark that for any $0 \leq i \leq j \leq n$, one can compute an "encryption" of $\alpha \cdot \mathbf{s}[i] \cdot \mathbf{s}[j]$ for any scalar $\alpha \in \mathbb{Z}_q$, simply by writing α in its binary representation, and taking the appropriate linear combination of the "encryptions" of $2^{\tau} \cdot \mathbf{s}[i] \cdot \mathbf{s}[j]$.

  The key-generation algorithm proceeds to choose a uniformly random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{e} \xleftarrow{\$} \chi^m$, and compute $\mathbf{b} := \mathbf{A}\mathbf{s}_0 + 2\mathbf{e}$.

  The secret key $sk = \mathbf{s}_D$, and the public key $pk = ((\mathbf{A}, \mathbf{b}), \Psi)$. We remark that the public key contains two parts with distinct purposes: while the pair $(\mathbf{A}, \mathbf{b})$ is used to encrypt messages, the value Ψ is used exclusively for the purpose of performing homomorphic operations.

[13] It is quite straightforward to generalize the scheme to work over a message space GF(t), where t is relatively prime to q. Since we mostly care about the binary case, we choose not to present this generalization.

[14] A knowledgeable reader may notice that the above is very similar to encryptions of $2^{\tau} \cdot \mathbf{s}_{d-1}[i] \cdot \mathbf{s}_{d-1}[j]$ via an LWE-based scheme. We note, though, that these "ciphertexts" may not correctly decrypt to $2^{\tau} \cdot \mathbf{s}_{d-1}[i] \cdot \mathbf{s}_{d-1}[j]$, due to an overlap of the messages and the noise.

- $\mathsf{SH.Enc}(pk, \mu)$: Recall that $pk = ((\mathbf{A}, \mathbf{b}), \Psi)$. To encrypt a message $\mu \in \mathrm{GF}(2)$, we use only the $(\mathbf{A}, \mathbf{b})$ component to encrypt (just like in Regev's scheme): namely, sample a vector $\mathbf{r} \stackrel{\$}{\leftarrow} \{0,1\}^m$ and set
$$\mathbf{v}{:=}\mathbf{A}^T\mathbf{r} \qquad \text{and} \qquad w{:=}\mathbf{b}^T\mathbf{r} + \mu \ .$$
We output the ciphertext $c{:=}((\mathbf{v}, w), 0)$.[15]

- $\mathsf{SH.Eval}(\star, c, c')$ where $\star \in \{+, \times\}$: We show how to homomorphically add and multiply two ciphertexts. This can then be used to evaluate arithmetic circuits over GF(2). We will later show in the analysis that the evaluated ciphertext is decryptable as long as the number of homomorphic operations performed is bounded.

  While we provide a full analysis for the correctness of our scheme in Section 3.2, we wish to provide some intuition as we present the homomorphic operations. Towards this goal, we put forward the invariant that a ciphertext $c = ((\mathbf{v}, w), d)$ that decrypts to a message μ, will always be such that $w - \langle \mathbf{v}, \mathbf{s}_d \rangle = \mu + 2 \cdot e \pmod q$, where e is "small". Needless to say, this allows decryption upon reducing further modulo 2.

  Let $c = ((\mathbf{v}, w), d)$ and $c' = ((\mathbf{v}', w'), d)$. Note that we assume that the value of d is the same for both operands, which is without loss of generality as we will see below.

  – *Addition.* To add c and c' as above, output
  $$c_{\mathsf{add}} = ((\mathbf{v}_{\mathsf{add}}, w_{\mathsf{add}}), d){:=}((\mathbf{v} + \mathbf{v}', w + w'), d) \ .$$
  Informally, one can see that
  $$w_{\mathsf{add}} - \langle \mathbf{v}_{\mathsf{add}}, \mathbf{s}_d \rangle = (w - \langle \mathbf{v}, \mathbf{s}_d \rangle) + (w' - \langle \mathbf{v}', \mathbf{s}_d \rangle) = (\mu + 2e) + (\mu' + 2e') = (\mu + \mu') + 2(e + e') \ ,$$
  and if the noise is not too big, taking the above modulo 2 will give $\mu + \mu'$ as the result (where addition is over GF(2)).

  – *Multiplication.* To multiply c, c' as above, we first consider the following n-variate *symbolic* polynomial, evaluated over a symbolic n-dimensional variable $\mathbf{x}$:
  $$p(\mathbf{x}) = p_{(w,\mathbf{v}),(w',\mathbf{v}')}(\mathbf{x}) \triangleq (w - \langle \mathbf{v}, \mathbf{x} \rangle) \cdot (w' - \langle \mathbf{v}', \mathbf{x} \rangle) \ .$$
  This is a quadratic polynomial that one can symbolically open the parenthesis and express as
  $$p(\mathbf{x}) = \sum_{0 \le i \le j \le n} h_{i,j} \cdot \mathbf{x}[i] \cdot \mathbf{x}[j] \ ,$$
  where $h_{i,j} \in \mathbb{Z}_q$.[16] Let $h_{i,j,\tau} \in \{0,1\}$ be the τ^{th} bit in the binary representation of $h_{i,j}$. Namely $h_{i,j} = \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,j,\tau} \cdot 2^\tau$. Then the above can be re-written as
  $$p(\mathbf{x}) = \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot 2^\tau \cdot \mathbf{x}[i] \cdot \mathbf{x}[j] \ .$$

[15] This ciphertext is a "label-0" ciphertext as indicated by the last element of c. The significance of the "label" will become clear when we describe the homomorphic evaluation algorithms.

[16] We once again remind the reader that because of the notational trick of setting $\mathbf{x}[0] \triangleq 1$, this expression captures the constant term in the product, as well as all the linear terms, thus homogenizing the polynomial $p(\mathbf{x})$.

The homomorphic multiplication algorithm will set

$$\mathbf{v}_{\mathsf{mult}} := \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot \mathbf{a}_{d+1,i,j,\tau} \ ,$$

and

$$w_{\mathsf{mult}} = \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot b_{d+1,i,j,\tau} \ .$$

where $\psi_{d+1,i,j,\tau} = (\mathbf{a}_{d+1,i,j,\tau}, b_{d+1,i,j,\tau})$ comes from the public parameters Ψ. The final output ciphertext will be

$$c_{\mathsf{mult}} := ((\mathbf{v}_{\mathsf{mult}}, w_{\mathsf{mult}}), d+1) \ .$$

To see the rationale behind this procedure, we note that

$$\begin{aligned}
w_{\mathsf{mult}} - \langle \mathbf{v}_{\mathsf{mult}}, \cdot \mathbf{s}_{d+1} \rangle &= \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot (b_{d+1,i,j,\tau} - \langle \mathbf{a}_{d+1,i,j,\tau}, \mathbf{s}_{d+1} \rangle) \\
&= \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot 2^{\tau} \cdot \mathbf{s}_d[i] \cdot \mathbf{s}_d[j] + 2 \cdot h_{i,j,\tau} \cdot e_{d+1,i,j,\tau} \\
&= p(\mathbf{s}_d) + \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{d+1,i,j,\tau} \\
&= (w - \langle \mathbf{v}, \cdot \mathbf{s}_d \rangle) \cdot (w' - \langle \mathbf{v}', \mathbf{s}_d \rangle) + \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{d+1,i,j,\tau} \\
&= (\mu + 2e)(\mu' + 2e') + \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{d+1,i,j,\tau} \\
&= \mu\mu' + 2 \left(\mu e' + \mu' e + 2ee' + \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot e_{d+1,i,j,\tau} \right) \ , \qquad (2)
\end{aligned}$$

which, taken modulo 2, gives $\mu\mu' \pmod 2$, if the noise term within the parenthesis is not too large. See Lemma 3.1 for a precise analysis of the noise-term.

Note that neither addition nor multiplication increases the number of elements in the ciphertext (which remains in $\mathbb{Z}_q^n \times \mathbb{Z}_q \times \{0, \ldots, D\}$ throughout). Homomorphic addition keeps the "label" of the ciphertext the same (at d) whereas homomorphic multiplication increases it by one (from d to $d+1$).

A remark on using these operations to evaluate a Boolean circuit is in order. Assume that the circuit is composed of addition (XOR) and multiplication (AND) gates. Given the encryptions of the input bits of the circuit (at level 0), we evaluate the circuit level by level. At each level ℓ, we take ciphertexts of the form $((\mathbf{v}, w), \ell - 1)$ (namely, label-$(\ell - 1)$ ciphertexts) and transform them into ciphertexts of the form $((\mathbf{v}, w), \ell)$ (namely, label-ℓ ciphertexts), using the homomorphic addition and multiplication routines above. This enables us to evaluate circuits of depth at most D. Moreover, it is easy to see that if a certain level of the circuit consists only of addition gates, then the ciphertext label stays the same. Thus, we can actually use the scheme to evaluate circuits of *multiplicative depth* at most D. Of course, this is only an *upper-bound*, and in particular, there may be circuits of multiplicative depth D that the scheme cannot evaluate, due to the rapid growth of the error. See Section 3.2 for a precise analysis.

