Efficient Fully Homomorphic Encryption from (Standard) LWE*

Zvika Brakerski† Vinod Vaikuntanathan‡

Abstract

A fully homomorphic encryption scheme allows anyone to transform an encryption of a message, m, into an encryption of any (efficient) function of that message, $f(m)$, without knowing the secret key.

We present a leveled fully homomorphic encryption scheme that is based solely on the (standard) learning with errors (LWE) assumption. (Leveled FHE schemes are initialized with a bound on the maximal evaluation depth. However, this restriction can be removed by assuming "weak circular security".)

Applying known results on LWE, the security of our scheme is based on the worst-case hardness of "short vector problems" on arbitrary lattices.

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-modulus reduction* technique, which shortens the ciphertexts and reduces the decryption complexity of our scheme, *without introducing additional assumptions.*

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 $k \cdot \mathrm{polylog}(k) + \log |\mathtt{DB}|$ bits per single-bit query, in order to achieve security against 2^k-time adversaries (based on the best known attacks against our underlying assumptions).

*A preliminary version appeared in *Proceedings of the* 52^{nd} *IEEE Foundations of Computer Science*, 2011.

†Stanford University. Email: zvika@stanford.edu. The majority of this work was done while the author was at the Weizmann Institute of Science, Israel, and was supported by ISF grant 710267, BSF grant 710613, and NSF contracts CCF-1018064 and CCF-0729011.

‡University of Toronto. Email: vinodv@cs.toronto.edu. Part of this work was done while the author was at Microsoft Research, Redmond. This work was also partially supported by an NSERC Discovery Grant, and by DARPA under Agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

1 Introduction

Fully-homomorphic encryption is one of the holy grails 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 breakthrough work demonstrated the first construction of fully homomorphic encryption in 2009. However, his solution involved new and relatively untested cryptographic assumptions. Our work aims to put fully homomorphic encryption on firm grounds by basing it on the hardness of standard, well-studied mathematical problems.

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 on *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 use to build fully homomorphic encryption in that they natively support both addition and multiplication (whereas lattices are closed under addition only). Indeed, all subsequent constructions of fully homomorphic encryption [SV10, DGHV10, BV11a] 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 on *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, 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 perhaps been 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 on the classical hardness of solving standard lattice problems in the worst-case. Specifically, our scheme is based on the learning with errors (LWE) assumption that is known to be at least as difficult as solving hard problems in general lattices. Thus our solution does not rely on lattices directly and is fairly natural to understand and implement. Under this assumption, we achieve a *leveled* fully homomorphic encryption scheme, which 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 3.)

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, BV11a]:

1. We introduce the *re-linearization* technique, and show how to use it to obtain a *somewhat*

[1] Roughly speaking, ideal lattices correspond to a geometric embedding of an ideal in a number field. See [LPR10] for a precise definition.

homomorphic encryption that does not require hardness assumptions on *ideals*.

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

We provide a detailed overview of these new techniques in Sections 1.1 and 1.2 below.

Interestingly, the ciphertexts of the resulting fully homomorphic scheme are very short. This is a desirable property which we use, in conjunction with other techniques, to achieve very efficient private information retrieval protocols. See also Section 1.3 below.

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 $\mathcal{C}$-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 $\mathcal{C}$-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[2] are already quite hard to come by (even without requiring that they are bootstrappable).[3] 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 decrypting ciphertexts. Letting c_1 and c_2 be encryptions of m_1 and m_2 respectively,

$$c_1 c_2 = (m_1 + xI)(m_2 + yI) = m_1 m_2 + (m_1 y + m_2 x + xyI)I = m_1 m_2 + zI$$

When decrypting, the ideal is annihilated and the product $m_1 m_2$ survives. Thus, $c_1 c_2$ is indeed an encryption of $m_1 m_2$, 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

[2] 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.

[3] 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. We will also look at *mildly non-compact* homomorphic encryption schemes where the ciphertexts grow as a *sub-linear* function of the size of the evaluated circuit. If we do allow the ciphertexts to grow linearly with the size of the evaluated circuit, a number of solutions are possible. See, e.g., [SYY99, GHV10a, MGH10].

the coefficients of $\mathbf{s}$ are computationally indistinguishable from uniformly random elements in $\mathbb{Z}_q$. Mathematically,

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

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).

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.[4] 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 a very strong 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.[5] 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 $2e + m \pmod q$.[6] If we choose e to be small enough, 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 further 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 function $f_{\mathbf{a},b} : \mathbb{Z}_q^n \to \mathbb{Z}_q$ defined as:

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

[4] The nearly exponential time is for a large enough error (i.e., one that is a $1/\mathrm{poly}(n)$ 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. $2^{n^{1-\epsilon}}$.

[5] We remark that using $2e$ instead of e as in the original formulation of LWE does not adversely impact security, so long as q is odd (since in that case 2 is a unit in $\mathbb{Z}_q$, and thus the transformation $(\mathbf{a}, b) \to (2\mathbf{a}, 2b)$ is injective). This property has been previously used in [GHV10b].

[6] Throughout this paper, we use $x \pmod q$ to denote the unique residue of x in the interval $[-q/2, q/2)$.

[7] Although our 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.

where $\mathbf{x} = (\mathbf{x}[1], \ldots, \mathbf{x}[n])$ denotes the variables, and the ciphertext $(\mathbf{a}, b)$ defines 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).[8]

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 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 the 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 "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

$$b_{i,j} = \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle + 2e_{i,j} + \mathbf{s}[i] \cdot \mathbf{s}[j] \approx \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle + \mathbf{s}[i] \cdot \mathbf{s}[j] \ .[9]$$

where the expression $x \approx y$ means that the absolute difference between x and y is small.

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

[8] The observation that an LWE-based ciphertext can be interpreted as a linear equation of the secret was also used in [BV11a].

[9] Actually, calling these "encryptions" is inaccurate: $\mathbf{s}[i] \cdot \mathbf{s}[j] \in \mathbb{Z}_q$ is not a single bit and therefore the "ciphertext" cannot be decrypted. However, we feel that thinking of these as encryptions may benefit the reader's intuition.

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 $h_{i,j,\tau} \in \{0,1\}$. That is, the solution is to publish the "encryptions" of all the 2^τ-multiples of $\mathbf{s}[i]\mathbf{s}[j]$, instead of just the encryptions of $\mathbf{s}[i]\mathbf{s}[j]$. 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 L secret keys (together with encryptions of quadratic terms of one secret key using the next secret key) allows us to perform up to L levels of multiplications without blowing up the ciphertext size. It is possible to achieve multiplicative depth $L = \epsilon \log n$ (which corresponds to a degree $D = n^\epsilon$ polynomial) 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. This scheme will be a building block towards our full construction and is formally presented in Section 4.1.

1.2 Dimension-Modulus Reduction: Fully Homomorphic Encryption Without Squashing

As explained above, the "bootstrapping" method for achieving full homomorphism requires a $\mathcal{C}$-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 subsequent 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 show how to "upgrade" our somewhat homomorphic scheme (explained in Section 1.1) into a scheme that enjoys the same amount of homomorphism but has a much smaller decryption circuit. All of this, without making any additional assumption (beyond LWE)!

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-modulus 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 the 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-modulus 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 $\mathbf{t}$.[10] 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" $\mathbb{Z}_q$ 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

$$b_{i,\tau} = \langle \mathbf{a}_{i,\tau}, \mathbf{t} \rangle + e + \left\lfloor \frac{p}{q} \cdot 2^\tau \cdot \mathbf{s}[i] \right\rceil \ .^{11}$$

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-modulus 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.[12]

In conclusion, dimension-modulus reduction allows us to achieve a bootstrappable scheme, based on the LWE assumption alone. We refer the reader to Section 4 for the formal presentation and full analysis of our entire solution. Specifically, dimension-modulus reduction is used for the scheme in Section 4.2.

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 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

[10] 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 function of $\mathbf{s}$. This only makes things easier.

[11] 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.

[12] 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.

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 Ramzan [GR05]. Of which, the latter achieves the best communication complexity of $O(\log^{3-o(1)}(N))$.[13] 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.

Fully homomorphic, or even somewhat homomorphic, encryption is known to imply polylogarithmic PIR protocols.[14] 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.[15] We show how, using our somewhat homomorphic scheme, in addition to new ideas, we can bring down communication complexity to a near optimal $\log N \cdot \mathrm{polyloglog}\, N$ (one cannot do better than $\log N$). To obtain the best parameters, one needs to assume $2^{\widetilde{\Omega}(k)}$-hardness of polynomial-factor approximation for short vector problems in arbitrary dimension k lattices, which is supported by current knowledge. 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 $O(\log N)$. This technique is in fact generic and can be used to reduce the communication complexity in other applications of homomorphic encryption

[13] 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.

[14] To be precise, one needs sub-exponentially secure such schemes.

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

(see Section 5.3).

The improvement we achieve in the length of the sender's response comes from the fact that our dimension-modulus reduction technique guarantees very short ciphertexts (essentially as short as non-homomorphic LWE based schemes). This translates into $\log N \cdot \text{polyloglog}\, N$ bits per ciphertext, and the communication complexity of our protocol follows. 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.

1.4 Dual-Regev Encryption and IBE

Gentry, Peikert and Vaikuntanathan [GPV08] present a "dual" LWE-based encryption scheme, where the roles of the key generation and the encryption procedure are reversed. Interestingly, their ciphertext takes the same form as in Regev's scheme (namely, $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m)$, albeit with higher dimensional vectors and different distribution of $\mathbf{a}, e$). All of our techniques can therefore be applied just as well to the dual scheme, resulting in a scheme with slightly longer parameters, but with other interesting properties.

Using the above observation, and using any of the known LWE based identity based encryption constructions (e.g. [GPV08, CHKP10, ABB10]), we can derive a limited form of fully homomorphic identity based encryption (FH-IBE): A scheme 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", which can be computed using the (individual) secret key.

While limited FH-IBE can be achieved generically by combining any IBE scheme with any fully homomorphic scheme,[16] the above construction is more natural and can hopefully be a stepping stone towards full-fledged FH-IBE. Indeed, very recently, Gentry, Sahai and Waters [GSW13] realized the aforementioned outline using a variant of our scheme.

1.5 Other Related Work

First Generation. The first generation of fully homomorphic encryption includes Gentry's scheme (and a variant thereof by Smart and Vercauteren [SV10] and an optimization by Stehle and Steinfeld [SS10]), as well as two followup works that followed similar principles [DGHV10, BV11a]. These works followed Gentry's blueprint and presented a somewhat homomorphic encryption schemes which were not bootstrappable, and required the use of squashing. Once squashing is applied, the scheme can be bootstrapped and *leveled* fully homomorphic encryption could be obtained. For a non-leveled scheme, a weak circular security assumption needed to be made. The security of these first generation schemes relied on that of the somewhat homomorphic component, in addition to the sparse subset sum assumption (and weak circular security to remove the "leveled" constraint).

The novelty in [DGHV10, BV11a] was the construction of a new *somewhat homomorphic* components, which were different from Gentry's. The first of these schemes is due to van Dijk, Gentry,

[16] We thank Benny Applebaum for pointing this to us.

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 [BV11a], 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.

Second Generation. This work puts forth a second generation of fully homomorphic schemes that do not require squashing, as described above.

In a recent independent work, Gentry and Halevi [GH11a] showed how the sparse subset sum assumption can be replaced by either the (decisional) 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.

Followup Work. We now describe some followup work that appeared since the publication of the conference version of this paper [BV11b]. First, the work of Brakerski, Gentry and Vaikuntanathan [BGV12] presented a considerable refinement and simplification of our modulus reduction technique, and used it to construct a *leveled* fully homomorphic encryption scheme without bootstrapping. Whereas we use modulus reduction in "one shot" to reduce the size of the ciphertext, their main insight is to use it in an "iterative" way to carefully manage the growth of noise in the ciphertext, and ultimately achieve leveled fully homomorphic encryption without using bootstrapping altogether. We point out in order to achieve a *true (non-leveled)* fully homomorphic encryption scheme, both our methods as well as the result of [BGV12] needs bootstrapping and consequently, "circular security" type assumptions. Removing these remains a very important open problem in the design of fully homomorphic encryption. Brakerski [Bra12] simplified this construction and improved the underlying assumptions.

Our re-linearization and dimension-modulus reduction techniques are quite general, and they can be applied to number of other somewhat homomorphic encryption schemes. We list two such examples below.

Most notably, the work of Gentry, Halevi and Smart [GHS11b, GHS11a] presents a number of clever optimization techniques that apply to the Ring LWE-based scheme from [BV11a] combined with the modulus reduction technique from this work, in order to encrypt and operate on many bits at once (the so-called SIMD mode).

