Fully Homomorphic Encryption Using Ideal Lattices

Craig Gentry
Stanford University and IBM Watson
cgentry@cs.stanford.edu

ABSTRACT

We propose a fully homomorphic encryption scheme – i.e., a scheme that allows one to evaluate circuits over encrypted data without being able to decrypt. Our solution comes in three steps. First, we provide a general result – that, to construct an encryption scheme that permits evaluation of *arbitrary circuits*, it suffices to construct an encryption scheme that can evaluate (slightly augmented versions of) its *own decryption circuit*; we call a scheme that can evaluate its (augmented) decryption circuit *bootstrappable*.

Next, we describe a public key encryption scheme using *ideal lattices* that is *almost* bootstrappable. Lattice-based cryptosystems typically have decryption algorithms with low circuit complexity, often dominated by an inner product computation that is in NC1. Also, *ideal* lattices provide both additive and *multiplicative* homomorphisms (modulo a public-key ideal in a polynomial ring that is represented as a lattice), as needed to evaluate general circuits.

Unfortunately, our initial scheme is not quite bootstrappable – i.e., the depth that the scheme can correctly evaluate can be logarithmic in the lattice dimension, just like the depth of the decryption circuit, but the latter is greater than the former. In the final step, we show how to modify the scheme to reduce the depth of the decryption circuit, and thereby obtain a bootstrappable encryption scheme, without reducing the depth that the scheme can evaluate. Abstractly, we accomplish this by enabling the *encrypter* to start the decryption process, leaving less work for the decrypter, much like the server leaves less work for the decrypter in a server-aided cryptosystem.

Categories and Subject Descriptors: E.3 [Data Encryption]: Public key cryptosystems

General Terms: Algorithms, Design, Security, Theory

1. INTRODUCTION

We propose a solution to the old open problem of constructing a *fully homomorphic encryption scheme*. This notion, originally called a *privacy homomorphism*, was introduced by Rivest, Adleman and Dertouzos [54] shortly after the invention of RSA by Rivest, Adleman and Shamir [55]. Basic RSA is a multiplicatively homomorphic encryption scheme – i.e., given RSA public key $\text{pk} = (N, e)$ and ciphertexts $\{\psi_i \leftarrow \pi_i^e \bmod N\}$, one can efficiently compute $\prod_i \psi_i = (\prod_i \pi_i)^e \bmod N$, a ciphertext that encrypts the product of the original plaintexts. Rivest et al. [54] asked a natural question: What can one do with an encryption scheme that is *fully* homomorphic: a scheme $\mathcal{E}$ with an efficient algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ that, for any valid public key pk, *any* circuit C (not just a circuit consisting of multiplication gates), and any ciphertexts $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\text{pk}, \pi_i)$, outputs

$$\psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\text{pk}, C, \psi_1, \ldots, \psi_t) \ ,$$

a valid encryption of $C(\pi_1, \ldots, \pi_t)$ under pk? Their answer: one can arbitrarily *compute on encrypted data* – i.e., one can process encrypted data (query it, write into it, do anything to it that can be efficiently expressed as a circuit) without the decryption key. As an application, they suggested private data banks: a user can store its data on an untrusted server in encrypted form, yet still allow the server to process, and respond to, the user's data queries (with responses more concise than the trivial solution: the server just sends all of the encrypted data back to the user to process). Since then, cryptographers have accumulated a list of "killer" applications for fully homomorphic encryption. However, prior to this proposal, *we did not have a viable construction.*

1.1 Homomorphic Encryption

A homomorphic public key encryption scheme $\mathcal{E}$ has four algorithms $\mathsf{KeyGen}_{\mathcal{E}}$, $\mathsf{Encrypt}_{\mathcal{E}}$, $\mathsf{Decrypt}_{\mathcal{E}}$, and an additional algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ that takes as input the public key pk, a circuit C from a permitted set $\mathcal{C}_{\mathcal{E}}$ of circuits, and a tuple of ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$; it outputs a ciphertext ψ. The computational complexity of all of these algorithms must be polynomial in security parameter λ and (in the case of $\mathsf{Evaluate}_{\mathcal{E}}$) the size of C. $\mathcal{E}$ is *correct* for circuits in $\mathcal{C}_{\mathcal{E}}$ if, for any key-pair (sk, pk) output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, any circuit $C \in \mathcal{C}_{\mathcal{E}}$, any plaintexts $\pi_1, \ldots, \pi_t$, and any ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$ with $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\text{pk}, \pi_i)$, it is the case that:

$$\psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\text{pk}, C, \Psi) \ \Rightarrow \ C(\pi_1, \ldots, \pi_t) = \mathsf{Decrypt}_{\mathcal{E}}(\text{sk}, \psi)$$

By itself, mere correctness does not exclude trivial schemes.[1] So, we require ciphertext size and decryption time to be up-

[1]In particular, we could define $\mathsf{Evaluate}_{\mathcal{E}}(\text{pk}, C, \Psi)$ to just output (C, Ψ) without "processing" the circuit or ciphertexts at all, and $\mathsf{Decrypt}_{\mathcal{E}}$ to decrypt the component ciphertexts and apply C to results.

STOC'09, May 31–June 2, 2009, Bethesda, Maryland, USA.

per bounded by a function of the security parameter λ, independently of C (or perhaps, as a relaxation, dependent on the *depth* of C, but not the *size*).

DEFINITION 1 (HOMOMORPHIC ENCRYPTION). *$\mathcal{E}$ is homomorphic for circuits in $\mathcal{C}_\mathcal{E}$ if $\mathcal{E}$ is correct for $\mathcal{C}_\mathcal{E}$ and $\mathsf{Decrypt}_\mathcal{E}$ can be expressed as a circuit $D_\mathcal{E}$ of size* $\mathrm{poly}(\lambda)$.

DEFINITION 2 (FULLY HOMOMORPHIC ENCRYPTION). *$\mathcal{E}$ is fully homomorphic if it is homomorphic for all circuits.*

DEFINITION 3 (LEVELED FULLY HOM. ENCRYPTION). *A family of schemes $\{\mathcal{E}^{(d)} : d \in \mathbb{Z}^+\}$ is leveled fully homomorphic if they all use the same decryption circuit, $\mathcal{E}^{(d)}$ is homomorphic for all circuits of depth at most d (that use some specified set of gates Γ), and the computational complexity of $\mathcal{E}^{(d)}$'s algorithms is polynomial in λ, d, and (in the case of $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$) the size of C.*

REMARK 1. One may desire – e.g., for the two-party computation setting – the additional property that $\mathsf{Encrypt}_\mathcal{E}$ and $\mathsf{Evaluate}_\mathcal{E}$ have the same output distribution, or that there is at least some post-hoc randomization procedure that induces the same output distribution. We discuss such *circuit privacy* in Section 7.

Here, we focus on constructing a scheme that is semantically secure against chosen plaintext attacks (or just "semantically secure"). Unfortunately a scheme that has nontrivial homomorphisms cannot be semantically secure against adaptive chosen ciphertext attacks (CCA2), since it is malleable. There are relaxed notions of CCA2 security [3, 16, 52], but they do not apply to a fully homomorphic scheme. However, constructing a CCA1-secure fully homomorphic encryption scheme is an interesting open problem.

1.2 Our Results

We construct a fully homomorphic encryption scheme using ideal lattices. The result can roughly be broken down into three steps: a general "bootstrapping" result, an "initial construction" using ideal lattices, and a technique to "squash the decryption circuit" to permit bootstrapping.

Our research began with the second step: a PKE scheme $\mathcal{E}_1$ described in Section 3 that uses ideal lattices and is homomorphic for shallow circuits. A ciphertext ψ has the form $\mathbf{v} + \mathbf{x}$ where $\mathbf{v}$ is in the ideal lattice and $\mathbf{x}$ is an "error" or "offset" vector that encodes the plaintext π. Interpreting ciphertext vectors as coefficient vectors of elements in a polynomial ring $\mathbb{Z}[x]/f(x)$, we add and multiply ciphertexts using ring operations – $\psi \leftarrow \psi_1 + \psi_2$ or $\psi \leftarrow \psi_1 \times \psi_2$ – and induce addition and multiplication of the underlying plaintexts. By itself, this scheme improves upon prior work. It compares favorably to Boneh-Goh-Nissim [11]; it is homomorphic for circuits with greater multiplicative depth while allowing essentially unlimited additions. The security of $\mathcal{E}_1$ is based on a natural decisional version of the closest vector problem for ideal lattices for ideals in a fixed ring.

$\mathcal{E}_1$ is homomorphic only for shallow circuits because the "error" vector *grows* with addition and (especially) multiplication operations; eventually, it becomes so long that it causes a decryption error. It is natural to ask: can we "refresh" a ciphertext whose error vector is almost too long to obtain a new ciphertext whose error vector is shorter? Obviously, we could refresh a ciphertext if we could completely decrypt it! This is the idea behind bootstrapping: we *do* decrypt the ciphertext, but homomorphically!