- $\mathsf{SH.Dec}(\mathbf{s}_D, c)$: When decrypting a ciphertext c, we assume w.l.o.g that $c = ((\mathbf{v}, w), D)$ (otherwise, one can perform "blank" homomorphic operations to increase the value of d). To decrypt, compute

$$\mu \quad \triangleq \quad (w - \langle \mathbf{v}, \mathbf{s}_D \rangle \pmod q) \pmod 2 \tag{3}$$

As a preliminary explanation, we note that the ciphertext of a message $m \in GF(2)$ produced by the encryption algorithm is of the form

$$((\mathbf{v} \triangleq \mathbf{A}^T\mathbf{r}, w \triangleq \mathbf{b}^T\mathbf{r} + \mu), 0)$$

where $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$. Rewriting this, we see that

$$w = \mathbf{b}^T\mathbf{r} + \mu = (\mathbf{A}\mathbf{s}_0 + 2\mathbf{e})^T\mathbf{r} + \mu = \mathbf{s}_0^T(\mathbf{A}^T\mathbf{r}) + 2\mathbf{e}^T\mathbf{r} + \mu = \langle \mathbf{v}, \mathbf{s}_0 \rangle + 2e + \mu \tag{4}$$

where $e \triangleq \mathbf{e}^T\mathbf{r}$ has small magnitude since the entries of both the vectors $\mathbf{e}$ and $\mathbf{r}$ are small. Performing a number of homomorphic operations on this ciphertext brings it to the form $((\mathbf{v}, w), D)$, where $w = \langle \mathbf{v}, \mathbf{s}_D \rangle + 2e + \mu$, where the "error" e has small magnitude. [17]

Now, the decryption algorithm computes

$$w - \langle \mathbf{v}, \mathbf{s}_D \rangle \pmod q = 2e + \mu \pmod q = 2e + \mu \tag{5}$$

where the last equality holds if $|e| < q/4 - 1$. In other words, if e is small enough, then reducing $2e + \mu$ modulo q has no effect on it. In this case, the decryption outputs $2e + \mu \pmod 2 = \mu$, which is the correct message. See the precise analysis below.

3.2 Correctness Analysis

We first show that the scheme is correct and capable of evaluating polynomials of total degree up to D, under some conditions on the parameters (see Lemma 3.1 below). The security of the scheme follows in a straightforward way from the learning with errors (LWE) assumption. Finally, we instantiate the parameters required for both correctness and security, and put them together in Theorem 3.4.

We first define the notion of a B-bounded error distribution over the integers: a distribution χ is called B-bounded if the absolute value of a sample from the distribution is at most B with all but an exponentially small probability (in the security parameter κ).

[17]Without loss of generality, if the label in the ciphertext is less than D, then we can perform "blank" homomorphic operations to bring it to D while increasing the error by only a modest amount.

Lemma 3.1 (Correctness)**.** *Let $n = n(\kappa), q = q(\kappa)$ and $D = D(\kappa)$ be as above. Assume that the error distribution χ is B-bounded for some $B = B(\kappa)$. The scheme correctly evaluates a polynomial $g = g(x_1, \ldots, x_\ell) \in \mathbb{Z}_2[x_1, \ldots, x_\ell]$ in ℓ variables with total degree at most D if*

$$\|\mathbf{g}\|_1 \cdot D \cdot B^{4D-1} \cdot (34n^3 \log q)^{2D} < q/4 - 1 \tag{6}$$

where $\|g\|_1$ denotes the ℓ_1-norm of the coefficient vector of g.

Proof. We first examine conditions under which a ciphertext decrypts correctly. Let $c = ((\mathbf{v}, w), d)$ be an encryption of the message μ, where $w = \langle \mathbf{v}, \mathbf{s}_d \rangle + 2e + \mu \pmod q$, and let

$$\eta(c) := 2e + \mu$$

denote the "noise" in c. Applying "blank homomorphic operations" to bring the label up from d to D adds an additional noise of at most $(D-d) \cdot n \log q \cdot B \leq DBn \log q$. Now, applying the rationale following Equation 5 above, we see that the decryption succeeds if

$$\eta(c) + DBn \log q < q/4 - 1 \tag{7}$$

Homomorphic operations generally increase the noise $\eta(c)$. To determine how many homomorphic operations we can support, we first analyze how much the homomorphic addition and multiplication operations increase the error in a ciphertext. Let $c = ((\mathbf{v}, w), d)$ and $c' = ((\mathbf{v}', w'), d)$ be encryptions of messages μ and μ', and let $\eta = \eta(c)$ and $\eta' = \eta(c')$ denote the errors in c and c', respectively. Then:

- *Addition.* Assuming that $\eta, \eta' \leq E$, we show that $\eta_{\mathsf{add}} := \eta(c_{\mathsf{add}}) \leq 2E$.

  The noise in $c_{\mathsf{add}} := ((\mathbf{v}_{\mathsf{add}}, w_{\mathsf{add}}), d)$ can be written as

  $$\eta_{\mathsf{add}} = (w_{\mathsf{add}} - \langle \mathbf{v}_{\mathsf{add}}, \mathbf{s}_d \rangle) = (w - \langle \mathbf{v}, \mathbf{s}_d \rangle) + (w' - \langle \mathbf{v}', \mathbf{s}_d \rangle) = \eta + \eta'$$

  Thus, $|\eta_{\mathsf{add}}| \leq |\eta| + |\eta'| \leq 2E$.

  This can be easily generalized to computing arbitrary linear combinations on encrypted inputs. In particular, given ℓ ciphertexts $c_1, \ldots, c_\ell$, where $\eta(c_i) \leq E$ for all i, computing homomorphically a linear combination with coefficients $\mathbf{g} = (g_1, \ldots, g_\ell)$ increases the error to at most

  $$\sum_{i=1}^{\ell} |g_i| \cdot E \triangleq \|\mathbf{g}\|_1 \cdot E \tag{8}$$

  where $\|\mathbf{g}\|_1$ denotes the L_1 norm of the vector $\mathbf{g}$.

- *Multiplication.* Assuming that $\eta, \eta' \leq E$, we show that $\eta_{\mathsf{mult}} := \eta(c_{\mathsf{mult}}) \leq 11E^2B \cdot n^2 \log q$.

  The noise in c_{mult} can be written as follows (as in Equation 2):

  $$\eta_{\mathsf{mult}} := \eta(c_{\mathsf{mult}}) = \mu\mu' + 2\left(\mu e' + \mu' e + 2ee' + \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot e_{d+1,i,j,\tau}\right)$$

where $h_{i,j,\tau} \in \{0,1\}$ and $|e_{d+1,i,j,\tau}| \leq B$. Thus,

$$\begin{aligned} |\eta_{\mathsf{mult}}| &\leq 1 + 2(|e| + |e'| + 2|ee'| + n^2 \cdot \log q \cdot |e_{d+1,i,j,\tau}|) \\ &\leq 1 + 2(2E + 2E^2 + n^2 \cdot \log q \cdot B) \\ &\leq 9E^2 + 2n^2 \log q \cdot B \leq 11E^2 B n^2 \log q \end{aligned}$$

This can be easily generalized to computing a product of at most D encrypted inputs. In particular, given D ciphertexts $c_1, \ldots, c_D$, where $\eta(c_i) \leq E$ for all i, computing the product homomorphically increases the error to at most

$$E^{2^{\lceil \log D \rceil}} \cdot (11Bn^2 \log q)^{2^{\lceil \log D \rceil} - 1} \leq E^{2D} \cdot (11Bn^2 \log q)^{2D-1} \tag{9}$$

We point out that this is a fairly loose bound, which can be improved by a more careful analysis.

We now turn to analyzing the evaluation of a polynomial $g(x_1, \ldots, x_\ell) \in \mathbb{Z}_2[x_1, \ldots, x_\ell]$ of degree at most D, given ℓ encryptions $c_i \leftarrow \mathsf{Enc}(pk, \mu_i)$. We compute $c^* \leftarrow \mathsf{Eval}(g, c_1, \ldots, c_\ell)$ as follows: first, we compute an encryption of all the monomials of g, and then we take their linear combination using the coefficients of the monomials.