Fully homomorphic encryption schemes operate on ciphertexts encrypted under the same key. Quite recently, Lopéz-Alt, Tromer and Vaikuntanathan [LTV12] constructed a *multi-key* fully homomorphic encryption scheme that can operate on encryptions under different, unrelated public keys. The resulting ciphertext after homomorphic evaluation can be decrypted using the knowledge of all the constituent secret keys. Their construction is based on the NTRU encryption scheme [HPS98], and uses our re-linearization and modulus reduction techniques to turn NTRU into a (multi-key) fully homomorphic encryption scheme.

As mentioned above, Gentry, Sahai and Waters [GSW13] recently showed how to achieve a fully homomorphic encryption scheme that does not require additional auxiliary information for

the homomorphic evaluation. This scheme uses matrices for ciphertexts instead of vectors. This allows to implement our outline above to achieve fully homomorphic identity based encryption, as well as attribute based encryption.

1.6 Paper Organization

Some preliminaries and notation are described in Section 2. We formally define somewhat and fully homomorphic encryption and present the bootstrapping theorem in Section 3. The main technical section of this paper is Section 4, where our scheme is presented and fully analyzed. Lastly, our private information retrieval protocol is presented in Section 5.

2 Preliminaries

Notations. Let $\mathcal{D}$ denote a distribution over some finite set S. Then, $x \xleftarrow{\$} \mathcal{D}$ will denote the fact that x is chosen from the distribution $\mathcal{D}$. When we write $x \xleftarrow{\$} S$, we simply mean that x is sampled from the uniform distribution over S. Unless explicitly mentioned, all logarithms are to base 2. All arithmetics are performed over the integers (or reals). Modular arithmetic will be explicitly pointed out.

In this work, we utilize "noise" distributions over integers. The only property of these distributions we use is their magnitude. Hence, we define a B-bounded distribution to be a distribution over the integers where the magnitude of a sample is bounded. A definition follows.

Definition 2.1 (B-bounded distributions)**.** *A distribution ensemble $\{\chi_n\}_{n\in\mathbb{N}}$, is called B-bounded if it is only supported over $\mathbb{Z} \cap [-B, B]$.*

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], stated in terms of distinguishing advantage.

Lemma 2.1 (matrix-vector leftover hash lemma)**.** *Let $\kappa \in \mathbb{N}$, $n \in \mathbb{N}$, $q \in \mathbb{N}$, and $m \geq n \log q + 2\kappa$. 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 for any function f*

$$\left|\Pr[f(\mathbf{A}, \mathbf{A}^T\mathbf{r}) = 1] - \Pr[f(\mathbf{A}, \mathbf{y}) = 1]\right| \leq 2^{-\kappa} \ .$$

Computational Model. Throughout this work, our model of efficient computation is families of polynomial time algorithms, which can be uniform or non-uniform. All algorithms and security reductions we present are uniform, and thus preserve the uniformity of the underlying security assumption. Our running time analysis will always be asymptotic. For the sake of readability, "an efficient algorithm $\mathcal{A}$" will refer to an ensemble indexed by the security parameter $\{\mathcal{A}_\kappa\}_{\kappa\in\mathbb{N}}$. Similarly "an algorithm $\mathcal{A}$ running in time t" will refer to an ensemble $\{\mathcal{A}_\kappa\}_{\kappa\in\mathbb{N}}$ in which each $\mathcal{A}_\kappa$ runs in time at most $t(\kappa)$. In a uniform setting $\mathcal{A}$ is a Turing machine, and $\mathcal{A}_\kappa = \mathcal{A}(1^\kappa)$.

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 $q \geq 2$, 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 $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly at random and a noise term $e \xleftarrow{\$} \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. 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 $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We denote by $\mathsf{DLWE}_{n,q,\chi}$ the variant where the adversary gets oracle access to $A_{\mathbf{s},\chi}$, and is not a-priori bounded in the number of samples.*

For an algorithm $\mathcal{B}$ and security parameter κ, we denote

$$\mathsf{DLWE}_{n,q,\chi}\mathrm{Adv}[\mathcal{B}] \triangleq \left| \Pr\left[\mathcal{B}^{A_{\mathbf{s},\chi}}(1^\kappa) = 1\right] - \Pr\left[\mathcal{B}^{U(\mathbb{Z}_q^n \times \mathbb{Z}_q)}(1^\kappa) = 1\right] \right| .$$

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 worst-case lattices. Specifically, these reductions take χ to be (discretized versions of) the Gaussian distribution. These distributions can easily be made B-bounded for an appropriate B by rejection sampling without effecting the validity of the reduction. 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 $q = q(n) \in \mathbb{N}$ be a product of co-prime numbers $q = \prod q_i$ such that for all i, $q_i = \mathrm{poly}(n)$, and let $B \geq n$. Then there exists an efficiently sampleable B-bounded distribution χ such that if there is an efficient algorithm that solves the (average-case) $\mathsf{DLWE}_{n,q,\chi}$ problem. Then there exist an efficient algorithm that approximates short vector problems in n-dimensional lattices to within a $\widetilde{O}(n\sqrt{n} \cdot q/B)$ factor. Specifically:*

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

We refer the reader to [Reg05, Pei09] for the formal definition of these lattice problems, as they have no direct connection to this work. We only note here 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 $2^{\tilde{O}(n/k)}$ [Sch87].

2.2 Symmetric Encryption

A symmetric encryption scheme $\mathsf{SYM} = (\mathsf{SYM.Keygen}, \mathsf{SYM.Enc}, \mathsf{SYM.Dec})$, over message space $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$, is a triple of PPT algorithms as follows. We always denote the security parameter by κ.

- **Key generation.** The algorithm $sk \leftarrow \mathsf{SYM.Keygen}(1^\kappa)$ takes a unary representation of the security parameter and outputs symmetric encryption/decryption key sk.
- **Encryption.** The algorithm $c \leftarrow \mathsf{SYM.Enc}_{sk}(\mu)$ takes the symmetric key sk and a message $\mu \in \mathcal{M}_\kappa$ and outputs a ciphertext c.
- **Decryption.** The algorithm $\mu^* \leftarrow \mathsf{SYM.Dec}_{sk}(c)$ takes the symmetric key sk and a ciphertext c and outputs a message $\mu^* \in \mathcal{M}_\kappa$.

Correctness and security against chosen plaintext attacks (IND-CPA security) are defined as follows.

Definition 2.3. *A symmetric scheme* SYM *is correct if for all* $\mu \in \mathcal{M}_\kappa$ *and all* $sk \leftarrow \mathsf{SYM.Keygen}(1^\kappa)$,

$$\Pr[\mathsf{SYM.Dec}_{sk}(\mathsf{SYM.Enc}_{sk}(\mu)) \neq \mu] = \mathrm{negl}(\kappa) \ ,$$

where the probability is over the coins of $\mathsf{SYM.Keygen}$, $\mathsf{SYM.Enc}$.

Definition 2.4. *Let* $t = t(\kappa)$, $\epsilon = \epsilon(\kappa)$ *be some functions. A symmetric scheme* SYM *is* (t, ϵ)*-IND-CPA secure if for any adversary* $\mathcal{A}$ *that runs in time* t *it holds that*

$$\left|\Pr[\mathcal{A}^{\mathsf{SYM.Enc}_{sk}(\cdot)}(1^\kappa) = 1] - \Pr[\mathcal{A}^{\mathsf{SYM.Enc}_{sk}(0)}(1^\kappa) = 1]\right| \leq \epsilon(\kappa) \ ,$$

where the probability is over $sk \leftarrow \mathsf{SYM.Keygen}(1^\kappa)$, *the coins of* $\mathsf{SYM.Enc}$ *and the coins of the adversary* $\mathcal{A}$.

Namely, no adversary can distinguish between an oracle that encrypts messages of its choice and an oracle that only returns encryptions of 0 (where 0 is some arbitrary element in the message space).

3 Homomorphic Encryption: Definitions and Tools

In this section we discuss the definition of homomorphic encryption and its properties, as well as some related subjects. We start by defining homomorphic and fully homomorphic encryption in Section 3.1. Then, in Section 3.2 we discuss Gentry's bootstrapping theorem.

We note that there are a number of ways to define homomorphic encryption and to describe the bootstrapping theorem. We chose the definitions that best fit our constructions and we urge even the knowledgeable reader to go over them so as to avoid confusion in interpreting our results.

3.1 Homomorphic Encryption – Definitions

We now define homomorphic encryption and its desired properties. Throughout this section (and this work) we use κ to indicate the security parameter. In addition, all schemes in this paper encrypt bit-by-bit and therefore our definitions only refer to this case. The generalization to an arbitrary message space is immediate.

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

- **Key generation.** The algorithm $(pk, evk, sk) \leftarrow \mathsf{HE.Keygen}(1^\kappa)$ takes a unary representation of the security parameter and outputs a public encryption key pk, a public evaluation key evk and a secret decryption key sk.
- **Encryption.** The algorithm $c \leftarrow \mathsf{HE.Enc}_{pk}(\mu)$ takes the public key pk and a single bit message $\mu \in \{0,1\}$ and outputs a ciphertext c.
- **Decryption.** The algorithm $\mu^* \leftarrow \mathsf{HE.Dec}_{sk}(c)$ takes the secret key sk and a ciphertext c and outputs a message $\mu^* \in \{0,1\}$.
- **Homomorphic evaluation.** The algorithm $c_f \leftarrow \mathsf{HE.Eval}_{evk}(f, c_1, \ldots, c_\ell)$ takes the evaluation key evk, a function $f : \{0,1\}^\ell \to \{0,1\}$ and a set of ℓ ciphertexts $c_1, \ldots, c_\ell$, and outputs a ciphertext c_f.

  The representation of the function f is an important issue. Since the representation can vary between schemes, we leave this issue outside of this syntactic definition. We remark, however, that in this work, f will be represented by an arithmetic circuit over GF(2).

We note that while one can treat the evaluation key as a part of the public key, as has been done in the literature so far, we feel that there is an expository value to treating it as a separate entity and to distinguishing between the public elements that are used for encryption and those that are used only for homomorphic evaluation.

The only security notion we consider in this work is semantic security, namely security w.r.t. passive adversaries. We use its widely known formulation as IND-CPA security, defined as follows.

Definition 3.1 (CPA security)**.** *A scheme* HE *is IND-CPA secure if for any polynomial time adversary* $\mathcal{A}$ *it holds that*

$$\mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}] \triangleq |\Pr[\mathcal{A}(pk, evk, \mathsf{HE.Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, evk, \mathsf{HE.Enc}_{pk}(1)) = 1]| = \mathrm{negl}(\kappa) \ ,$$

where $(pk, evk, sk) \leftarrow \mathsf{HE.Keygen}(1^\kappa)$.

In fact, based on the best known about lattices, the schemes we present in this paper will be secure against even stronger adversaries. In order for our reductions to make sense for such adversaries as well, we also consider a parameterized version of CPA security. There, we allow the adversary to run in time t (which is typically super-polynomial) and succeed with probability ϵ (which is typically sub-polynomial).

Definition 3.2 ((t,ϵ)-CPA security)**.** *A scheme* HE *is* (t,ϵ)*-IND-CPA secure if for any adversary* $\mathcal{A}$ *that runs in time* t *for* $t = t(\kappa)$ *it holds that*

$$\mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}] \triangleq |\Pr[\mathcal{A}(pk, evk, \mathsf{HE.Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, evk, \mathsf{HE.Enc}_{pk}(1)) = 1]| \leq \epsilon = \epsilon(\kappa) \ ,$$

where $(pk, evk, sk) \leftarrow \mathsf{HE.Keygen}(1^\kappa)$.

We move on to define the homomorphism property. Note that we do not define the "correctness" of the scheme as a separate property, but rather (some form of) correctness will follow from our homomorphism properties.

We start by defining $\mathcal{C}$-homomorphism, which is homomorphism with respect to a specified class $\mathcal{C}$ of functions. This notion is sometimes also referred to as "somewhat homomorphism".

Definition 3.3 ($\mathcal{C}$-homomorphism)**.** *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be a class of functions (together with their respective representations). A scheme* HE *is $\mathcal{C}$-homomorphic (or, homomorphic for the class $\mathcal{C}$) if for any sequence of functions $f_\kappa \in \mathcal{C}_\kappa$ and respective inputs $\mu_1, \ldots, \mu_\ell \in \{0,1\}$ (where $\ell = \ell(\kappa)$), it holds that*

$$\Pr\left[\mathsf{HE.Dec}_{sk}(\mathsf{HE.Eval}_{evk}(f, c_1, \ldots, c_\ell)) \neq f(\mu_1, \ldots, \mu_\ell)\right] = \mathrm{negl}(\kappa) \ ,$$

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

We point out two important properties that the above definition *does not* require. First of all, we do not require that the ciphertexts c_i are decryptable themselves, only that they become decryptable after homomorphic evaluation.[17] Secondly, we do not require that the output of HE.Eval can undergo additional homomorphic evaluation.[18]

Before we define full homomorphism, let us define the notion of *compactness*.