Specifically, suppose $\mathcal{E}$ is bootstrappable with plaintext space $\mathcal{P}$ is $\{0,1\}$, and that circuits are boolean. Suppose we have a ciphertext ψ_1 that encrypts π under pk_1, which we want to refresh. So that we can decrypt it homomorphically, suppose we also have sk_1, the secret key for pk_1, encrypted under a second public key pk_2: let $\overline{\mathrm{sk}_{1j}}$ be the encrypted secret key bits. Consider the following algorithm.

$\mathsf{Recrypt}_\mathcal{E}(\mathrm{pk}_2, D_\mathcal{E}, \langle\overline{\mathrm{sk}_{1j}}\rangle, \psi_1)$.

Set $\overline{\psi_{1j}} \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_\mathcal{E}(\mathrm{pk}_2, \psi_{1j})$
Output $\psi_2 \leftarrow \mathsf{Evaluate}_\mathcal{E}(\mathrm{pk}_2, D_\mathcal{E}, \langle\langle\overline{\mathrm{sk}_{1j}}\rangle, \langle\overline{\psi_{1j}}\rangle\rangle)$

Above, $\mathsf{Evaluate}$ takes in the bits of sk_1 and ψ_1, each encrypted under pk_2. Then, $\mathcal{E}$ is used to evaluate the decryption circuit homomorphically. The output ψ_2 is thus an encryption under pk_2 of $\mathsf{Decrypt}_\mathcal{E}(\mathrm{sk}_1, \psi_1) = \pi$.[2] Applying the decryption circuit $D_\mathcal{E}$ removes the error vector associated to the first ciphertext, but $\mathsf{Evaluate}_\mathcal{E}$ simultaneously introduces a new error vector. Intuitively, we have made progress as long as the second error vector is shorter.

Suppose $\mathcal{C}_\mathcal{E}$ contains not just $D_\mathcal{E}$ but also the augmentation of $D_\mathcal{E}$ by NAND (i.e., a NAND gate connecting two copies of $D_\mathcal{E}$). Then, we say $\mathcal{E}$ is *bootstrappable*.

THEOREM 1 (INFORMAL). *One can construct a (semantically secure) family $\{\mathcal{E}^{(d)}\}$ of leveled fully homomorphic encryption schemes from any (semantically secure) bootstrappable encryption scheme $\mathcal{E}$.*

$\mathcal{E}^{(d)}$'s public key contains $d+1$ $\mathcal{E}$ public keys, basically one for each level of the circuit, and an acyclic chain of encrypted secret keys. It evaluates a d-depth NAND-circuit as follows: given ciphertexts encrypted under pk_i associated to wires at the ith level of the circuit, it $\mathsf{Recrypt}$s them so that they become encrypted under pk_{i-1}, applies the NAND gates at that level, and recurses for $i-1$. If $\mathcal{E}$ securely encrypts key-dependent messages (is KDM-secure) [9, 24, 12] – i.e., roughly, if providing a ciphertext that encrypts a function of the secret key does not hurt security – then the $\mathcal{E}$ public keys need not be distinct; we can have a "loop," even a "self-loop," of encrypted secret keys, and the derived scheme is fully homomorphic. See Section 2 for formal details.

In Section 4, we describe a scheme $\mathcal{E}_2$ that "tweaks" $\mathcal{E}_1$, analyze the complexity of its decryption circuit, and explain why we are unable to prove that it is bootstrappable. In Section 5, we describe a transformation of $\mathcal{E}_2$, labeled $\mathcal{E}_3$, where the $\mathcal{E}_3$ public key contains a set of vectors with a secret sparse subset whose sum is related to $\mathcal{E}_2$ secret key. This allows us to re-express decryption as a sparse subset vector sum rather than a full matrix-vector product, which requires a circuit of lower depth, permitting bootstrapping. It also entails an additional assumption: roughly, that it is hard to distinguish a set that has a sparse subset with a known sum from a set that is uniform. Similar assumptions have been analyzed in connection with server-aided decryption.

THEOREM 2 (INFORMAL). *$\mathcal{E}_3$ is bootstrappable and is semantically secure (under two assumptions).*

[2] $\mathsf{Recrypt}$ implies a one-way proxy re-encryption scheme [10].

1.3 Other Related Work

Homomorphic encryption schemes that are not semantically secure, like basic RSA, may also have stronger attacks on their one-wayness. Boneh and Lipton [13] proved that any algebraic privacy homomorphism over a ring $\mathbb{Z}_n$ can be broken in sub-exponential time under a (reasonable) number theoretic assumption, if the scheme is deterministic or otherwise offers an equality oracle. See also [35] and [18].

Goldwasser and Micali [23] proposed the first semantically secure homomorphic encryption scheme (and also introduced the notion of semantic security). Their scheme is "additively homomorphic" over $\mathbb{Z}_2$; in our terminology, its set $\mathcal{C}_{\mathcal{E}}$ of permitted circuits contains circuits with XOR gates. Other additively homomorphic encryption schemes with proofs of semantic security are Benaloh [8], Naccache-Stern [42], Okamoto-Uchiyama [46], Paillier [47], and Damgard-Jurik [19]. Some additively homomorphic encryption schemes use lattices or linear codes [22, 50, 27, 36, 37, 4]. ElGamal [20] is multiplicatively homomorphic.

Semantically secure schemes that allow both addition and multiplication include Boneh-Goh-Nissim [11] (quadratic formulas) and "Polly Cracker" by Fellows and Koblitz [21, 29, 30, 31] (arbitrary circuits but with exponential ciphertext-size blow-up). Sanders, Young and Yung [56, 7] (SYY) use circuit-private additively homomorphic encryption to construct a circuit-private scheme that can handle NC1 circuits. Ishai and Paskin [26] do this for branching programs, which covers NC1 circuits (by Barrington [6]), and ciphertexts in their scheme are much shorter – proportional to the *length* of the branching program rather than its *size*, though the *computation* is proportional to the size.

2. BOOTSTRAPPABLE ENCRYPTION

To be bootstrappable, $\mathcal{E}$ needs to be able to evaluate not only its decryption circuit, which merely allows recryptions of the same plaintext, but also slightly augmented versions of it, so that we can perform nontrivial operations on plaintexts and make progress through a circuit.

Definition 4 (Augmented Decryption Circuit). *Let Γ be a set of gates with inputs and output in plaintext space $\mathcal{P}$ including the identity gate (input and output are the same). For gate $g \in \Gamma$, the* g-augmented decryption circuit *consists of a g-gate connecting multiple copies of $D_{\mathcal{E}}$ (the number of copies equals the number of inputs to g), where $D_{\mathcal{E}}$ takes a secret key and ciphertext as input formatted as elements of $\mathcal{P}^{\ell(\lambda)}$. We denote the set of g-augmented decryption circuits, $g \in \Gamma$, by $D_{\mathcal{E}}(\Gamma)$.*

Definition 5 (Bootstrappable Encryption). *Let $\mathcal{C}_{\mathcal{E}}$ be a set of circuits with respect to which $\mathcal{E}$ is homomorphic. We say that $\mathcal{E}$ is* bootstrappable *with respect to Γ if*

$$D_{\mathcal{E}}(\Gamma) \subseteq \mathcal{C}_{\mathcal{E}} \ .$$

For a family of schemes $\{\mathcal{E}^{(d)}\}$ derived from $\mathcal{E}$ that we will define momentarily, we have the following theorems.

Theorem 3. *Let $\mathcal{E}$ be bootstrappable with respect to a set of gates Γ. Then the family $\{\mathcal{E}^{(d)}\}$ is leveled fully homomorphic (for circuits with gates in Γ).*

Theorem 4. *Let $\mathcal{A}$ be an algorithm that breaks the semantic security of $\mathcal{E}^{(d)}$ with advantage ϵ. Then, there is an algorithm $\mathcal{B}$ that breaks the semantic security of $\mathcal{E}$ with probability at least $\epsilon/\ell(d+1)$, for ℓ as in Definition 4, and time* $\mathrm{poly}(\ell, d)$ *times that of $\mathcal{A}$.*

Note that Γ is arbitrary in Theorem 3. For example, each gate in Γ could be a circuit of (ANDs, ORs, NOTs) of depth m and fan-in 2; for this value of Γ, Theorem 3 implies the scheme correctly evaluates "normal" circuits of depth $d \cdot m$.

Now we specify the leveled fully homomorphic scheme. Let $\mathcal{E}$ be bootstrappable with respect to a set of gates Γ. For any integer $d \geq 1$, we use $\mathcal{E}$ to construct a scheme $\mathcal{E}^{(d)} = (\mathsf{KeyGen}_{\mathcal{E}^{(d)}}, \mathsf{Encrypt}_{\mathcal{E}^{(d)}}, \mathsf{Evaluate}_{\mathcal{E}^{(d)}}, \mathsf{Decrypt}_{\mathcal{E}^{(d)}})$ that can handle all circuits of depth d with gates in Γ. When we refer to the level of a wire in C, we mean the level of the gate for which the wire is an input. We use the notation $D_{\mathcal{E}}(\Gamma, \delta)$ to refer to the set of circuits that equal a δ-depth circuit with gates in Γ augmented by $D_{\mathcal{E}}$ (copies of $D_{\mathcal{E}}$ become inputs to the δ-depth circuit).

