Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based

Craig Gentry* Amit Sahai† Brent Waters‡

June 8, 2013

Abstract

We describe a comparatively simple fully homomorphic encryption (FHE) scheme based on the learning with errors (LWE) problem. In previous LWE-based FHE schemes, multiplication is a complicated and expensive step involving "relinearization". In this work, we propose a new technique for building FHE schemes that we call the *approximate eigenvector* method. In our scheme, for the most part, homomorphic addition and multiplication are just matrix addition and multiplication. This makes our scheme both asymptotically faster and (we believe) easier to understand.

In previous schemes, the homomorphic evaluator needs to obtain the user's "evaluation key", which consists of a chain of encrypted secret keys. Our scheme has no evaluation key. The evaluator can do homomorphic operations without knowing the user's public key at all, except for some basic parameters. This fact helps us construct the first identity-based FHE scheme. Using similar techniques, we show how to compile a recent attribute-based encryption scheme for circuits by Gorbunov et al. into an attribute-based FHE scheme that permits data encrypted under the same index to be processed homomorphically.

1 Introduction

Fully homomorphic encryption (FHE) schemes [RAD78, Gen09, Gen10, vDGHV10] [SV10, GH11b, CMNT11, BV11a, BV11b, GH11a, BGV12, CNT12, GHS12a, GHS12b] [LATV12, Bra12] "have been simplified enough so that their description can fit, well, in a blog post" [BB12b, BB12a]. In this paper, we try to make FHE even simpler.

*IBM Research. cbgentry@us.ibm.com. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20202. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

†UCLA. sahai@cs.ucla.edu. Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

‡UT Austin. bwaters@cs.utexas.edu. Supported by NSF CNS-0915361 and CNS-0952692, CNS-1228599 DARPA via Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

1.1 Previous FHE Schemes Based on Learning with Errors

Currently, perhaps the simplest leveled[1] FHE scheme based on the learning with errors (LWE) assumption [Reg05] is by Brakerski [Bra12]. In fact, Barak and Brakerski do give a remarkably clear exposition of this scheme in a blog post [BB12a]. However, while the scheme's key generation, encryption, decryption, and homomorphic addition procedures are easy to describe, they note that "multiplication is more tricky".

In Brakerski's scheme, similar to previous FHE schemes based on LWE [BV11b, BGV12], the ciphertext $\vec{c}$ and secret key $\vec{s}$ are n-dimensional vectors whose dot product $\langle \vec{c}, \vec{s} \rangle \approx \mu$ equals the message μ, up to some small "error" that is removed by rounding. Homomorphic multiplication uses an identity regarding dot products of tensor products of vectors: namely, $\langle \vec{u}_1 \otimes \vec{u}_2, \vec{v}_1 \otimes \vec{v}_2 \rangle = \langle \vec{u}_1, \vec{v}_1 \rangle \cdot \langle \vec{u}_2, \vec{v}_2 \rangle$. Thus, if ciphertexts $\vec{c}_1$ and $\vec{c}_2$ satisfy $\langle \vec{c}_1, \vec{s} \rangle \approx \mu_1$ and $\langle \vec{c}_2, \vec{s} \rangle \approx \mu_2$, then $\langle \vec{c}_1 \otimes \vec{c}_2, \vec{s} \otimes \vec{s} \rangle \approx \mu_1 \cdot \mu_2$, where $\vec{c}_1 \otimes \vec{c}_2$ is interpreted as the new ciphertext and $\vec{s} \otimes \vec{s}$ as the new secret key, each having dimension $\Theta(n^2)$. Since multiplying-by-tensoring blows up the ciphertext size, it can only be used for a constant number of steps. For efficiency, the evaluator must *relinearize* [BV11b] the ciphertext after tensoring. Relinearization is a procedure that takes the *long* ciphertext that encrypts $\mu_1 \cdot \mu_2$ under the long key $\vec{s} \otimes \vec{s}$, and compresses it into a *normal-sized* n-dimensional ciphertext that encrypts $\mu_1 \cdot \mu_2$ under a normal-sized n-dimensional key $\vec{s}'$. To relinearize, the evaluator multiplies the long ciphertext vector by a special $n \times \Theta(n^2)$ relinearization matrix. This relinearization matrix is part of the "evaluation key" that the evaluator must obtain from the public key to perform homomorphic evaluation.

The relinearization step [BV11b] is ingenious and is perhaps the main insight that led to FHE based on LWE. However, relinearization is not particularly natural, nor is it easy to give an intuitive description of how and why it works. Moreover, relinearization is expensive. Each relinearization matrix has size $\Omega(n^3)$, and the public key must contain L of them to evaluate circuits of maximum multiplicative depth L. Computationally, relinearization requires $\Omega(n^3)$ operations, where each operation has cost polynomial in L.

This situation raises the question: Can we construct a LWE-based FHE scheme with a *natural* multiplication procedure? For ciphertexts c_1 and c_2, can we construct a scheme where homomorphic addition and multiplication are just $c_1 + c_2$ and $c_1 \cdot c_2$, where '+' and '$\cdot$' are natural algebraic operations over some ring, and where the new ciphertexts have the "same form" as the old ones; for example, $c_1 \cdot c_2$ is not a "long" ciphertext? Can we eliminate the need for an "evaluation key" in general, and the relinearization matrices in particular? If so, LWE-based FHE might become easier to explain. If we can simplify LWE-based FHE while also improving its efficiency and supporting new applications, then even better.

1.2 Our Results

Our main results are:

- **Conceptually simpler FHE based on LWE:** We fully describe our scheme here in the Introduction, and think our new approach will prove valuable pedagogically and theoretically.

[1] "Leveled" FHE is a relaxation of "pure" FHE [Gen09]. For fixed parameters, a pure FHE scheme can evaluate arbitrary circuits. In a leveled FHE scheme, the parameters of the scheme may depend on the *depth*, but not the *size*, of the circuits that the scheme can evaluate. We focus on leveled FHE schemes, and typically omit the term "leveled". One can transform our leveled FHE schemes to pure ones by using Gentry's bootstrapping theorem and assuming "circular security" [Gen09].

- **Asymptotically faster FHE based on LWE:** We eliminate relinearization and the large relinearization matrices, with their $\Omega(n^3)$ complexity. Instead, ciphertexts are matrices that are added and multiplied naturally. In principle, matrix multiplication uses sub-cubic computation: e.g., Strassen and Williams achieved $n^{2.807}$ and $n^{2.3727}$ respectively [Str69, Wil12].

- **Identity-based FHE:** We solve an open problem mentioned in previous works [Nac10, GHV10, Bra12, CHT13] – namely, to construct an identity-based FHE scheme, in which there are no user-specific keys that must be obtained by the encrypter or evaluator. Informally speaking, in an identity-based FHE scheme, a user that has only the public parameters should be able to perform *both* encryption and homomorphism operations. The homomorphism operations should allow a user to take two ciphertexts encrypted to the same target identity, and homomorphically combine them to produce another ciphertext under the same target identity. Previously, only "weak" identity-based FHE schemes were known, where the evaluator needs a user-specific evaluation key, and thus the homomorphism is *not* exploitable by a user that only has the public parameters. Our scheme solves the problem by eliminating evaluation keys entirely.

  We obtain our identity-based FHE scheme by presenting a "compiler" that transforms any LWE-based IBE scheme in the literature that satisfies certain properties, into a fully homomorphic identity-based encryption scheme. Several LWE-based IBE schemes in the literature satisfy the properties needed for our compiler [GPV08, ABB10a, ABB10b, CHKP10].

- **Attribute-based FHE:** Recently Gorbunov et al. [GVW13] constructed an attribute-based encryption (ABE) for circuits based on LWE. Our compiler for LWE-based IBE also works for their ABE scheme, with relatively minor modifications. We obtain an ABE scheme in which messages encrypted under the same index can be processed homomorphically without any evaluation key in a polynomial depth circuit, and still be decrypted by any party that was entitled to decrypt the original ciphertexts.[2]

Our FHE scheme retains advantages of other LWE-based FHE schemes, such as making bootstrapping optional [BGV12], (with bootstrapping) basing security on LWE for quasi-polynomial factors versus sub-exponential factors [BGV12], eliminating "modulus switching" [Bra12], and basing security directly on the hardness of classical GapSVP [Bra12].

We do not want to oversell our asymptotic result; we now provide some additional context: In general, FHE schemes based on LWE have much worse performance (certainly asymptotically) than schemes based on ring LWE (RLWE) [LPR10, BV11a, GHS12a], and even RLWE-based schemes cannot yet be considered practical [GHS12b]. Moreover, sub-cubic matrix multiplication algorithms may not beat cubic ones by much in practice. Rather, we view our asymptotic result mainly as evidence of how fundamentally new our techniques are. We note that it is straightforward to construct an RLWE-based version of our scheme, but its performance is worse than the best known RLWE-based schemes [BGV12, Bra12, GHS12a, GHS12b] by log factors. On the other hand, while our techniques may not reduce evaluation complexity as much as we would like, they reduce the space complexity significantly (from quasi-cubic to quasi-quadratic), which is a significant issue for LWE-based FHE schemes in practice.

[2]Independently, Garg et al. [GGH+13b] also recently constructed an ABE scheme for circuits using multilinear maps [GGH13a, CLT13], but our techniques do not work as effectively with their scheme.

As with all current FHE schemes without bootstrapping, the parameters and per-gate complexity of evaluation depend on the multiplicative depth L of the circuit. "Bootstrapping" [Gen09], together with an assumption of circular security, remains the only known way of making these performance metrics independent of L, and while the overhead of bootstrapping is high, it becomes an attractive option once L passes some threshold. However, our scheme loses some of its advantages once bootstrapping is used. First, to apply bootstrapping, the evaluator needs to obtain the user's secret key encrypted under its public key – in effect, an evaluation key – and therefore we no longer achieve identity-based/attribute-based FHE in this context. Second, this encrypted secret key has quasi-cubic size in our scheme, and while this can be mitigated by public key compression techniques [CNT12], it eliminates the space complexity advantages of our scheme. Essentially, bootstrapping returns us to the realm of "unnatural" operations, with all of its disadvantages. It remains a fascinating open problem to find some "natural" alternative to bootstrapping, and (relatedly) to achieve "pure" FHE without an assumption of circular security.

1.3 An Overview of Our FHE Scheme

Our main insight is that we can achieve LWE-based homomorphic encryption where homomorphic addition and multiplication correspond directly to matrix addition and multiplication.

1.3.1 Homomorphic Operations

Let us skip key generation and encryption for the moment, and jump directly to the homomorphic operations (and decryption).

In our scheme, for some modulus q and dimension parameter N to be specified later, a ciphertext C is a $N \times N$ matrix over $\mathbb{Z}_q$, with "small" entries (much smaller than q) and the secret key $\vec{v}$ is a N-dimensional vector over $\mathbb{Z}_q$ with at least one "big" coefficient v_i. We restrict the message μ to be a "small" integer. We say C encrypts μ when $C \cdot \vec{v} = \mu \cdot \vec{v} + \vec{e}$, where $\vec{e}$ is a "small" error vector. To decrypt, we extract the i-th row C_i from C, compute $x \leftarrow \langle C_i, \vec{v} \rangle = \mu \cdot v_i + e_i$, and output $\mu = \lfloor x/v_i \rceil$. In a nutshell, the essence of our scheme is that the secret key $\vec{v}$ is an *approximate eigenvector* of the ciphertext matrix C, and the message μ is the *eigenvalue*.

Now, let us see why matrix addition and multiplication are correct homomorphic operations. Suppose C_1 and C_2 encrypt μ_1 and μ_2 in that $C_i \cdot \vec{v} = \mu_i \cdot \vec{v} + \vec{e}_i$ for small $\vec{e}_i$. Let $C^+ = C_1 + C_2$ and $C^\times = C_1 \cdot C_2$. For addition, we have $C^+ \cdot \vec{v} = (\mu_1 + \mu_2) \cdot \vec{v} + (\vec{e}_1 + \vec{e}_2)$, where the error likely has grown a little, as usual in FHE schemes. But assuming the error is still "small", the sum of the ciphertext matrices encrypts the sum of the messages. For multiplication, we have

$$\begin{aligned} C^\times \cdot \vec{v} &= C_1 \cdot (\mu_2 \cdot \vec{v} + \vec{e}_2) = \mu_2 \cdot (\mu_1 \cdot \vec{v} + \vec{e}_1) + C_1 \cdot \vec{e}_2 = \mu_1 \cdot \mu_2 \cdot \vec{v} + \mu_2 \cdot \vec{e}_1 + C_1 \cdot \vec{e}_2 \\ &= \mu_1 \cdot \mu_2 \cdot \vec{v} + \mathit{small} \end{aligned}$$

where the final error vector is hopefully "small", since μ_2, C_1, $\vec{e}_1$, and $\vec{e}_2$ are all small. If so, the product of the ciphertext matrices encrypts the product of the messages. Interestingly, $C_2 \cdot C_1$ is also an encryption of $\mu_1 \cdot \mu_2$, even though matrix multiplication is not commutative.