Now, by Equation 4, the error in a fresh ciphertext c_i produced by the encryption algorithm is $2\mathbf{e}^T\mathbf{r} + \mu$, where $|\mathbf{e}[i]| \leq B$ and $\mathbf{r}[i] \in \{0,1\}$. Thus,

$$|\eta(c_i)| \leq 2n \cdot B + 1 \leq 3nB$$

Applying equations 8 and 9, we get that the resulting error (after homomorphically evaluating a polynomial g as above) is

$$|\eta(c^*)| \leq \|\mathbf{g}\|_1 \cdot B^{4D-1} \cdot (33n^3 \log q)^{2D}$$

As observed above, the ciphertext c^* decrypts correctly to $g(\mu_1, \ldots, \mu_\ell)$ if

$$|\eta(c^*)| + DBn \log q \leq \|\mathbf{g}\|_1 \cdot D \cdot B^{4D-1} \cdot (34n^3 \log q)^{2D} < q/4 - 1 \ . \qquad \square$$

Since $\|\mathbf{g}\|_1 \leq \binom{\ell}{D} \leq \ell^D$ for any degree-D, ℓ-variate polynomial g with 0-1 coefficients, the following corollary is immediate:

Corollary 3.2. *The conclusion of Lemma 3.1 holds if we replace Condition 6 by*

$$D \cdot \ell^D \cdot B^{4D-1} \cdot (34n^3 \log q)^{2D} < q/4 - 1 \ . \tag{10}$$

A Remark on a Generalization of the Correctness Proof. As such, the correctness proof above holds only when evaluating a multi-variate polynomial of bounded degree. There are times when we wish to evaluate general Boolean circuits (composed of, say, XOR and AND gates) whose representation as a multi-variate polynomial has exponentially many terms. Consider the function $g(x_1, \ldots, x_\ell) = (x_1 + x_2)(x_3 + x_4) \ldots (x_{\ell-1} + x_\ell)$ which, written down as a sum of monomials has $2^{\ell/2}$ terms, but is efficiently representable in the form of a Boolean circuit. Thus, the homomorphic evaluation algorithm has no choice but to evaluate the Boolean circuit directly. The correctness proof can be extended to this case as well, using the notion of *the norm of a circuit*, as defined by Barak [Bar11]. We omit this generalization from the paper.

3.3 Security Analysis

We now turn to showing the security of the scheme under the (decisional) LWE assumption.

Lemma 3.3 (Security). *Let $n = n(\kappa), q = q(\kappa)$, $D = D(\kappa)$, and $\chi = \chi(\kappa)$ be as above. The scheme is semantically secure under the $\mathsf{DLWE}_{n,q,\chi}$ assumption. In particular, if the $\mathsf{DLWE}_{n,q,\chi}$ problem is (t, ϵ)-hard, then the scheme is $(t - \mathrm{poly}(\kappa), \epsilon \cdot \mathrm{poly}(\kappa))$-semantically secure.*

Proof. The somewhat homomorphic scheme is exactly Regev's encryption scheme (except for a slight difference in the way the public key is computed) with the main difference being the inclusion of the public parameters Ψ. Our proof first shows that these public parameters can be replaced indistinguishably by uniformly random elements, and then invokes an argument similar to the security proof of Regev's encryption scheme.

More precisely, we show IND-CPA security by a series of experiments (or hybrids). Let $\mathcal{A}$ be an adversary that runs in time t and has an advantage of ϵ in the CPA-security game. We consider the following series of hybrids.

- **Hybrid H_{D+1}:** This is the identical to the IND-CPA game, where the adversary gets a public key $pk \triangleq ((\mathbf{A}, \mathbf{b}), \Psi)$ computed using the $\mathsf{SH.Keygen}$ algorithm, and the encryption of a message μ_0 or μ_1 computed by the $\mathsf{SH.Enc}$ algorithm. By definition,

$$\mathrm{Adv}_{H_{D+1}}[\mathcal{A}] \triangleq \big|\Pr[\mathcal{A}(pk, \mathsf{SH.Enc}(pk, m_0) = 1] - \Pr[\mathcal{A}(pk, \mathsf{SH.Enc}(pk, m_1) = 1]\big| = \epsilon \ .$$

- **Hybrid H_ℓ, for $\ell \in [1 \ldots D]$:** Hybrid H_ℓ is identical to $H_{\ell+1}$ except that we change each of the components $\psi_{\ell,i,j,\tau}$ for all $i, j \in [0 \ldots n]$ and $\tau \in [0, \ldots, \lfloor \log q \rfloor]$. Instead of computing $\psi_{\ell,i,j,\tau}$ correctly as $(\mathbf{a}_{\ell,i,j,\tau}, \langle \mathbf{a}_{\ell,i,j,\tau}, \mathbf{s}_\ell \rangle + 2e_{\ell,i,j,\tau} + 2^\tau \cdot \mathbf{s}_{\ell-1}[i] \cdot \mathbf{s}_{\ell-1}[j]) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, we sample $\psi_{\ell,i,j,\tau}$ uniformly at random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

  We claim that there exists an adversary $\mathcal{B}$ that solves the DLWE problem that runs in time $t + \mathrm{poly}(\kappa)$ and whose advantage is

$$\mathsf{DLWE}\mathrm{Adv}[\mathcal{B}] = \big|\mathrm{Adv}_{H_\ell}[\mathcal{A}] - \mathrm{Adv}_{H_{\ell+1}}[\mathcal{A}]\big| \ .$$

  Note that in the hybrid H_1, the component Ψ in the public key consists of uniformly random elements in the appropriate domains. Thus, the view of the adversary in H_1 looks precisely like its view in the Regev encryption scheme.

- **Hybrid H_0:** Hybrid H_0 is identical to H_1 except that the vector $\mathbf{b}$ in the public key is chosen uniformly at random from $\mathbb{Z}_q^m$, rather than being computed as $\mathbf{A} \cdot \mathbf{s}_0 + 2\mathbf{e}$. By the decisional LWE assumption, hybrids H_0 and H_1 are indistinguishable. Namely, there exists an adversary $\mathcal{B}$ that solves the DLWE problem that runs in time $t + \mathrm{poly}(\kappa)$ and whose advantage is

$$\mathsf{DLWE}\mathrm{Adv}[\mathcal{B}] = |\mathrm{Adv}_{H_1}[\mathcal{A}] - \mathrm{Adv}_{H_0}[\mathcal{A}]| \ .$$

  The adversary $\mathcal{B}$ gets as input a pair $(\mathbf{A}, \mathbf{y})$ where $\mathbf{y}$ is either an LWE instance of the form $\mathbf{y} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, or it is uniformly random. The adversary $\mathcal{B}$ construct the public key as $(\mathbf{A}, \mathbf{b}{:=}2 \cdot \mathbf{y} \pmod q)$ and runs the experiment with this public key. On the one hand, $\mathbf{y}$ is in fact an LWE instance, then $\mathbf{b} = 2 \cdot \mathbf{y} = \mathbf{A} \cdot (2\mathbf{s}) + 2\mathbf{e} = \mathbf{A}\mathbf{s}' + 2\mathbf{e} \pmod q$, where $\mathbf{s}'$ is a uniformly distributed element of $\mathbb{Z}_q^n$. In this case, the experiment is identical to H_1. On the other hand, if $\mathbf{y}$ is uniformly random, then so is $\mathbf{b} = 2\mathbf{y}$, and the experiment is identical to H_0. Thus, the advantage of $\mathcal{B}$ in the LWE distinguishing game is the same as the advantage of $\mathcal{A}$ in distinguishing between the hybrids H_0 and H_1.

- **Hybrid H_{rand}:** Hybrid H_{rand} is identical to H_0 except that the ciphertext is chosen uniformly at random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$, rather than being computed as $(\mathbf{A}^T \cdot \mathbf{r}, \mathbf{b}^T \cdot \mathbf{r} + \mu_b)$.

  We now claim that
  $$|\mathrm{Adv}_{H_0}[\mathcal{A}] - \mathrm{Adv}_{H_{\mathsf{rand}}}[\mathcal{A}]| \le 2^{-\kappa} \ .$$
  This is true essentially because of Leftover hash lemma (Lemma 2.1), since $m > n \log q + \kappa$.

Note that in H_{rand}, all the elements of both the public key and the ciphertext are uniformly random and independent of the message. Thus,

$$\mathrm{Adv}_{H_{\mathsf{rand}}}[\mathcal{A}] = 0$$

Putting these together, we get that the adversary $\mathcal{B}$ solves the DLWE problem with advantage at least

$$\mathsf{DLWE}\mathrm{Adv}[\mathcal{B}] \ge (\epsilon - 2^{-\kappa})/D \ .$$

which finishes the proof. □

3.4 Instantiating the Parameters

Putting together the parameters of Lemma 3.1 and 3.3, we have:

Theorem 3.4. *For every $\epsilon \in (0,1)$ and $C_1 \in \mathbb{N}$, there is a constant $C \in \mathbb{N}$ such that the following holds. Setting the parameters $n \ge \kappa$, $m = n \log q + 2\kappa$, $q = 2^{n^\epsilon}$, and χ to be any n-bounded, efficiently sampleable, distribution, the scheme is secure under the $\mathsf{DLWE}_{n,2^{n^\epsilon},\chi}$ assumption, and is capable of evaluating multi-variate polynomials on n^{C_1} variables of degree*

$$D \le C \cdot n^\epsilon / \log n$$

Proof. Setting $C = 2/(C_1 + 10 + 2\log 34 + 2\epsilon)$, we claim that the scheme can evaluate polynomials of degree $D = C \cdot n^\epsilon / \log n$. Substituting this value of D into Equation 10 in Corollary 3.2, and setting $B = n$, we get

$$\begin{aligned}
D \cdot \ell^D \cdot B^{4D-1} \cdot (34 n^3 \log q)^{2D} &= D \cdot n^{C_1 D} \cdot n^{4D-1} \cdot (34 n^{3+\epsilon})^{2D} \\
&= n^{(C_1+10+2\epsilon)D} \cdot \frac{34^{2D} \cdot D}{n} \\
&\le n^{(C_1+10+2\epsilon)D} \cdot 34^{2D} \qquad \text{(since } D \le n) \\
&\le n^{(C_1+10+2\log 34+2\epsilon)D} \qquad \text{(since } n \ge 2) \\
&\le n^{\frac{n^\epsilon}{\log n} \cdot C \cdot (C_1+10+2\log 34+2\epsilon)} \qquad \text{(substituting for } D) \\
&\le 2^{2n^\epsilon} \qquad \text{(substituting for } C) \\
&< q/4 - 1
\end{aligned}$$