$\mathsf{KeyGen}_{\mathcal{E}^{(d)}}(\lambda, d)$. Takes as input a security parameter λ and a positive integer d. For $\ell = \ell(\lambda)$ as in Definition 4, it sets

$$(\mathrm{sk}_i, \mathrm{pk}_i) \xleftarrow{R} \mathsf{KeyGen}_{\mathcal{E}}(\lambda) \qquad \text{for } \ i \in [0, d]$$
$$\overline{\mathrm{sk}_{ij}} \xleftarrow{R} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}_{i-1}, \mathrm{sk}_{ij}) \qquad \text{for } \ i \in [1, d], j \in [1, \ell]$$

where $\mathrm{sk}_{i1}, \ldots, \mathrm{sk}_{i\ell}$ is the representation of sk_i as elements of $\mathcal{P}$. It outputs the secret key $\mathrm{sk}^{(d)} \leftarrow \mathrm{sk}_0$ and the public key $\mathrm{pk}^{(d)} \leftarrow (\langle \mathrm{pk}_i \rangle, \langle \overline{\mathrm{sk}_{ij}} \rangle)$. For $\delta \leq d$, let $\mathcal{E}^{(\delta)}$ refer to the scheme using $\mathrm{sk}^{(\delta)} \leftarrow \mathrm{sk}_0$ and $\mathrm{pk}^{(\delta)} \leftarrow (\langle \mathrm{pk}_i \rangle_{i \in [0,\delta]}, \langle \overline{\mathrm{sk}_{ij}} \rangle_{i \in [1,\delta]})$.

$\mathsf{Encrypt}_{\mathcal{E}^{(d)}}(\mathrm{pk}^{(d)}, \pi)$. Takes as input a public key $\mathrm{pk}^{(d)}$ and a plaintext $\pi \in \mathcal{P}$. It outputs a ciphertext $\psi \xleftarrow{R} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}_d, \pi)$.

$\mathsf{Decrypt}_{\mathcal{E}^{(d)}}(\mathrm{sk}^{(d)}, \psi)$. Takes as input a secret key $\mathrm{sk}^{(d)}$ and a ciphertext ψ (which should be an encryption under pk_0). It outputs $\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}_0, \psi)$.

$\mathsf{Evaluate}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C_\delta, \Psi_\delta)$. Takes as input a public key $\mathrm{pk}^{(\delta)}$, a circuit C_δ of depth at most δ with gates in Γ, and a tuple of input ciphertexts Ψ_δ (where each input ciphertext should be under pk_δ). We assume that each wire in C_δ connects gates at consecutive levels; if not, add identity gates to make it so. If $\delta = 0$, it outputs Ψ_0 and terminates. Otherwise, it does the following:

- Sets $(C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1}) \leftarrow \mathsf{Augment}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C_\delta, \Psi_\delta)$.
- Sets $(C_{\delta-1}, \Psi_{\delta-1}) \leftarrow \mathsf{Reduce}_{\mathcal{E}^{(\delta-1)}}(\mathrm{pk}^{(\delta-1)}, C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1})$.
- Runs $\mathsf{Evaluate}_{\mathcal{E}^{(\delta-1)}}(\mathrm{pk}^{(\delta-1)}, C_{\delta-1}, \Psi_{\delta-1})$.

$\mathsf{Augment}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C_\delta, \Psi_\delta)$. Takes as input a public key $\mathrm{pk}^{(\delta)}$, a circuit C_δ of depth at most δ with gates in Γ, and a tuple of input ciphertexts Ψ_δ (where each input ciphertext should be under pk_δ). It augments C_δ with $D_{\mathcal{E}}$; call the resulting circuit $C^\dagger_{\delta-1}$. Let $\Psi^\dagger_{\delta-1}$ be the tuple of ciphertexts formed by replacing each input ciphertext $\psi \in \Psi_\delta$ by the tuple $\langle \langle \overline{\mathrm{sk}_{\delta j}} \rangle, \langle \overline{\psi_j} \rangle \rangle$, where $\overline{\psi_j} \leftarrow \mathsf{Encrypt}_{\mathcal{E}^{(\delta-1)}}(\mathrm{pk}^{(\delta-1)}, \psi_j)$ and the ψ_j's form the properly-formatted representation of ψ as elements of $\mathcal{P}$. It outputs $(C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1})$.

$\mathsf{Reduce}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C^\dagger_\delta, \Psi^\dagger_\delta)$. Takes as input a public key $\mathrm{pk}^{(\delta)}$, a tuple of input ciphertexts $\Psi^\dagger_\delta$ (where each input ciphertext should be an output of $\mathsf{Encrypt}_{\mathcal{E}^{(\delta)}}$), and a circuit $C^\dagger_\delta \in D_{\mathcal{E}}(\Gamma, \delta+1)$. It sets C_δ to be the sub-circuit of $C^\dagger_\delta$ consisting of the first δ levels. It sets Ψ_δ to be the induced input

ciphertexts of C_δ, where the ciphertext $\psi_\delta^{(w)}$ associated to wire w at level δ is set to $\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}_\delta, C_\delta^{(w)}, \Psi_\delta^{(w)})$, where $C_\delta^{(w)}$ is the sub-circuit of $C_\delta^\dagger$ with output wire w, and $\Psi_\delta^{(w)}$ are the input ciphertexts for $C_\delta^{(w)}$. It outputs (C_δ, Ψ_δ).

Regarding the computational complexity of $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$:

THEOREM 5. *For a circuit C of depth at most d and size s (defined here as the number of wires), the computational complexity of applying $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$ to C is dominated by at most $s \cdot \ell \cdot d$ applications of $\mathsf{Encrypt}_{\mathcal{E}}$ and at most $s \cdot d$ applications of $\mathsf{Evaluate}_{\mathcal{E}}$ to circuits in $D_{\mathcal{E}}(\Gamma)$, where ℓ is as in Definition 4.*

As mentioned above, if $\mathcal{E}$ is KDM-secure, we can shorten the public key to include pk and and an encryption of sk under pk, a "self-loop" rather than an acyclic chain of encrypted secret keys.[3] This scheme is fully homomorphic, with security following from KDM-security, and correctness following from correctness of the above scheme.

3. AN INITIAL CONSTRUCTION

Our goal now is to construct an encryption scheme that is bootstrappable with respect to a universal set of gates Γ – e.g., where Γ includes NAND and the trivial gate. Here, we describe an initial attempt. In Section 4, we describe some tweaks to this construction and find that the decryption complexity is still too large to permit bootstrappability. We finally achieve bootstrappability in Section 5.

3.1 The Initial Construction, Abstractly

We start by describing our initial attempt simply in terms of rings and ideals; we bring in ideal lattices later. In our initial scheme $\mathcal{E}$, we use a fixed ring R that is set appropriately according to a security parameter λ. We also use a fixed basis $\mathbf{B}_I$ of a ideal $I \subset R$, and an algorithm $\mathsf{IdealGen}(R, \mathbf{B}_I)$ that outputs public and secret bases $\mathbf{B}_J^{\mathrm{pk}}$ and $\mathbf{B}_J^{\mathrm{sk}}$ of some (variable) ideal J, such that $I + J = R$ – i.e., I and J are relatively prime. (Actually, $\mathbf{B}_J^{\mathrm{sk}}$ can be a basis of a (fractional) ideal that contains J, rather than of J.) We assume that if $\mathbf{t} \in R$ and $\mathbf{B}_M$ is a basis for ideal $M \subset R$, then the value $\mathbf{t} \bmod \mathbf{B}_M$ is unique and can be computed efficiently – i.e., the coset $\mathbf{t} + M$ has a unique, efficiently-computable "distinguished representative" with respect to the basis $\mathbf{B}_M$. We use the notation $R \bmod \mathbf{B}_M$ to denote the set of distinguished representatives of $r + M$ over $r \in R$, with respect to the particular basis $\mathbf{B}_M$ of M. We also use an algorithm $\mathsf{Samp}(\mathbf{x}, \mathbf{B}_I, R, \mathbf{B}_J)$ that samples from the coset $\mathbf{x} + I$.

In the scheme, $\mathsf{Evaluate}$ takes as input a circuit C whose gates perform operations modulo $\mathbf{B}_I$. For example, an $\mathsf{Add}_{\mathbf{B}_I}$ gate in C takes two terms in $R \bmod \mathbf{B}_I$, and outputs a third term in $R \bmod \mathbf{B}_I$, which equals the sum of the first two terms modulo I.

$\mathsf{KeyGen}(R, \mathbf{B}_I)$. Takes as input a ring R and basis $\mathbf{B}_I$ of I. It sets $(\mathbf{B}_J^{\mathrm{sk}}, \mathbf{B}_J^{\mathrm{pk}}) \xleftarrow{R} \mathsf{IdealGen}(R, \mathbf{B}_I)$. The plaintext space $\mathcal{P}$ is (a subset of) $R \bmod \mathbf{B}_I$. The public key pk includes R, $\mathbf{B}_I$, $\mathbf{B}_J^{\mathrm{pk}}$, and Samp. The secret key sk also includes $\mathbf{B}_J^{\mathrm{sk}}$.