To simplify further, it might be helpful to imagine an *error-free* version of the scheme, where $C_i \cdot \vec{v} = \mu_i \cdot \vec{v}$ *exactly*. In this case, the key $\vec{v}$ is an (exact) eigenvector of ciphertext matrices, and the message μ_i is the eigenvalue. In general, if matrices C_1 and C_2 have a common eigenvector $\vec{v}$ with eigenvalues μ_1 and μ_2, then $C_1 \cdot C_2$ and $C_2 \cdot C_1$ have eigenvector $\vec{v}$ with eigenvalue $\mu_1 \cdot \mu_2$.

Of course, in our scheme, the secret key $\vec{v}$ is only an *approximate* eigenvector, not an *exact* one. Introducing error is necessary to base the security of our scheme on LWE. The cost of making $\vec{v}$ only an *approximate* eigenvector is that certain terms in our scheme must be "small" to ensure that homomorphic operations do not disrupt the essential form of the ciphertexts. We call our new approach to LWE-based (homomorphic) encryption the *approximate eigenvector* method.

1.3.2 Bounding the Error and Somewhat Homomorphic Encryption

Although we have not fully specified the scheme, let us go ahead and estimate how homomorphic it is. The scheme above works correctly until the coefficients of the error vector begin to approach q in magnitude. How many homomorphic operations can we perform before that happens?

Suppose C_1 and C_2 are B-bounded ciphertexts, in the sense that μ_i and the coefficients of C_i and $\vec{e}_i$ all have magnitude at most some bound B. Then, C^+ is $2B$-bounded, and $C^\times$ is $(N+1)B^2$-bounded. In short, the error level grows worse than B^{2^L}, *doubly exponentially with the multiplicative depth* L of the circuit being evaluated. Alternatively, if one wants to consider the *degree* (rather than *depth*) of functions that can be evaluated, if we evaluate a multivariate polynomial $P(x_1, \ldots, x_t)$ of total degree d, on B-bounded ciphertexts as input, the final ciphertext is $|P|(N+1)^{d-1}B^d$-bounded, where $|P|$ is the ℓ_1-norm of P's coefficient vector. Taking q to comfortably exceed this bound, we (roughly) can evaluate polynomials of degree $\log_{NB} q$. Since q/B must be subexponential (at most) in N for security reasons, our scheme-so-far can only evaluate polynomials of (sublinear) polynomial degree in N (only logarithmic depth). In short, our scheme-so-far is a *somewhat homomorphic encryption* (SWHE) scheme [Gen09] that can evaluate log-depth or polynomial degree. Though not yet fully homomorphic, it is by far the most homomorphic LWE-based encryption scheme that uses only "natural" homomorphic operations.

1.3.3 Flattening Ciphertexts and Fully Homomorphic Encryption

To obtain a leveled FHE scheme that can evaluate circuits of polynomial *depth* without bootstrapping or techniques like relinearization, we need to ensure better bounds on the growth of the error. Let us say that a ciphertext C is B-strongly-bounded if its associated μ and the coefficients of C all have magnitude *at most 1*, while the coefficients of its $\vec{e}$ all have magnitude at most B. If we evaluate a NAND gate on B-strongly-bounded ciphertexts C_1, C_2 to obtain a new ciphertext $C_3 \leftarrow I_N - C_1 \cdot C_2$ (where I_N is the N-dimensional identity matrix), then the message remains in $\{0, 1\}$, and the coefficients of C_3's error vector have magnitude at most $(N+1)B$. If we could somehow additionally ensure that C_3's coefficients have magnitude at most 1 so that strong-boundedness is preserved, then we could evaluate a circuit of *depth* L while keeping the error magnitude at most $(N+1)^L B$. Setting q/B to be subexponential in N, we could evaluate a circuit of polynomial *depth* rather than merely polynomial *degree*. In short, we would have a leveled FHE scheme.

Here we describe a operation called ciphertext *flattening* that keeps ciphertexts strongly bounded, so that we obtain leveled FHE.

Flattening uses some simple transformations from [BV11b, BGV12, Bra12] that modify vectors without affecting dot products. Let $\vec{a}$, $\vec{b}$ be vectors of some dimension k over $\mathbb{Z}_q$. Let $\ell = \lfloor \log_2 q \rfloor + 1$ and $N = k \cdot \ell$. Let $\mathsf{BitDecomp}(\vec{a})$ be the N-dimensional vector $(a_{1,0}, \ldots, a_{1,\ell-1}, \ldots, a_{k,0}, \ldots, a_{k,\ell-1})$, where $a_{i,j}$ is the j-th bit in a_i's binary representation, bits ordered least significant to most significant. For $\vec{a}' = (a_{1,0}, \ldots, a_{1,\ell-1}, \ldots, a_{k,0}, \ldots, a_{k,\ell-1})$, let $\mathsf{BitDecomp}^{-1}(\vec{a}') = (\sum 2^j \cdot a_{1,j}, \ldots, \sum 2^j \cdot a_{k,j})$ be the inverse of $\mathsf{BitDecomp}$, but well-defined even when the input is not a 0/1 vector. For

N-dimensional $\vec{a}'$, let $\mathsf{Flatten}(\vec{a}') = \mathsf{BitDecomp}(\mathsf{BitDecomp}^{-1}(\vec{a}'))$, a N-dimensional vector with 0/1 coefficients. When A is a matrix, let $\mathsf{BitDecomp}(A)$, $\mathsf{BitDecomp}^{-1}$, or $\mathsf{Flatten}(A)$ be the matrix formed by applying the operation to each row of A separately. Finally, let $\mathsf{Powersof2}(\vec{b}) = (b_1, 2b_1, \ldots, 2^{\ell-1}b_1, \ldots, b_k, 2b_k, \ldots, 2^{\ell-1}b_k)$, a N-dimensional vector. Here are some obvious facts:

- $\langle \mathsf{BitDecomp}(\vec{a}), \mathsf{Powersof2}(\vec{b}) \rangle = \langle \vec{a}, \vec{b} \rangle$.
- For any N-dimensional $\vec{a}'$, $\langle \vec{a}', \mathsf{Powersof2}(\vec{b}) \rangle = \langle \mathsf{BitDecomp}^{-1}(\vec{a}'), \vec{b} \rangle = \langle \mathsf{Flatten}(\vec{a}'), \mathsf{Powersof2}(\vec{b}) \rangle$.

An interesting feature of $\mathsf{Flatten}$ is that it makes the coefficients of a vector or matrix *small*, without affecting its product with $\mathsf{Powersof2}(\vec{b})$, and without knowing $\vec{b}$.

To facilitate ciphertext flattening, we give a special form to our secret key $\vec{v}$. Specifically, we set $\vec{v} = \mathsf{Powersof2}(\vec{s})$ for some secret vector $\vec{s}$ (to be specified later). This form is consistent with our earlier requirement that $\vec{v}$ have some big coefficient v_i for decryption; indeed, since $\vec{v}$'s coefficients go up by $\lfloor \log_2 q \rfloor$ powers of 2, it *must* have a big coefficient suitable to recover $\mu \in \{0, 1\}$.

Now, for any $N \times N$ matrix C, we have $\mathsf{Flatten}(C) \cdot \vec{v} = C \cdot \vec{v}$. So, after we compute an initial ciphertext $C_3 \leftarrow I_N - C_1 \cdot C_2$ for the NAND gate, we set $C^{\mathsf{NAND}} = \mathsf{Flatten}(C_3)$ to obtain a ciphertext that has 0/1 coefficients and is strongly bounded. Thus, we obtain leveled FHE without relinearization, under a fixed approximate eigenvector secret key.

1.3.4 Key Generation, Encryption, and Reduction to LWE

Let us finally circle back to key generation and encryption. We want to base security on LWE. So, for key generation, we generate an LWE instance. For suitable parameters $q, n, m = O(n \log q)$, an LWE instance over $\mathbb{Z}_q$ consists of a $m \times (n+1)$ matrix A such that there exists a $(n+1)$-dimensional vector $\vec{s}$ whose first coefficient is 1 where $\vec{e} = A \cdot \vec{s}$ is a "small" error vector. (See Section 2 for a formal definition of LWE.) In our scheme, A is public and $\vec{s}$ is secret. We set our approximate eigenvector to be $\vec{v} = \mathsf{Powersof2}(\vec{s})$, a vector of dimension $N = (n+1) \cdot \ell$ for $\ell = \lfloor \log_2 q \rfloor + 1$.

To encrypt $\mu \in \mathbb{Z}_q$, the encrypter generates a random $N \times m$ matrix R with 0/1 entries, and sets $C = \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(R \cdot A))$, where I_N is the N-dimensional identity matrix. Since $\mathsf{Flatten}$ does not affect the product with $\vec{v}$, we have:

$$C \cdot \vec{v} = \mu \cdot \vec{v} + \mathsf{BitDecomp}(R \cdot A) \cdot \vec{v} = \mu \cdot \vec{v} + R \cdot A \cdot \vec{s} = \mu \cdot \vec{v} + small$$

$\mathsf{Flatten}$ ensures that the coefficients of C are small, and therefore that C has the proper form of a ciphertext that permits our homomorphic operations. Decryption works as mentioned previously.

To show that security is based on LWE, it is now enough to show that C is statistically independent of μ when A is a uniformly random $m \times (n+1)$ matrix over $\mathbb{Z}_q$. Let $C' = \mathsf{BitDecomp}^{-1}(C)$. Recall that C is $\mathsf{Flatten}$'d, and so $C = \mathsf{Flatten}(C) = \mathsf{BitDecomp}(C')$. Therefore, C reveals nothing more than C'. But $C' = \mathsf{BitDecomp}^{-1}(\mu \cdot I_N) + R \cdot A$, and $R \cdot A$ is statistically uniform by the leftover hash lemma when $m = O(n \log q)$ is chosen appropriately.

1.4 Roadmap

After finishing some preliminaries in Section 2, we describe our new FHE construction more formally in Section 3. In Section 4, we provide an overview of our identity-based and attribute-based FHE schemes.

2 Preliminaries

2.1 The Learning with Errors (LWE) Problem and GapSVP

The learning with errors (LWE) problem was introduced by Regev [Reg05].

Definition 1 (LWE)**.** *For security parameter λ, let $n = n(\lambda)$ be an integer dimension, let $q = q(\lambda) \geq 2$ be an integer, and let $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The $\mathrm{LWE}_{n,q,\chi}$ problem is to distinguish the following two distributions: In the first distribution, one samples $(\vec{a}_i, b_i)$ uniformly from $\mathbb{Z}_q^{n+1}$. In the second distribution, one first draws $\vec{s} \leftarrow \mathbb{Z}_q^n$ uniformly and then samples $(\vec{a}_i, b_i) \in \mathbb{Z}_q^{n+1}$ by sampling $\vec{a}_i \leftarrow \mathbb{Z}_q^n$ uniformly, $e_i \leftarrow \chi$, and setting $b_i = \langle \vec{a}_i, \vec{s} \rangle + e_i$. The $\mathrm{LWE}_{n,q,\chi}$ assumption is that the $\mathrm{LWE}_{n,q,\chi}$ problem is infeasible.*

Sometimes it is convenient to view the vectors $b_i \| \vec{a}_i$ as the rows of a matrix A, and to redefine $\vec{s}$ as $(1, -\vec{s})$. Then, either A is uniform, or there is a vector $\vec{s}$ whose first coefficient is 1 such that $A \cdot \vec{s} = \vec{e}$, where the coefficients of $\vec{e}$ come from the distribution χ.