Definition 3.4 (compactness)**.** *A homomorphic scheme* HE *is compact if there exists a polynomial $s = s(\kappa)$ such that the output length of $\mathsf{HE.Eval}(\cdots)$ is at most s bits long (regardless of f or the number of inputs).*

Note that a $\mathcal{C}$-homomorphic scheme is not necessarily compact.

We give the minimal definition of fully homomorphic encryption, which suffices for most applications.

Definition 3.5 (fully homomorphic encryption)**.** *A scheme* HE *is fully homomorphic if it is both compact and homomorphic for the class of all arithmetic circuits over $GF(2)$.*

As in the definition of $\mathcal{C}$ homomorphism, one can require that the outputs of HE.Eval can again be used as inputs for homomorphic evaluation ("multi-hop homomorphism"). Indeed, any bootstrappable scheme (see Section 3.2 below), including ours, has this additional property. However, due to the complexity of the formal definition in this case, we refrain from presenting a formal definition.

An important relaxation of fully homomorphic encryption is the following.

Definition 3.6 (leveled fully homomorphic encryption)**.** *A leveled fully homomorphic encryption scheme is a homomorphic scheme where the* HE.Keygen *gets an additional input 1^L (now $(pk, evk, sk) \leftarrow \mathsf{HE.Keygen}(1^\kappa, 1^L)$) and the resulting scheme is homomorphic for all depth-L binary arithmetic circuits. The bound $s(\kappa)$ on the ciphertext length must remain independent of L.*

In most cases, the only parameter of the scheme that becomes dependent on L is the bit-length of the evaluation key evk.

[17] Jumping ahead, while this may seem strange at first, this notion of somewhat homomorphism is all that is really required in order to bootstrap into full homomorphism and it also makes our schemes easier to describe. Lastly, note that one can always perform a "blank" homomorphic operation and then decrypt, so functionality is not hurt.

[18] This is termed "1-hop homomorphism" in [GHV10a].

3.2 Gentry's Bootstrapping Technique

In this section we formally define the notion of a bootstrappable encryption scheme and present Gentry's bootstrapping theorem [Gen09b, Gen09a] which implies that a bootstrappable scheme can be converted into a fully homomorphic one.

Definition 3.7 (bootstrappable encryption scheme)**.** *Let* HE *be* $\mathcal{C}$*-homomorphic, and let* f_{add} *and* f_{mult} *be the the augmented decryption functions of the scheme defined as*

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

where c_1, c_2 *are either properly encrypted ciphertexts of the scheme, or outputs of the homomorphic evaluation function, applied to such.*

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

We describe two variants of Gentry's bootstrapping theorem. The first implies leveled fully homomorphic encryption but requires no additional assumption; where the second makes an additional *(weak) circular security* assumption and achieves the stronger (non-leveled) variant of Definition 3.5.

The first variant follows.

Theorem 3.1 ([Gen09b, Gen09a])**.** *Let* HE *be a bootstrappable scheme, then there exists a leveled fully homomorphic encryption scheme as per Definition 3.6.*

Specifically, the leveled homomorphic scheme is such that only the length of the evaluation key depends on the level L. All other parameters of the scheme are distributed identically regardless of the value of L.

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

Definition 3.8 (weak circular security)**.** *A public key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is weakly circular secure if it is IND-CPA secure even for an adversary with auxiliary information containing encryptions of all secret key bits:* $\{\mathsf{Enc}_{pk}(sk[i])\}_i$*.*

Namely, no polynomial time adversary can distinguish an encryption of 0 from an encryption of 1 even given the additional information.

We can now state the second theorem.

Theorem 3.2 ([Gen09b, Gen09a])**.** *Let* HE *be a bootstrappable scheme that is also weakly circular secure. Then there is a fully homomorphic encryption scheme as per Definition 3.5.*

Finally, we want to make a statement regarding the ciphertext length of a bootstrapped scheme. The following is implicit in [Gen09b, Gen09a].

Lemma 3.3. *If a scheme* FH *is obtained from applying either Theorem 3.1 or Theorem 3.2 to a bootstrappable scheme* HE*, then both* $\mathsf{FH.Enc}$ *and* $\mathsf{FH.Eval}$ *produce ciphertexts of the same length as* $\mathsf{HE.Eval}$ *(regardless of the length of the ciphertext produced by* $\mathsf{HE.Enc}$*).*

4 The New Fully Homomorphic Encryption Scheme

In this section, we present our fully homomorphic encryption scheme and analyze its security and performance. We present our scheme in a gradual manner. First, in Section 4.1 we present an LWE-based somewhat homomorphic scheme, SH, that will serve as building block for our construction (that scheme by itself is not sufficient to achieve full homomorphism). The main technique used here is re-linearization. Our bootstrappable scheme, BTS, which utilizes dimension-modulus reduction, is presented in Section 4.2. We then turn to analyze the properties of BTS. In Section 4.3 we prove the security of the scheme based on LWE and discuss the worst case hardness that is implied by known reductions. In Section 4.4 we analyze the homomorphic properties of SH and BTS which enables us to prove (in Section 4.5) that the bootstrapping theorem is indeed applicable to BTS, and obtain a fully homomorphic scheme based on LWE. We then discuss the parameters and efficiency of our scheme.

4.1 The Scheme SH: A Somewhat Homomorphic Encryption Scheme

We present a somewhat homomorphic public-key encryption scheme, based on our re-linearization technique, whose message space is $\mathrm{GF}(2)$.[19] 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 $q \in \mathbb{N}$ (note that q needs not be prime) 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 $L \in \mathbb{N}$ which is an upper bound on the maximal multiplicative depth 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, $m \geq n \log q + 2\kappa$ is a polynomial in n, the modulus is an odd number $q \in [2^{n^\epsilon}, 2 \cdot 2^{n^\epsilon})$ is sub-exponential in n (where $\epsilon \in (0,1)$ is some constant), χ is some noise distribution that produces small samples (say, of magnitude at most n) in $\mathbb{Z}_q$, and the depth bound is $L \approx \epsilon \log n$.

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

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

  where $\mathbf{a}_{\ell,i,j,\tau} \xleftarrow{\$} \mathbb{Z}_q^n$, $e_{\ell,i,j,\tau} \xleftarrow{\$} \chi$ (recall that, according to our notational convention, $\mathbf{s}_{\ell-1}[0] \triangleq 1$). We define $\Psi \triangleq \{\psi_{\ell,i,j,\tau}\}_{\ell,i,j,\tau}$ to be the set of all these values.[20] At this point, it may not yet be clear what the purpose of the 2^τ factors is; indeed, this will be explained later when we explain homomorphic multiplication.

  Recalling our intuition from Section 1.1, the keys $\mathbf{s}_0, \ldots, \mathbf{s}_L$ will correspond to the L levels of homomorphic evaluation, and $\psi_{\ell,i,j,\tau}$ are auxiliary parameters that allow to re-linearize level $\ell - 1$ ciphertexts after homomorphic evaluation.

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

[20] A knowledgeable reader may notice that the above is similar to encryptions of $2^\tau \cdot \mathbf{s}_{\ell-1}[i] \cdot \mathbf{s}_{\ell-1}[j] \pmod q$ via an LWE-based scheme, except this "ciphertext" is not decryptable since the "message" is not a single bit value.

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 $\mathbf{b} := \mathbf{A}\mathbf{s}_0 + 2\mathbf{e} \pmod{q} \in \mathbb{Z}_q^m$.

It then outputs the secret key $sk = \mathbf{s}_L$, the evaluation key $evk = \Psi$, and the public key $pk = (\mathbf{A}, \mathbf{b})$.[21]

- Encryption $\mathsf{SH.Enc}_{pk}(\mu)$: Recall that $pk = (\mathbf{A}, \mathbf{b})$. To encrypt a message $\mu \in \mathrm{GF}(2)$, sample a vector $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$ and set (just like in Regev's scheme)

$$\mathbf{v} := \mathbf{A}^T\mathbf{r} \pmod{q} \qquad \text{and} \qquad w := \mathbf{b}^T\mathbf{r} + \mu \pmod{q} .$$

  The output ciphertext contains the pair $(\mathbf{v}, w)$, in addition to a "level tag" which is used during homomorphic evaluation and indicates the "multiplicative depth" where the ciphertext has been generating. For freshly encrypted ciphertext, therefore, the level tag is zero. Formally, the encryption algorithm outputs $c := ((\mathbf{v}, w), 0)$.

- Homomorphic evaluation $\mathsf{SH.Eval}_{evk}(f, c_1, \ldots, c_t)$ where $f : \{0,1\}^t \to \{0,1\}$: We require that f is represented by a binary arithmetic circuit with '+' gates of arbitrary fan-in and '×' gates with fan-in 2. We further require that the circuit is *layered*, namely that it is composed of homogenous layers of either all '+' gates or all '×' gates (it is easy to see that any arithmetic circuit can be converted to this form). Lastly, we require that the multiplicative depth of the circuit (the total number of '×' layers) is exactly L.[22]

  We homomorphically evaluate the circuit f gate by gate. Namely, we will show how to perform homomorphic addition (of arbitrarily many ciphertexts) and homomorphic multiplication (of two ciphertexts). Combining the two, we will be able to evaluate any such function f.

Ciphertext structure during evaluation. During the homomorphic evaluation, we will generate ciphertexts of the form $c = ((\mathbf{v}, w), \ell)$, where the tag ℓ indicates the multiplicative level at which the ciphertext has been generated (hence fresh ciphertexts are tagged with 0). The requirement that f is layered will make sure that throughout the homomorphic evaluation all inputs to a gate have the same tag. In addition, we will keep the invariant that the output of each gate evaluation $c = ((\mathbf{v}, w), \ell)$, is such that

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

where $\mu \in \mathrm{GF}(2)$ is the correct plaintext output of the gate, and e is a noise term that depends on the gate's input ciphertexts. Note that it always holds that $\ell \leq L$ due to the bound on the multiplicative depth, and that the output of the homomorphic evaluation of the entire circuit is expected to have $\ell = L$.

Homomorphic evaluation of gates:

[21] The public key pk is essentially identical to the public key in Regev's scheme.

[22] Jumping ahead, in the analysis we will only prove correctness for a specific sub-class of these circuits.

– *Addition gates.* Homomorphic evaluation of a '+' gate on inputs $c_1, \ldots, c_t$, where $c_i = ((\mathbf{v}_i, w_i), \ell)$, is performed by outputting

$$c_{\mathsf{add}} = ((\mathbf{v}_{\mathsf{add}}, w_{\mathsf{add}}), \ell) := \left(\left(\sum_i \mathbf{v}_i, \sum_i w_i \right), \ell \right) .$$

Informally, one can see that

$$w_{\mathsf{add}} - \langle \mathbf{v}_{\mathsf{add}}, \mathbf{s}_\ell \rangle = \sum_i (w_i - \langle \mathbf{v}_i, \mathbf{s}_\ell \rangle) = \sum_i (\mu_i + 2e_i) = \sum_i \mu_i + 2 \sum_i e_i \pmod q ,$$

where μ_i is the plaintext corresponding to c_i. So long as the term $\sum_i \mu_i + 2\sum_i e_i$ is less than $q/2$ (in absolute value), it will hold that c_{add} will be decrypted to $\mu_{\mathsf{add}} = \sum_i \mu_i \pmod 2$ as desired.

– *Multiplication gates.* We show how to multiply ciphertexts c, c' where $c = ((\mathbf{v}, w), \ell)$ and $c' = ((\mathbf{v}', w'), \ell)$ (recall that multiplication gates have fan-in 2), to obtain an output ciphertext $c_{\mathsf{mult}} = ((\mathbf{v}_{\mathsf{mult}}, w_{\mathsf{mult}}), \ell + 1)$. Note that the level tag increases by 1.
We first consider an n-variate *symbolic* polynomial over the unknown vector $\mathbf{x}$:

$$\phi(\mathbf{x}) = \phi_{(w,\mathbf{v}),(w',\mathbf{v}')}(\mathbf{x}) \triangleq (w - \langle \mathbf{v}, \mathbf{x} \rangle) \cdot (w' - \langle \mathbf{v}', \mathbf{x} \rangle) \pmod q . \tag{3}$$

We symbolically open the parenthesis of this quadratic polynomial, and express it as

$$\phi(\mathbf{x}) = \sum_{0 \le i \le j \le n} h_{i,j} \cdot \mathbf{x}[i] \cdot \mathbf{x}[j] \pmod q ,$$

where $h_{i,j} \in \mathbb{Z}_q$ are known (we can compute them from $(\mathbf{v}, w), (\mathbf{v}', w')$ by opening parenthesis in Eq. (3)).[23]
For technical reasons (related to keeping the error growth under control), we want to express $\phi(\cdot)$ as a polynomial with small coefficients. We consider the binary representation of $h_{i,j}$, letting $h_{i,j,\tau}$ be the τ^{th} bit in this representation. In other words

$$h_{i,j} = \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,j,\tau} \cdot 2^\tau \pmod q ,$$

for $h_{i,j,\tau} \in \{0, 1\}$.
We can express ϕ therefore as

$$\phi(\mathbf{x}) = \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot \left(2^\tau \cdot \mathbf{x}[i] \cdot \mathbf{x}[j]\right) \pmod q .[24]$$

We recall that the evaluation key $evk = \Psi$ contains elements of the form $\psi_{\ell,i,j,\tau} = (\mathbf{a}_{\ell,i,j,\tau}, b_{\ell,i,j,\tau})$ such that

$$2^\tau \mathbf{s}_\ell[i] \mathbf{s}_\ell[j] \approx b_{\ell+1,i,j,\tau} - \langle \mathbf{a}_{\ell+1,i,j,\tau}, \mathbf{s}_{\ell+1} \rangle \pmod q .$$

[23] 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 $\phi(\mathbf{x})$.
[24] This can be interpreted as a polynomial with small coefficients whose variables are $(2^\tau \cdot \mathbf{x}[i] \cdot \mathbf{x}[j])$.