[3]Initially, we used $\mathcal{E}^*$ (with the self-loop) and tried to prove KDM-security. Canetti [15] proposed the acyclic, leveled approach to make this unnecessary.

$\mathsf{Encrypt}(\mathrm{pk}, \pi)$. Takes as input the public key pk and plaintext $\pi \in \mathcal{P}$. It sets $\psi' \leftarrow \mathsf{Samp}(\pi, \mathbf{B}_I, R, \mathbf{B}_J^{\mathrm{pk}})$ and outputs $\psi \leftarrow \psi' \bmod \mathbf{B}_J^{\mathrm{pk}}$.

$\mathsf{Decrypt}(\mathrm{sk}, \psi)$. Takes as input the secret key sk and a ciphertext ψ. It outputs

$$\pi \leftarrow (\psi \bmod \mathbf{B}_J^{\mathrm{sk}}) \bmod \mathbf{B}_I$$

$\mathsf{Evaluate}(\mathrm{pk}, C, \Psi)$. Takes as input the public key pk, a circuit C in some permitted set $\mathcal{C}_{\mathcal{E}}$ of circuits composed of $\mathsf{Add}_{\mathbf{B}_I}$ and $\mathsf{Mult}_{\mathbf{B}_I}$ gates and a set of input ciphertexts Ψ. It invokes Add and Mult, given below, in the proper sequence to compute the output ciphertext ψ. (We will describe $\mathcal{C}_{\mathcal{E}}$ when we consider correctness below. If desired, one could use different arithmetic gates.)

$\mathsf{Add}(\mathrm{pk}, \psi_1, \psi_2)$. Outputs $\psi_1 + \psi_2 \bmod \mathbf{B}_J^{\mathrm{pk}}$.

$\mathsf{Mult}(\mathrm{pk}, \psi_1, \psi_2)$. Outputs $\psi_1 \times \psi_2 \bmod \mathbf{B}_J^{\mathrm{pk}}$.

Now, let us consider correctness, which is a highly nontrivial issue in this paper. The following definitions provide structure for our analysis.

To begin, we observe that the scheme is actually using two different circuits. First, $\mathsf{Evaluate}$ takes a mod-$\mathbf{B}_I$ circuit C as input. This circuit is implicitly applied to plaintexts. Second, $\mathsf{Evaluate}$ applies a circuit related to C, which we call the *generalized circuit*, to the ciphertexts; this circuit uses the ring operations (not modulo I).

DEFINITION 6 (GENERALIZED CIRCUIT). *Let C be a mod-$\mathbf{B}_I$ circuit. We say generalized circuit $g(C)$ of C is the circuit formed by replacing C's $\mathsf{Add}_{\mathbf{B}_I}$ and $\mathsf{Mult}_{\mathbf{B}_I}$ operations with addition '+' and multiplication '×' in the ring R.*

Here are a few more definitions relevant to Theorem 6 below, which concerns correctness.

DEFINITION 7 (X_{Enc} AND X_{Dec}). *Let X_{Enc} be the image of Samp. Notice that all ciphertexts output by $\mathsf{Encrypt}$ are in $X_{\mathsf{Enc}} + J$. Let X_{Dec} equal $R \bmod \mathbf{B}_J^{\mathrm{sk}}$, the distinguished representatives of cosets of J wrt the secret basis $\mathbf{B}_J^{\mathrm{sk}}$.*

DEFINITION 8 (PERMITTED CIRCUITS). *Let*

$$\mathcal{C}_{\mathcal{E}}{}' = \{C \ : \ \forall (x_1, \ldots, x_t) \in {X_{\mathsf{Enc}}}^t, g(C)(x_1, \ldots, x_t) \in X_{\mathsf{Dec}}\}$$

In other words, $\mathcal{C}_{\mathcal{E}}{}'$ is the set of mod-$\mathbf{B}_I$ circuits that, when generalized, the output is always in X_{Dec} if the inputs are in X_{Enc}. (The value t will of course depend on C.) If $\mathcal{C}_{\mathcal{E}} \subseteq \mathcal{C}_{\mathcal{E}}{}'$, we say that $\mathcal{C}_{\mathcal{E}}$ is a set of permitted circuits.

DEFINITION 9. *ψ is a* valid ciphertext *wrt $\mathcal{E}$ public key* pk *and permitted circuits $\mathcal{C}_{\mathcal{E}}$ if it equals $\mathsf{Evaluate}(\mathrm{pk}, C, \Psi)$ for some $C \in \mathcal{C}_{\mathcal{E}}$, where each $\psi \in \Psi$ is in the image of $\mathsf{Encrypt}$.*

THEOREM 6. *Assume $\mathcal{C}_{\mathcal{E}}$ is a set of permitted circuits containing the identity circuit. $\mathcal{E}$ is correct for $\mathcal{C}_{\mathcal{E}}$ – i.e., $\mathsf{Decrypt}$ correctly decrypts valid ciphertexts.*

PROOF. For $\Psi = \{\psi_1, \ldots, \psi_t\}$, $\psi_k = \pi_k + i_k + j_k$, where $\pi_k \in \mathcal{P}$, $i_k \in I$, $j_k \in J$, and $\pi_k + i_k \in X_{\mathsf{Enc}}$, we have

$$\begin{aligned} \mathsf{Evaluate}(\mathrm{pk}, C, \Psi) &= g(C)(\Psi) \bmod \mathbf{B}_J^{\mathrm{pk}} \\ &\in g(C)(\pi_1 + i_1, \ldots, \pi_t + i_t) + J \end{aligned}$$

If $C \in \mathcal{C}_\mathcal{E}$, we have $g(C)(X_{\mathsf{Enc}}, \ldots, X_{\mathsf{Enc}}) \in X_{\mathsf{Dec}}$; so,

$$\begin{aligned}&\mathsf{Decrypt}(\mathrm{sk}, \mathsf{Evaluate}(\mathrm{pk}, C, \Psi))\\ &\quad = g(C)(\pi_1 + i_1, \ldots, \pi_t + i_t) \bmod \mathbf{B}_I\\ &\quad = g(C)(\pi_1, \ldots, \pi_t) \bmod \mathbf{B}_I\\ &\quad = C(\pi_1, \ldots, \pi_t)\end{aligned}$$

as required. □

The bottom line is that we have proven that $\mathcal{E}$ is correct for permitted circuits, and our goal now is to maximize this set. The permitted circuits are defined somewhat indirectly; they are the circuits for which the "error" $g(C)(x_1, \ldots, x_t)$ of the output ciphertext is small (i.e., lies inside X_{Dec}) when the input ciphertexts are in the image of $\mathsf{Encrypt}_\mathcal{E}$. When we begin to instantiate the abstract scheme with lattices and give geometric interpretations of X_{Enc} and X_{Dec}, the problem of maximizing $\mathcal{C}_\mathcal{E}$ will have a geometric flavor.

3.2 Security of the Abstract Construction

For an abstract "instantiation" of Samp that we describe momentarily, we provide a simple proof of semantic security based on the following abstract problem.

Definition 10 (Ideal Coset Problem (ICP)). *Fix R, $\mathbf{B}_I$, algorithm $\mathsf{IdealGen}$, and an algorithm Samp_1 that efficiently samples R. The challenger sets $b \xleftarrow{R} \{0,1\}$ and $(\mathbf{B}_J^{\mathrm{sk}}, \mathbf{B}_J^{\mathrm{pk}}) \xleftarrow{R} \mathsf{IdealGen}(R, \mathbf{B}_I)$. If $b = 0$, it sets $\mathbf{r} \xleftarrow{R} \mathsf{Samp}_1(R)$ and $\mathbf{t} \leftarrow \mathbf{r} \bmod \mathbf{B}_J^{\mathrm{pk}}$. If $b = 1$, it samples $\mathbf{t}$ uniformly from $R \bmod \mathbf{B}_J^{\mathrm{pk}}$. The problem: guess b given $(\mathbf{t}, \mathbf{B}_J^{\mathrm{pk}})$.*

Basically the ICP asks one to decide whether $\mathbf{t}$ is uniform modulo J, or whether it was chosen according to a known "clumpier" distribution induced by Samp_1.

To get a proof based on the ICP, define Samp as follows. Let $\mathbf{s} \in I$ be such that the ideal $(\mathbf{s})$ is relatively prime to J. (Assume for now that such an $\mathbf{s}$ can be efficiently generated.)

$\mathsf{Samp}(\mathbf{x}, \mathbf{B}_I, R, \mathbf{B}_J^{\mathrm{pk}})$. Run $\mathbf{r} \xleftarrow{R} \mathsf{Samp}_1(R)$. Output $\mathbf{x} + \mathbf{r} \times \mathbf{s}$.

Obviously, the output is in $\mathbf{x} + I$ since $\mathbf{s} \in I$. The value $\mathbf{s}$ could be a fixed parameter wired into the algorithm Samp, or generated dynamically from $\mathbf{B}_I$ and $\mathbf{B}_J^{\mathrm{pk}}$ according to a prescribed distribution.