For lattice dimension parameter n and number d, GapSVP_γ is the problem of distinguishing whether a n-dimensional lattice has a vector shorter than d or no vector shorter than $\gamma(n) \cdot d$. The two theorems below capture reductions, quantum and classical, from GapSVP to LWE for certain parameters. We state the result in terms of B-bounded distributions.

Definition 2 (B-bounded distributions)**.** *A distribution ensemble $\{\chi_n\}_{n \in \mathbb{N}}$, supported over the integers, is called B-bounded if*

$$\Pr_{e \leftarrow \chi_n} [|e| > B] = \mathsf{negl}(n) \ .$$

Theorem 1 ([Reg05, Pei09, MM11, MP12], stated as Corollary 2.1 from [Bra12])**.** *Let $q = q(n) \in \mathbb{N}$ be either a prime power or a product of small (size $\mathrm{poly}(n)$) distinct primes, and let $B \geq \omega(\log n) \cdot \sqrt{n}$. Then there exists an efficient sampleable B-bounded distribution χ such that if there is an efficient algorithm that solves the average-case LWE problem for parameters n, q, χ, then:*

- *There is an efficient quantum algorithm that solves $GapSVP_{\tilde{O}(nq/B)}$ on any n-dimensional lattice.*
- *If $q \geq \tilde{O}(2^{n/2})$, then there is an efficient classical algorithm for $GapSVP_{\tilde{O}(nq/B)}$ on any n-dimensional lattice.*

In both cases, if one also considers distinguishers with sub-polynomial advantage, then we require $B \geq \tilde{O}(n)$ and the resulting approximation factor is slightly larger than $\tilde{O}(n^{1.5} q/B)$.

Theorem 2 (Informal Theorem 1.1 of [BLP+13])**.** *Solving n-dimensional LWE with $\mathrm{poly}(n)$ modulus implies an equally efficient solution to a worst-case lattice problem (e.g., GapSVP) in dimension $\sqrt{n}$.*

2.2 Identity-Based and Attribute-Based Homomorphic Encryption

In a homomorphic encryption scheme $\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, the message space is some ring, and Eval homomorphically evaluates arithmetic circuits over this ring (with addition and multiplication gates). We omit formal definitions and theorems regarding homomorphic encryption, which can be found in referenced papers.

An identity-based HE scheme $\mathsf{IBHE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ has all of the properties of a normal IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ [Sha84, BF03]. Setup generates master

keys $(\mathsf{MSK}, \mathsf{MPK})$, $\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{ID})$ outputs a secret key $\mathsf{sk}_{\mathsf{ID}}$ for identity ID, $\mathsf{Enc}(\mathsf{MPK}, \mathsf{ID}, m)$ outputs an encryption c of m under ID, and $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c)$ decrypts c (if it is under ID). Standard security properties apply. For example, an IBE scheme is expected to be *collusion-resistant* – in particular, the adversary can ask for many secret keys, as long as the challenge ciphertext is under an unqueried identity.

For some function family $\mathcal{F}$, IBHE's procedure $c \leftarrow \mathsf{Eval}(\mathsf{MPK}, \mathsf{ID}, f, c_1, \ldots, c_t)$ homomorphically evaluates any $f \in \mathcal{F}$ on ciphertexts $\{c_i \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{ID}, m_i)\}$ under the same ID. Ultimately, $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) = f(m_1, \ldots, m_t)$. We define identity-based (leveled) *fully* homomorphic encryption (IBFHE) in the expected way.

The definition of IBHE can be extended to a multi-identity setting – specifically, Eval could work over ciphertexts under multiple identities. For security to make sense, Dec would require cooperation of all parties whose identities were used in Eval. In this paper, we restrict our attention to the single-identity setting.

An attribute-based HE scheme $\mathsf{ABHE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ has all of the properties of a normal ABE scheme $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ [SW05, GPSW06]. For some relation R, some function family $\mathcal{F}$ and any $f \in \mathcal{F}$, and any ciphertexts $\{c_i \leftarrow \mathsf{Enc}(\mathsf{MPK}, x, m_i)\}$ encrypted under common index x, the ciphertext $c \leftarrow \mathsf{Eval}(\mathsf{MPK}, x, f, c_1, \ldots, c_t)$ can be decrypted (to $f(m_1, \ldots, m_t)$) using a key sk_y for any y for which $R(x, y) = 1$. In an ABE scheme *for circuits*, R can be a circuit of polynomial depth. We define attribute-based (leveled) *fully* homomorphic encryption (ABFHE) in the expected way.

Similar to IBHE, ABHE can be extended so that Eval operates on ciphertexts under multiple indices $x_1, \ldots, x_k$. Regarding decryption, there are different possibilities. For example, the result can only be decrypted using some sk_y for which $R(x_1, y) = \cdots = R(x_k, y) = 1$. Alternatively, the result can be cooperatively decrypted using $\mathsf{sk}_{y_1}, \ldots, \mathsf{sk}_{y_\ell}$ such that for every x_i there is some j such that $R(x_i, y_j) = 1$. We restrict our attention to the single-index setting.

2.3 Other Preliminaries

For n, q, and $\ell = \lfloor \log q \rfloor + 1$, we define the procedures $\mathsf{BitDecomp}$, $\mathsf{BitDecomp}^{-1}$, $\mathsf{Flatten}$ and $\mathsf{Powersof2}$ as described in the Introduction. I_N denotes the N-dimensional identity matrix.

3 Our LWE-Based FHE Scheme

3.1 Basic Encryption Scheme

Here, we formally describe our basic encryption scheme (without homomorphic operations). This description matches the description outlined in the Introduction. In our description, we split up KeyGen into three parts Setup, $\mathsf{SecretKeyGen}$ and $\mathsf{PublicKeyGen}$. We provide two decryption algorithms Dec and MPDec. Dec is sufficient to recover the message μ when it is in a small space (e.g., $\{0, 1\}$). MPDec is an algorithm by Micciancio and Peikert [MP12] that can recover any $\mu \in \mathbb{Z}_q$.

- $\mathsf{Setup}(1^\lambda, 1^L)$: Choose a modulus q of $\kappa = \kappa(\lambda, L)$ bits, lattice dimension parameter $n = n(\lambda, L)$, and error distribution $\chi = \chi(\lambda, L)$ appropriately for LWE that achieves at least 2^λ security against known attacks. Also, choose parameter $m = m(\lambda, L) = O(n \log q)$. Let $params = (n, q, \chi, m)$. Let $\ell = \lfloor \log q \rfloor + 1$ and $N = (n + 1) \cdot \ell$.

- $\mathsf{SecretKeyGen}(params)$: Sample $\vec{t} \leftarrow \mathbb{Z}_q^n$. Output $sk = \vec{s} \leftarrow (1, -t_1, \ldots, -t_n) \in \mathbb{Z}_q^{n+1}$. Let $\vec{v} = \mathsf{Powersof2}(\vec{s})$.

- $\mathsf{PublicKeyGen}(params, sk)$: Generate a matrix $B \leftarrow \mathbb{Z}_q^{m \times n}$ uniformly and a vector $\vec{e} \leftarrow \chi^m$. Set $\vec{b} = B \cdot \vec{t} + \vec{e}$. Set A to be the $(n+1)$-column matrix consisting of $\vec{b}$ followed by the n columns of B. Set the public key $pk = A$. (*Remark:* Observe that $A \cdot \vec{s} = \vec{e}$.)

- $\mathsf{Enc}(params, pk, \mu)$: To encrypt a message $\mu \in \mathbb{Z}_q$, sample a uniform matrix $R \in \{0,1\}^{N \times m}$ and output the ciphertext C given below.

$$C = \mathsf{Flatten}\big(\mu \cdot I_N + \mathsf{BitDecomp}(R \cdot A)\big) \in \mathbb{Z}_q^{N \times N}.$$

- $\mathsf{Dec}(params, sk, C)$: Observe that the first ℓ coefficients of $\vec{v}$ are $1, 2, \ldots, 2^{\ell-1}$. Among these coefficients, let $v_i = 2^i$ be in $(q/4, q/2]$. Let C_i be the i-th row of C. Compute $x_i \leftarrow \langle C_i, \vec{v} \rangle$. Output $\mu' = \lfloor x_i / v_i \rceil$.

- $\mathsf{MPDec}(params, sk, C)$ (for q a power of 2): Observe that $q = 2^{\ell-1}$ and the first $\ell - 1$ coefficients of $\vec{v}$ are $1, 2, \ldots, 2^{\ell-2}$, and therefore if $C \cdot \vec{v} = \mu \cdot \vec{v} + small$, then the first $\ell - 1$ coefficients of $C \cdot \vec{v}$ are $\mu \cdot \vec{g} + small$, where $\vec{g} = (1, 2, \ldots, 2^{\ell-2})$. Recover $\mathrm{LSB}(\mu)$ from $\mu \cdot 2^{\ell-2} + small$, then recover the next-least-significant-bit from $(\mu - \mathrm{LSB}(\mu)) \cdot 2^{\ell-3} + small$, etc. (See [MP12] for the general q case.)

Dec is a BitDecomp'd version of Regev's decryption procedure, applied to one row of the ciphertext, which is a BitDecomp'd Regev ciphertext. (The extra rows will come into play in the homomorphic operations). If C is properly generated, then by the elementary properties of BitDecomp and Powersof2, we have

$$C \cdot \vec{v} = \mu \cdot \vec{v} + R \cdot A \cdot \vec{s} = \mu \cdot \vec{v} + R \cdot \vec{e}.$$

Dec only uses the i-th coefficient of the above expression, which is $x_i = \mu \cdot v_i + \langle R_i, \vec{e} \rangle$. The error $\langle R_i, \vec{e} \rangle$ has magnitude at most $\|\vec{e}\|_1$. In general, if $x_i = \mu \cdot v_i + e'$ for some error e' of magnitude at most $q/8$, and if $v_i \in (q/4, q/2]$, then x_i / v_i differs from μ by at most $(q/8)/v_i < 1/2$, and Dec uses rounding to output the correct value of μ. (In the basic scheme, we set χ to ensure that the error is so bounded with overwhelming probability.)

For the basic scheme (without homomorphic operations), one can take n to be quasi-linear in the security parameter λ and $\kappa = O(\log n)$. When allowing homomorphic operations, L represents the circuit complexity of the functions that the scheme correctly evaluates (roughly, L is the multiplicative depth); we provide a detailed analysis later of how L affects the other parameters.

3.2 Security

Observe that $\mathsf{BitDecomp}^{-1}(C) = \mu \cdot G + R \cdot A$, where $G = \mathsf{BitDecomp}^{-1}(I_N)$ is (the transpose of) the "primitive matrix" used by Micciancio and Peikert [MP12] in their construction of lattice trapdoors, and the rows of $R \cdot A$ are simply Regev [Reg05] encryptions of 0 for dimension n. Assuming $\mathsf{BitDecomp}^{-1}(C)$ hides μ, C does as well, since C can be derived by applying BitDecomp. Thus, the security of our basic encryption scheme follows directly from the following lemma, used to prove the security of Regev's encryption scheme [Reg05].

Lemma 1 (Implicit in [Reg05]). *Let params* $= (n, q, \chi, m)$ *be such that the* $\mathrm{LWE}_{n,q,\chi}$ *assumption holds. Then, for* $m = O(n \log q)$ *and* A, R *as generated above, the joint distribution* $(A, R \cdot A)$ *is computationally indistinguishable from uniform over* $\mathbb{Z}_q^{m \times (n+1)} \times \mathbb{Z}_q^{N \times (n+1)}$.

Concretely, it suffices to take $m > 2n \log q$ [Reg05].

Like Brakerski [Bra12], we can also base security on GapSVP via a classical reduction from LWE [Pei09, BLP+13]. Specifically, Peikert [Pei09] gives a classical reduction of GapSVP to LWE, with the caveat that q must be exponential in n. Brakerski notes that exponential q was unusable in previous FHE schemes, since the ratio of q to the error level B of "fresh" ciphertexts cannot be exponential in n for security reasons (since LLL [LLL82] could be used to break such a scheme), and since B must be very small to permit many homomorphic operations. As in Brakerski's scheme, we do not have that problem. The error bound B of fresh ciphertexts in our scheme does not need to be small to permit many homomorphic operations; we only require q/B to be sub-exponential, and we can therefore permit q to be exponential. Alternatively, we can use a sub-exponential q and base security on GapSVP via Brakerski et al.'s [BLP+13] recent classical reduction to LWE that works even for polynomial-size moduli, with the caveat that, in their reduction, the dimension of the GapSVP instances may be much smaller than the dimension of the LWE instances.