Since this choice of D satisfies Condition 10 in Corollary 3.2, our scheme can evaluate n^{C_1}-variate polynomials of degree at most $C \cdot n^\epsilon / \log n$. □

Basing the Security on Worst-case Hardness. Using known connections between the (decisional) LWE assumption and the worst-case hardness of standard problems on lattices (see Corollary 2.2), we can base the security of our scheme on the worst-case hardness of lattice problems. In particular, invoking Regev's worst-case to average-case reduction [Reg05], the scheme is secure for the parameters in Theorem 3.4, assuming that the shortest independent vector problem (SIVP) or the gap shortest vector problem, gapSVP is hard to approximate to within a factor of $\tilde{O}(n \cdot q/\sigma) = \tilde{O}(\sqrt{n} \cdot 2^{n^\epsilon})$ by quantum algorithms running in time $\mathrm{poly}(n, q)$. Thus, the larger the degree of the polynomials that our scheme supports, the stronger the complexity assumption we need to make.

We can also invoke the classical worst-case to average-case reduction of Peikert [Pei09], basing the security of our scheme on the classical worst-case hardness of the ζ-to-γ decisional shortest vector problem $\mathsf{gapSVP}_{\zeta,\gamma}$ (defined in [Pei09]), where $\gamma = \tilde{O}(n \cdot q/\sigma) = \tilde{O}(\sqrt{n} \cdot 2^{n^\epsilon})$, and $\zeta = \tilde{O}(q\sqrt{n}) = \tilde{O}(\sqrt{n} \cdot 2^{n^\epsilon})$. Peikert also has a classical reduction from more standard lattice problems such as gapSVP, but his reduction from gapSVP requires the modulus q to be exponential in n. Unfortunately, such a large choice of q seems incompatible with the multiplicative homomorphic properties of our scheme.

Efficiency of the Scheme. The keys and the ciphertext in the scheme are the same as in Regev's public-key encryption scheme [Reg05], except for three differences that impact its efficiency: first, the modulus q is sub-exponential in n, whereas in Regev's scheme, q is a small polynomial in n. Secondly, the scheme relies on the LWE assumption with a sub-exponentially small ratio between the error and the modulus. This in turn necessitates choosing a large security parameter κ and dimension n.[18] Finally, the public key contains the "homomorphism parameters" that contribute considerably to its size.

In short, the public key has $O(mn \log q) + O(n^2 \log q \cdot D \cdot n \log q) = O(n^2 \log^2 q) + O(n^{3+\epsilon} \log^2 q)$ bits. The secret key has $O(n \log q)$ bits, and so does the ciphertext that encrypts a bit. Even though the ciphertext seems long, the dimension reduction technique in the next section enables compressing it into a rather compact $O(\kappa \log \kappa)$ bits (where κ is the security parameter). The encryption and decryption operations involve computing simple linear combinations over $\mathbb{Z}_q$ and are quite efficient as well.

4 Fully Homomorphic Encryption via Dimension Reduction

In this section we "upgrade" the scheme of Section 3 to have very short ciphertexts, and low decryption complexity. We do this by reducing the dimension and the modulus of the LWE instance underlying the ciphertext. We present our bootstrappable scheme in Section 4.1, and analyze it in Section 4.2.

4.1 A Bootstrappable Scheme

Our bootstrappable scheme builds upon the somewhat homomorphic scheme of Section 3. Our scheme allows to apply *one* function only (of bounded complexity) on a given ciphertext, resulting in a ciphertext that is decryptable but not "homomorphable" — we cannot apply additional

[18] All known somewhat homomorphic encryption schemes suffer from these limitations as well.

homomorphic operations to it. Using the terminology of [GHV10a], our scheme is "1-hop somewhat homomorphic". It is a straightforward observation that 1-hop somewhat homomorphism is sufficient for bootstraping into a fully homomorphic scheme, so long as the scheme can evaluate (a little more than) its own decryption circuit.

Our bootstrappable scheme is parameterized by (n, m, q, χ, D), which are parameters for the somewhat homomorphic scheme, and in addition by parameters $(k, p, \hat{\chi})$ which are the "reduced" parameters. $n, q \in \mathbb{N}$ are referred to as the "long" dimension and modulus respectively, while k, p are the "short" dimension and modulus. $\chi, \hat{\chi}$ are the long and short noise distributions, over $\mathbb{Z}_q, \mathbb{Z}_p$, respectively. The parameter $m \in \mathbb{N}$ is used towards public key generation. The maximal degree for an evaluated function is $D \in \mathbb{N}$.

While we discuss parameter values below, we encourage the reader to consider the following settings as a running example: $k = \kappa$, $n = k^3$, $q \approx 2^{\sqrt{n}}$, $D = n^{1/2-\epsilon} > k^{1.4}$, $p = O(Dn^2 \log q) = \mathrm{poly}(k)$, $m = O(n \log q)$.

- $\mathsf{BTS.Keygen}(1^\kappa)$: The key generation process starts by running $\mathsf{SH.Keygen}(1^\kappa)$ to obtain the secret key $\mathbf{s}_D$ and public key $((\mathbf{A}, \mathbf{b}), \Psi)$ of the somewhat homomorphic scheme. Recall that $\mathbf{s}_D \in \mathbb{Z}_q^n$, $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, and $\Psi \in (\mathbb{Z}_q^n \times \mathbb{Z}_q)^{(n+1)^2 \cdot \lceil \log q \rceil \cdot D}$.

  We proceed by sampling the "short" secret key $\hat{\mathbf{s}} \xleftarrow{\$} \mathbb{Z}_p^k$ and computing additional homomorphism parameters: For all $i \in [n]$, $\tau \in \{0, \ldots, \lfloor \log q \rfloor\}$ we sample $\hat{\mathbf{a}}_{i,\tau} \xleftarrow{\$} \mathbb{Z}_p^k$, $\hat{e}_{i,\tau} \xleftarrow{\$} \hat{\chi}$, and compute

  $$\hat{b}_{i,\tau} := \langle \hat{\mathbf{a}}_{i,\tau}, \hat{\mathbf{s}} \rangle + \hat{e}_{i,\tau} + \left\lfloor \frac{p}{q} \cdot \left(2^\tau \cdot \mathbf{s}_D[i]\right) \right\rceil \pmod{p} .$$

  We then set $\hat{\psi}_{i,\tau} := \left(\hat{\mathbf{a}}_{i,\tau}, \hat{b}_{i,\tau}\right) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$, and

  $$\hat{\Psi} := \{\hat{\psi}_{i,\tau}\}_{i \in [n], \tau \in \{0, \ldots, \lfloor \log q \rfloor\}} .$$

  Finally, we output the secret key $sk = \hat{\mathbf{s}}$ and public key $pk = ((\mathbf{A}, \mathbf{b}), \Psi, \hat{\Psi})$. As in the somewhat homomorphic scheme, $(\mathbf{A}, \mathbf{b})$ are required to encrypt, while $\Psi, \hat{\Psi}$ are used for homomorphism.

- $\mathsf{BTS.Enc}(pk, \mu)$: Encryption is essentially identical to the encryption algorithm of our somewhat homomorphic scheme. Recall that $pk = ((\mathbf{A}, \mathbf{b}), \Psi, \hat{\Psi})$. To encrypt a message $\mu \in \mathrm{GF}(2)$, we use only the $(\mathbf{A}, \mathbf{b})$. We sample a vector $\mathbf{r} \xleftarrow{\$} \{0, 1\}^m$ and set

  $$\mathbf{v} := \mathbf{A}^T \cdot \mathbf{r} \ ; \qquad w := \mathbf{b}^T \cdot \mathbf{r} + \mu .$$

  We output the ciphertext $c := ((\mathbf{v}, w), 0)$.

  We remark that despite our "promise" for a fully homomorphic scheme with short ciphertexts, the encryption algorithm generates fairly long ciphertexts in $\mathbb{Z}_q^n \times \mathbb{Z}_q \times \{0, \ldots, D\}$. However, after the dimension reduction operation described next, the ciphertexts of the resulting fully homomorphic scheme will be in $\mathbb{Z}_p^k \times \mathbb{Z}_p$ and thus much shorter.

- $\mathsf{BTS.Eval}(f, c_1, \ldots, c_\ell)$, where $f : \{0,1\}^\ell \to \{0,1\}$ is a degree-D boolean circuit and $c_1, \ldots, c_\ell$ are legal ciphertexts, is performed as follows.