Theorem 7. *Suppose that there is an algorithm $\mathcal{A}$ that breaks the semantic security of $\mathcal{E}$ with advantage ϵ when it uses Samp. Then, there is an algorithm $\mathcal{B}$, running in about the same time as $\mathcal{A}$, that solves the ICP with advantage $\epsilon/2$.*

Proof. The challenger sends $\mathcal{B}$ a ICP instance $(\mathbf{t}, \mathbf{B}_J^{\mathrm{pk}})$. $\mathcal{B}$ sets $\mathbf{s}$, and sets the other components of pk in the obvious way using the ICP instance. When $\mathcal{A}$ requests a challenge ciphertext on one of $\pi_0, \pi_1 \in \mathcal{P}$, $\mathcal{B}$ sets a bit $\beta \xleftarrow{R} \{0,1\}$ and sends back $\psi \leftarrow \pi_\beta + \mathbf{t} \times \mathbf{s} \bmod \mathbf{B}_J^{\mathrm{pk}}$. $\mathcal{A}$ sends back a guess β', and $\mathcal{B}$ guesses $b' \leftarrow \beta \oplus \beta'$.

If $b = 0$, we claim that $\mathcal{B}$'s simulation is perfect; in particular, the challenge ciphertext has the correct distribution. When $b = 0$, we have that $\mathbf{t} = \mathbf{r} + j$, where $\mathbf{r}$ was chosen according to Samp_1 and $j \in J$. So, $\psi \leftarrow \pi_\beta + \mathbf{t} \times \mathbf{s} = \pi_\beta + \mathbf{r} \times \mathbf{s} \bmod \mathbf{B}_J^{\mathrm{pk}}$; the ciphertext is thus well-formed. In this case $\mathcal{A}$ should have advantage ϵ, which translates into an advantage of ϵ for $\mathcal{B}$.

If $b = 1$, then $\mathbf{t}$ is uniformly random modulo J. Since the ideal $(\mathbf{s})$ is relatively prime to J, $\mathbf{t} \times \mathbf{s}$ is uniformly random modulo J, and consequently ψ is a uniformly random element of $R \bmod \mathbf{B}_J^{\mathrm{pk}}$ that is independent of β. In this case $\mathcal{A}$'s advantage is 0. Overall, $\mathcal{B}$'s advantage is $\epsilon/2$. □

3.3 Background on Ideal Lattices

Let $R = \mathbb{Z}[x]/f(x)$, where $f(x) \in \mathbb{Z}[x]$ is monic and of degree n. We view an element $\mathbf{v} \in R$ both as a ring element and *as a vector* – specifically, the coefficient vector $\mathbf{v} \in \mathbb{Z}^n$. The ideal $(\mathbf{v})$ generated by $\mathbf{v}$ directly corresponds to the lattice generated by the column vectors $\{\mathbf{v} \times x^i \bmod f(x) : i \in [0, n-1]\}$; we call this the *rotation basis* of the *ideal lattice* $(\mathbf{v})$. Generally speaking, an ideal $I \subset R$ need not be *principal* – i.e., have a single generator – and a basis $\mathbf{B}_I$ of I need not be a rotation basis. R corresponds to the ideal (1) or $(\mathbf{e}_1)$, and to $\mathbb{Z}^n$. The Hermite normal form (HNF) of a lattice I is an upper triangular basis that can be efficiently computable from any other basis of I, making it well-suited to be a public key [39]. The index $[R : I]$ equals $|\det(\mathbf{B}_I)|$; since this is invariant, we refer to $\det(I)$.

As required by our abstract scheme, for any basis $\mathbf{B}_I$ of an ideal $I \subseteq R$, the term $\mathbf{t} \bmod \mathbf{B}_I$ is uniquely defined as the vector $\mathbf{t}' \in \mathcal{P}(\mathbf{B}_I)$ such that $\mathbf{t} - \mathbf{t}' \in I$, where $\mathcal{P}(\mathbf{B}_I)$ is the half-open parallelepiped $\mathcal{P}(\mathbf{B}) \leftarrow \{\sum_{i=1}^n x_i \mathbf{b}_i : x_i \in [-1/2, 1/2)\}$. The vector $\mathbf{t} \bmod \mathbf{B}_I$ is efficiently computable as $\mathbf{t} - \mathbf{B} \cdot \lfloor \mathbf{B}^{-1} \cdot \mathbf{t} \rceil$, where $\lfloor \cdot \rceil$ rounds the coefficients of a vector to the nearest integer. We define the length $\|\mathbf{B}_I\|$ of a basis to be $\max\{\|\mathbf{b}_i\| : \mathbf{b}_i \in \mathbf{B}_I\}$.

Cryptographic work on ideal lattices includes NTRU [25] and more recent work [41, 32, 33, 48, 49].

3.4 The Initial Construction, Concretely

When we implement the abstract construction using a polynomial ring and ideal lattices as described in Section 3.3, the sets X_{Enc} and X_{Dec} become subsets of $\mathbb{Z}^n$. We recharacterize these sets geometrically as follows.

Definition 11 (r_{Enc} and r_{Dec}). *Let r_{Enc} be the smallest value such that $X_{\mathsf{Enc}} \subseteq \mathcal{B}(r_{\mathsf{Enc}})$, where $\mathcal{B}(r)$ is the ball of radius r. Let r_{Dec} be the largest such that $X_{\mathsf{Dec}} \supseteq \mathcal{B}(r_{\mathsf{Dec}})$.*

Now, let us define a set of permitted circuits $\mathcal{C}_\mathcal{E}$ as follows:

$$\mathcal{C}_\mathcal{E} = \{C \ : \ \forall (x_1, \ldots, x_t) \in \mathcal{B}(r_{\mathsf{Enc}})^t, g(C)(x_1, \ldots, x_t) \in \mathcal{B}(r_{\mathsf{Dec}})\}$$

$\mathcal{C}_\mathcal{E}$ is defined like the maximal set $\mathcal{C}_\mathcal{E}'$ of permitted circuits in Definition 8, but we have replaced X_{Enc} and X_{Dec} with $\mathcal{B}(r_{\mathsf{Enc}})$ and $\mathcal{B}(r_{\mathsf{Dec}})$. Clearly, $\mathcal{C}_\mathcal{E} \subseteq \mathcal{C}_\mathcal{E}'$. (At several points later in the paper, we narrow our set of permitted circuits again so as to enable a less complex decryption algorithm.)

For fixed values of r_{Enc} and r_{Dec}, what is $\mathcal{C}_\mathcal{E}$? This is a geometric problem, and we can bound the Euclidean length $\|g(C)(\mathbf{x}_1, \ldots, \mathbf{x}_t)\|$ by bounding the lengths of $\|\mathbf{u} + \mathbf{v}\|$ and $\|\mathbf{u} \times \mathbf{v}\|$ in terms of $\|\mathbf{u}\|$ and $\|\mathbf{v}\|$. For addition, this is easy: using the triangle inequality, we have $\|\mathbf{u} + \mathbf{v}\| \le \|\mathbf{u}\| + \|\mathbf{v}\|$ for $\mathbf{u}, \mathbf{v} \in R$. For multiplication, we can prove that $\|\mathbf{u} \times \mathbf{v}\| \le \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| \cdot \|\mathbf{v}\|$, where $\gamma_{\mathsf{Mult}}(R)$ is some factor that is dependent only on the ring R. (See [32] for a different definition of the expansion factor for multiplication.)

The following theorem characterizes the "error expansion" that a circuit can cause based on the circuit's depth.

Theorem 8. *Suppose $r_{\mathsf{E}} \ge 1$ and that circuit C's additive fan-in is $\gamma_{\mathsf{Mult}}(R)$, multiplicative is 2, and depth is at most*

$$\log\log r_{\mathsf{D}} - \log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{E}})$$

Then, $C(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in \mathcal{B}(r_{\mathsf{D}})$ for all $\mathbf{x}_1, \ldots, \mathbf{x}_t \in \mathcal{B}(r_{\mathsf{E}})$.

PROOF. For a d-depth circuit, let r_i be an upper-bound on the Euclidean norm of the values at level i, given that $r_d = r_{\mathsf{E}}$. By the triangle inequality, an addition (or subtraction) gate at level i outputs some $\mathbf{v} \in R$ such that $\|\mathbf{v}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i$. A multiplication gate at level i outputs some $\mathbf{v} \in R$ such that $\|\mathbf{v}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i^2$. In either case, $r_{i-1} \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i^2$, and thus $r_0 \leq (\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{E}})^{2^d}$. The result follows. □

An (oversimplified) bottom line from Theorem 8 is that, to maximize the depth of circuits that $\mathcal{E}$ can correctly evaluate (see Theorem 6), we should minimize $\gamma_{\mathsf{Mult}}(R)$ and r_{Enc}, and maximize r_{Dec}. Most of the remainder of this subsection consists of proposals toward this goal.