3.3 Homomorphic Operations

Recall that we already described some basic "homomorphic" operations $\mathsf{BitDecomp}$, $\mathsf{BitDecomp}^{-1}$, $\mathsf{Flatten}$, and $\mathsf{Powersof2}$. These will play an important role in analyzing the homomorphic operations supported by our scheme. We remark that $\mathsf{BitDecomp}$ could alternatively decompose with respect to bases other than 2, or according to the Chinese Remainder Theorem.

We provide additional homomorphic operations $\mathsf{MultConst}$, Add, Mult, NAND as follows.

- $\mathsf{MultConst}(C, \alpha)$: To multiply a ciphertext $C \in \mathbb{Z}_q^{N \times N}$ by known constant $\alpha \in \mathbb{Z}_q$, set $M_\alpha \leftarrow \mathsf{Flatten}(\alpha \cdot I_N)$ and output $\mathsf{Flatten}(M_\alpha \cdot C)$. Observe that:

$$\begin{aligned} \mathsf{MultConst}(C, \alpha) \cdot \vec{v} &= M_\alpha \cdot C \cdot \vec{v} = M_\alpha \cdot (\mu \cdot \vec{v} + \vec{e}) = \mu \cdot (M_\alpha \cdot \vec{v}) + M_\alpha \cdot \vec{e} \\ &= \alpha \cdot \mu \cdot \vec{v} + M_\alpha \cdot \vec{e} \end{aligned}$$

  Thus, the error increases by a factor of at most N, regardless of what element $\alpha \in \mathbb{Z}_q$ is used for multiplication. As in "classical" additively homomorphic encryption schemes, we could alternatively perform multiplication-by-constant α by recursively applying Add. But this multiplies the error size by at least α, whereas $\mathsf{MultConst}$ increases the error by at most a factor of N, regardless of α. An example application of $\mathsf{MultConst}$ is that we can perform homomorphic fast Fourier transformations (FFTs) natively over $\mathbb{Z}_q$ without error growth dependent on q. Previously, the error growth depended on the size of the field underlying the FFT [GHS12a, GHS12b], restricting the choice of field.

- $\mathsf{Add}(C_1, C_2)$: To add ciphertexts $C_1, C_2 \in \mathbb{Z}_q^{N \times N}$, output $\mathsf{Flatten}(C_1 + C_2)$. The correctness of this operation is immediate. Note that the addition of messages is over the full base ring $\mathbb{Z}_q$.

- $\mathsf{Mult}(C_1, C_2)$: To multiply ciphertexts $C_1, C_2 \in \mathbb{Z}_q^{N \times N}$, output $\mathsf{Flatten}(C_1 \cdot C_2)$. Observe that:

$$\begin{aligned} \mathsf{Mult}(C_1, C_2) \cdot \vec{v} &= C_1 \cdot C_2 \cdot \vec{v} = C_1 \cdot (\mu_2 \cdot \vec{v} + \vec{e}_2) + \mu_2 \cdot (\mu_1 \vec{v} + \vec{e}_1) + C_1 \cdot \vec{e}_2 \\ &= \mu_1 \cdot \mu_2 \cdot \vec{v} + \mu_2 \cdot \vec{e}_1 + C_1 \cdot \vec{e}_2 \end{aligned}$$

As in Add, the multiplication operator is over the full base field $\mathbb{Z}_q$. In Mult, the new error depends on the old errors, the ciphertext C_1, and the message μ_2. The dependence on the old errors seems unavoidable (and normal for LWE-based HE schemes), and observe that C_1 contributes at most a factor N blowup of error, since all components of C_1 are restricted to $\{0,1\}$. The error growth based on the message μ_2, however, presents a concern. In general, we must address this concern by using homomorphic operations in a way that restricts the message space to small messages. One way to do this is to consider Boolean circuits using only NAND operations: this would restrict the message space to $\{0,1\}$. We elaborate below.

- $\mathsf{NAND}(C_1, C_2)$: To NAND ciphertexts $C_1, C_2 \in \mathbb{Z}_q^{N \times N}$ that are known to encrypt messages $\mu_1, \mu_2 \in \{0,1\}$, output $\mathsf{Flatten}(I_N - C_1 \cdot C_2)$. Observe that:

$$\mathsf{NAND}(C_1, C_2) \cdot \vec{v} = (I_N - C_1 \cdot C_2) \cdot \vec{v} = (1 - \mu_1 \cdot \mu_2) \cdot \vec{v} - \mu_2 \cdot \vec{e}_1 - C_1 \cdot \vec{e}_2$$

  Note here that the NAND homomorphic operation maintains the invariant that if the input messages are in $\{0,1\}$, then the output ciphertext will also encryption of $\{0,1\}$, thus guaranteeing small messages. Note that since $\mu_2 \in \{0,1\}$, the error is increased by a factor of at most $N+1$.

Circuits. By iteratively applying the homomorphic operations above, different types of (bounded-depth) circuits may be homomorphically computed while maintaining correctness of decryption.

The simplest case to analyze is the case of Boolean circuits computed over encryptions of $\{0,1\}$ values. In this case, the circuit can be converted to use only NAND gates, and through appropriate leveled application of the NAND homomorphic operation, the final ciphertext's error will be bounded by $(N+1)^L \cdot B$, where L is the NAND-depth of the circuit, and B is the original bound on the error of a fresh encryption of $\{0,1\}$.

More generally, with more care, we may consider arithmetic circuits over $\mathbb{Z}_q$ that make use of gates that perform addition, multiplication, or multiplication by a known constant. However, as we have seen in the case of multiplication gates, the error growth may depend on the values being encrypted in intermediate computations. One way to deal with this is to focus on situations where (1) all input values are known to encrypt values bounded by some value T, and (2) the arithmetic circuit is chosen to guarantee that all intermediate values are also bounded by T' whenever the circuit inputs are constrained to values bounded by T. In such a situation, the final ciphertext's error will be bounded by $(N+T')^L \cdot B$, where L is the depth of the arithmetic circuit, and B is the original bound on the error of fresh encryptions of values smaller than T. For example, in this way, we can homomorphically evaluate polynomials of degree d in this large-message-space variant when the initial messages are bounded by roughly $q^{1/d}$, achieving a scheme that is "somewhat homomorphic" [Gen09]. Another example application would be to convert encryptions of a polynomially bounded set of small values to encryptions of binary values, by using an appropriate arithmetic circuit for the conversion. Once converted to encryptions of binary values, a NAND-based Boolean circuit could be used for further computations.

3.4 Parameters, Performance and Optimizations

Suppose that $\mathsf{Flatten}$'d ciphertexts C_1, C_2 encrypt $\mu_1, \mu_2 \in \{0,1\}$ under approximate eigenvector $\vec{v}$ with B-bounded error – that is, $C_i \cdot \vec{v} = \mu_i \cdot \vec{v} + \vec{e}_i$ where $|\vec{e}_i|_\infty \leq B$. Then $C^{\mathsf{NAND}} \leftarrow \mathsf{NAND}(C_1, C_2)$ encrypts $\mathsf{NAND}(\mu_1, \mu_2) \in \{0,1\}$ under $\vec{v}$ with $(N+1)B$-bounded error. As long as $q/B > 8(N+1)^L$,

we can evaluate a depth-L circuit of NANDs over B-bounded ciphertexts to obtain a $q/8$-bounded ciphertext, which Dec will decrypt correctly.

As in previous LWE-based FHE schemes, n (hence N) must increase linearly with $\log(q/B)$ to maintain fixed 2^λ security against known attacks, so q/B grows more like $\exp(L \log L)$. We will brush such issues under the rug and view n as a fixed parameter. Choosing χ so that B is not too large, and since in practice there is no reason to have $\kappa = \log q$ grow super-linearly with n, we have $\kappa = O(L \log N) = O(L(\log n + \log \kappa)) = O(L \log n)$, similar to [BGV12, Bra12]. Given that the NAND procedure is dominated by multiplication of two $N \times N$ matrices for $N = O(n\kappa) = \tilde{O}(nL)$, we have the following theorem to characterize the performance of our FHE scheme.

Theorem 3. *For dimension parameter n and depth parameter L, our FHE scheme evaluates depth-L circuits of* NAND *gates with $\tilde{O}((nL)^\omega)$ field operations per gate, where $\omega < 2.3727$ is the matrix multiplication exponent.*

This compares favorably with previous LWE-based FHE schemes, which all have at least $\tilde{O}(n^3 L)$ field operations per gate [BV11b, BGV12, Bra12].

Theorem 3 hides some factors, both good and bad. On the good size, it hides the fact that ciphertext matrices in our scheme have 0/1 entries, and therefore can be multiplied faster than if they were general matrices over $\mathbb{Z}_q$. In previous LWE-based FHE schemes, the field operations involve multiplying a small number with a general number of $\mathbb{Z}_q$, which has complexity $\tilde{O}(\kappa) = \tilde{O}(L)$. So, previous LWE-based FHE schemes have real complexity $\tilde{O}(n^3 L^2)$ whereas ours remains $\tilde{O}((nL)^\omega)$. On the bad side, Theorem 3 hides logarithmic factors in the dimension of the ciphertext matrices, since $N = O(n\kappa) = O(nL \log n)$. We note that typically n will dominate L, since for very deep circuits, one would want to use Gentry's bootstrapping technique [Gen09] to make the per-gate computation *independent* of L.

Since bootstrapping involves homomorphically evaluating the decryption function, and since Dec is essentially Regev decryption [Reg05], bootstrapping works as in previous LWE-based FHE schemes. In particular, we can use techniques from [BV11b] to reduce the dimension and modulus-size of the ciphertext before bootstrapping, so that the complexity of decryption (and hence bootstrapping) is completely independent of the depth L of the circuit that was evaluated to arrive at that ciphertext. Regev decryption can be evaluated in $O(\log n)$ depth. Due to the logarithmic depth, one can take q/B to be quasi-polynomial in n, and base security on LWE for quasi-polynomial factors.

4 Our Identity-Based and Attribute-Based FHE Schemes

Identity-based encryption (IBE) [Sha84, BF03] and attribute-based encryption (ABE) [SW05, GPSW06] are designed to provide more flexible access control of encrypted data than a traditional public key infrastructure. Traditionally, IBE and ABE do not offer any computation over the encrypted data. However, access control of encrypted data remains important even (or especially) when the data is encrypted homomorphically. (See [CHT13] for a nice discussion of applications.)

Unfortunately, while there are some IBE schemes that allow simple homomorphic operations [GHV10, CHT13], it has remained a stubborn open problem [Nac10, GHV10, Bra12, CHT13] to construct an IBE scheme that allows fully or even "somewhat" homomorphic encryption. Previously it was mentioned [Bra12, CHT13]) that instead of building an FHE scheme on Regev's encryption scheme as we do in Section 3, one can alternatively use the "dual-Regev" system [GPV08], for

which it is known how to generate identity-based keys (see also [ABB10a, ABB10b, CHKP10]). However, making the encryption/decryption keys identity-based only solves half of the problem, and yields only a "weak" form of identity-based FHE. In all previous FHE schemes, there is also an "evaluation key" required for homomorphic evaluation. This evaluation key is user-specific and is not "identity-based", in the sense that it cannot be computed non-interactively from the user's identity. But having to *obtain* this evaluation key undermines the main appeal of IBE: its non-interactivity. Thus, identity-based FHE (IBFHE) has remained wide open, and attribute-based FHE (ABFHE) seems even more difficult to construct.

Interestingly, however, our new FHE scheme does not have evaluation keys. To perform evaluation, the evaluator only needs to know some basic parameters of the scheme (like n, m and ℓ).

The absence of evaluation keys allows us to construct the first IBFHE scheme. We describe a simple "compiler" that transforms any LWE-based IBE scheme (that satisfies certain natural properties) into a IBFHE. All LWE-based IBE schemes that we know of (e.g., [GPV08, ABB10a, ABB10b, CHKP10]) can be described so as to have the required properties.