The homomorphic multiplication algorithm will thus set

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

and

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

The final output ciphertext will be

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

Note that the level tag is increased by one as expected. Let us now verify that our invariant as per Eq. (2) still holds for the new ciphertext:

$$\begin{aligned}
w_{\mathsf{mult}} - \langle \mathbf{v}_{\mathsf{mult}}, \mathbf{s}_{\ell+1} \rangle &= \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0,\ldots,\lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot (b_{\ell+1,i,j,\tau} - \langle \mathbf{a}_{\ell+1,i,j,\tau}, \mathbf{s}_{\ell+1} \rangle) \\
&= \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0,\ldots,\lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot 2^{\tau} \cdot \mathbf{s}_{\ell}[i] \cdot \mathbf{s}_{\ell}[j] + 2 \cdot h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \\
&= \phi(\mathbf{s}_{\ell}) + \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0,\ldots,\lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \\
&= (w - \langle \mathbf{v}, \mathbf{s}_{\ell} \rangle) \cdot (w' - \langle \mathbf{v}', \mathbf{s}_{\ell} \rangle) + \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0,\ldots,\lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \\
&= (\mu + 2e)(\mu' + 2e') + \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0,\ldots,\lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \\
&= \mu\mu' + 2 \left(\mu e' + \mu' e + 2ee' + \sum_{\substack{0 \le i \le j \le n \\ \tau \in \{0,\ldots,\lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \right) , \quad (4)
\end{aligned}$$

where all equalities are modulo q.

Indeed, we get the plaintext output $\mu\mu'$ in addition to a noise term that is inherited from the input ciphertexts and from the evaluation key.

- Decryption $\mathsf{SH.Dec}_{\mathbf{s}_L}(c)$: To decrypt a ciphertext $c = ((\mathbf{v}, w), L)$ (recall that we are only required to decrypt ciphertexts that are output by $\mathsf{SH.Eval}(\cdots)$ and those will always have level tag L), compute

$$(w - \langle \mathbf{v}, \mathbf{s}_L \rangle \pmod{q}) \pmod{2} . \quad (5)$$

4.2 The Scheme BTS: A Bootstrappable Scheme

We now utilize the dimension-modulus reduction technique to present the scheme BTS, which uses SH as building block and inherits its homomorphic properties. However, BTS has much shorter ciphertexts and lower decryption complexity, which will enable us to apply the bootstrapping theorem to obtain full homomorphism.

Our bootstrappable scheme is parameterized by (n, m, q, χ, L), which are the parameters for SH, and additional parameters $(k, p, \hat{\chi})$ which are the "smaller" 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 $\mathbb{Z}_q$ and $\mathbb{Z}_p$, respectively. The parameter $m \in \mathbb{N}$ is used towards public key generation. The parameter L is an upper bound on the multiplicative depth of the evaluated function.

While we discuss parameter values below, we encourage the reader to consider the following (non-optimal, but easier to understand) settings as a running example: $k = \kappa$, $n = k^4$, $q \approx 2^{\sqrt{n}}$, $L = 1/3 \log n = 4/3 \log k$, $p = (n^2 \log q) \cdot \mathrm{poly}(k) = \mathrm{poly}(k)$, $m = O(n \log q)$. The distributions $\chi, \hat{\chi}$ can be thought of as being n- and k-bounded, respectively.

- Key generation $\mathsf{BTS.Keygen}(1^\kappa)$: Run $\mathsf{SH.Keygen}(1^\kappa)$ to obtain the secret key $\mathbf{s}_L$, evaluation key Ψ and public key $(\mathbf{A}, \mathbf{b})$ of SH.

  Recall that $\mathbf{s}_L \in \mathbb{Z}_q^n$, $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, and $\Psi \in (\mathbb{Z}_q^n \times \mathbb{Z}_q)^{(n+1)^2 \cdot (\lfloor \log q \rfloor + 1) \cdot L}$.

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

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

  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

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

  This is very similar to the generation of Ψ in the scheme SH, but now $\hat{\psi}_{i,\tau}$ "encodes" scaled linear terms, rather than quadratic terms.

  Finally, output the secret key $sk = \hat{\mathbf{s}}$, evaluation key $evk = (\Psi, \hat{\Psi})$ and public key $pk = (\mathbf{A}, \mathbf{b})$. Note that the public key is identical to that of SH.

- Encryption $\mathsf{BTS.Enc}_{pk}(\mu)$: Use the same encryption algorithm as SH. To encrypt a bit $\mu \in \{0, 1\}$, compute $c \leftarrow \mathsf{SH.Enc}_{(\mathbf{A},\mathbf{b})}(\mu)$ and output c as the ciphertext.

- Homomorphic evaluation $\mathsf{BTS.Eval}_{evk}(f, c_1, \ldots, c_t)$, where $f : \{0,1\}^t \to \{0,1\}$: Recall that $evk = (\Psi, \hat{\Psi})$. To perform homomorphic evaluation, we will use the homomorphic evaluation function of SH. We thus require that f is represented by a binary arithmetic circuit which is a legal input for $\mathsf{SH.Eval}$.

  The first step in the homomorphic evaluation is computing

  $$c_f \leftarrow \mathsf{SH.Eval}_\Psi(f, c_1, \ldots, c_t) \ .$$

This results in a ciphertext of the form $c_f = ((\mathbf{v}, w), L) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \times \{L\}$.

Next, we reduce the dimension and modulus of c_f to k, p as follows. 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

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

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

$$\begin{aligned} \hat{\mathbf{v}} &:= 2 \cdot \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \hat{\mathbf{a}}_{i,\tau} \pmod{p} \in \mathbb{Z}_p^k \\ \hat{w} &:= 2 \cdot \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \hat{b}_{i,\tau} \pmod{p} \in \mathbb{Z}_p . \end{aligned}$$

The output of $\mathsf{BTS.Eval}$ is the new ciphertext $\hat{c} \in \mathbb{Z}_p^k \times \mathbb{Z}_p$. Note that the bit-length of $\hat{c}$ is $(k+1) \log p$.

Recall the invariant we enforce on the structure of ciphertexts of SH (see Eq. 2). We show that a similar invariant holds for $\hat{c}$: Namely, that if c_f is such that $w - \langle \mathbf{v}, \mathbf{s}_L \rangle = \mu + 2e \pmod{q}$, then

$$\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle = \mu + 2\hat{e} \pmod{p} ,$$

where $\hat{e}$ is proportional to $\frac{p}{q}e$ (an appropriately scaled version of e) plus some additional noise.

To see the above, recall that $(p+1)/2$ is the inverse of 2 modulo p, and notice that[25]

$$\begin{aligned} \frac{p+1}{2} (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) &= \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \left(\hat{b}_{i,\tau} - \langle \hat{\mathbf{a}}_{i,\tau}, \hat{\mathbf{s}} \rangle \right) \pmod{p} \\ &= \sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \left(\hat{e}_{i,\tau} + \left\lfloor \frac{p}{q} \cdot \left(2^{\tau} \cdot \mathbf{s}_L[i] \right) \right\rceil \right) \pmod{p} \\ &= \phi(\mathbf{s}_L) + \underbrace{\sum_{i=0}^{n} \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \left(\hat{e}_{i,\tau} + \hat{\omega}_{i,\tau} \right)}_{\triangleq \ \delta_1} \pmod{p} , \end{aligned} \tag{6}$$

[25] While the following sequence of derivations might seem like an indirect way to prove what we need, the way we choose to do it will be useful later.

where we define

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

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 Eq. (6)) is "small" as well.

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

$$\begin{aligned}
\phi(\mathbf{s}_L) &\triangleq \frac{p}{q} \cdot \left(\frac{q+1}{2} \cdot (w - \langle \mathbf{v}, \mathbf{s}_L \rangle) \right) \pmod p \\
&= \frac{p}{q} \cdot \left(\frac{q+1}{2} \cdot (2e + \mu + Mq) \right) \pmod p \qquad (\text{where } M \in \mathbb{Z}) \\
&= \frac{p}{q} \cdot \left(\frac{q+1}{2}\mu + e + M'q \right) \pmod p \qquad (\text{where } M' \in \mathbb{Z}) \\
&= \frac{p}{q} \cdot \frac{q+1}{2}\mu + \frac{p}{q} \cdot e \pmod p \\
&= \frac{p+1}{2} \cdot \mu + \underbrace{(\frac{p}{q} - 1) \cdot \frac{\mu}{2} + \frac{p}{q} \cdot e}_{\triangleq \delta_2} \pmod p \\
&= \frac{p+1}{2} \cdot \mu + \delta_2 \qquad (7)
\end{aligned}$$

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 Eq. (6) and (7), we see that

$$\frac{p+1}{2}(\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) = \frac{p+1}{2} \cdot \mu + (\delta_1 + \delta_2) . \qquad (8)$$

Multiplying by 2, we have

$$\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle = \mu + 2(\delta_1 + \delta_2) . \qquad (9)$$

Now defining $\hat{e} \triangleq \delta_1 + \delta_2$, the invariant follows.

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

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

- Decryption $\mathsf{BTS.Dec}_{\hat{\mathbf{s}}}(\hat{c})$: To decrypt $\hat{c} = (\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ (recall, again, that we only need to decrypt ciphertexts that are output by $\mathsf{BTS.Eval}$), compute

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

If indeed $\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle = \mu + 2\hat{e} \pmod p$ then $\mu^* = \mu$ so long as $\hat{e}$ is small enough.

4.3 Security Analysis

In this section, we analyze the security of BTS based on LWE and then, using known connections, based on worst case hardness of lattice problems.

The following theorem asserts the security of BTS based on two DLWE problems: One with modulus q, dimension n and noise χ, and one with modulus p, dimension k and noise $\hat{\chi}$.

Theorem 4.1 (security)**.** *Let $n = n(\kappa), k = k(\kappa), q = q(\kappa), p = p(\kappa)$ and $L = L(\kappa)$ be functions of the security parameter. Let $\chi, \hat{\chi}$ be some distributions over the integers, and define $m \triangleq n \log q + 2\kappa$.*

The scheme BTS *is CPA 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 (t, ϵ)-hard, then the scheme is $(t - \mathrm{poly}(\kappa), 2(L+3) \cdot (2^{-\kappa} + \epsilon))$-semantically secure, for some polynomial* $\mathrm{poly}(\cdot)$.

Essentially, the view of a CPA adversary for our scheme is very similar to Regev's scheme, with the exception that our adversary also gets to see the evaluation key. However, the evaluation key contains a sequence of LWE instances which, based on our assumption, are indistinguishable from uniform. Therefore our reduction will perform a sequence of L hybrids to replace the Ψ component of the evaluation key with a set of completely uniform elements. Then, an additional hybrid will imply the same for $\hat{\Psi}$. Once this is done, we will use the known proof techniques from Regev's scheme and get the security of our scheme. A formal proof follows.

Proof. As explained above, we prove by a sequence of hybrids. Let $\mathcal{A}$ be an IND-CPA adversary for BTS that runs in time t. We consider a sequence of hybrids, in each hybrid the adversary $\mathcal{A}$ is given pk, evk, c, where pk, evk are generated according to a distribution specific to the hybrid, and c is generated by applying the encryption algorithm using pk on a message μ. We let $\Pr_{H,\mu}[\mathcal{A}]$ denote the probability that the adversary $\mathcal{A}$ outputs 1 in the experiment defined by hybrid H where the message μ is encrypted.

- **Hybrid $\hat{H}_{L+1}$:** This is the identical to the IND-CPA game, where the adversary gets properly distributed keys pk, evk, generated by BTS.Keygen, and an encryption of either 0 or 1 computed using BTS.Enc. By definition,

$$\mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}] = \left|\Pr_{\hat{H}_{L+1},0}[\mathcal{A}] - \Pr_{\hat{H}_{L+1},1}[\mathcal{A}]\right| . \tag{10}$$

- **Hybrid H_{L+1}:** This hybrid is identical to $\hat{H}_{L+1}$, except in the generation of $\hat{\Psi}$. In this hybrid, $\hat{\Psi}$ is not generated as prescribed, but is rather sampled uniformly. Namely, for all i, τ we set $\hat{\psi}_{i,\tau} \xleftarrow{\$} \mathbb{Z}_p^k \times \mathbb{Z}_p$.