First, though, we observe that semantic security places a limit on how large we can take $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$. In concrete terms, the ICP becomes (roughly): decide whether $\mathbf{t}$ is within a small distance (less than r_{Enc}) of the lattice J (according to a distribution determined by Samp_1), or is uniformly random modulo J. This is a fairly natural decisional version of the closest vector problem, though we are unaware of an equivalent problem mentioned in the literature. Certainly $r_{\mathsf{Dec}} < \lambda_1(J)$, the latter being the length of the shortest nonzero vector in J. On other hand, we cannot let $\lambda_1(J)/r_{\mathsf{Enc}}$ be too large (e.g., 2^n); otherwise, lattice reduction techniques, such as LLL [28, 5, 57], make it feasible to recover the J-vector closest to $\mathbf{t}$, which breaks the ICP. However, given known attacks, this problem is infeasible for a 2^{n^c} approximation factor when $c < 1$. Overall, the ratio $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$ can also be sub-exponential. When $r_{\mathsf{Dec}} = 2^{n^{c_1}}$ and $\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}} = 2^{n^{c_2}}$, then Theorems 6 and 8 imply that the scheme can correctly evaluate circuits of depth $(c_1 - c_2) \log n$.

Regarding $\gamma_{\mathsf{Mult}}(R)$, there are many f for which $\gamma_{\mathsf{Mult}}(R)$ is only polynomial in n:

THEOREM 9. *Let $R = \mathbb{Z}[x]/f(x)$ and suppose $f(x) = x^n - h(x)$ where $h(x)$ has degree at most $n - (n-1)/k$ for $k \geq 2$. Then, $\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot (\sqrt{(k-1)n}\|f\|)^k)$.*

As an example, $f(x) = x^n \pm 1$ achieves $\gamma_{\mathsf{Mult}}(R) = \sqrt{n}$. Aside from Theorem 9, we do not specify how to choose R; let us assume it is chosen so that $\gamma_{\mathsf{Mult}}(R)$ is polynomial in n.

Regarding r_{Enc}, recall that $X_{\mathsf{Enc}} \subseteq \mathcal{B}(r_{\mathsf{Enc}})$ is the image of Samp, where our security proof (Theorem 7) holds when $\mathsf{Samp}(\mathbf{x}, \mathbf{B}_I, R, \mathbf{B}_J^{\mathrm{pk}})$ runs $\mathbf{r} \leftarrow \mathsf{Samp}_1(R)$ and outputs $\mathbf{x} + \mathbf{r} \times \mathbf{s}$, where $\mathbf{s} \in I$ and the ideal $(\mathbf{s})$ is relatively prime to J. For simplicity, suppose that I is the *principal* ideal $(\mathbf{s})$. Let ℓ be an upper bound on the length of $\mathbf{r}$, drawn according to Samp_1. We have

$$r_{\mathsf{Enc}} = \max\{\|\mathbf{x} + \mathbf{r} \times \mathbf{s}\|\} \leq n \cdot \|\mathbf{B}_I\| + \sqrt{n} \cdot \ell \cdot \|\mathbf{B}_I\|$$

Toward minimizing r_{Enc}, we can choose $\mathbf{s}$ to be short – e.g., use $\mathbf{s} = 2 \cdot \mathbf{e}_1$. The size of ℓ is a security issue. We need it to be large enough so that the min-entropy of $\mathbf{t} \bmod \mathbf{B}_J^{\mathrm{pk}}$ in the ICP is large; taking ℓ to be polynomial in n suffices. Overall, we can take r_{Enc} to be polynomial in n. We note that, even in this case, the plaintext space may be as large as $[R : I] = \det(I)$, which can be exponential in n.

Now, r_{Dec}, as the radius of the largest sphere centered at $\mathbf{0}$ that is circumscribed by $\mathbf{B}_J^{\mathrm{sk}}$, obviously depends crucially on the secret basis. The important property of $\mathbf{B}_J^{\mathrm{sk}}$ is its shape – i.e., we want the parallelepiped $\mathcal{P}(\mathbf{B}_J^{\mathrm{sk}})$ to be "fat" enough to contain a large sphere. This property is easier to formalize in terms of the inverse matrix $(\mathbf{B}_J^{\mathrm{sk}})^{-1}$, whose transpose is a basis (or independent set) of the fractional ideal J^{-1}. (The fractional ideal J^{-1} contains the elements in $\mathbb{Q}[x]/f(x)$ that, when multiplied by any element in J, the product is in R.)

LEMMA 1. *Let $\mathbf{B}$ be a lattice basis and $\mathbf{B}^* = (\mathbf{B}^{-1})^T$. Let r be the radius of the largest sphere, centered at $\mathbf{0}$, circumscribed by $\mathcal{P}(\mathbf{B})$ (permitting tangential overlap). Then, $r = 1/(2 \cdot \|\mathbf{B}^*\|)$. In particular,*

$$r_{\mathsf{Dec}} = 1/(2 \cdot \|((\mathbf{B}_J^{\mathrm{sk}})^{-1})^T\|)$$

Suppose $\|\mathbf{t}\| < r$; then each coefficient of $\mathbf{B}^{-1} \cdot \mathbf{t}$ has magnitude at most $1/2$.

PROOF. Each coefficient of $\mathbf{B}^{-1} \cdot \mathbf{t}$ is an inner product of $\mathbf{t}$ with a column vector of $\mathbf{B}^*$, and therefore has magnitude at most $\|\mathbf{t}\| \cdot \|\mathbf{B}^*\| < 1/2$. This implies $\lfloor \mathbf{B}^{-1} \cdot \mathbf{t} \rceil = 0$, and thus $\mathbf{t} = (\mathbf{t} \bmod \mathbf{B})$, and thus $\mathbf{t} \in \mathcal{P}(\mathbf{B})$. Let $\mathbf{v}$ be the longest vector in $\mathbf{B}^*$, and let $\mathbf{x}$ be parallel to $\mathbf{v}$. Then, $\lfloor \mathbf{B}^{-1} \cdot \mathbf{x} \rceil \neq 0$ – i.e., $\mathbf{x} \neq \mathcal{P}(\mathbf{B})$ – when $|\langle \mathbf{v}, \mathbf{x} \rangle| > 1/2 \Leftrightarrow \|\mathbf{x}\| > 1/(2 \cdot \|\mathbf{B}^*\|)$. □

It is easy to imagine ad hoc ways of instantiating $\mathsf{IdealGen}$ – e.g., one could generate a random vector $\mathbf{v}$ and simply set $\mathbf{B}_J^{\mathrm{sk}}$ to be the rotation basis of $\mathbf{v}$, and set $\mathbf{B}_J^{\mathrm{pk}}$ to be the HNF of $(\mathbf{v})$. Very roughly speaking, if $\mathbf{v}$ is generated as a vector that is very "nearly parallel" to $\mathbf{e}_1$ (i.e., the vector $(1, 0, \ldots, 0)$), then the rotational basis will have r_{Dec} within a small factor of $\|\mathbf{v}\|/2$. This method can be made to ensure that r_{Dec} is within a polynomial factor of $\lambda_1(J)$.

On the other hand, one may prefer to generate J according to a "nice" average-case distribution that permits a worst-case / average-case reduction [1, 2, 53]. This is a technical issue that we tackle in a separate work. We stress that our analysis below regarding the decryption circuit does not rely on any of the concrete proposals in this subsection – e.g., the analysis does not require I to be a principal ideal.

4. BOOTSTRAPPABLE YET?

Here, we describe two "tweaks" to the scheme to lower the decryption complexity in preparation for bootstrapping; we label the tweaked scheme $\mathcal{E}_2$. Next, we analyze the decryption complexity (and find that it is too large). However, some of the analysis applies to the final scheme.

We separate $\mathcal{E}_2$ from $\mathcal{E}_1$ because the former already improves upon previous work, irrespective of bootstrapping. The Boneh-Goh-Nissim (BGN) pairing-based cryptosystem [11] was the first to permit efficient evaluation of quadratic formulas that may have an arbitrary number of monomials. However, BGN has a small plaintext space – $\log \lambda$ bits for security parameter λ. $\mathcal{E}_1$ allows both greater multiplicative depth in the circuit (while allowing essentially an arbitrary number of additions) and also a larger plaintext space.

4.1 Some Tweaks to the Initial Construction

In each of the following tweaks, we slightly narrow our set of permitted circuits $\mathcal{C}_{\mathcal{E}}$. Recall from Section 3.4 that $\mathcal{E}_1$'s definition of $\mathcal{C}_{\mathcal{E}}$ used $\mathcal{B}(r_{\mathsf{Enc}})$ and $\mathcal{B}(r_{\mathsf{Dec}})$.

Tweak 1: Redefine the set of permitted circuits $\mathcal{C}_{\mathcal{E}}$, replacing $\mathcal{B}(r_{\mathsf{Dec}})$ with $\mathcal{B}(r_{\mathsf{Dec}}/2)$.

Purpose: To ensure that ciphertext vectors are closer to the lattice J than they strictly need to be, so that we will need less "precision" to ensure the correctness of decryption.