1. **Property 1 (Ciphertext and decryption key vectors)**: The decryption key for identity ID, and a ciphertext for ID, are vectors $\vec{s}_{\mathsf{ID}}, \vec{c}_{\mathsf{ID}} \in \mathbb{Z}_q^{n'}$ for some n'. The first coefficient of $\vec{s}_{\mathsf{ID}}$ is 1.
2. **Property 2 (Small Dot Product)**: If $\vec{c}_{\mathsf{ID}}$ encrypts 0, then $\langle \vec{c}_{\mathsf{ID}}, \vec{s}_{\mathsf{ID}} \rangle$ is "small".
3. **Property 3 (Security)**: Encryptions of 0 are indistinguishable from uniform vectors over $\mathbb{Z}_q$ (under LWE).

Theorem 4. *We can compile an IBE scheme* E *with the above properties into a related IBFHE scheme.*

Proof. The IBFHE uses E's Setup and KeyGen algorithms, supplementing E's MPK with the basic parameters for our FHE scheme (such as m, ℓ). Let $N = (n+1) \cdot \ell$ for $\ell = \lfloor \log q \rfloor + 1$, as usual. To encrypt $\mu \in \{0,1\}$, the encrypter generates N encryptions of 0 using $\mathsf{E.Enc}$, sets C'_{ID} to be the $N \times (n+1)$ matrix whose rows are these ciphertexts, and outputs $C_{\mathsf{ID}} = \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(C'_{\mathsf{ID}}))$. Suppose $\vec{s}_{\mathsf{ID}}$ is the decryption key for ID, as above, and let $\vec{v}_{\mathsf{ID}} = \mathsf{Powersof2}(\vec{s}_{\mathsf{ID}})$. The decrypter runs our FHE decryption algorithm $\mathsf{Dec}(\vec{v}_{\mathsf{ID}}, C_{\mathsf{ID}})$ to recover μ. Homomorphic operations are as in Section 3.3.

Decryption is correct, since $C_{\mathsf{ID}} \cdot \vec{v}_{\mathsf{ID}} = \mu \cdot \vec{v}_{\mathsf{ID}} + C'_{\mathsf{ID}} \cdot \vec{s}_{\mathsf{ID}} = \mu \cdot \vec{v}_{\mathsf{ID}} + \mathit{small}$, where $C'_{\mathsf{ID}} \cdot \vec{s}_{\mathsf{ID}}$ is a small vector by Property 2. In this setting Dec recovers $\mu \in \{0,1\}$. Any adversary that breaks the semantic security of our IBFHE scheme can distinguish C'_{ID} from a uniform matrix over $\mathbb{Z}_q$, and therefore distinguish LWE by Property 3. □

For ABFHE, our approach begins by re-interpreting the decryption process in the Gorbunov et al. (GVW) ABE scheme [GVW13]. To decrypt a ABE ciphertext under x with sk_y for which $R(x,y) = 1$, we view the decrypter as deriving a "sub-key" $\vec{s}_{x,y}$ associated to x. This sub-key will satisfy something similar to Property 2 above – i.e., if $\vec{c}_x$ encrypts 0 under x, then $\langle \vec{c}_x, \vec{s}_{x,y} \rangle$ is "small". Viewing GVW in this way allows us to apply our compiler above.

We provide more details of our identity-based and attribute-based FHE constructions in Appendices A and B.

Acknowledgments

We gratefully thank Shai Halevi for his collaboration on this work. We also thank Boaz Barak and the anonymous CRYPTO reviewers for their many helpful comments.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *CRYPTO*, pages 98–115, 2010.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108, 1996.

[BB12a] Boaz Barak and Zvika Brakerski. Building the swiss army knife. Windows on Theory Blog, http://windowsontheory.org/2012/05/02/building-the-swiss-army-knife, 2012.

[BB12b] Boaz Barak and Zvika Brakerski. The swiss army knife of cryptography. Windows on Theory Blog, http://windowsontheory.org/2012/05/01/the-swiss-army-knife-of-cryptography, 2012.

[BF03] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. Extended abstract in Crypto'01.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science (ITCS'12)*, 2012. Available at `http://eprint.iacr.org/2011/277`.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013.

[Boy13] Xavier Boyen. Attribute-based functional encryption on lattices. In *TCC*, pages 122–142, 2013.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, pages 868–886, 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, volume 6841, page 501, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011. References are to full version: `http://eprint.iacr.org/2011/344`.

[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.

[CHT13] Michael Clear, Arthur Hughes, and Hitesh Tewari. Homomorphic encryption with access policies: Characterization and new constructions. In *AFRICACRYPT*, pages 61–87, 2013.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2013:183, 2013. To appear in CRYPTO 2013.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *CRYPTO*, pages 487–504, 2011.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer, 2012.

[Gen09] 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.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. *IACR Cryptology ePrint Archive*, 2013:128, 2013. To appear in CRYPTO 2013.

[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*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.

[GHL+11] Craig Gentry, Shai Halevi, Vadim Lyubashevsky, Christopher Peikert, Joseph Silverman, and Nigel Smart, 2011. Observation.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, pages 465–482, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In *CRYPTO*, pages 850–867, 2012.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple bgn-type cryptosystem from lwe. In *EUROCRYPT*, pages 506–522, 2010.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, pages 89–98, 2006.

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

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554, 2013.

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481, 2002.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.

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

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In *CRYPTO*, pages 465–484, 2011.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

[Nac10] David Naccache. Is theoretical cryptography any good in practice? Invited talk at Crypto/CHES 2010, available at http://www.iacr.org/workshops/ches/ches2010, 2010.

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

[RAD78] Ron Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[Reg10] Oded Regev. The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204. IEEE Computer Society, 2010.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography - PKC'10*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, 2005.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *STOC*, pages 887–898, 2012.

A Our Identity-Based FHE Construction: More Details

In this section, as an example, we review Cash et al.'s LWE-based IBE scheme [CHKP10], and show that it has the properties required by our compiler described in Section 4. Since Cash et al. actually describe a *hierarchical* IBE (HIBE) scheme, our compiler turns it into a hierarchical IBFHE scheme.

A.1 (Hierarchical) IBFHE: Definition

An identity-based encryption (IBE) scheme IBE has four algorithms Setup, KeyGen, Enc, Dec [Sha84, BF03]. Setup generates master keys $(\mathsf{MSK}, \mathsf{MPK})$, $\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{ID})$ outputs a secret key $\mathsf{sk}_{\mathsf{ID}}$ for identity ID, $\mathsf{Enc}(\mathsf{MPK}, \mathsf{ID}, \mu)$ outputs an encryption c of μ under ID, and $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c)$ decrypts c (if it is under ID). (Often KeyGen is instead called "$\mathsf{Extract}$" to avoid confusion with the key generation that occurs in Setup.)

In a d-hierarchical IBE (HIBE) scheme [HL02, GS02], identities are vectors of length at most d, and there is fifth algorithm called Derive. Given identities $\mathsf{ID}, \mathsf{ID}'$ where ID is a proper prefix of ID' and a secret key $\mathsf{sk}_{\mathsf{ID}}$ for ID, $\mathsf{Derive}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{ID}, \mathsf{ID}')$ outputs a secret key $\mathsf{sk}_{\mathsf{ID}'}$ for ID'.

In a d-hierarchical IBFHE (HIBFHE) scheme, there is a sixth algorithm Eval. For some function family $\mathcal{F}$, $c \leftarrow \mathsf{Eval}(\mathsf{MPK}, \mathsf{ID}, f, c_1, \ldots, c_t)$ homomorphically evaluates any $f \in \mathcal{F}$ on ciphertexts $\{c_i \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{ID}, \mu_i)\}$ under the same ID. Since the scheme is (leveled) fully homomorphic, $\mathcal{F}$ consists of all circuits of depth L for some L, consisting of (e.g.) NAND gates. In the hierarchical context, it is also fine if ID is a prefix of all of the identity tuples associated to the original ciphertexts.

Correctness for d-HIBFHE means the following: For any $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}$ and any legal sequence of calls to KeyGen and Derive to obtain a secret key $\mathsf{sk}_{\mathsf{ID}}$ for any identity $\mathsf{ID} = (id_1, \ldots, id_k)$, for any t and any ciphertexts $\{c_i \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{ID}_i, \mu_i) : i \in [t]\}$ generated under d-hierarchical identities ID_i that all have ID as a prefix, for any t-ary function $f \in \mathcal{F}$, $c \leftarrow \mathsf{Eval}(\mathsf{MPK}, \mathsf{ID}, f, c_1, \ldots, c_t)$ is a ciphertext that satisfies $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) = f(\mu_1, \ldots, \mu_t)$. We could consider a weaker version of HIBFHE where $\mathsf{ID} = \mathsf{ID}_i$ for all i, but our construction will satisfy the stronger version.

Notions of security for HIBFHE are the same as for HIBE, with the understanding that Eval is an additional (public) algorithm that the adversary may try to exploit in its attack. We refer to prior papers on HIBE for details about HIBE security; here, we highlight a few details. The

adversary may be *selective* or *adaptive* in its choice of which target identity to use in its challenge ciphertext. In either case, the adversary is permitted to query secret keys for many identities as long as none of these identities are prefixes of the target identity. However, in a selective-ID attack, the adversary must specify the target identity before receiving MPK. Also, a HIBE scheme may be anonymous. In an anonymous HIBE scheme, it is hard for an adversary to distinguish under which of two equal-length identities a ciphertext was constructed. Similarly, one can define a notion of anonymous HIBFHE. In a HIBFHE scheme, of course Eval cannot take as input any identity information. In fact, our HIBE to HIBFHE compiler does not require any identity information, and therefore can be used to compile anonymous HIBE schemes (such as Cash et al.'s scheme) into anonymous HIBFHE schemes.

A.2 Review of Cash et al.'s Hierarchical IBE Construction

Perhaps the key lemma of Cash et al. [CHKP10] is the following:

Lemma 2 (Lemma 3.3 in full version of [CHKP10])**.** *There is a deterministic polynomial-time algorithm* $\mathsf{ExtBasis}$ *with the following properties: given an arbitrary* $A \in \mathbb{Z}_q^{n \times m}$ *whose columns generate the entire group* $\mathbb{Z}_q^n$*, an arbitrary basis* $S \in \mathbb{Z}^{m \times m}$ *of* $\Lambda^\perp(A)$*, and an arbitrary* $\bar{A} \in \mathbb{Z}_q^{n \times \bar{m}}$*,* $\mathsf{ExtBasis}(S, A' = A \| \bar{A})$ *outputs a basis* S' *of* $\Lambda^\perp(A') \subseteq \mathbb{Z}^{m+\bar{m}}$ *such that* $\|\tilde{S}'\| = \|\tilde{S}\|$*.*

Above, $\Lambda^\perp(A) = \{\vec{x} \in \mathbb{Z}^m : A \cdot \vec{x} = \vec{0} \in \mathbb{Z}_q^n\}$ is the space of vectors orthogonal modulo q to all of the rows of A, and $\tilde{S}'$ and $\tilde{S}$ are the Gram-Schmidt orthogonalizations of S' and S. Assuming the vectors in S are short, we may sometimes refer to S as a matrix of *short integer solutions* associated to the matrix A. These short vectors are "solutions" in the sense that $A \cdot S = 0 \bmod q$. The above lemma basically says that if we are given short integer solutions for A, we can extend them to obtain short integer solutions for $A' = A \| \bar{A}$, where $\bar{A}$ can be any matrix (of appropriate dimension). There are efficient algorithms to generate a statistically uniform A together with a basis S of short integer solutions for A (see [Ajt96] and more recently [MP12]). Gentry, Peikert and Vaikuntanathan (Theorem 4.1 in [GPV08]) showed that, given a basis S of short integer solutions for A (where A's columns are full-rank) and any vector $\vec{z} \in \mathbb{Z}_q^n$, there is an efficient algorithm $\mathsf{SampleD}(S, A, \vec{z})$ that samples a short vector $\vec{t} \in \mathbb{Z}^m$ such that $A \cdot \vec{t} = \vec{z} \bmod q$.

With this machinery, we now sketch Cash et al.'s scheme (with minor variations). We limit ourselves to their "binary tree encryption" (BTE) scheme, where identities are tuples of bits. HIBE schemes whose identities use larger alphabets can be mapped to a BTE scheme using a collision-resistant hash function.