  We evaluate the function f gate by gate, using the procedures $\mathsf{SH.Eval}(+, \cdot, \cdot)$, $\mathsf{SH.Eval}(\times, \cdot, \cdot)$, from our somewhat homomorphic scheme. This process concludes in a ciphertext $c = ((\mathbf{v}, w), D) \in (\mathbb{Z}_q^n \times \mathbb{Z}_q) \times \{D\}$.

  The next step is reducing the dimension of c to obtain a ciphertext that is decryptable using $\hat{\mathbf{s}}$. We consider the following function from $\mathbb{Z}^n$ into the rationals modulo p

$$\phi(\mathbf{x}) \triangleq \phi_{\mathbf{v},w}(\mathbf{x}) \triangleq \frac{p}{q} \cdot \left(\frac{q+1}{2} \cdot (w - \langle \mathbf{v}, \mathbf{x} \rangle) \right) \pmod{p} .$$

  Rearranging, one can find $h_0, \ldots, h_n \in \mathbb{Z}_q$ such that

$$\phi(\mathbf{x}) = \sum_{i=0}^{n} h_i \cdot (\frac{p}{q} \cdot \mathbf{x}[i]) \pmod{p} ,$$

  Let $h_{i,\tau}$ be the τ^{th} bit of h_i, for all $\tau \in \{0, \ldots, \lfloor \log q \rfloor\}$. Then

$$\phi(\mathbf{x}) = \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot (\frac{p}{q} \cdot 2^\tau \cdot \mathbf{x}[i]) .$$

  Using the parameters in $\hat{\Psi}$, we create a new ciphertext $\hat{c} = (\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ by setting

$$\begin{aligned} \hat{\mathbf{v}} &:= \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \hat{\mathbf{a}}_{i,\tau} \pmod{p} \in \mathbb{Z}_p^k \\ \hat{w} &:= \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \hat{b}_{i,\tau} \pmod{p} \in \mathbb{Z}_p . \end{aligned}$$

  The output of $\mathsf{BTS.Eval}$ is the new ciphertext $\hat{c}$.

  To see the rationale behind this procedure, note that

$$\begin{aligned} \hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle &= \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \left(\hat{b}_{i,\tau} - \langle \hat{\mathbf{a}}_{i,\tau}, \hat{\mathbf{s}} \rangle \right) \pmod{p} \\ &= \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \left(\hat{e}_{i,\tau} + \left\lfloor \frac{p}{q} \cdot \left(2^\tau \cdot \mathbf{s}_D[i]\right) \right\rceil \right) \pmod{p} \\ &= \phi(\mathbf{s}_D) + \underbrace{\sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \left(\hat{e}_{i,\tau} + \hat{\omega}_{i,\tau} \right)}_{\triangleq \ \delta_1} \pmod{p} , \end{aligned} \quad (11)$$

  where we define

$$\hat{\omega}_{i,\tau} \triangleq \left\lfloor \frac{p}{q} \cdot \left(2^\tau \cdot \mathbf{s}_D[i]\right) \right\rceil - \frac{p}{q} \cdot \left(2^\tau \cdot \mathbf{s}_D[i]\right) ,$$

and notice that $|\hat{\omega}_{i,\tau}| \leq 1/2$. Since $h_{i,\tau} \in \{0,1\}$ and $\hat{e}_{i,\tau}$ is small, δ_1 (defined in equation 11) is "small" as well.

Now, letting $w = \langle \mathbf{v}, \mathbf{s}_D \rangle + 2e + \mu \pmod{q}$, we wish to examine $\phi(\mathbf{s}_D) \triangleq \phi_{(\mathbf{v},w)}(\mathbf{s}_D)$ more closely, as follows.

$$\begin{aligned}
\phi(\mathbf{s}_D) &\triangleq \frac{p}{q} \cdot \left(\frac{q+1}{2} \cdot (w - \langle \mathbf{v}, \mathbf{s}_D \rangle) \right) \pmod{p} \\
&= \frac{p}{q} \cdot \left(\frac{q+1}{2} \cdot (2e + \mu + Mq) \right) \pmod{p} \qquad \text{(where } M \in \mathbb{Z}\text{)} \\
&= \frac{p}{q} \cdot \left(\frac{q+1}{2} \mu + e + M'q \right) \pmod{p} \qquad \text{(where } M' \in \mathbb{Z}\text{)} \\
&= \frac{p}{q} \cdot \frac{q+1}{2} \mu + e \pmod{p} \\
&= \frac{p+1}{2} \cdot \mu + \underbrace{(\frac{p}{q} - 1) \cdot \frac{\mu}{2} + e}_{\triangleq \delta_2} \pmod{p} \\
&= \frac{p+1}{2} \cdot \mu + \delta_2 \qquad (12)
\end{aligned}$$

and notice that if $p \leq q$ (as is the case in our setting), $|\delta_2| \leq \frac{p}{q} |e| + \frac{1}{2}$.

Putting together Equations 11 and 12, we see that

$$\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle = \frac{p+1}{2} \cdot \mu + (\delta_1 + \delta_2) \approx \frac{p+1}{2} \cdot \mu \ . \qquad (13)$$

In other words, $\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle$ is either very close to 0 or very close to $(p+1)/2$, depending on whether the encrypted bit is 0 or 1, respectively.

Finally, we remark that the ciphertext $\hat{c}$ has bit-length $(k+1)\log p$.

- $\mathsf{BTS.Dec}(\hat{\mathbf{s}}, \hat{c})$: Assume w.l.o.g that $\hat{c} = (\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ (if the ciphertext looks like a "somewhat homomorphic" ciphertext, one can perform "blank" homomorphic operations to convert it to this form). To decrypt, compute

$$\mu := (2 \cdot (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) \bmod p) \pmod{2} \ .$$

Roughly speaking, the decryption algorithm is correct because, by Equation 13, $2 \cdot (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) \bmod p = 2(\delta_1 + \delta_2) + \mu \bmod p$. This gives us the message μ upon reducing modulo 2 if the error $\delta_1 + \delta_2$ is a "small" integer.

It is important to notice that, while not immediate from its definition, $\delta_1 + \delta_2$ is indeed always an integer. To see this, note that it can be represented as a difference between integers:

$$\delta_1 + \delta_2 = (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) - \frac{p+1}{2} \cdot \mu \ .$$

This implies that $2(\delta_1 + \delta_2)$ is an even integer and is eliminated by reducing modulo 2.

4.2 Analysis of the Scheme

Lemma 4.1 (Correctness). *Let $n = n(\kappa), k = k(\kappa), q = q(\kappa), p = p(\kappa)$ and $D = D(\kappa)$ be as above. Assume that the error distributions χ and $\hat{\chi}$ are both B- and $\hat{B}$-bounded respectively, for some $B = B(\kappa)$ and $\hat{B} = \hat{B}(\kappa)$. The scheme correctly evaluates a polynomial $g = g(x_1, \ldots, x_\ell) \in \mathbb{Z}_2[x_1, \ldots, x_\ell]$ with total degree at most D if*

$$\frac{p}{q} \cdot \left(\|\mathbf{g}\|_1 \cdot D \cdot B^{4D-1} \cdot (34n^3 \log q)^{2D} \right) + 8 \cdot \hat{B} \cdot n \log q < p/8 - 2 \tag{14}$$

where $\|\mathbf{g}\|_1$ denotes the ℓ_1-norm of the coefficient vector of g.

Proof. To start with, similar to Lemma 3.1 and using Equation 13, we can see that decryption succeeds if $2(\delta_1 + \delta_2) + \mu < p/4 - 1$. It then suffices to bound δ_1 and δ_2 separately.

Keeping in mind that $h_{i,\tau} \in \{0, 1\}$ and $|\hat{\omega}_{i,\tau}| \leq 1/2$, we see that

$$\begin{aligned} |\delta_1| &= \left| \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau}(\hat{e}_{i,\tau} + \hat{\omega}_{i,\tau}) \right| \\ &\leq (n+1)(\lfloor \log q \rfloor + 1) \cdot \max(|\hat{e}_{i,\tau}| + 1/2) \\ &\leq 8\hat{B}n \log q \ . \end{aligned}$$

Since $\mu \in \{0, 1\}$ and $q \geq p$,

$$|\delta_2| \triangleq \left| e \cdot \frac{p}{q} + \frac{\mu}{2} \cdot \left(\frac{p}{q} - 1 \right) \right| \leq |e| \cdot \frac{p}{q} + 1$$

Now, the error e comes from the somewhat homomorphic ciphertext. By Lemma 3.1, we know that upon evaluating a multivariate polynomial g of degree at most D, we have

$$|e| \leq \|\mathbf{g}\|_1 \cdot D \cdot B^{4D-1} \cdot (34n^3 \log q)^{2D}$$

Putting all this together gives us the statement of the lemma. □

Semantic security of the scheme follows from DLWE using an argument very similar to Lemma 3.3. In particular, the main difference between the somewhat homomorphic scheme and the bootstrappable scheme of this section is the additional public parameters $\hat{\Psi}$ which, informally speaking, constitute an encryption of $\mathbf{s}_D \in \mathbb{Z}_q^n$ using the secret key $\hat{\mathbf{s}} \in \mathbb{Z}_p^k$. By the decisional LWE assumption $\mathsf{DLWE}_{k,p,\hat{\chi}}$, these additional encryptions are indistinguishable from uniformly random elements. From then on, semantic security is guaranteed under the $\mathsf{DLWE}_{n,q,\chi}$ assumption by Lemma 3.3 (since the public key doesn't contain additional information compared to the somewhat homomorphic scheme). In particular, we have:

Lemma 4.2 (Security). *Let $n = n(\kappa), k = k(\kappa), q = q(\kappa), p = p(\kappa)$ and $D = D(\kappa)$ be as above. The scheme is semantically secure under the $\mathsf{DLWE}_{n,q,\chi}$ and the $\mathsf{DLWE}_{k,p,\hat{\chi}}$ assumptions. In particular, if both the $\mathsf{DLWE}_{n,q,\chi}$ and the $\mathsf{DLWE}_{k,p,\hat{\chi}}$ problems are (τ, ϵ)-hard, then the scheme is $(\tau - \mathrm{poly}(\kappa), \epsilon \cdot \mathrm{poly}(\kappa))$-semantically secure.*

4.3 Fully Homomorphic Encryption

In this section, we show that the scheme is bootstrappable (namely, the scheme is capable of evaluating its own decryption circuit, plus some). Invoking the bootstrapping theorem of Gentry [Gen09b, Gen09a], our bootstrappable scheme can be turned into a fully homomorphic encryption scheme.

Theorem 4.3. *For every $\epsilon \in (0,1)$, there is a constant $C \in \mathbb{N}$ such that the following holds. Setting the parameters $k \geq \kappa$, $n \geq k^C$, $m = n \log q + 2\kappa$, $q = 2^{n^\epsilon}$, $p \geq 128 \cdot kn \log q$, and χ (resp. $\hat{\chi}$) to be any n-bounded (resp. k-bounded), efficiently sampleable, distribution, the scheme is secure under the $\mathsf{DLWE}_{n,q,\chi}$ and the $\mathsf{DLWE}_{k,p,\hat{\chi}}$ assumptions, and is a bootstrappable encryption scheme.*

Thus, applying Theorems 2.3, 2.4, we get the following corollary.

Corollary 4.4. *For every polynomial function d (in the security parameter), there exists an $\mathsf{DLWE}_{n,q,\chi}, \mathsf{DLWE}_{k,p,\hat{\chi}}$ based fully homomorphic encryption scheme that can evaluate* any Boolean function *of polynomial-size and depth at most d.*

Furthermore, if the scheme from Theorem 4.3 is weakly circular secure, as per Definition 2.4, then there exists fully homomorphic encryption scheme that can evaluate any Boolean function *of polynomial-size.*

We note that while we cannot prove the circular security of our scheme, it is not known of *any* semantically secure scheme that is not weakly circular secure. Specifically, we are not aware of any improvements in the known attacks on LWE if encryptions of the bits of the secret key are given. Thus, at the state of our current knowledge, weak circular security of our scheme is plausible.

Proof of Theorem 4.3. By an analysis exactly analogous to the proof of Theorem 3.4, we can see that the scheme is correct, and supports evaluation of polynomials of degree at most $C_1 \cdot n^\epsilon / \log n$ for some constant $C_1 > 0$. Security follows directly from Lemma 4.2.

To see bootstrappability, we use Lemma 4.5 which states that the decryption circuit for the scheme can be implemented as a circuit with depth $C_2 \cdot \log k$ and degree at most $\mathrm{poly}(k) = k^{C_2}$ for some constant $C_2 > 0$. To show that the scheme is bootstrappable, we need to set the constant C in the theorem statement so that the scheme can evaluate the augmented decryption circuit $C_{c_1,c_2}(s) = \mathsf{BTS.Dec}_s(c_1) \star \mathsf{BTS.Dec}_s(c_2)$, where $\star \in \{+, \times\}$. The augmented decryption circuits have degree at most k^{2C_2}.

Now, setting $C = (2C_2+1)/\epsilon$ and $n = k^C$, the scheme is bootstrappable since

$$n^\epsilon / \log n = (k^{(2C_2+1)/\epsilon})^\epsilon / \log k = k^{2C_2+1} / \log k \geq k^{2C_2}$$

By Theorem 2.3, for every $d \in \mathbb{N}$, we can construct a fully homomorphic encryption scheme that can evaluate *any Boolean function* of polynomial-size and depth at most d. □

The following lemma about the decryption complexity of our scheme is elementary.

Lemma 4.5. *Let $k \in \mathbb{N}$, $p = \mathrm{poly}(k)$ and let $f_p : \mathbb{Z}_p \times \mathbb{Z}_p^k \times \mathbb{Z}_p^k \to \mathbb{Z}_p$ be the function that maps*

$$(\hat{w}, \hat{\mathbf{v}}, \hat{\mathbf{s}}) \mapsto (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) \pmod{p} .$$

Then, there is a constant $C > 0$ such that f_p can be implemented by a Boolean circuit *with AND and XOR gates with depth $C \cdot \log k$ and degree $\mathrm{poly}(k) = k^C$.*

Optimizing the Decryption Complexity. It is possible to optimize the constant C in the statement of Lemma 4.5, and thus Theorem 4.3, resulting in a smaller value of the "large dimension" n. We omit a detailed analysis of this optimization, and only mention that it is possible to achieve $C = 2$.

Judiciously Choosing the Parameters. The security of our scheme relies on the LWE assumption with two settings of parameters – the "long" parameter setting (n, q, χ) and the "short" parameter setting $(k, p, \hat{\chi})$. We show how to trade-off the parameters judiciously to get optimal security against the best known attacks. For the parameters k, n, p, q, χ and $\hat{\chi}$ in Theorem 4.3, the best known algorithm to solve the $\mathsf{LWE}_{k,p,\hat{\chi}}$ problem runs in time $2^{\tilde{O}(k)}$, whereas the solver for $\mathsf{LWE}_{n,q,\chi}$ problem runs in time $2^{\tilde{O}(n^{1-\epsilon})}$.[19] The best trade-off between the two assumptions is obtained when these two running times are equal which, in turn, gives us security against adversaries that run in time $2^{\tilde{O}(k)}$. This happens when $n^{1-\epsilon} \approx k$, i.e., when $\epsilon = 1 - (1/C)$ where recall that $C = \frac{2C_2+1}{\epsilon}$ is the constant from Theorem 4.3. Solving for ϵ, we get $\epsilon = 1 - \frac{1}{2C_2+2}$.

Efficiency. The public key for the scheme (including the public parameters) is roughly the same size as the somewhat homomorphic scheme, plus $O(n^2 \log^2 q)$ bits for the "dimension reduction parameters". The secret key and the ciphertext are both considerably shorter. The secret key has $k \log p \approx \kappa \log \kappa$ bits, and the ciphertext has $(k+1)\log p \approx \kappa \log \kappa$ bits as well. Since the ciphertext and the secret key are both very short, the decryption algorithm is quite fast. The encryption as we described needs to output longer "homomorphable ciphertexts" and has to run in time $\mathrm{poly}(n, \log q)$. However, it is easy to see that an encryption scheme that outputs shorter ciphertexts in dimension k modulo p suffices since such a ciphertext can be "bootstrapped up" into a homomorphable ciphertext. Thus, all ciphertexts in our scheme are short, having $(k+1)\log p = O(\kappa \log \kappa)$ bits.

5 LWE-Based Private Information Retrieval

In this section, we present a single-server private information retrieval (PIR) protocol with nearly optimal communication complexity. First, we present the definitions of PIR in Section 5.1. Then, in Section 5.2, we show a generic construction from somewhat homomorphic encryption. Finally, in Section 5.3, we instantiate the generic construction using our own scheme from Section 4 and analyze its parameters.

5.1 Definitions of Single Server PIR

We define single server private information retrieval in the public-key setting. In this setting, there is a public key associated with the receiver (who holds the respective secret key). This public key is independent of the query and of the database, and can be generated and sent (or posted) before the interaction begins, and be used many times. Thus, the size of the public key is not counted towards communication complexity of the scheme. We formalize this by an efficient setup procedure that runs before the protocol starts and generate this public key.

[19]Roughly speaking, solving LWE with an error-to-modulus ratio 2^{-k} can be done in time $2^{\tilde{O}(n/k)}$ time, and no significantly better algorithm is known.

Letting κ be the security parameter and let $N \in \mathbb{N}$ be the database size, a PIR protocol in the public-key setting is defined by a tuple of polynomial-time computable algorithms $(\mathsf{PIR.Setup}, \mathsf{PIR.Query}, \mathsf{PIR.Response}, \mathsf{PIR.Decode})$ as follows:

0. **Setup.** The protocol begins in an off-line setup phase that does not depend on the index to be queried or on the contents of the database.

   The receiver runs the setup algorithm

   $$(params, setupstate) \leftarrow \mathsf{PIR.Setup}(1^\kappa) \ .$$

   It thus obtains a public set of parameters *params* (the public key) that is sent to the sender, and a secret state *setupstate* that is kept private.

   Once the setup phase is complete, the receiver and sender can run the remainder of the protocol an unbounded number of times.