Recall that Decrypt computes $(\psi \bmod \mathbf{B}_J^{\mathrm{sk}}) \bmod \mathbf{B}_I$, where $\psi \bmod \mathbf{B}_J^{\mathrm{sk}} = \psi - \mathbf{B}_J^{\mathrm{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi \rceil$. If we permitted the coefficients of $(\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi$ to be very close to half-integers, we would need high precision to ensure correct rounding. However, after Tweak 1, we have the following lemma:

LEMMA 2. *If ψ is a valid ciphertext after Tweak 1, then each coefficient of $(\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi$ is within 1/4 of an integer.*

PROOF. Observe that $\psi \in \mathcal{B}(r_{\mathsf{Dec}}/2) + J$. Let $\psi = \mathbf{x} + \mathbf{j}$ for $\mathbf{x} \in \mathcal{B}(r_{\mathsf{Dec}}/2)$ and $\mathbf{j} \in J$. We have $(\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi = (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \mathbf{x} + (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \mathbf{j}$, where the former term has coefficients of magnitude at most 1/4 by Lemma 1 and the latter is an integer vector. □

Per Theorem 8, the new maximum evaluation depth of the scheme after Tweak 1 is $\log\log(r_{\mathsf{Dec}}/2) - \log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$, which is less than the original amount by only a sub-constant additive factor.

Tweak 2 is more technical and less "essential" than the first tweak. It replaces matrix-vector multiplications with ring multiplications. This is nice, but the two computations are essentially equally parallelizable, and their circuits have essentially the same depth. So, really, this tweak is primarily useful for reducing the public key size and the per-gate computation during bootstrapping in our ultimate scheme.

Tweak 2: From $\mathbf{B}_I$ and $\mathbf{B}_J^{\mathrm{sk}}$, compute a short vector $\mathbf{v}_J^{\mathrm{sk}} \in J^{-1}$ such that there exists $\mathbf{u} \in 1+I$ with $\mathbf{u} \times (\mathbf{v}_J^{\mathrm{sk}})^{-1} \in 1+I$. Also, redefine $\mathcal{C}_{\mathcal{E}}$ again, now using $\mathcal{B}(2 \cdot r_{\mathsf{Dec}}/(n^{1.5} \cdot \gamma_{\mathsf{Mult}}(R)^2 \cdot \|\mathbf{B}_I\|))$.

Purpose: To modify Decrypt from $\psi - \mathbf{B}_J^{\mathrm{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi \rceil \bmod \mathbf{B}_I$ to the simpler expression $\psi - \lfloor \mathbf{v}_J^{\mathrm{sk}} \times \psi \rceil \bmod \mathbf{B}_I$, where $\mathbf{v}_J^{\mathrm{sk}} \in \mathbb{Q}[x]/f(x)$.

To use both tweaks, we need to reduce the radius of the sphere in Tweak 2 by an additional factor of 2.

Tweak 2 requires us to reduce the permitted distance of ciphertexts from the J-lattice much more than Tweak 1. Still, it does not affect our maximum evaluation depth very much when $\gamma_{\mathsf{Mult}}(R)$ and $\|\mathbf{B}_I\|$ are polynomial in n, and $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$ is super-polynomial.

The following two lemmas say that we can perform Tweak 2 efficiently and that it has the expected effect – i.e., the new Decrypt equation works correctly.

LEMMA 3. *From $\mathbf{B}_I$ and $\mathbf{B}_J^{\mathrm{sk}}$, we can compute in polynomial time a vector $\mathbf{v}_J^{\mathrm{sk}} \in J^{-1}$ and a vector $\mathbf{u} \in 1+I$ such that $\mathbf{u} \times (\mathbf{v}_J^{\mathrm{sk}})^{-1} \in 1+I$. Moreover, $\|\mathbf{v}_J^{\mathrm{sk}}\| \le (n/2) \cdot \gamma_{\mathsf{Mult}}(R) \cdot \|((\mathbf{B}_J^{\mathrm{sk}})^{-1})^T\| \cdot \|\mathbf{B}_I\|$.*

LEMMA 4. *Suppose ψ is a valid ciphertext after Tweak 2 which decrypts to π. Then, $\pi = \psi - \lfloor \mathbf{v}_J^{\mathrm{sk}} \times \psi \rceil \bmod \mathbf{B}_I$.*

4.2 Decryption Complexity of Tweaked Scheme

To decrypt, we compute

$$(\psi - \mathbf{B}_J^{\mathrm{sk1}} \cdot \lfloor \mathbf{B}_J^{\mathrm{sk2}} \cdot \psi \rceil) \bmod \mathbf{B}_I$$

where $\psi \in \mathbb{Z}^n$, $\mathbf{B}_J^{\mathrm{sk1}} \in \mathbb{Z}^{n \times n}$, $\mathbf{B}_J^{\mathrm{sk2}} \in \mathbb{Q}^{n \times n}$, and $\mathbf{B}_I$ is a basis of an ideal I of $R = \mathbb{Z}[x]/f(x)$. From Tweak 1, we have the promise that the coefficients of $\mathbf{B}_J^{\mathrm{sk2}} \cdot \psi$ are all within 1/4 of an integer. Optionally, Tweak 2 ensures that $\mathbf{B}_J^{\mathrm{sk1}}$ is the identity matrix and $\mathbf{B}_J^{\mathrm{sk2}}$ is a rotation matrix. We split the decryption computation into the following sequence of steps:

Step 1: Generate n vectors with sum $\mathbf{B}_J^{\mathrm{sk2}} \cdot \psi$

Step 2: From the n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, generate *integer* vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{n+1}$ with sum $\lfloor \sum \mathbf{x}_i \rceil$.

Step 3: Compute $\pi \leftarrow \psi - \mathbf{B}_J^{\mathrm{sk1}} \cdot (\sum \mathbf{y}_i) \bmod \mathbf{B}_I$

In Step 1, to obtain the n vectors, we perform the multiplications associated to the inner products associated to the matrix-vector multiplication; we do not discuss this step in detail, since (we will find that) Step 2 is already too expensive to permit bootstrappability.

Regarding Step 2, recall that Evaluate takes as input circuits with mod-$\mathbf{B}_I$ gates – i.e., arithmetic gates whose inputs and outputs are in $R \bmod \mathbf{B}_I$. However, in general "base-$\mathbf{B}_I$," it is unclear how to handle the *carries* needed for adding the vectors in Step 2 without resorting to high-degree polynomials that require more multiplicative depth that our initial construction can evaluate. To handle this complication, we use the plaintext space $\mathcal{P} = \{0,1\} \bmod \mathbf{B}_I$, regardless of the underlying ideal I. We *emulate* boolean operations, where carries are constant-depth. (This plaintext space restriction is unnecessary if we use the initial construction without bootstrapping.) We have the following lemma.

LEMMA 5. *For $i \in [1,t]$, let $a_i = (\ldots, a_{i,1}, a_{i,0}, a_{i,-1}, \ldots)$ be a real number given in binary representation mod $\mathbf{B}_I$ with the promise that $\sum_i a_i \bmod 1 \in [-1/4, 1/4]$. There is a mod-$\mathbf{B}_I$ circuit C for generating $t+1$ integers $z_1, \ldots, z_{t+1}$ (also represented in binary) whose sum is $\lfloor \sum_i a_i \rceil$, such that if the generalized circuit $g(C)$'s inputs are in $\mathcal{B}(r_{in})$, then its outputs are in $\mathcal{B}(r_{out})$ for:*

$$r_{out} \le (\gamma_{\mathsf{Mult}}(R) \cdot n \cdot \|\mathbf{B}_I\| \cdot (1 + \gamma_{\mathsf{Mult}}(R) \cdot r_{in})^t \cdot t)^{\mathrm{polylog}(t)}$$

For $\|\mathbf{B}_I\| \le r_{in}$, $t \le n$, and $\gamma_{\mathsf{Mult}}(R) = n^{\Omega(1)}$, we have:

$$r_{out} \le (\gamma_{\mathsf{Mult}}(R) \cdot r_{in})^{t + \mathrm{polylog}(t)}$$

PROOF (FRAGMENT). Let a_i^* be the integer part of a_i and let $a_i^\dagger = (a_{i,-1}, a_{i,-2}, \ldots)$ be the fractional part. Let $T = \lceil \log t \rceil + 2$. Let $b_i = (a_{i,-1}^\dagger, \ldots, a_{i,-T}^\dagger)$. First, we claim that $\lfloor \sum a_i^\dagger \rceil = \lfloor \sum b_i \rceil$. This claim follows from the promise that $\sum_i a_i^\dagger$ is within 1/4 of an integer, and that

$$\left| \sum_i a_i^\dagger - \sum_i b_i \right| = \left| \sum_i^{j \in [T+1, \infty]} 2^{-j} \cdot a_{i,-j} \right| < 1/4$$

Our $t+1$ output integers will be $a_1^*, \ldots, a_t^*, \lfloor \sum b_i \rceil$.

Our strategy for computing $\lfloor \sum b_i \rceil$ is first to compute, for each $j \in [1,T]$, the binary representation c_j of the Hamming weight of $(b_{1,-j}, \ldots, b_{t,-j})$. Then, we finish by computing the sum $\lfloor \sum_{j=1}^{T} 2^{-j} \cdot c_j \rceil$; this latter term is much easier to compute than the original term, since it only consists of T numbers, rather than t.