  Consider $\widehat{\mathcal{B}}_\mu(1^k)$, an adversary for the $\mathsf{DLWE}_{k,p,\hat{\chi}}$ problem defined as follows (for $\mu \in \{0,1\}$). The algorithm $\widehat{\mathcal{B}}$ will sample all vectors $\mathbf{s}_0, \ldots, \mathbf{s}_L$ by itself and generate pk, Ψ. Now, note that given $\mathbf{s}_L$, it is possible to generate $\hat{\Psi}$ efficiently using polynomially many samples from the distribution $A_{\hat{\mathbf{s}},\hat{\chi}}$. Furthermore, if $A_{\hat{\mathbf{s}},\hat{\chi}}$ is replaced with a uniform oracle, the same process will result in a completely uniform $\hat{\Psi}$.

  The algorithm $\widehat{\mathcal{B}}$ will draw sufficiently many samples from its oracle, and generate $\hat{\Psi}$ as if this oracle is indeed $A_{\hat{\mathbf{s}},\hat{\chi}}$. Finally, it will return $\mathcal{A}(pk, evk, \mathsf{BTS.Enc}_{pk}(\mu))$ (i.e. call $\mathcal{A}$ on the keys it generated and an encryption of μ).

It holds that

$$\begin{aligned}\Pr[\widehat{\mathcal{B}}_{\mu}^{A_{\hat{\mathbf{s}},\hat{\chi}}}(1^{\kappa})] &= \Pr_{\hat{H}_{L+1},\mu}[\mathcal{A}] \\ \Pr[\widehat{\mathcal{B}}_{\mu}^{U(\mathbb{Z}_p^k\times\mathbb{Z}_p)}(1^{\kappa})] &= \Pr_{H_{L+1},\mu}[\mathcal{A}]\ . \end{aligned} \tag{11}$$

- **Hybrid H_ℓ, for $\ell \in [L]$:** Hybrid H_ℓ is identical to $H_{\ell+1}$, except for a change in the Ψ component of the evaluation key. Specifically, we change each of the components $\psi_{\ell,i,j,\tau}$ for all i, j, τ: Instead of computing $\psi_{\ell,i,j,\tau}$ as prescribed (i.e., $(\mathbf{a}_{\ell,i,j,\tau}, \langle \mathbf{a}_{\ell,i,j,\tau}, \mathbf{s}_\ell\rangle + 2e_{\ell,i,j,\tau} + 2^\tau \cdot \mathbf{s}_{\ell-1}[i]\cdot\mathbf{s}_{\ell-1}[j]))$, we sample it uniformly. Namely, we set $\psi_{\ell,i,j,\tau} \xleftarrow{\$} \mathbb{Z}_q^n \times \mathbb{Z}_q$.

  We now define a procedure $\mathcal{B}_{\ell,\mu}$, for $\ell \in [L]$, $\mu \in \{0,1\}$, that attempts to solve $\mathsf{DLWE}_{n,q,\chi}$. The procedure $\mathcal{B}_{\ell,\mu}$ will work as follows: It will first sample vectors $\mathbf{s}_0, \ldots, \mathbf{s}_{\ell-1} \xleftarrow{\$} \mathbb{Z}_q^n$, and will generate pk and the values $\psi_{\ell',i,j,\tau}$, for all $\ell' < \ell$ properly, according to the prescribed distribution. It will further sample $\psi_{\ell',i,j,\tau}$ for $\ell' > \ell$ uniformly in $\mathbb{Z}_q^n \times \mathbb{Z}_q$ and $\hat{\psi}_{i,\tau}$ uniformly in $\mathbb{Z}_p^k \times \mathbb{Z}_p$.

  Now, we notice that given $\mathbf{s}_{\ell-1}$, it is possible to generate $\psi_{\ell,i,j,\tau}$, for all i, j, τ, given oracle access to $A_{\mathbf{s}_\ell,\chi}$. This is less obvious than in the previous hybrid, since we require the noise element in ψ to be even. However, notice that for an odd q, if $x \xleftarrow{\$} \mathbb{Z}_q$, then $2x$ is uniform in $\mathbb{Z}_q$ as well. Therefore, taking a sample $(\mathbf{a}, b) \xleftarrow{\$} A_{\mathbf{s}_\ell,\chi}$, and considering $(2\mathbf{a}, 2b) \pmod q$, we have that $2b = \langle 2\mathbf{a}, \mathbf{s}_\ell\rangle + 2e \pmod q$. Given such samples, and access to $\mathbf{s}_{\ell-1}$, the generation of $\psi_{\ell,i,j,\tau}$ is straightforward. Furthermore, if $A_{\mathbf{s}_\ell,\chi}$ is replaced with a uniform oracle, then the above process will result in uniformly distributed elements $\psi_{\ell,i,j,\tau}$.

  The procedure $\mathcal{B}_{\ell,\mu}$ will draw samples from its oracle as if it was indeed $A_{\mathbf{s}_\ell,\chi}$ and complete the generation of Ψ as described above. Finally, given the generated pk, evk, the procedure will return $\mathcal{A}(pk, evk, \mathsf{BTS.Enc}_{pk}(\mu))$ (i.e. call $\mathcal{A}$ on the keys it generated and an encryption of μ).

  It holds that

$$\begin{aligned}\Pr[\mathcal{B}_{\ell,\mu}^{A_{\mathbf{s}_\ell,\chi}}(1^{\kappa})] &= \Pr_{H_{\ell+1},\mu}[\mathcal{A}] \\ \Pr[\mathcal{B}_{\ell,\mu}^{U(\mathbb{Z}_q^n\times\mathbb{Z}_q)}(1^{\kappa})] &= \Pr_{H_\ell,\mu}[\mathcal{A}]\ . \end{aligned} \tag{12}$$

  Note that in the hybrid H_1, the evaluation key $evk = (\Psi, \hat{\Psi})$ is completely uniform, and hence the adversary $\mathcal{A}$ gets the same information as in Regev's 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 $\mathbf{A}\cdot\mathbf{s}_0 + 2\mathbf{e}$.

  We define the procedure $\mathcal{B}_{0,\mu}$ for the problem $\mathsf{DLWE}_{n,q,\chi}$, where $\mu \in \{0,1\}$. This procedure will start by sampling $evk = (\Psi, \hat{\Psi})$ uniformly. Now, we notice that given an oracle to $A_{\mathbf{s}_0,\chi}$, it is possible to efficiently generate pk (this requires multiplying the samples by 2 as in the previous hybrid). Furthermore, if a uniform oracle is given instead, this will result in completely uniform $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$.

  The procedure $\mathcal{B}_{0,\mu}$ will draw samples from its oracle as if it is indeed $A_{\mathbf{s}_0,\chi}$, to generate pk. Finally, given the generated pk, evk, the procedure will return $\mathcal{A}(pk, evk, \mathsf{BTS.Enc}_{pk}(\mu))$ (i.e. call $\mathcal{A}$ on the keys it generated and an encryption of μ).

It holds that

$$\begin{aligned}\Pr[\mathcal{B}_{0,\mu}^{A_{\mathbf{s}_0,\chi}}(1^\kappa)] &= \Pr_{H_1,\mu}[\mathcal{A}] \\ \Pr[\mathcal{B}_{0,\mu}^{U(\mathbb{Z}_q^n\times\mathbb{Z}_q)}(1^\kappa)] &= \Pr_{H_0,\mu}[\mathcal{A}]\ .\end{aligned} \tag{13}$$

Furthermore, notice that since an encryption of μ is computed as $(\mathbf{A}^T\cdot\mathbf{r}, \mathbf{b}^T\cdot\mathbf{r}+\mu)$, and since $m > (n+1)\log q + 2\kappa$, we can apply the Leftover hash lemma (Lemma 2.1) which implies that

$$|\Pr_{H_0,0}[\mathcal{A}] - \Pr_{H_0,1}[\mathcal{A}]| \leq 2^{-\kappa}\ . \tag{14}$$

Putting together Eq. (11), (12), (13), we get that for any $\mu \in \{0,1\}$

$$\Pr_{\hat{H}_{L+1},\mu}[\mathcal{A}] - \Pr_{H_0,\mu}[\mathcal{A}] = \left(\Pr[\widehat{\mathcal{B}}_\mu^{A_{\hat{\mathbf{s}},\hat{\chi}}}(1^\kappa)] - \Pr[\widehat{\mathcal{B}}_\mu^{U(\mathbb{Z}_p^k\times\mathbb{Z}_p)}(1^\kappa)]\right) + \sum_{\ell=0}^{L+1}\left(\Pr[\mathcal{B}_{\ell,\mu}^{A_{\mathbf{s},\chi}}] - \Pr[\mathcal{B}_{\ell,\mu}^{U(\mathbb{Z}_q^n\times\mathbb{Z}_q)}]\right)\ . \tag{15}$$

This leads to the following definitions of adversaries $\widehat{\mathcal{B}}, \mathcal{B}$ for $\mathsf{DLWE}_{k,p,\hat{\chi}}$, $\mathsf{DLWE}_{n,q,\chi}$ (respectively):

- The adversary $\widehat{\mathcal{B}}$ for $\mathsf{DLWE}_{k,p,\hat{\chi}}$, will sample $\mu \xleftarrow{\$} \{0,1\}$ and return $\mu \oplus \widehat{\mathcal{B}}_\mu^{(\cdot)}(1^\kappa)$ (namely, it will run $\widehat{\mathcal{B}}$ and either return the same result or flip it based on the value of μ). Note that the running time is a $\mathrm{poly}(\kappa)$ plus the running time of $\mathcal{A}$.
- The adversary $\mathcal{B}$ for $\mathsf{DLWE}_{n,q,\chi}$, will sample $\mu \xleftarrow{\$} \{0,1\}$ and $\ell \xleftarrow{\$} \{0,\ldots,L+1\}$ and return $\mu \oplus \mathcal{B}_{\ell,\mu}^{(\cdot)}(1^\kappa)$. The running time, again, is $\mathrm{poly}(\kappa)$ plus the running time of $\mathcal{A}$.

It follows that

$$\left|\left(\Pr_{\hat{H}_{L+1},0}[\mathcal{A}] - \Pr_{\hat{H}_{L+1},1}[\mathcal{A}]\right) - (\Pr_{H_0,0}[\mathcal{A}] - \Pr_{H_0,1}[\mathcal{A}])\right| \leq 2\mathsf{DLWE}_{k,p,\hat{\chi}}\mathrm{Adv}[\widehat{\mathcal{B}}] + 2(L+2)\mathsf{DLWE}_{n,q,\chi}\mathrm{Adv}[\mathcal{B}]\ . \tag{16}$$

On the other hand, putting together Eq. (10), (14), we get that

$$\left|\left(\Pr_{\hat{H}_{L+1},0}[\mathcal{A}] - \Pr_{\hat{H}_{L+1},1}[\mathcal{A}]\right) - (\Pr_{H_0,0}[\mathcal{A}] - \Pr_{H_0,1}[\mathcal{A}])\right| \geq \mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}] - 2^{-\kappa}\ . \tag{17}$$

In conclusion,

$$\mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}] \leq 2\mathsf{DLWE}_{k,p,\hat{\chi}}\mathrm{Adv}[\widehat{\mathcal{B}}] + 2(L+2)\mathsf{DLWE}_{n,q,\chi}\mathrm{Adv}[\mathcal{B}] + 2^{-\kappa}\ ,$$

and the theorem follows. □

Specific Parameters and Worst-Case Hardness. The parameters we require for homomorphism (see Theorem 4.2 below) are as follows. We require that $q = 2^{n^\epsilon}$ for some $\epsilon \in (0,1)$, χ is n-bounded, $p = 16nk\log(2q)$ and $\hat{\chi}$ is k-bounded. In order to achieve the best lattice reduction, we will choose q as a product of polynomially bounded co-prime numbers. Applying known results (see Corollary 2.2), $\mathsf{DLWE}_{n,q,\chi}$ translates into approximating short-vector problems in worst case n-dimensional lattices to within a factor of $\widetilde{O}\left(\sqrt{n}\cdot 2^{n^\epsilon}\right)$, while $\mathsf{DLWE}_{k,p,\hat{\chi}}$ translates to approximating k-dimensional lattice problems to within $\widetilde{O}\left(n^{1+\epsilon}\cdot k^{1.5}\right)$ factor.[26] These problems are essentially incomparable as the hardness of the problem increases as the dimension increases on one hand, but decreases as the approximation factor increases on the other. The best known algorithms solve the first problem in time (roughly) $2^{\widetilde{O}(n^{1-\epsilon})}$, and the second in time $2^{\widetilde{O}(k)}$.