$\mathsf{Setup}(d)$: Generate statistically uniform $A_0 \in \mathbb{Z}_q^{n \times m}$ together with a short basis $S_0 \in \mathbb{Z}^{m \times m}$ of $\Lambda^\perp(A_0)$. For each $(i, b) \in [d] \times \{0, 1\}$, generate a uniform and independent matrix $A_{i,b} \in \mathbb{Z}_q^{n \times m}$. Choose $\vec{z} \in \mathbb{Z}_q^n$ uniformly at random. Output $\mathsf{MPK} = (A_0, \{A_{i,b}\}, \vec{z}, d)$ and $\mathsf{MSK} = S_0$.

$\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{ID})$: Let $\mathsf{ID} = (id_1, \ldots, id_k) \in \{0, 1\}^k$ for $k \leq d$. Define $A_{\mathsf{ID}} = A_0 \| A_{1,id_1} \| \cdots \| A_{k,id_k}$. Generate $S_{\mathsf{ID}} \leftarrow \mathsf{ExtBasis}(S_0, A_{\mathsf{ID}})$, a basis of short integer solutions for A_{ID}.

Once it obtains S_{ID}, the user itself does the following. It sets $\vec{t}_{\mathsf{ID}} \leftarrow \mathsf{SampleD}(S_{\mathsf{ID}}, A_{\mathsf{ID}}, \vec{z})$ and $\vec{s}_{\mathsf{ID}} = (1, -\vec{t}_{\mathsf{ID}})$. Let $A'_{\mathsf{ID}} = \vec{z} \| A_{\mathsf{ID}}$. Observe that $A'_{\mathsf{ID}} \cdot \vec{s}_{\mathsf{ID}} = \vec{0} \bmod q$.

$\mathsf{Derive}(S_{\mathsf{ID}}, \mathsf{ID}, \mathsf{ID}')$: Similar to KeyGen, it uses $\mathsf{ExtBasis}$ to generate a secret key for lower-level identity ID'.

$\mathsf{Enc}(\mathsf{MPK}, \mathsf{ID}, \mu \in \{0,1\})$: Let $\mathsf{ID} = (id_1, \ldots, id_k) \in \{0,1\}^k$ for $k \leq d$. Define A'_{ID} as above. Suppose A'_{ID} has m' columns. Let $\vec{\mu} \in \mathbb{Z}_q^{m'}$ be the vector of all 0's except with $\mu \cdot \lfloor q/2 \rfloor$ in the first coefficient. Choose random $\vec{r} \in \mathbb{Z}_q^n$ and small error vector $\vec{e} \in \mathbb{Z}^{m'}$. Output $\vec{c}_{\mathsf{ID}} \leftarrow \vec{r} \cdot A'_{\mathsf{ID}} + \vec{e} + \vec{\mu} \in \mathbb{Z}_q^{m'}$.

$\mathsf{Dec}(\vec{s}_{\mathsf{ID}}, \vec{c}_{\mathsf{ID}})$: Pad $\vec{s}_{\mathsf{ID}}$ with 0's if necessary to make it the same length as $\vec{c}_{\mathsf{ID}}$. Set $\delta \leftarrow \langle \vec{c}_{\mathsf{ID}}, \vec{s}_{\mathsf{ID}} \rangle \in \mathbb{Z}_q$. If δ is small, output '$\mu = 0$'; if $\delta - q/2 \bmod q$ is small, output '$\mu = 1$'; otherwise, output '$\perp$'.

Regarding correctness, observe that we have $\langle \vec{c}_{\mathsf{ID}}, \vec{s}_{\mathsf{ID}} \rangle = \vec{r} \cdot A'_{\mathsf{ID}} \cdot \vec{s}_{\mathsf{ID}} + \langle \vec{e} + \vec{\mu}, \vec{s}_{\mathsf{ID}} \rangle = \mu \cdot \lfloor q/2 \rfloor + small$. Regarding security, Cash et al. show that encryptions of 0 (i.e., vectors of the form $\vec{r} \cdot A'_{\mathsf{ID}} + \vec{e}$) are indistinguishable from uniformly random vectors under the LWE assumption in an adaptive-ID attack model.

A.3 Hierarchical IBFHE: Applying Our Compiler to Cash et al.'s HIBE

Our compiler from Section 4 applies immediately to Cash et al.'s (non-hierarchical) IBE scheme. It is clear from the description above that their scheme has the three properties required by our compiler.

For the hierarchical setting, we need to address case where Dec uses $\vec{s}_{\mathsf{ID}}$ for some ID that is a (possibly proper) prefix of all of the identities $\{\mathsf{ID}_i\}$ associated to ciphertexts $\{\vec{c}_i\}$ used in Eval. Our compiler basically works as before, except that all ciphertext matrices need to be padded with 0's so that they all have the same square dimension, and the secret key vector is also padded with 0's to match this dimension.

In more detail, let ID_1 be a prefix of ID_2. Applying our compiler, a ciphertext for ID_2 has the form $C_{\mathsf{ID}_2} = \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(C'_{\mathsf{ID}_2}))$, where the rows of C'_{ID_2} are Cash et al. encryptions of 0. In Cash et al.'s scheme, if ID_1 is a prefix of ID_2, and $\vec{c}_{\mathsf{ID}_2}$ is an encryption of 0 under ID_2, it holds that $\langle \vec{c}_{\mathsf{ID}_2}, \vec{s}_{\mathsf{ID}_1} \rangle$ is small, when $\vec{s}_{\mathsf{ID}_1}$ is appropriately padded with 0's. Therefore, if we set $\vec{v}_{\mathsf{ID}_1} \leftarrow \mathsf{Powersof2}(\vec{s}_{\mathsf{ID}_1})$, we have that

$$C_{\mathsf{ID}_2} \cdot \vec{v}_{\mathsf{ID}_1} = \mu \cdot \vec{v}_{\mathsf{ID}_1} + C'_{\mathsf{ID}_2} \cdot \vec{s}_{\mathsf{ID}_1} = \mu \cdot \vec{v}_{\mathsf{ID}_1} + small.$$

Thus C_{ID_2} has the proper form of an approximate eigenvector encryption of μ under ID_1, and homomorphic operations proceed as usual.

Regarding security, it is easy to see that an attack on the semantic security of this scheme would imply an algorithm to distinguish Cash et al. encryptions of 0 from random vectors, breaking LWE. Our HIBFHE scheme inherits the adaptive-ID security of Cash et al.'s HIBE.

B Our Attribute-Based FHE Construction: More Details

In this section, we review a recent attribute-based encryption (ABE) scheme by Gorbunov, Vaikuntanathan and Wee (GVW) [GVW13], and show that we can re-interpret their decryption procedure so that we can apply our compiler from Section 4 to obtain an attribute-based FHE (ABFHE) scheme.[3]

[3] We believe our techniques would work on Boyen's recent LWE-based ABE scheme [Boy13] as well, but do not pursue this further.

B.1 ABFHE: Definition

An attribute-based encryption (ABE) scheme [SW05, GPSW06] is associated to some efficiently computable relation $R(x, y)$, $x \in \{0,1\}^k$, $y \in \{0,1\}^\ell$, and has four algorithms Setup, KeyGen, Enc, Dec. Setup generates master keys $(\mathsf{MSK}, \mathsf{MPK})$, $\mathsf{KeyGen}(\mathsf{MSK}, y)$ outputs a secret key sk_y for string $y \in \{0,1\}^\ell$, $\mathsf{Enc}(\mathsf{MPK}, x, \mu)$ outputs an encryption c of μ under string $x \in \{0,1\}^k$, and $\mathsf{Dec}(\mathsf{sk}_y, c)$ decrypts c (if c is under x where $R(x, y) = 1$). In an ABE scheme *for circuits*, R can be any efficiently computable relation. Recently, the first ABE schemes for circuits were constructed [GGH+13b, GVW13].

An ABFHE scheme has a fifth algorithm Eval associated to some function family $\mathcal{F}$. For any $f \in \mathcal{F}$, $c \leftarrow \mathsf{Eval}(\mathsf{MPK}, x, f, c_1, \ldots, c_t)$ homomorphically evaluates any f on ciphertexts $\{c_i \leftarrow \mathsf{Enc}(\mathsf{MPK}, x, \mu_i)\}$ under the same string x. Since the scheme is (leveled) fully homomorphic, $\mathcal{F}$ consists of all circuits of depth L for some L, consisting of (e.g.) NAND gates.

Correctness for ABFHE means the following: For any $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}$, for any secret key $\mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, y)$ for any $y \in \{0,1\}^\ell$, for any t and any ciphertexts $\{c_i \leftarrow \mathsf{Enc}(\mathsf{MPK}, x, \mu_i) : i \in [t]\}$ generated under string $x \in \{0,1\}^k$ for which $R(x, y) = 1$, for any t-ary function $f \in \mathcal{F}$, $c \leftarrow \mathsf{Eval}(\mathsf{MPK}, x, f, c_1, \ldots, c_t)$ is a ciphertext that satisfies $\mathsf{Dec}(\mathsf{sk}_y, c) = f(\mu_1, \ldots, \mu_t)$.

Notions of security for ABFHE are the same as for ABE, with the understanding that Eval is an additional (public) algorithm that the adversary may try to exploit in its attack. As in ABE schemes, the adversary may be *selective* or *adaptive* in its choice of which string x^* to use in its challenge ciphertext. In either case, the adversary is permitted to query secret keys for many strings y_i as long as $R(x^*, y_i) = 0$ for all i. In a selective attack, the adversary must specify x^* before receiving MPK.

B.2 Review of the GVW ABE Construction

Gorbunov et al. describe a selectively-secure ABE scheme for circuits. We describe their ABE scheme in a rather different way than [GVW13]. The main difference is that we split decryption into a two-step process. In the first step, assuming $R(x, y) = 1$, the keyholder uses sk_y to generate a sub-key $\vec{s}_{x,y}$ for x. This sub-key is a short vector. Moreover, if $\vec{c}_x$ encrypts 0 under x, then $\langle \vec{c}_x, \vec{s}_{x,y} \rangle$ is "small". The second part of decryption begins with computing this dot product. Splitting GVW decryption in this way allows us to apply a compiler similar to our compiler for the IBE case.

We re-use some notation from the description of Cash et al.'s IBE scheme in Appendix A. In particular, for a matrix $A \in \mathbb{Z}_q^{n \times m}$, we let $\Lambda^\perp(A) = \{\vec{x} \in \mathbb{Z}^m : A \cdot \vec{x} = \vec{0} \in \mathbb{Z}_q^n\}$ denote the space of vectors orthogonal modulo q to all of the rows of A. When $S \in \mathbb{Z}^{m \times m}$ is a basis of short (much smaller than q) vectors that generate $\Lambda^\perp(A)$, we may refer to S as a basis of *short integer solutions* for A in the sense that $A \cdot S = 0 \bmod q$. There are efficient algorithms to generate a statistically uniform A together with a basis S of short integer solutions for A (see [Ajt96] and more recently [MP12]). Gentry, Peikert and Vaikuntanathan (Theorem 4.1 in [GPV08]) showed that, given a basis S of short integer solutions for A (where A's columns are full-rank) and any vector $\vec{z} \in \mathbb{Z}_q^n$, there is an efficient algorithm $\mathsf{SampleD}(S, A, \vec{z})$ that samples a short vector $\vec{t} \in \mathbb{Z}^m$ such that $A \cdot \vec{t} = \vec{z} \bmod q$.

The GVW scheme uses recoding matrices. We say that matrix $R^{2m \times m}$ recodes matrix $A_0 \in \mathbb{Z}_q^{n \times m}$ in terms of matrices $A_1 \| A_2 \in \mathbb{Z}_q^{n \times 2m}$ if $[A_1 \| A_2] \cdot R = A_0 \bmod q$, and moreover the entries of R are small. We can efficiently generate a recoding matrix from A_0 to $A_1 \| A_2$ given a basis of short integer solutions for $A_1 \| A_2$ using $\mathsf{SampleD}$.