1. **Query.** When the receiver wishes to receive the i^{th} element in the database $\mathtt{DB}[i]$, it runs

   $$(query, qstate) \leftarrow \mathsf{PIR.Query}(1^\kappa, setupstate, i) \ .$$

   The query message *query* is then sent to the sender and *qstate* is a query-specific secret information that is kept private.

2. **Answer.** The sender has access to a database $\mathtt{DB} \in \{0,1\}^N$. Upon receiving the query message *query* from the receiver, it runs the "answering" algorithm

   $$resp \leftarrow \mathsf{PIR.Response}(1^\kappa, \mathtt{DB}, params, query) \ .$$

   The response *resp* is then sent back to the receiver.

3. **Decode.** Upon receiving *resp*, the receiver decodes the response by running

   $$x \leftarrow \mathsf{PIR.Decode}(1^\kappa, setupstate, qstate, resp) \ .$$

   The output $x \in \{0,1\}$ is the output of the protocol.

We note that while in general a multi-round interactive protocol is required for each database query, the protocols we present are of the simple form of a query message followed by a response message. Hence, we chose to present the simple syntax above.

The communication complexity of the protocol is defined to be $|query| + |resp|$. Namely, the number of bits being exchanged to transfer a single database element (excluding the setup phase). We sometime analyze the query length and the response length separately.

Correctness and security are defined as follows.

- **Correctness.** For all $\kappa \in \mathbb{N}$, $\mathtt{DB} \in \{0,1\}^*$ where $N \triangleq |\mathtt{DB}|$, and $i \in [N]$, it holds that

  $$\Pr[\mathsf{PIR.Decode}(1^\kappa, setupstate, qstate, resp) \neq \mathtt{DB}[i]] = \mathrm{negl}(\kappa) \ ,$$

  where $(params, setupstate) \leftarrow \mathsf{PIR.Setup}(1^\kappa)$, $(query, qstate) \leftarrow \mathsf{PIR.Query}(1^\kappa, setupstate, i)$ and $resp \leftarrow \mathsf{PIR.Response}(1^\kappa, \mathtt{DB}, params, query)$.

- (τ,ϵ)**-Privacy.** For all $\kappa \in \mathbb{N}$, $N \in \mathbb{N}$ and for any adversary $\mathcal{A}$ running in time $\tau = \tau_{\kappa,N}$ it holds that

  $$\max_{\substack{\mathbf{i}=(i_1,\ldots,i_\tau),\\ \mathbf{j}=(j_1,\ldots,j_\tau)\in[N]^\tau}} \left|\Pr[\mathcal{A}(params,\mathbf{i},query_{\mathbf{i}})=1]-\Pr[\mathcal{A}(params,\mathbf{j},query_{\mathbf{j}})=1]\right| \le \epsilon \ \big(= \epsilon_{\kappa,N} \big) \ ,$$

  where $(params, setupstate) \leftarrow \mathsf{PIR.Setup}(1^\kappa)$, $(query_{i_\ell}, qstate_{i_\ell}) \leftarrow \mathsf{PIR.Query}(1^\kappa, setupstate, i_\ell)$ and $(query_{j_\ell}, qstate_{j_\ell}) \leftarrow \mathsf{PIR.Query}(1^\kappa, setupstate, j_\ell)$, for all $\ell \in [\tau]$.

We note that the definition of privacy above differs from the one usually found in literature. The standard definition refers to vectors $\mathbf{i}, \mathbf{j}$ of dimension 1. That is, only allow the adversary to see one query to the database. A hybrid argument can show that with proper degradation in parameters, this guarantees some security also for the case of many queries. However in the public-key setting, where the same public key is used for all queries, this hybrid argument no longer works. Thus, we must require that the adversary is allowed to view many query strings.[20] In fact, one could consider even stronger attacks in the public-key setting, which is outside the scope of this work

The definition of privacy deserves some further discussion. We note that we did not define the ranges of parameters for (τ,ϵ) for which the protocol is considered "private". Indeed there are several meaningful ways to define what it means for a protocol to be private. Let us discuss two options and provide corresponding definitions.

i. The first approach is to argue that the resources of the adversary are similar to those of an honest server (we can think of an adversary as a "server gone bad"). Thus, in this approach the adversary can run in polynomial time in N, κ and must still not succeed with non-negligible probability in N, κ. We say that a scheme is (i)-private if it is $(p(\kappa, N), 1/p(\kappa, N))$-private for any polynomial $p(\cdot,\cdot)$.

ii. The second approach argues that the security parameter is the "real" measure for privacy. Thus the protocol needs to be exponentially secure in the security parameter. Thus a scheme is (ii)-private if it is $(2^{\Omega(\kappa)}, 2^{-\Omega(\kappa)})$-private.

5.2 PIR via Somewhat Homomorphic and Symmetric Encryption

In this section we describe a generic PIR protocol that uses a somewhat homomorphic encryption and symmetric encryption as building blocks. This protocol has the useful property that the somewhat homomorphic scheme is not used to encrypt the index to the database. Rather, we use the symmetric scheme to encrypt the index, and have the server homomorphically decrypt it during query evaluation.

Our PIR protocol relies on two building blocks – a semantically secure symmetric encryption scheme $\mathcal{E}_{\rm sym} = (\mathsf{SYM.Keygen}, \mathsf{SYM.Enc}, \mathsf{SYM.Dec})$, and a somewhat homomorphic encryption scheme $\mathcal{E}_{\rm sh} = (\mathsf{SH.Keygen}, \mathsf{SH.Enc}, \mathsf{SH.Dec}, \mathsf{SH.Eval})$. The level of somewhat homomorphism required for the protocol depends on the symmetric scheme being used (in particular, the decryption complexity of the symmetric scheme). We recall that our scheme from Section 4 can be instantiated

[20]We feel that our definition captures the essence of an attack on a PIR protocol more than the standard one-time definition, even in the usual setting. As we mention above, converting between the definitions incurs a linear blowup in the adversary's advantage so a (τ,ϵ)-private scheme according to the old definition is only $(\tau,\tau\epsilon)$-private according to ours.

as a somewhat homomorphic scheme with respect to functions computable by any polynomial-size circuit without relying on any circular security assumptions. However, a clever selection of the symmetric scheme to be used can make our methodology applicable also for schemes that are somewhat homomorphic up to an absolute bound (such as a single instance of the scheme from Section 4 – without bootstrapping).

We present the functions ($\mathsf{PIR.Setup}, \mathsf{PIR.Query}, \mathsf{PIR.Response}, \mathsf{PIR.Decode}$) of our protocol (as defined in Section 5.1).

- $\mathsf{PIR.Setup}(1^\kappa)$: In the setup procedure, we generate a symmetric key $symsk \leftarrow \mathsf{SYM.Keygen}(1^\kappa)$ and a key pair for the somewhat homomorphic scheme $(shsk, shpk) \leftarrow \mathsf{SH.Keygen}(1^\kappa)$.

  The symmetric key is then encrypted using the homomorphic public key to create a ciphertext
  $$c_{symsk} \leftarrow \mathsf{SH.Enc}_{shpk}(symsk) \ .$$
  The setup procedure then outputs the public parameters
  $$params := (shpk, c_{symsk}) \ ,$$
  and the secret state
  $$setupstate := (shsk, symsk) \ .$$

- $\mathsf{PIR.Query}(1^\kappa, setupstate, i)$: To generate a query string, we just encrypt i using the symmetric scheme. Recall that $setupstate = (shsk, symsk)$, then
  $$query \leftarrow \mathsf{SYM.Enc}_{symsk}(i) \ .$$
  In our scheme, no additional information needs to be saved per query: $qstate := \phi$.

- $\mathsf{PIR.Response}(1^\kappa, \mathtt{DB}, params, query)$: Upon receiving a query, a response is computed as follows. Recall that $params = (shpk, c_{symsk})$ and consider the function h defined as follows:
  $$h(x) \triangleq \mathtt{DB}[\mathsf{SYM.Dec}_x(query)] \ ,$$
  namely the function h uses its input as a symmetric key to decrypt the query, and then uses the plaintext to index the database and retrieve the appropriate value. Note that $h(symsk) = \mathtt{DB}[i]$, where i is the index embedded in $query$.

  While $\mathsf{PIR.Response}$ does not know $symsk$, it does know c_{symsk} and thus can homomorphically evaluate $h(symsk)$ and set
  $$resp \leftarrow \mathsf{SH.Eval}_{shpk}(h, c_{symsk}) \ .$$
  Note that $resp$ should correspond to a decryptable ciphertext of $\mathtt{DB}[i]$.

- $\mathsf{PIR.Decode}(1^\kappa, setupstate, qstate, resp)$: We recall that $setupstate = (shsk, symsk)$ and that $qstate$ is null. To decode the answer to the query, we decrypt the ciphertext associated with $resp$, outputting
  $$b \leftarrow \mathsf{SH.Dec}_{shsk}(resp) \ .$$

Correctness and privacy are easily reduced to those of the underlying primitives in the following lemmas.