The relation between n and k is determined based on the required homomorphic properties. In this work, we only prove there there exists a constant C such that setting $n = k^{C/\epsilon}$ implies fully homomorphic encryption. Given the value of C, setting $\epsilon \approx 1 - \frac{1}{C+1}$ will make the two problems equally hard (at least based on the current state of the art).

4.4 Homomorphic Properties of SH And BTS

In this section we analyze the homomorphic properties of SH and BTS. Both schemes have essentially the same homomorphic properties but BTS has the additional advantage of having low decryption complexity (as analyzed in Section 4.5). Thus, BTS would be our main focus, and the properties of SH will follow as a by-product of our analysis.

We start by formally defining the class of functions for which we prove homomorphism and proceed by stating the homomorphic properties and proving them.

The Function Class $\mathsf{Arith}[L,T]$. We define the function class for which we prove somewhat homomorphism of our scheme. Essentially, this is the class of arithmetic circuits over GF(2) with bounded fan-in and bounded depth, with an additional final "collation": a high fan-in addition gate at the last level. We require that the circuit is structured in a canonical "layered" manner as we describe below.

Definition 4.1. *Let $L = L(\kappa), T = T(\kappa)$ be functions of the security parameter. The class $\mathsf{Arith}[L,T]$ is the class of arithmetic circuits over GF(2), with $\{+,\times\}$ gates, with the following structure. Each circuit contains exactly $2L+1$ layers of gates (numbered $1,\ldots,2L+1$ starting from the input level), gates of layer $i+1$ are fed only by gates of layer i. The odd layers contain only '$+$' gates and the even layers contain only '$\times$' gates. The gates at layers $1,\ldots,2L$ have fan-in 2, while the final addition gate in layer $2L+1$ is allowed to have fan-in T.*

We note that $\mathsf{Arith}[L,T]$ conforms with the requirements on the evaluated function imposed by SH.Eval and BTS.Eval. Indeed, the multiplicative depth of any circuit in $\mathsf{Arith}[L,T]$ is exactly L, and hence, homomorphic evaluation is well defined on any such function.

To motivate the choice of this function class, we first note that any arithmetic circuit of fan-in 2 and depth D can be trivially converted into a circuit in $\mathsf{Arith}[D,1]$.[27] This will be useful for

[26] We do not mention the specific lattice problem or the specific type of reduction (quantum vs. classical) since, as one can observe from Corollary 2.2, the approximation factor we get is essentially the same for all problems, and the state of the art is roughly the same as well.

[27] One way to do this is to separate each level of the circuit into two levels – an addition level and a multiplication

the purpose of bootstrapping. Jumping ahead, the collation gate will be useful for constructing a private information retrieval protocol, where we will need to evaluate polynomials with a very large number of monomials and fairly low degree. The collation gate will thus be used for the final aggregation of monomials.

Our goal is now to prove that with the appropriate choice of parameters, SH and BTS are $\mathsf{Arith}[L,T]$-homomorphic.

Theorem 4.2. *Let $n = n(\kappa) \geq 5$ be any polynomial, $q \geq 2^{n^\epsilon} \geq 3$ for some $\epsilon \in (0,1)$ be odd, χ be any n-bounded distribution, and $m = (n+1)\log q + 2\kappa$. Let $k = \kappa$, $p = 16nk\log(2q)$ (odd) and $\hat{\chi}$ be any k-bounded distribution. Then* SH *and* BTS *are both* $\mathsf{Arith}[L = \Omega(\epsilon \log n), T = \sqrt{q}]$*-homomorphic.*

Not surprisingly, the homomorphism class depends only on n and not on k. This is because, recalling the definition of BTS.Eval, the homomorphism property is inherited from SH.Eval. We note that it is possible to further generalize the class of circuits that we can homomorphically evaluate (for example, circuits with high multiplicative depth but low multiplicative degree), however since this is not required for our results, and since the proof will use the exact same tools, we choose not to further complicate the theorem statement and proof.

To prove the theorem, we introduce a sequence of lemmas as follows. Recall that the encryption algorithms of both schemes are identical, and that BTS.Eval first calls SH.Eval on all its inputs. We first analyze the growth of the noise in the execution of SH.Eval in Lemma 4.3 (which will imply the theorem for SH), and then, in Lemma 4.4, we complete the noise calculation of BTS.Eval, which will complete the proof of the theorem.

To track the growth of the noise, we define, for any ciphertext $c = ((\mathbf{v}, w), \ell)$ a noise measure $\eta(c) \in \mathbb{Z}$ as follows. We let $e \in \mathbb{Z}$ be the smallest integer (in absolute value) such that

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

and define $\eta(c) \triangleq \mu + 2e$ (note that $\eta(c)$ is defined over the integers, and not modulo q). We note that so long as $|\eta(c)| < q/2$, the ciphertext is decryptable. We can now bound the error in the execution by bounding $\eta(c_f)$ of the output ciphertext.

Lemma 4.3. *Let $n = n(\kappa) \geq 5$, $q = q(\kappa) \geq 3$, χ be B-bounded and $L = L(\kappa)$ and let $f \in \mathsf{Arith}[L,T]$, $f : \{0,1\}^t \to \{0,1\}$ (for some $t = t(\kappa)$). Then for any input $\mu_1, \ldots, \mu_t \in \{0,1\}$, if we let $(pk, evk, sk) \leftarrow \mathsf{SH.Keygen}(1^\kappa)$, $c_i \leftarrow \mathsf{BTS.Enc}_{pk}(\mu_i) = \mathsf{SH.Enc}_{pk}(\mu_i)$ and we further let $c_f = ((\mathbf{v}, w), L) \leftarrow \mathsf{SH.Eval}_{evk}(f, c_1, \ldots, c_t)$ be the encryption of $f(\mu_1, \ldots, \mu_t)$, it holds that*

$$|\eta(c_f)| \leq T \cdot (16nB\log q)^{2^L} \ .$$

Proof. Recall that all samples of χ are of magnitude at most B. We track the growth of noise as the homomorphic evaluation proceeds.

- **Fresh ciphertexts.** Our starting point is level-0 ciphertexts $((\mathbf{v}, w), 0)$ that are generated by the encryption algorithm. By definition of the encryption algorithm we have that

$$w - \langle \mathbf{v}, \mathbf{s}_0 \rangle = \mathbf{r}^T \cdot \mathbf{b} + \mu - \mathbf{r}^T \cdot \mathbf{A} \cdot \mathbf{s}_0 = \mu + \mathbf{r}^T \cdot (\mathbf{b} - \mathbf{A}\mathbf{s}_0) = \mu + 2\mathbf{r}^T \cdot \mathbf{e} \pmod q \ .$$

level – and finally, adding a dummy fan-in-1 addition gate at the top. This gives us a $2D+1$ depth circuit with alternating addition and multiplication levels, or, in other words, the transformed circuit belongs to $\mathsf{Arith}[D, 1]$.

Since $\left|\mu + 2\mathbf{r}^T \cdot \mathbf{e}\right| \leq 1 + 2nB$, it follows that

$$|\eta(c)| \leq 2nB + 1 \ . \tag{18}$$

- **Homomorphic addition gates.** When evaluating '+' on ciphertexts $c_1, \ldots, c_t$ to obtain c_{add}, we just sum their $(\mathbf{v}, w)$ values. Therefore

$$|\eta(c_{\mathsf{add}})| \leq \sum_i |\eta(c_i)| \ .$$

- **Homomorphic multiplication gates.** When evaluating '$\times$' on $c = ((\mathbf{v}, w), \ell)$, $c' = ((\mathbf{v}', w'), \ell)$ to obtain $c_{\mathsf{mult}} = ((\mathbf{v}_{\mathsf{mult}}, w_{\mathsf{mult}}), \ell + 1)$, we get that by Eq. (4)

$$w_{\mathsf{mult}} - \langle \mathbf{v}_{\mathsf{mult}}, \mathbf{s}_{\ell+1} \rangle = \eta(c) \cdot \eta(c') + 2 \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \ldots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \pmod{q} \ .$$

  It follows that

$$|\eta(c_{\mathsf{mult}})| \leq |\eta(c)| \cdot \left|\eta(c')\right| + 2 \cdot \frac{(n+1)(n+2)}{2} \cdot B(\log q + 1) \ .$$

  If we define

$$E \triangleq \max\left\{ |\eta(c)|, \left|\eta(c')\right|, (n+2)\sqrt{B \log(2q)} \right\} \ ,$$

  then $|\eta(c_{\mathsf{mult}})| \leq 2E^2$.

Let

$$E_0 = \max\left\{ 2nB + 1, (n+2)\sqrt{B \log(2q)} \right\} \leq 2nB \log q$$

be an upper bound on $|\eta(c)|$ of fresh ciphertexts.

Then it holds that a bound $E_{2\ell}$ on $|\eta(c)|$ of the outputs of layer 2ℓ (recall that the even layers contain multiplication gates) is obtained by

$$E_{2\ell} \leq 2(2E_{2(\ell-1)})^2 \ .$$

and therefore, recursively,

$$E_{2L} \leq (8E_0)^{2^L} \leq (16nB \log q)^{2^L} \ .$$

And after the final collation gate it holds that

$$|\eta(c_f)| \leq T \cdot (16nB \log q)^{2^L} \ . \qquad \square$$

We now similarly define $\hat{\eta}(\hat{c})$ for $\hat{c} = (\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ that encrypts μ. We let $\hat{e} \in \mathbb{Z}$ be the smallest integer (in absolute value) such that

$$\mu + 2\hat{e} = \hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle \pmod{p} \ ,$$

and define $\hat{\eta}(\hat{c}) \triangleq \mu + 2\hat{e}$ (note that, as before, $\hat{\eta}$ is defined over the integers, and not modulo p). So long as $|\hat{\eta}(\hat{c})| < p/2$, BTS.Dec will decrypt $\hat{c}$ correctly. In the next lemma, we bound $|\hat{\eta}(\hat{c})|$ of the output of BTS.Eval.

Lemma 4.4. *Let $n = n(\kappa) \geq 5$, $q = q(\kappa) \geq 3$, χ be B-bounded and $L = L(\kappa)$. Let $p = p(\kappa)$, $k = k(\kappa)$ and $\hat{\chi}$ be $\hat{B}$-bounded. Consider a homomorphic evaluation $\hat{c} \leftarrow \mathsf{BTS.Eval}_{evk}(f, c_1, \ldots, c_t)$ and the terms δ_1, δ_2 defined in Eq. (6) and (7), respectively. Let $c_f \in \mathbb{Z}_q^n \times \mathbb{Z}_q \times \{L\}$ be the intermediate value returned by the call to* SH.Eval. *Then*

$$|\delta_1 + \delta_2| \leq \frac{p}{2q} |\eta(c_f)| + 2n\hat{B} \log(2q) \ .$$

Proof. Recall that all samples from $\hat{\chi}$ are of magnitude at most $\hat{B}$. By definition (recall that δ_1, δ_2 have been defined over the rationals), we have that

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

and

$$\begin{aligned} |\delta_2| &= \left| (\frac{p}{q} - 1) \cdot \frac{\mu}{2} + \frac{p}{q} \cdot e \right| \\ &= \left| \frac{p}{q} \cdot \frac{\mu + 2e}{2} - \frac{\mu}{2} \right| \\ &\leq \frac{p}{2q} |\eta(c_f)| + 1/2 \ . \end{aligned}$$

Adding the terms together, the result follows. □

We can now finally prove Theorem 4.2.

Proof of Theorem 4.2. Let us consider the homomorphism claim about BTS (the argument for SH will follow as by-product): A sufficient condition for ciphertext $\hat{c} = (\hat{\mathbf{v}}, \hat{w})$ to decrypt correctly is that $\hat{e} < p/4$. By Lemma 4.4, it is sufficient to prove that

$$p/4 > \frac{p}{2q} |\eta(c_f)| + 2n\hat{B} \log(2q) \geq \frac{p}{2q} |\eta(c_f)| + p/8 \ .$$

Thus it is sufficient to prove that

$$|\eta(c_f)| < q/4 \ .$$

We note that if we prove this, then it also follows that c_f is decryptable and hence the claim about the homomorphism of SH holds as well.

Plugging in the bound from Lemma 4.3, we get

$$T \cdot (16nB \log q)^{2^L} < q/4 \ ,$$

and plugging in all the parameters and $T = \sqrt{q}$, we need

$$(16n^{2+\epsilon})^{2^L} < 2^{n^\epsilon/2}/4$$

which clearly holds for some $L = \Omega(\epsilon \log n)$. □

4.5 Bootstrapping and Full Homomorphism

We now show how to apply Gentry's bootstrapping theorem (Theorems 3.1, 3.2) to achieve full homomorphism. In order to do this, we first need to bound the complexity of an augmented decryption circuit. Since our decryption is essentially a computation of inner product, we bound the complexity of this operation.