An important tool in GVW is what we will call "recoding of recodings", a recursive process which uses a recoding of A_0 in terms of some lower-level matrices together with recodings of the lower-level matrices in terms of even-lower-level ones to recode A_0 strictly in terms of the even-lower-level matrices. For example, suppose we have recodings R_0, R_1, R_2 such that $[A_1 \| A_2] \cdot R_0 = A_0 \bmod q$, and $[A_3 \| A_4] \cdot R_1 = A_1 \bmod q$, and $[A_4 \| A_5] \cdot R_2 = A_2 \bmod q$. The idea, in English, is that since each column of A_0 is expressed (via R_0) as a short integer linear combination of columns in $A_1 \| A_2$, and since each column of $A_1 \| A_2$ is expressed (via R_1 and R_2) as a short integer linear combination of columns from $A_3 \| A_4 \| A_5$, we can use R_0, R_1, R_2 to efficiently construct a new recoding matrix R'_0 that represents each column of A_0 as a (somewhat) short linear combination strictly of the columns of $A_3 \| A_4 \| A_5$ (no columns from $A_1 \| A_2$ are used anymore). Mathematically, if we let $R_{0,top}$ be the top m rows of R_0 and $R_{0,bot}$ be the bottom m rows, we have $A_0 = A_1 \cdot R_{0,top} + A_2 \cdot R_{0,bot}$. Similarly, we have $A_1 = A_3 \cdot R_{1,top} + A_4 \cdot R_{1,bot}$ and $A_2 = A_4 \cdot R_{2,top} + A_5 \cdot R_{2,bot}$. Plugging the latter two equations into the first equation, we obtain

$$\begin{aligned} A_0 &= (A_3 \cdot R_{1,top} + A_4 \cdot R_{1,bot}) \cdot R_{0,top} + (A_4 \cdot R_{2,top} + A_5 \cdot R_{2,bot}) \cdot R_{0,bot} \\ &= A_3 \cdot (R_{1,top} \cdot R_{0,top}) + A_4 \cdot (R_{1,bot} \cdot R_{0,top} + R_{2,top} \cdot R_{0,bot}) + A_5 \cdot (R_{2,bot} \cdot R_{0,bot}) \\ &= [A_3 \| A_4 \| A_5] \cdot R'_0, \end{aligned}$$

where R'_0 is the transposed concatenation of $R_{1,top} \cdot R_{0,top}$, $R_{1,bot} \cdot R_{0,top} + R_{2,top} \cdot R_{0,bot}$, and $R_{2,bot} \cdot R_{0,bot}$, and is the "recoding of recodings" of A_0 strictly in terms of $A_3 \| A_4 \| A_5$. This new recoding R'_0 is only "somewhat" short, since it is quadratic in the original "short" recodings. If we were to recursively recode A_0 in terms of matrices that are d levels lower, the resulting recoding matrix would have entries that are roughly the d-th power of the entries of the initial recoding matrices.

In GVW, y defines some predicate $P_y : \{0,1\}^k \to \{0,1\}$, represented by a boolean circuit of depth at most d. We are now ready to describe the GVW scheme, with minor variations, restricting to the case of 1-bit messages.

$\mathsf{Setup}(k, d)$: Generate statistically uniform matrices $A_{0,1} \in \mathbb{Z}_q^{n \times 1}, \{A_{i,b} \in \mathbb{Z}_q^{n \times m} : i \in [k], b \in \{0,1\}\}$, together with bases of short integer solutions $\{S_{i,b}\}$. Output $\mathsf{MPK} = (A_{0,1}, \{A_{i,b}\}, k, d)$ and $\mathsf{MSK} = \{S_{i,b}\}$.

$\mathsf{KeyGen}(\mathsf{MSK}, y)$: Let $P_y : \{0,1\}^k \to \{0,1\}$ be the d-depth boolean circuit associated to y. Associate $A_{0,1}$ to the output wire of P_y (with an assignment of 1) and associate $A_{i,b}$ to the i-th input wire of P_y (with an assignment of b). For each internal wire $w \in P_y$ and $b \in \{0,1\}$, generate uniform and independent matrices $A_{w,b} \in \mathbb{Z}_q^{n \times m}$ together with a basis $S_{w,b}$ of short integer solutions. For each gate g in P_y, do the following. Suppose g's wires are w_l (left), w_r (right) and w_o (output). For each possible pair of inputs $(b_l, b_r) \in \{0,1\}^2$ to g, use the short integer solution matrices to generate a recoding matrix R_{g,b_l,b_r} from $A_{w_o,g(b_l,b_r)}$ to $A_{w_l,b_l} \| A_{w_r,b_r}$. The key sk_y consists of the recoding matrices $\{R_{g,b_l,b_r}\}$.

$\mathsf{Enc}(\mathsf{MPK}, x, \mu \in \{0,1\})$: Let $A_x = A_{1,x_1} \| \cdots \| A_{k,x_k}$ and $A'_x = A_{0,1} \| A_x$, the latter of which has $1 + km$ columns. Let $\vec{\mu} \in \mathbb{Z}_q^{1+km}$ be the vector of all 0's except that the first coefficient is $\mu \cdot \lfloor q/2 \rceil$. Choose random $\vec{r} \in \mathbb{Z}_q^n$ and small error vector $\vec{e} \in \mathbb{Z}^{1+km}$. Output x and $\vec{c}_x \leftarrow \vec{r} \cdot A'_x + \vec{e} + \vec{\mu} \in \mathbb{Z}_q^{1+km}$.

$\mathsf{Dec}(\mathsf{sk}_y, x, \vec{c}_x)$: Evaluate $P_y(x)$. If $P_y(x) = 0$, then output '$\perp$'. Otherwise, run $\vec{t}_{x,y} \leftarrow \mathsf{Recode}(\mathsf{sk}_y, x)$ to obtain a recoding of $A_{0,1}$ in terms of A_x – that is, $\vec{t}_{x,y} \in \mathbb{Z}_q^{km}$ is a short vector such that

$A_{0,1} = A_x \cdot \vec{t}_{x,y} \bmod q$. Let $\vec{s}_{x,y} = (1, -\vec{t}_{x,y}) \in \mathbb{Z}_q^{1+km}$. Compute $\delta \leftarrow \langle \vec{c}_x, \vec{s}_{x,y} \rangle \in \mathbb{Z}_q$. If δ is small, output '$\mu = 0$'; if $\delta - q/2 \bmod q$ is small, output '$\mu = 1$'; otherwise, output '$\perp$'.

$\mathsf{Recode}(\mathsf{sk}_y, x)$: Evaluate $P_y(x)$ to obtain a bit $b_w^{(x)}$ for each wire w in P_y. Partition the wires of the circuit into $k \leq d$ levels $W_0, \ldots, W_k$, where W_0 is the output level and W_k is the input level. To begin, sk_y contains a way to recode $A_{0,1}$ in terms of $A_{w_l, b_{w_l}^{(x)}} \| A_{w_r, b_{w_r}^{(x)}}$ where $w_l, w_r \in W_1$ are the left and right input wires of the output gate of P_y. Similarly and recursively, for each $i \in \{2, \ldots, k\}$, use the recoding matrices from sk_y and the "recoding of recodings" technique to transform the recoding of $A_{0,1}$ under $\cup_{w \in W_{i-1}} A_{w, b_w^{(x)}}$ into a recoding under $\cup_{w \in W_i} A_{w, b_w^{(x)}}$. Finally, at the input level, output the recoding of $A_{0,1}$ in terms of A_x – i.e., a short vector $\vec{t}_{x,y}$ such that $A_x \cdot \vec{t}_{x,y} = A_{0,1} \bmod q$.

Regarding correctness, observe that $\vec{s}_{x,y} \in \mathbb{Z}_q^{1+km}$ is a short vector that satisfies $A'_x \cdot \vec{s}_{x,y} = \vec{0}$. Therefore, $\langle \vec{c}_x, \vec{s}_{x,y} \rangle = \vec{r} \cdot A'_x \cdot \vec{s}_{x,y} + \langle \vec{e} + \vec{\mu}, \vec{s}_{x,y} \rangle = \mu \cdot \lfloor q/2 \rceil + small$.

Regarding security, the details are unimportant for us, except Gorbunov et al. show that, assuming LWE, GVW encryptions of 0 under an index x are computationally indistinguishable from uniform to an adversary given MPK and only permitted to query keys for predicates P_y for which $P_y(x) = 0$.

B.3 ABFHE: Applying Our Compiler to the GVW ABE

Once the decrypter runs $\vec{t}_{x,y} \leftarrow \mathsf{Recode}(\mathsf{sk}_y, x)$ and computes the "sub-key" $\vec{s}_{x,y} \leftarrow (1, -\vec{t}_{x,y})$, we are again in a setting where we can apply our compiler from Section 4. Namely, the following three properties are met:

1. **Property 1 (Ciphertext and decryption key vectors)**: The sub-key for index x for key sk_y, and the ciphertext for index x, are vectors $\vec{s}_{x,y}, \vec{c}_x \in \mathbb{Z}_q^{m'}$ for some m'. The first coefficient of $\vec{s}_{x,y}$ is 1.
2. **Property 2 (Small Dot Product)**: If $\vec{c}_x$ encrypts 0, then $\langle \vec{c}_x, \vec{s}_{x,y} \rangle$ is "small".
3. **Property 3 (Security)**: Encryptions of 0 are indistinguishable from uniform vectors over $\mathbb{Z}_q$ (under LWE).

The compiler works exactly as before. In detail, our ABFHE uses the ABE scheme's Setup and KeyGen algorithms. Let $N = m' \cdot \ell$ for $\ell = \lfloor \log q \rfloor + 1$. To encrypt $\mu \in \{0, 1\}$ under x, the encrypter generates N ABE encryptions of 0 under x, sets C'_x to be the $N \times m'$ matrix whose rows are these ciphertexts, and outputs $C_x = \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(C'_x))$. Suppose $\vec{s}_{x,y}$ is the sub-key for x and sk_y, as above, and let $\vec{v}_{x,y} = \mathsf{Powersof2}(\vec{s}_{x,y})$. The decrypter runs our FHE decryption algorithm $\mathsf{Dec}(\vec{v}_{x,y}, C_x)$ to recover μ. Homomorphic operations are as in Section 3.3.

Decryption is correct, since $C_x \cdot \vec{v}_{x,y} = \mu \cdot \vec{v}_{x,y} + C'_x \cdot \vec{s}_{x,y} = \mu \cdot \vec{v}_{x,y} + small$, where $C'_x \cdot \vec{s}_{x,y}$ is a small vector by Property 2. In this setting Dec recovers $\mu \in \{0, 1\}$. Any adversary that breaks the semantic security of our ABFHE scheme can distinguish C'_x from a uniform matrix over $\mathbb{Z}_q$, and therefore distinguish LWE by Property 3.

C Approximate Eigenvector FHE Based on Other Assumptions

C.1 Ring LWE

The ring learning with errors (RLWE) problem was introduced by Lyubaskevsky, Peikert and Regev (LPR) [LPR10]. It is a natural adaptation of the LWE problem from vectors over $\mathbb{Z}_q$ to vectors

over polynomial rings $\mathbb{Z}_q[x]/f(x)$. The similarity between LWE and RLWE makes it very easy to map our LWE-based FHE scheme to a RLWE-based scheme.

Our RLWE-based scheme is more efficient than the LWE-based one, but not (yet) as efficient as previous RLWE-based FHE schemes. The reason for this relative inefficiency is that, while previous RLWE-based FHE schemes after [BV11a] (such as [BGV12, Bra12]) use "relinearization", this relinearization is much more efficient for RLWE-based FHE schemes than LWE-based ones. In particular, since RLWE-based ciphertext vectors in previous schemes have constant dimension, the relinearization matrices are only constant size (up to a $\log q$ factor). Nonetheless, we describe our approximate eigenvector FHE scheme under RLWE here in the hope that it will stimulate new ideas.

First, let us recall RLWE. We use an simplified special-case version of the problem that is easier to work with [Reg10, BV11a].

Definition 3 (RLWE)**.** *For security parameter λ, let $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2. Let $q = q(\lambda) \geq 2$ be an integer. Let $R = \mathbb{Z}[x]/(f(x))$ and let $R_q = R/qR$. Let $\chi = \chi(\lambda)$ be a distribution over R. The $\mathrm{RLWE}_{d,q,\chi}$ problem is to distinguish the following two distributions: In the first distribution, one samples (a_i, b_i) uniformly from R_q^2. In the second distribution, one first draws $s \leftarrow R_q$ uniformly and then samples $(a_i, b_i) \in R_q^2$ by sampling $a_i \leftarrow \mathbb{R}_q$ uniformly, $e_i \leftarrow \chi$, and setting $b_i = a_i \cdot s + e_i$. The $\mathrm{RLWE}_{d,q,\chi}$ assumption is that the $\mathrm{RLWE}_{d,q,\chi}$ problem is infeasible.*

Typically, one chooses the noise distribution χ according to a Gaussian distribution with deviation small relative to q. This Gaussian distribution may need to be "ellipsoidal" for certain reductions to go through [LPR10]. It has been shown for RLWE that one can equivalently assume that s is alternatively sampled from the noise distribution χ [LPR10].