Lemma 5.1 (correctness)**.** *If our symmetric scheme $\mathcal{E}_{sym}$, and our somewhat homomorphic scheme $\mathcal{E}_{sh}$ are correct and if the somewhat homomorphic scheme can evaluate the function h defined above, then our PIR protocol is correct.*

The proof in this case is immediate from the syntax of the protocol.

Lemma 5.2 (privacy)**.** *If our somewhat homomorphic scheme is $(\tau \cdot \mathrm{poly}(\kappa), \epsilon_1)$-CPA secure and our symmetric scheme is $(\tau + \mathrm{poly}(\kappa), \epsilon_2)$-CPA secure, then our PIR protocol is $(\tau, \epsilon_1 + \epsilon_2)$-private.*

Proof. We prove this by a series of hybrids (or experiments). Let $\mathcal{A}$ be an adversary that runs in time τ against the privacy of our protocol and has advantage ϵ. We consider the behavior of $\mathcal{A}$ in a number of hybrids H_0, H_1, H_2 as defined below. We let $\mathrm{Adv}_{H_i}[\mathcal{A}]$ denote the advantage of $\mathcal{A}$ in hybrid H_i.

- **Hybrid H_0.** This is identical to the original privacy game of the scheme. By definition

$$\mathrm{Adv}_{H_0}[\mathcal{A}] = \epsilon \ .$$

- **Hybrid H_1.** We now change the game so that instead of computing $c_{symsk} \leftarrow \mathsf{SH.Enc}_{shpk}(symsk)$ in $\mathsf{PIR.Setup}$, we will set $c_{symsk} \leftarrow \mathsf{SH.Enc}_{shpk}(0)$.

  There exists an adversary $\mathcal{B}$ for the CPA-security of the somewhat homomorphic scheme that runs in time $\tau \cdot \mathrm{poly}(\kappa)$ and whose advantage is

$$\mathrm{CPAAdv}[\mathcal{B}] = |\mathrm{Adv}_{H_0}[\mathcal{A}] - \mathrm{Adv}_{H_1}[\mathcal{A}]| \ .$$

  It follows that

$$|\mathrm{Adv}_{H_0}[\mathcal{A}] - \mathrm{Adv}_{H_1}[\mathcal{A}]| \leq \epsilon_1 \ .$$

- **Hybrid H_2.** We now change the game so that instead of setting $query \leftarrow \mathsf{SYM.Enc}_{symsk}(i)$ in $\mathsf{PIR.Query}$, we will set $query \leftarrow \mathsf{SYM.Enc}_{symsk}(0)$.

  There exists an adversary $\mathcal{C}$ for the CPA-security of the symmetric scheme that runs in time $\tau + \mathrm{poly}(\kappa)$ and whose advantage is

$$\mathrm{CPAAdv}[\mathcal{C}] = |\mathrm{Adv}_{H_1}[\mathcal{A}] - \mathrm{Adv}_{H_2}[\mathcal{A}]| \ .$$

  It follows that

$$|\mathrm{Adv}_{H_1}[\mathcal{A}] - \mathrm{Adv}_{H_2}[\mathcal{A}]| \leq \epsilon_2 \ .$$

  However, in H_2, the view of the adversary is independent of the queried indices. Therefore

$$\mathrm{Adv}_{H_2}[\mathcal{A}] = 0 \ .$$

It follows that $\epsilon \leq \epsilon_1 + \epsilon_2$ as required. □

Lastly, let us analyze the communication complexity of our protocol. It follows by definition that the query size is the length of an encryption of $\{0,1\}^{\lceil \log N \rceil}$ bits using our symmetric scheme, and the response is the encryption of a single bit using our somewhat homomorphic scheme.

5.3 Instantiating the Components: The PIR Protocol

We show how to implement the primitives required in Section 5.2 in a number of different ways.

The first idea is to use an optimized, symmetric-key LWE-based encryption as the symmetric encryption scheme in the PIR protocol. Specifically, using the same parameters k, p as in our our bootstrappable scheme, we get a scheme whose decryption is almost identical to that of our bootstrappable scheme. In particular, we apply an optimization of [PVW08, ACPS09] to get ciphertexts of length $O(\log N) + O(k \log k)$ to encrypt $\log N$ bits of the index. Roughly speaking, the optimization is based on two observations: first, rather than encrypting a single bit using an element of $\mathbb{Z}_p$, we can "pack in" $O(\log p)$ bits, if we set the error in the LWE instances to be correspondingly smaller (but still a $1/\text{poly}(k)$ fraction of p). Secondly, observe that in a symmetric ciphertext $(\mathbf{v}, w) \in \mathbb{Z}_p^n \times \mathbb{Z}_p$, most of the space is consumed by the vector $\mathbf{v}$. The observation of [PVW08, ACPS09] is that $\mathbf{v}$ can be re-used to encrypt multiple messages using different secret keys $\mathbf{s}_1, \ldots, \mathbf{s}_\ell$. Using these optimizations, the resulting PIR protocol has query length of $O(k \log k + \log N)$ bits and response length $O(k \log k)$ for $k = \text{poly}(\kappa)$.

Since the decryption of the symmetric scheme has the same degree as that of our own scheme, hence $\text{poly}(k)$, we can use our bootstrappable scheme "out of the box", with an appropriately chosen n, to evaluate the database function h.

For the best currently known attacks on LWE (see [MR09, LP11, RS10]), this protocol is $(2^{\Omega(k/\text{polylog} k)}, 2^{-\Omega(k/\text{polylog} k)})$-private. Thus, going back to our definitions in Section 5.1, and setting $k = \kappa \cdot \text{polylog}(\kappa)$, we get a (ii)-private PIR scheme with a total communication complexity of $O(\log N) + O(\kappa \cdot \text{polylog}(\kappa))$; and a (i)-private scheme with communication complexity $\log N \cdot \text{polyloglog}(N)$ by setting $\kappa = \log N \cdot \text{polyloglog}(N) = \omega(\log N)$.

A second instantiation aims to bring the (ii)-private communication complexity down to $\log N + \kappa \cdot \text{polylog}(\kappa)$. This can be done in a number of different ways, for example, by instantiating the symmetric encryption scheme above with an optimal symmetric encryption scheme with ciphertexts of length $\log N + \kappa \cdot \text{polylog}(\kappa)$. Such a scheme can be based on a pseudo-random function (PRF) constructed from LWE by applying the GGM transformation [GGM86].

In this case, we do not have a polynomial upper bound on the degree of h (since the degree of evaluating the PRF depends on the actual implementation). However, for any specific PRF, we can bootstrap our scheme as in Corollary 4.4 to support the required circuit depth.

Finally, let us note that the parameters produced in the setup phase of our protocol are of length $\text{poly}(\kappa)$. Thus our protocol can be trivially modified to work in a setting without setup, with communication complexity $\log N + \text{poly}(\kappa)$ (under the (ii)-private notion) and $\text{polylog}(N)$ (under the (i)-private notion).

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In Gilbert [Gil10], pages 553–572.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.

[Ajt98] Miklós Ajtai. The shortest vector problem in $_2$ is p-hard for randomized reductions (extended abstract). In *STOC*, pages 10–19, 1998.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.

[Bar11] Boaz Barak, 2011. Personal Communication.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography - TCC'05*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, 2011. To appear.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Gilbert [Gil10], pages 523–552.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414, 1999.

[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [Gil10], pages 24–43. Full Version in `http://eprint.iacr.org/2009/616.pdf`.

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

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Mitzenmacher [Mit09], pages 169–178.

[Gen10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In Rabin [Rab10], pages 116–137.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GH11a] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. Cryptology ePrint Archive, Report 2011/279, 2011. `http://eprint.iacr.org/`.

[GH11b] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.

[GHV10a] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. *i*-hop homomorphic encryption and rerandomizable yao circuits. In Rabin [Rab10], pages 155–172.

[GHV10b] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple bgn-type cryptosystem from lwe. In Gilbert [Gil10], pages 506–522.

[Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377. ACM, 1982.

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

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815. Springer, 2005.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *STOC*, pages 12–24. ACM, 1989.

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

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2005.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovsz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. 10.1007/BF01457454.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In Aggelos Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [Gil10], pages 1–23. Draft of full version was provided by the authors.

[MGH10] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with *d*-operand multiplications. In Rabin [Rab10], pages 138–154.

[Mic00] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035, 2000.

[Mic10] Daniele Micciancio. A first glimpse of cryptography's holy grail. *Commun. ACM*, 53:96–96, March 2010.

[Mit09] Michael Mitzenmacher, editor. *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009.* ACM, 2009.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-Quantum Cryptography.* Springer, 2009.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In Leonard J. Schulman, editor, *STOC*, pages 351–358. ACM, 2010.

[OS07] Rafail Ostrovsky and William E. Skeith III. A survey of single-database private information retrieval: Techniques and applications. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411. Springer, 2007.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Mitzenmacher [Mit09], pages 333–342.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[Rab10] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science.* Springer, 2010.

[RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.

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

[RS10] Markus Rckert and Michael Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2010/137, 2010. `http://eprint.iacr.org/`.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394. Springer, 2010.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for nc^1. In *FOCS*, pages 554–567, 1999.