Lemma 4.5. *Let $(\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$. There exists an arithmetic circuit with fan-in 2 gates and $O(\log k + \log\log p)$ depth, that on input $\hat{\mathbf{s}} \in \mathbb{Z}_p^k$ (in binary representation) computes*

$$(\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle \pmod p) \pmod 2 \ .$$

Proof. We let $\hat{\mathbf{s}}[i](j)$ denote the j^{th} bit of the binary representation of $\hat{\mathbf{s}}[i] \in \mathbb{Z}_p$. We notice that

$$\begin{aligned} \hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle &= \hat{w} - \sum_{i=1}^{k} \hat{\mathbf{s}}[i]\hat{\mathbf{v}}[i] \pmod p \\ &= \hat{w} - \sum_{i=1}^{k} \sum_{j=0}^{\lfloor \log p \rfloor} \hat{\mathbf{s}}[i](j) \cdot (2^j \cdot \hat{\mathbf{v}}[i]) \pmod p \ . \end{aligned}$$

Therefore computing $\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle \pmod p$ is equivalent to summing up $k(1 + \lfloor \log p \rfloor) + 1$ numbers in $\mathbb{Z}_p$, and then taking the result modulo p. The summation (over the integers) can be done in depth $O(\log k + \log\log p)$ using the standard "3 to 2" method (see e.g. [KR88]). In order to take modulo p, one needs to subtract, in parallel, all possible multiples of p (there are at most $O(k \log p)$ options) and check if the result is in $\mathbb{Z}_p$. This requires depth $O(\log k + \log\log p)$ again. Then a selection tree of depth $O(\log k + \log\log p)$ is used to choose the correct result. Once this is done, outputting the least significant bit implements the final modulo 2 operation.

The total depth is thus $O(\log k + \log\log p)$ as required. □

We can now apply the bootstrapping theorem to obtain a fully homomorphic scheme.

Lemma 4.6. *There exists $C \in \mathbb{N}$ such that setting $n = k^{C/\epsilon}$ and the rest of the parameters as in Theorem 4.2,* BTS *is bootstrappable as per Definition 3.7.*

Proof. Lemma 4.5 guarantees that the decryption circuit is in $\mathsf{Arith}[O(\log k), 1]$ (note that $\log\log p = o(\log k)$), since the augmented decryption circuit just adds 1 to the depth, it follows that the augmented decryption circuits are also in $\mathsf{Arith}[O(\log k), 1]$.

Theorem 4.2, on the other hand, guarantees homomorphism for any $\mathsf{Arith}[\Omega(\epsilon \log n), \sqrt{q}]$ function. Taking a large enough C, it will hold that $\mathsf{Arith}[O(\log k), 1] \subseteq \mathsf{Arith}[\Omega(\epsilon \log n), \sqrt{q}]$ and the lemma follows. □

Finally, we conclude that there exists an LWE based fully homomorphic encryption based on Theorem 4.1 and Lemma 4.6.

Corollary 4.7. *There exists a leveled fully homomorphic encryption based on the $\mathsf{DLWE}_{n,q,\chi}$ and $\mathsf{DLWE}_{k,p,\hat{\chi}}$ assumptions.*

Furthermore, if BTS *is weakly circular secure (see Definition 3.8), then there exists a fully homomorphic encryption based on the same assumptions.*

Efficiency of the Scheme. Interestingly, our scheme is comparable to *non-homomorphic* LWE based schemes (e.g. Regev's) in terms of encryption, decryption and ciphertext sizes. Namely, so long as one doesn't use the homomorphic properties of the scheme, she does not need to "pay" for it. To see why this is the case, we observe that our scheme's secret key has length $k \log p = O(\kappa \log \kappa)$ and the ciphertext length is $(k+1)\log p = O(\kappa \log \kappa)$. The decryption algorithm is essentially the same as Regev's. As far as encryption is concerned, it may seem more costly. The public key as we describe it contains $(n+1)((n+1)\log q + 2\kappa)\log q$ bits, and encryption requires performing operations over $\mathbb{Z}_q$. However, we note that one can think of sampling a public key $(\hat{\mathbf{A}}, \hat{\mathbf{b}})$ where $\hat{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_p^{m \times k}$, $\hat{\mathbf{b}} = \hat{\mathbf{A}}\hat{\mathbf{s}} + 2\hat{\mathbf{e}} \in \mathbb{Z}_p^m$ (where $m = ((k+1)\log p + 2\kappa)$). This will enable generating short ciphertexts that will be "bootstrapped up" during the homomorphic evaluation. If such short public key is used, then encryption also becomes comparable to Regev's scheme.

Homomorphic evaluation is where the high price is paid. the evaluation key has size $O(Ln^2 \log^2 q + n \log q \log p) = \widetilde{O}(n^{2+2\epsilon})$. Considering the fact that $n = \kappa^{C/\epsilon}$, this accumulates to a fairly long evaluation key, especially considering that in a leveled scheme, this size increases linearly with the depth of the circuit to be evaluated. The bright side, as we mention above, is that *evk* only needs to be known to the homomorphic evaluator and is not needed for encryption or decryption.

Circuit Privacy. A property that is sometimes desired in the context of fully homomorphic encryption is *circuit privacy*. A scheme is circuit private if the output of a homomorphic evaluation, reveals no information on the evaluated function (other than the output of the function on the encrypted message). Circuit privacy for our scheme can be achieved by adding additional noise to the ciphertext c_f, right before applying dimension-modulus reduction. Similar techniques were used in previous schemes [Gen09a, Gen09b, DGHV10] and thus we feel that a more elaborate discussion is unnecessary here.

5 LWE-Based Private Information Retrieval

In this section, we present a single-server private information retrieval (PIR) protocol with nearly optimal communication complexity.

A connection between homomorphic encryption and PIR already appears in the literature. We formalize it in Section 5.2 using our notation. We then, in Section 5.3, present a generic method for reducing the "outgoing" communication complexity of homomorphic encryption, using hybrid encryption. Finally, in Section 5.4, we instantiate all components using our homomorphic encryption scheme BTS, and analyze the resulting PIR protocol. (We present our notations and definitions in Section 5.1.)

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, it can be generated and sent (or posted) before the interaction begins, and may 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.

Letting κ be the security parameter and let $N \in \mathbb{N}$ be the database size (we allow an arbitrary relation between N and κ, although it is customary to assume that $N \leq 2^{\mathrm{poly}(\kappa)}$). An efficient PIR protocol PIR, 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 (note that the only efficiency requirement is that the algorithms run in polynomial time in their respective inputs):

0. **Setup.** The protocol begins in an off-line setup phase that does not depend on the index to be queried nor on the contents of the database (only on its size).

   The receiver runs the setup algorithm

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

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

  We note that in this work we present PIR protocols with *perfect correctness*. Namely, where $\mathsf{PIR.Decode}(1^\kappa, setupstate, qstate, resp) = \mathtt{DB}[i]$ always holds.

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

$$\max_{\substack{\mathbf{i}=(i_1,\ldots,i_t),\\ \mathbf{j}=(j_1,\ldots,j_t)\in[N]^t}} \left|\Pr[\mathcal{A}(params,\mathbf{i},query_{\mathbf{i}})=1]-\Pr[\mathcal{A}(params,\mathbf{j},query_{\mathbf{j}})=1]\right| \leq \epsilon \left(=\epsilon_{\kappa,N}\right),$$

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

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.[28] 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 (t,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.

Comparing the two notions heavily depends on the context and the relation between N and κ. While the first one is stronger for large values of N, the latter is stronger when N becomes smaller (sub-exponential in κ). If $N = \mathrm{poly}(\kappa)$, the latter implies sub-exponential hardness. Further discussion is beyond the scope of this work.

5.2 PIR from Homomorphic Encryption

In this section we describe the standard construction of PIR from homomorphic encryption. This construction is fairly straightforward and appears in prior literature (e.g. [Gen09a]), we bring it here for the sake of completeness.

Given a homomorphic encryption scheme $\mathsf{HE} = (\mathsf{HE.Keygen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ with message space $[N]$ and sufficient homomorphic capacity (as described below), a PIR scheme $\mathsf{PIR} = (\mathsf{PIR.Setup}, \mathsf{PIR.Query}, \mathsf{PIR.Response}, \mathsf{PIR.Decode})$ is defined as follows.

[28] 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 (t,ϵ)-private scheme according to the old definition is only $(t, t\epsilon)$-private according to ours.

- $\mathsf{PIR.Setup}(1^\kappa, N)$: In the setup procedure, we generate keys for the somewhat homomorphic scheme $(hpk, hevk, hsk) \leftarrow \mathsf{HE.Keygen}(1^\kappa)$. The parameter N is implicitly required in order to initialize the homomorphic scheme with the required homomorphic capabilities.

  The setup procedure then outputs the public parameters

  $$params := (hevk) \ ,$$

  and the secret state

  $$setupstate := (hpk, hsk) \ .$$

- $\mathsf{PIR.Query}(1^\kappa, setupstate, i)$: To generate a query string, we just encrypt i. Recall that $setupstate = (hpk, hsk)$, then

  $$query \leftarrow \mathsf{HE.Enc}_{hpk}(i) \ .$$

  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 $params = (hevk)$. The response is computed by homomorphically evaluating the function $\mathtt{DB}[\cdot]$ that takes an index i and outputs $\mathtt{DB}[i]$

  $$resp \leftarrow \mathsf{HE.Eval}_{hevk}(\mathtt{DB}[\cdot], query) \ .$$

  Note that $resp$ should correspond to a decryptable ciphertext of $\mathtt{DB}[i]$.

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

  $$b \leftarrow \mathsf{HE.Dec}_{hsk}(resp) \ .$$

The following lemmas (which are folklore) state the correctness and security of the resulting protocol.

Lemma 5.1. *Let* HE *be a* $\mathcal{C}$*-homomorphic encryption scheme such that* $\mathcal{C} \supseteq \{\mathtt{DB}[\cdot] : \mathtt{DB} \in \{0,1\}^N\}$*. Then the above PIR protocol is correct.*

Proof. By definition it holds that

$$\Pr[\mathsf{PIR.Decode}(1^\kappa, setupstate, qstate, resp) \neq \mathtt{DB}[i]] = \Pr[\mathsf{HE.Eval}(\mathtt{DB}[\cdot], \mathsf{HE.Enc}_{hpk}(i)) \neq \mathtt{DB}[i]] \ ,$$

and the latter is negligible by correct homomorphism. □

The next lemma establishes security. Note that we assume that the native message space of HE is $[N]$. If the message space is $\{0,1\}$ then an additional reduction is required (essentially to show that encrypting the index bit-by-bit will leave the scheme CPA secure).

Lemma 5.2. *Let* HE *be a* (t, ϵ)*-IND-CPA secure. Then the PIR protocol above is* $(t, t \cdot \epsilon)$*-private.*

Proof. This follows by a standard hybrid argument, replacing the adversary's queries by encryptions of 0 one by one, hence the t factor in the success probability. □

5.3 Hybrid Homomorphic Encryption

In this section we show how to use hybrid encryption in order to reduce the ciphertext length of homomorphic encryption schemes. The basic idea is fairly simple, given a message μ to be encrypted, rather than using the homomorphic encryption scheme, draw a key $symsk$ for a symmetric encryption scheme, and encrypt μ using $symsk$. Then, encrypt $symsk$ itself under the homomorphic encryption scheme. This way, instead of encrypting μ, which can be very long, using the homomorphic scheme, we encrypt it using a symmetric scheme with potentially much shorter ciphertext size, and all we need to encrypt using the homomorphic scheme is $symsk$ which is independent of the message length. This idea can be improved further: the encryption of $symsk$ can be done once and for all during the key generation, and put in the evaluation key of the new scheme. To perform homomorphic operations, the evaluation function will first use the encryption of $symsk$ to implicitly decrypt the incoming ciphertext, obtaining an encryption of μ, but now under the homomorphic scheme. Homomorphic evaluation can then proceed as usual. It is important to note that the output of the homomorphic operation is in the form of a ciphertext of the original homomorphic scheme, and not the symmetric scheme. Therefore, we only save on the "outgoing" communication complexity. A formal presentation and analysis follows.

Given a homomorphic encryption scheme $\mathsf{HE} = (\mathsf{HE.Keygen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$, and a symmetric encryption scheme $\mathsf{SYM} = (\mathsf{SYM.Keygen}, \mathsf{SYM.Enc}, \mathsf{SYM.Dec})$ with message space $\mathcal{M}$, we present a new homomorphic encryption scheme $\mathsf{HE}' = (\mathsf{HE}'.\mathsf{Keygen}, \mathsf{HE}'.\mathsf{Enc}, \mathsf{HE}'.\mathsf{Dec}, \mathsf{HE}'.\mathsf{Eval})$ with message space $\mathcal{M}$, as follows.

- $\mathsf{HE}'.\mathsf{Keygen}(1^\kappa)$: We start by generating a symmetric key $symsk \leftarrow \mathsf{SYM.Keygen}(1^\kappa)$, and keys for the somewhat homomorphic scheme $(hpk, hevk, hsk) \leftarrow \mathsf{HE.Keygen}(1^\kappa)$.