In this paper, we prefer to view the vectors $b_i \| a_i$ as the rows of a matrix A, and define $\vec{s}$ as $(1, -s)$. Then, either A is uniform, or there is a two-dimensional vector $\vec{s}$ whose first coefficient is 1 such that $A \cdot \vec{s} = \vec{e}$, where the coefficients of $\vec{e}$ come from the distribution χ.

The RLWE problem is useful, because the well-established shortest vector problem (SVP) over ideal lattices can be reduced to it, specifically:

Theorem 5 (Lyubashevsky-Peikert-Regev [LPR10])**.** *For any d that is a power of 2, ring $R = \mathbb{Z}[x]/(x^d + 1)$, prime integer $q = q(d) = 1 \bmod d$, and $B = \omega(\sqrt{d \log d})$, there is an efficiently samplable distribution χ that outputs elements of R of length at most B with overwhelming probability, such that if there exists an efficient algorithm that solves $\mathrm{RLWE}_{d,q,\chi}$, then there is an efficient quantum algorithm for solving $d^{\omega(1)} \cdot (q/B)$-approximate worst-case SVP for ideal lattices over R.*

Toward constructing our FHE scheme, we begin with the LPR encryption scheme [LPR10]. The public key in LPR is simply a RLWE instance; in our case, just a 1×2 matrix A over R_q. The secret key is $\vec{s} = (1, s_1) \in R_q^2$, where $A \cdot \vec{s} = e$ is small. We assume s_1 is also small (chosen according to distribution χ). To encrypt 0, one samples small $r \in R_q$ and short vector $\vec{e}' \in \mathbb{R}_q^2$ according to distribution χ, and outputs $\vec{c} \leftarrow r \cdot A + \vec{e}' \in R_q^2$. By a standard hybrid argument, RLWE implies that $\vec{c}$ is indistinguishable from a random vector in R_q^2. To encrypt $\mu \in \{0, 1\}$ in LPR, one adds $\mu \cdot \lfloor q/2 \rceil$ to the first coefficient of $\vec{c}$. Decryption computes $\langle \vec{c}, \vec{s} \rangle = r \cdot e + \langle \vec{e}', \vec{s} \rangle + \mu \cdot \lfloor q/2 \rceil$, and outputs '$\mu = 0$' or '$\mu = 1$' depending on whether or not the result is small.

(One drawback of the above approach to RLWE-based encryption, for our purposes, is that it needs $\|\vec{e}'\| \cdot \|\vec{s}\| < q$, and therefore the error distribution χ essentially needs to be B-bounded for $B \approx \sqrt{q}$. For various reasons (e.g., to base security on classical GapSVP), we would like to permit

B to be larger. To fix this "problem", we can alternatively use an encryption approach more similar to Regev and the scheme described in Section 3.1, in which the size of the ciphertext error does not depend on the size of the secret key, and in particular s_1 can be uniform in R_q.)

Recall that we transform Regev's encryption scheme into our approximate eigenvector encryption scheme by computing a ciphertext as $C \leftarrow \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(C'))$, where the rows of C' are Regev encryptions of 0. We do a similar thing here to adapt the LPR encryption scheme to the approximate eigenvector framework. For $\ell = \lfloor \log q \rfloor + 1$, we set $N = 2 \cdot \ell$. The encrypter generates N LPR encryptions of 0, sets this as matrix C', and sets $C \leftarrow \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(C'))$, where $\mathsf{BitDecomp}(a) = (a_0, \ldots, a_{\ell-1}) \in R_q^\ell$ where each a_i is an element of R that when represented as a polynomial of degree $d-1$ has coefficients that are all in $\{0, 1\}$. (Alternatively, the elements a_i could be size-reduced with respect to the *canonical embedding* [LPR10] rather than with respect to the *coefficient embedding*.) The approximate eigenvector of the scheme becomes $\vec{v} \leftarrow \mathsf{Powersof2}(\vec{s})$, where as before $\mathsf{Powersof2}(\vec{a}) = (a_0, 2a_0, \ldots, 2^{\ell-1}a_0, a_1, \ldots...)$; if $\vec{a} \in R_q^k$, then $\mathsf{Powersof2}(\vec{a}) \in R_q^{k \cdot \ell}$. Decryption computes $C \cdot \vec{v} = \mu \cdot \vec{v} + \mathsf{BitDecomp}(C') \cdot \vec{v} = \mu \cdot \vec{v} + C' \cdot \vec{s} = \mu \cdot \vec{v} + small$. The security of the scheme follows from the fact that LPR encryptions of 0 are indistinguishable from random vectors under the RLWE assumption.

Homomorphic operations proceed as expected, where adding or multiplying ciphertext matrices (followed by $\mathsf{Flatten}$) gives a small ciphertext that encrypts the sum or product of the original messages. If we restrict the messages to $\mu \in \{0, 1\}$, then for reasons similar to the LWE setting, the noise level of the ciphertext remains tightly constrained so that we obtain a leveled FHE scheme.

We can adapt NTRU-based FHE [GHL+11, LATV12] to the approximate-eigenvector framework in a similar fashion.

C.2 Approximate GCD

Van Dijk et al. [vDGHV10] constructed a FHE scheme based on the *approximate gcd* problem.

Definition 4 (Approximate GCD)**.** *Let ρ, η, γ be some parameters. For a specific η-bit odd positive integer p, let $\mathcal{D}_{\gamma,\rho}(p)$ be the distribution induced by sampling integer q uniformly from $[0, 2^\gamma/p)$, integer r uniformly from $(-2^\rho, 2^\rho)$, and outputting $x \leftarrow p \cdot q + r$. The (ρ, η, γ)-approximate-gcd problem is: given polynomially samples from $\mathcal{D}_{\gamma,\rho}(p)$ for a randomly chosen η-bit odd positive integer p, output p.*

In their original scheme, they suggested parameters $\rho = O(\lambda)$, $\eta = \tilde{O}(\lambda^2)$, $\gamma = \tilde{O}(\lambda^5)$.

Work building on [vDGHV10] (e.g., [CMNT11, CNT12]) typically uses a stronger assumption.

Definition 5 (Error-Free Approximate GCD)**.** *Like Approximate GCD, except that the instance also includes a single x_0 that is sampled from $D_{\gamma,0}(p)$. In other words, the instance includes a single integer x_0 that is an exact multiple of p.*

This assumption is much stronger, since breaking it reduces to factoring integers, but it still seems hard for the suggested parameters above.

One can construct a very simple somewhat homomorphic encryption scheme based on error-free approximate gcd. (Van Dijk et al. use bootstrapping to make the somewhat homomorphic scheme leveled fully homomorphic.) KeyGen outputs sk as a η-bit odd positive integer p and pk as integers $x_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$, with x_0 an exact multiple of p. $\mathsf{Encrypt}(\mathsf{pk}, \mu \in \{0, 1\})$ outputs $c \leftarrow \mu + 2r + 2\sum_{i \in S} x_i \bmod x_0$, where r is a random integer in $(-2^{\rho'}, 2^{\rho'})$ for ρ' slightly larger than

ρ and S is a random subset of indices. $\mathsf{Dec}(\mathsf{sk}, c)$ outputs $(c \bmod p) \bmod 2$. Eval evaluates addition (XOR) and multiplication (AND) gates simply by adding and multiplying the ciphertexts modulo x_0. Van Dijk et al. provide a search-to-decision reduction that effectively shows that encryptions of 0 are indistinguishable from random numbers of equivalent bit-length. (Their proof is actually for a scheme based on approximate gcd rather than error-free approximate gcd, but the proof adapts easily to the latter context.)

To adapt the error-free approximate gcd scheme to the approximate-eigenvector framework, we first re-interpret the the error-free approximate gcd scheme so that the secret key becomes a vector, and decryption involves a dot product. Let $\ell = \lfloor \log x_0 \rfloor + 1$. Let $\vec{c} \leftarrow \mathsf{BitDecomp}(c)$ be the ℓ-bit decomposition of a ciphertext c, and let $\vec{v} \leftarrow \mathsf{Powersof2}(1)$ have coefficients $\{2^i : i \in [0, \ell - 1]\}$. Then, we have $c = \langle \vec{c}, \vec{v} \rangle$. Recall that an encryption of 0 in the error-free approximate gcd scheme is a near-multiple of p, and therefore $\langle \vec{c}, \vec{v} \rangle \bmod p$ is very small.

Now, to get an approximate-eigenvector scheme based on error-free approximate gcd, we use our standard tricks. To encrypt μ, we construct ℓ van Dijk et al. encryptions of 0, arranged as a $\ell \times 1$ matrix C', and then set our ciphertext to be $C \leftarrow \mathsf{Flatten}(\mu \cdot I_\ell + \mathsf{BitDecomp}(C'))$. (The $\mathsf{BitDecomp}^{-1}$ procedure inside $\mathsf{Flatten}$ is defined modulo x_0.) Decryption computes $C \cdot \vec{v} = \mu \cdot \vec{v} + \mathsf{BitDecomp}(C') \cdot \vec{v} = \mu \cdot \vec{v} + C' = \mu \cdot \vec{v} + \mathit{small} \bmod p$, and then recovers μ. Security follows from the fact that C reveals nothing more than $\mathsf{BitDecomp}^{-1}(C)$, which looks random modulo x_0, since C' looks random modulo x_0. Homomorphic operations are as in Section 3.3. If we restrict the messages to $\mu \in \{0, 1\}$, then for reasons similar to the LWE setting, the noise level of the ciphertext remains tightly constrained so that we obtain a leveled FHE scheme.

With a little more work, we can obtain an approximate-eigenvector FHE scheme based on a variant of (non-error-free) approximate gcd. Specifically, van Dijk et al. provide a variant of their scheme in which the public key contains additional near-multiples of p – namely, for $i = 0, \ldots, \gamma$, the values $x'_i \leftarrow 2(q'_i \cdot p + r'_i)$, where r'_i comes from the usual noise distribution $(-2^\rho, 2^\rho)$, but the quotient q'_i is an integer sampled from $[2^{\gamma+i-1}/p, 2^{\gamma+i}/p]$. Thus, the values $x'_i \in [2^{\gamma+i}, 2^{\gamma+i+1}]$ form a "ladder" of near-multiples of p, each one about twice the size of the previous one, until finally the largest one has size about $2^{2\gamma}$. After a multiplication of two ciphertexts, we can now size-reduce the resulting ciphertext (which has size about $2^{2\gamma}$) by subtracting off an appropriate subset sum of the terms in the ladder of near-multiples, to obtain a ciphertext of normal size (about 2^γ) that encrypts the same value. Since this variant is based on a variant of approximate gcd where the instance includes such a ladder of near multiples, it may be subject to additional attacks, but at least no error-free multiples are provided that would allow an immediate reduction to factoring.

In our setting, we use van Dijk et al.'s ladder of near-multiples as follows. Suppose $C \in \mathbb{Z}^{\ell \times \ell}$ is a ciphertext matrix whose entries are no longer in $\{0, 1\}$ (e.g., because it the sum or product of two previous ciphertext matrices). Set $C' \leftarrow \mathsf{BitDecomp}^{-1}(C)$ without any modular reduction, so that a row $C_i = (C_{i,0}, \ldots, C_{i,\ell-1})$ is mapped to $C'_i \leftarrow \sum_j C_{i,j} 2^j \in \mathbb{Z}$. Now use the ladder of near-multiples to size-reduce each C'_i, to obtain a new ciphertext $C^\dagger \in \mathbb{Z}^\ell$. Since we subtracted "encryptions of 0" from each entry, $C' \approx C^\dagger \bmod p$, up to a small (additive) error. Now that the entries of $C^\dagger$ each have only about γ bits, $\mathsf{BitDecomp}(C^\dagger)$ gives a vector with coefficients in $\{0, 1\}$ that encrypts the same value as C, with only a small difference in error.