  The symmetric key is then encrypted using the homomorphic public key to create a ciphertext

  $$c_{symsk} \leftarrow \mathsf{HE.Enc}_{hpk}(symsk) \ .$$

  We note that if HE is bit encryption scheme, then $symsk$ is encrypted bit by bit.

  The keys of the new scheme are $hpk' := hpk$, $hevk' := (hevk, c_{symsk})$, $hsk' := hsk$.

- $\mathsf{HE}'.\mathsf{Enc}_{hpk'}(\mu)$: To encrypt a message $m \in \mathcal{M}_k$, we just encrypt it using the symmetric scheme.

  $$c \leftarrow \mathsf{SYM.Enc}_{symsk}(\mu) \ .$$

- $\mathsf{HE}'.\mathsf{Eval}_{hevk'}(f, c_1, \ldots, c_t)$: The scheme will only support homomorphic evaluation of $c_1, \ldots, c_t$ that are freshly encrypted ciphertexts (this is only for the sake of simplicity, this obstacle can be overcome fairly simply). We recall that $hevk' = (c_{symsk}, hevk)$.

  Consider the function

  $$\mathsf{Compose}_{f,c_1,\ldots,c_t}(x) \triangleq f(\mathsf{SYM.Dec}_x(c_1), \ldots, \mathsf{SYM.Dec}_x(c_t)) \ ,$$

  and return

  $$\mathsf{HE.Eval}_{hevk}(\mathsf{Compose}_{f,c_1,\ldots,c_t}, c_{symsk}) \ .$$

  Note that in order for the output to be meaningful, the function $\mathsf{Compose}_{f,c_1,\ldots,c_t}(\cdot)$ must be homomorphically evaluable by the original scheme HE. Further note that while the inputs to this operation are SYM ciphertexts, the output is an HE ciphertext.

- $\mathsf{HE}'.\mathsf{Dec}_{hsk'}(c)$: To decrypt, we recall that $hsk' = hsk$. We use the standard homomorphic decryption to return

$$\mathsf{HE}.\mathsf{Dec}_{hsk}(c) \ .$$

  Note that freshly encrypted ciphertexts cannot be correctly decrypted using this procedure. Rather, this only applies to ciphertexts that are the output of a homomorphic evaluation. (Typically this will not be a problem since one can always evaluate the identity function and then decrypt.)

The following lemmas state the homomorphism and security properties of HE'.

Lemma 5.3. *Let* $\mathsf{SYM}, \mathsf{HE}, \mathsf{HE}'$ *be as above such that* HE *is* $\mathcal{C}$*-homomorphic and* $\mathcal{T}$ *is* SYM*'s ciphertext space. Then* HE' *is* $\mathcal{C}'$*-homomorphic for*

$$\mathcal{C}' \triangleq \{f : \forall c_1, \ldots, c_t \in \mathcal{T}.\ \mathsf{Compose}_{f,c_1,\ldots,c_t} \in \mathcal{C}\} \ .$$

Proof. This follows immediately by definition. Letting $(hpk', hevk', hsk') \leftarrow \mathsf{HE}'.\mathsf{Keygen}(1^\kappa)$, $f \in \mathcal{C}'$, $c_i \leftarrow \mathsf{HE}'.\mathsf{Enc}_{hpk'}(\mu_i)$, it holds that

$$\mathsf{HE}'.\mathsf{Eval}_{hevk'}(f, c_1, \ldots, c_t) = \mathsf{HE}.\mathsf{Eval}_{hevk}(\mathsf{Compose}_{f,c_1,\ldots,c_t}, c_{symsk}) \ ,$$

and therefore by the union bound

$$\begin{aligned}\Pr[\mathsf{HE}'.\mathsf{Eval}_{hevk'}(f, c_1, \ldots, c_t) \neq f(\mu_1, \ldots, \mu_t)] \leq \\ \Pr[\mathsf{HE}.\mathsf{Eval}_{hevk}(\mathsf{Compose}_{f,c_1,\ldots,c_t}, c_{symsk}) \neq \mathsf{Compose}_{f,c_1,\ldots,c_t}(symsk)] \\ + \Pr[\mathsf{Compose}_{f,c_1,\ldots,c_t}(symsk) \neq f(\mu_1, \ldots, \mu_t)] \ .\end{aligned}$$

The first term is negligible by the $\mathcal{C}$-homomorphism of HE, whereas the second is negligible by the correctness of SYM. □

Lemma 5.4. *Let* $\mathsf{SYM}, \mathsf{HE}, \mathsf{HE}'$ *be as above such that* $\mathsf{HE}, \mathsf{SYM}$ *are* (t, ϵ)*-IND-CPA secure. Then* HE' *is* $(t, \mathrm{poly}(\kappa) \cdot \epsilon)$*-IND-CPA secure, for some polynomial* $\mathrm{poly}(\cdot)$*.*

Proof. The proof is by a standard hybrid argument. We consider a CPA adversary for HE' and track its distinguishing advantage while changing the distributions of keys.

- **Hybrid** H_0**.** In this hybrid the distributions of all elements in HE' are as prescribed, and the adversary's advantage is as it is against HE'.

- **Hybrid** H_1**.** We change the key generation process and set $c_{symsk} \leftarrow \mathsf{HE}.\mathsf{Enc}_{hpk}(0)$ instead of $\mathsf{HE}.\mathsf{Enc}_{hpk}(symsk)$. A hybrid argument over the bits of $symsk$, using the CPA security of HE, shows that the advantage of any adversary here cannot deviate by more than $\mathrm{poly}(\kappa) \cdot \epsilon$ from the advantage in H_0 (where the polynomial is the length of $symsk$).

  We further note that in H_1, the CPA advantage of any adversary can be translated to the same advantage against SYM. It follows that this advantage is at most ϵ. The result thus follows. □

5.4 Instantiating the Components: Our PIR Protocol

Combining Sections 5.2 and 5.3, an efficient PIR protocol will follow from a combination of an adequate symmetric encryption scheme and a homomorphic encryption sheme. We now show how to instantiate these primitives in two different ways. In the first, a specific LWE-based encryption scheme is used as the symmetric encryption component. This will allow to use the scheme BTS as the homomorphic component. The second is more general and uses pseudorandom functions to instantiate the symmetric encryption. Since we do not have an explicit bound on the decryption depth of such circuit (in particular, it can be $\omega(\log k)$), the scheme BTS may not be able to handle its decryption circuit. However, one can use a *bootstrapped* version of BTS to evaluate any circuit. Hence, the latter solution requires bootstrapping and is therefore not as clean, but on the other hand it provides the best asymptotic communication complexity.

We emphasize that the edge of our protocols comes from two factors: First, the hybrid method, presented in Section 5.3, allows to reduce the query communication complexity. Secondly, our scheme is the first where the ciphertext length is quasi-linear in the security parameter (i.e. ciphertext length $\kappa \cdot \mathrm{polylog}(\kappa)$ to achieve security 2^κ), which allows low communication complexity for the response.

An Explicit LWE-Based Solution. The first idea is to use an optimized, symmetric-key LWE-based encryption as the symmetric encryption scheme in the PIR protocol, together with our scheme BTS as the homomorphic scheme. Specifically, using the same parameters k, p as in our bootstrappable scheme, we get a symmetric 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 $1/\mathrm{poly}(k)$ fraction of p). Secondly, observe that in a symmetric ciphertext $(\mathbf{v}, w) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$, 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 $k = \mathrm{poly}(\kappa)$. The following corollary summarizes the properties of this scheme.

Corollary 5.5 ([PVW08, ACPS09])**.** *Let $p, k, \hat{\chi}$ be as in Theorem 4.2. Then there exists a $\mathsf{DLWE}_{k,p,\hat{\chi}}$-secure symmetric encryption scheme whose ciphertext length is $O(k \log k + \ell)$ for ℓ-bit messages, and whose decryption circuit has the same depth as that of $\mathsf{BTS.Dec}$.*

We now prove that indeed the function $\mathsf{Compose}_{\mathsf{DB}[\cdot],c}$, where c is an encryption of an index, is indeed within the homomorphic capacity of BTS. Recall the definition of the class Arith from Section 4 and the definition of $\mathsf{Compose}$ from Section 5.3.

Lemma 5.6. *Let SYM be the scheme from Corollary 5.5, then for all $i \in [N]$, $c \leftarrow \mathsf{SYM.Enc}_{symsk}(i)$, it holds that $\mathsf{Compose}_{\mathsf{DB}[\cdot],c} \in \mathsf{Arith}[O(\log k) + \log\log N, N]$.*

Proof. We present a $\mathsf{Arith}[O(\log k) + \log\log N, N]$ circuit that implements $\mathsf{Compose}_{\mathsf{DB}[\cdot],c}(x)$. First, we decrypt the value of c to obtain an index i. Then, we compute the function $\sum_{j \in [N]} \mathsf{DB}[j] \cdot \mathbb{1}_{i=j}$ (where $\mathbb{1}_E$ is an indicator variable for the event E). The decryption circuit is implemented in

depth $O(\log k)$ as in Lemma 4.5. The function $\mathbb{1}_{i=j}$ is implemented using a comparison tree of depth $\log \log N$. Finally, a collation gate of fan-in N is used to compute the final sum. The result follows. □

This means that we can choose n in BTS to be large enough so that $\mathsf{Compose}_{\mathsf{DB}[\cdot],c}$ is correctly evaluated.

Theorem 5.7. *There exists a PIR protocol with communication complexity $O(k \log k + \log N)$ based on the $\mathsf{DLWE}_{n,q,\chi}$ and $\mathsf{DLWE}_{k,p,\hat{\chi}}$ assumptions, for $n = \mathrm{poly}(k)$ and the remainder of the parameters as in Theorem 4.2.*

Proof. We choose n such that $L = \Omega(\epsilon \log n) > O(\log k) + \log \log N$ and such that $\sqrt{q} = 2^{n^\epsilon/2} \geq N$. This will result in $n = \mathrm{poly}(k, \log N)$ (recall that the communication complexity depends only on k). The result follows from Theorem 4.2 and Theorem 4.3. □

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

An Almost Optimal Solution Using Pseudorandom Functions. A second instantiation aims to bring the (*ii*)-private communication complexity down to $\log N + \kappa \cdot \mathrm{polylog}(\kappa)$. This can be done by instantiating the symmetric encryption scheme above with an optimal symmetric encryption scheme with ciphertexts of length $\log N + \kappa \cdot \mathrm{polylog}(\kappa)$. Such a scheme follows immediately given any pseudo-random function (PRF).

If we want to base security solely on LWE, we can use the LWE-based PRF that is obtained by applying the GGM transformation [GGM86] to an LWE based pseudorandom generator. Note that using such instantiation, we cannot argue that $\mathsf{Compose}_{\mathsf{DB}[\cdot],c} \in \mathsf{Arith}[L, T]$ for reasonable L, T (since the complexity of evaluating the PRF might be high). However, we can use our leveled fully homomorphic scheme to support the required circuit depth of any function, and in particular the aforementioned PRF.

The Complexity of Transmitting the Public Parameters. Finally, we note that the parameters produced in the setup phase of our protocol are of length $\mathrm{poly}(\kappa, \log N)$. Thus our protocol can be trivially modified to work in a setting without setup, with communication complexity $\mathrm{poly}(\kappa, \log N)$ (under the (*ii*)-private notion) and $\mathrm{polylog}(N)$ (under the (*i*)-private notion).

Acknowledgments. We thank Shafi Goldwasser for her seemingly unlimited enthusiasm for and technical advice on this work, as well as numerous comments that considerably improved the presentation. We also thank Microsoft Research for hosting the first author during the course of this work.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[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.

[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.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science (ITCS)*, 2012.

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

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, 2011. To appear.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, pages 97–106, 2011.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[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 *EUROCRYPT*, pages 24–43, 2010. 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 *STOC*, pages 169–178, 2009.

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

[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. In *FOCS*, pages 107–109, 2011.

[GH11b] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, pages 129–148, 2011.

[GHS11a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:680, 2011.

[GHS11b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. *IACR Cryptology ePrint Archive*, 2011:566, 2011.

[GHV10a] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable Yao circuits. In *CRYPTO*, pages 155–172, 2010.

[GHV10b] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, pages 506–522, 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.

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

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[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.

[KR88] Richard M. Karp and Vijaya Ramachandran. A survey of parallel algorithms for shared-memory machines. Technical Report UCB/CSD-88-408, EECS Department, University of California, Berkeley, Mar 1988.

[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 *EUROCRYPT*, pages 1–23, 2010. Draft of full version was provided by the authors.

[LTV12] Adriana Lopéz-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation with a cloud via multi-key fully homomorphic encryption. To Appear in STOC 2012, 2012.

[MGH10] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with d-operand multiplications. In *CRYPTO*, pages 138–154, 2010.

[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.

[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 *STOC*, pages 333–342, 2009.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

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

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

[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.