This strategy is straightforward when $I = (2 \cdot \mathbf{e}_1)$ and the plaintext space is $\{0,1\} \bmod I$. The binary representation of the Hamming weight of $(b_1, \ldots, b_t)$ is given by

$$(e_{2^{\lfloor \log t \rfloor}}(b_1, \ldots, b_t) \bmod 2, \ldots, e_{2^0}(b_1, \ldots, b_t) \bmod 2)$$

where $e_i(x_1, \ldots, x_t)$ is the ith elementary symmetric polynomial over $x_1, \ldots, x_t$. (See Lemma 4 of [14].) These polynomials can be computed in time $\mathrm{poly}(t)$, and the fact that r_{out} is exponential in t comes from the fact that the degree of these polynomials is upper bounded by t. There are minor complications when $I \ne (2 \cdot \mathbf{e}_1)$ (omitted). □

Unfortunately, Step 2 uses $t = n$, implying $r_{\mathsf{Dec}}/r_{\mathsf{Enc}} \geq r_{out}/r_{in} \geq 2^n$, and therefore the above analysis cannot show that the initial construction is both bootstrappable and secure. However, Lemma 5 will be relevant to our final scheme, as will the following lemma regarding Step 3:

LEMMA 6. *Using a constant depth circuit having polynomial fan-in* $\mathsf{Add}_{\mathbf{B}_I}$ *gates and constant fan-in* $\mathsf{Mult}_{\mathbf{B}_I}$ *gates, we can compute* $\psi - \mathbf{B}_J^{\mathrm{sk1}} \cdot (\sum \mathbf{y}_i) \bmod \mathbf{B}_I$ *from a binary representation (using the bits* $\{0,1\} \bmod \mathbf{B}_I$*) of the terms of the expression.*

The proof of Lemma 6 involves converting the binary representation of the terms to a more "natural" mod-$\mathbf{B}_I$ representation, at which point the computation is trivially obviously constant depth. As a toy example for intuition, suppose we have mod-13 gates, where the numbers $0, \ldots, 12$ are represented by 13 different "frequencies" (not in terms of a binary representation), and Add_{13} and Mult_{13} perform addition and multiplication modulo 13 "automatically." To obtain the *natural* representation of r mod 13 as one of the 13 frequencies given the binary representation $...b_1b_0$ (each bit in $\{0,1\}$ mod 13), we precompute the $a_j \leftarrow 2^j \bmod 13$ and output $\mathsf{Add}_{13}(..., \mathsf{Mult}_{13}(a_1, b_1), \mathsf{Mult}_{13}(a_0, b_0))$. This takes constant depth even if r has a polynomial number of bits using polynomial-fan-in Add_{13} gates. Essentially the same considerations apply in the proof of Lemma 6. The simplest case is where $I = (2)$ and the conversion is unnecessary.

5. SQUASHING THE DEC. CIRCUIT

The circuit complexity of decryption in $\mathcal{E}_2$ is too high to permit bootstrapping. The culprit is Step 2 – i.e., adding n numbers. Here, we describe how transform the scheme so that $\mathsf{Decrypt}$ adds only a sub-linear quantity of numbers. Importantly, this transformation does not affect the set of permitted circuits at all. However, the transformation does induce an additional hardness assumption. By choosing parameters appropriately, the scheme becomes bootstrappable.

5.1 The Transformation, Abstractly

At a high level, our transformation works by splitting the initial decryption algorithm into two phases – an initial computationally intensive preprocessing phase performed without the secret key (by the encrypter), followed by a computationally lightweight phase using the secret key (by the decrypter). In short, the encrypter preprocesses its own initial ciphertext, leaving less work for the decrypter to do. Interestingly, this two-phase approach to decryption is precisely what one finds in *server aided cryptography*.

In detail, let $\mathcal{E}^*$ be an initial encryption scheme (which concretely will become $\mathcal{E}_2$). From $\mathcal{E}^*$, we construct a modified scheme $\mathcal{E}$ that uses two new algorithms, $\mathsf{SplitKey}_{\mathcal{E}}$ and $\mathsf{ExpandCT}_{\mathcal{E}}$, that will remain abstract for now.

$\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$. Runs $(\mathrm{pk}^*, \mathrm{sk}^*) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}^*}(\lambda)$ and $(\mathrm{sk}, \tau) \xleftarrow{\mathrm{R}} \mathsf{SplitKey}_{\mathcal{E}}(\mathrm{sk}^*, \mathrm{pk}^*)$. The secret key is sk. The public key pk is (pk^*, τ).

$\mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi)$. Runs $\psi^* \leftarrow \mathsf{Encrypt}_{\mathcal{E}^*}(\mathrm{pk}^*, \pi)$. It then sets ψ to include ψ^* and the output of $\mathsf{ExpandCT}_{\mathcal{E}}(\mathrm{pk}, \psi^*)$. ($\mathsf{ExpandCT}_{\mathcal{E}}$ makes heavy use of tag τ.)

$\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi)$. Uses sk and the expanded ciphertext to decrypt more efficiently. $\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi)$ should work whenever $\mathsf{Decrypt}_{\mathcal{E}^*}(\mathrm{sk}^*, \psi^*)$ works.

$\mathsf{Add}_{\mathcal{E}}(\mathrm{pk}, \psi_1, \psi_2)$. Extracts (ψ_1^*, ψ_2^*) from (ψ_1, ψ_2), computes $\psi^* \leftarrow \mathsf{Add}_{\mathcal{E}^*}(\mathrm{pk}^*, \psi_1^*, \psi_2^*)$, and outputs ψ which includes ψ^* and the output of $\mathsf{ExpandCT}_{\mathcal{E}}(\mathrm{pk}, \psi^*)$. $\mathsf{Mult}_{\mathcal{E}}(\mathrm{pk}, \psi_1, \psi_2)$ is analogous.

The security of the transformation relies on the following problem, which is completely abstract at this point.

DEFINITION 12 ($\mathsf{SplitKey}$ DISTINGUISHING PROBLEM). *The challenger sets* $(\mathrm{sk}^*, \mathrm{pk}^*) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}^*}$ *and* $b \xleftarrow{\mathrm{R}} \{0,1\}$. *If* $b = 0$, *it sets* $(\mathrm{sk}, \tau) \xleftarrow{\mathrm{R}} \mathsf{SplitKey}(\mathrm{sk}^*, \mathrm{pk}^*)$. *If* $b = 1$, *it sets* $(\mathrm{sk}, \tau) \xleftarrow{\mathrm{R}} \mathsf{SplitKey}(\perp, \mathrm{pk}^*)$, *where* $\perp$ *is a special symbol. The problem: guess* b *given* $(\tau, \mathrm{sk}^*, \mathrm{pk}^*)$.

THEOREM 10. *Suppose that there is an algorithm* $\mathcal{A}$ *that breaks the semantic security of* $\mathcal{E}$ *above with advantage* ϵ. *Then, there exist algorithms* $\mathcal{B}_0$ *and* $\mathcal{B}_1$, *running in about the same time as* $\mathcal{A}$, *such that either* $\mathcal{B}_0$*'s advantage against the* $\mathsf{SplitKey}$ *Distinguishing Problem or* $\mathcal{B}_1$*'s advantage against the semantic security of* $\mathcal{E}^*$ *is at least* $\epsilon/3$.

5.2 The Transformation, Concretely

Let $(\mathrm{sk}^*, \mathrm{pk}^*)$ be an $\mathcal{E}_2$ key pair. Let $\gamma_{set}(n)$ and $\gamma_{subset}(n)$ be functions, where the former is $\omega(n)$ and $\mathrm{poly}(n)$ and the latter is $\omega(1)$ and $o(n)$. Here are the concrete instantiations $\mathsf{SplitKey}_{\mathcal{E}}$, $\mathsf{ExpandCT}_{\mathcal{E}}$, and $\mathsf{Decrypt}_{\mathcal{E}}$ used to construct $\mathcal{E}_3$.

$\mathsf{SplitKey}_{\mathcal{E}}(\mathrm{sk}^\dagger, \mathrm{pk}^*)$. Takes as input $\mathrm{sk}^\dagger$, which may be either sk^* or $\perp$. If the former, it extracts the vector $\mathbf{v}_J^{\mathrm{sk}^*}$ from sk^*; if the latter, it sets $\mathbf{v}_J^{\mathrm{sk}^*} \leftarrow \mathbf{0}$. It outputs (sk, τ), where:

- τ is a set of $\gamma_{set}(n)$ vectors $\mathbf{t}_1, \ldots, \mathbf{t}_{\gamma_{set}(n)}$ that are uniformly random in $J^{-1} \bmod \mathbf{B}_I$, except there exists a subset $S \subseteq \{1, \ldots, \gamma_{set}(n)\}$ of cardinality $\gamma_{subset}(n)$ such that $\sum_{i \in S} \mathbf{t}_i \in \mathbf{v}_J^{\mathrm{sk}^*} + I$.
- sk is a matrix $\gamma_{subset}(n) \times \gamma_{set}(n)$ matrix M of 0's and 1's, where $M_{ij} = 1$ iff j is the ith member of S.

$\mathsf{ExpandCT}_{\mathcal{E}}(\mathrm{pk}, \psi^*)$. Outputs $\mathbf{c}_i \leftarrow \mathbf{t}_i \times \psi^* \bmod \mathbf{B}_I$ for $i \in [1, \gamma_{set}(n)]$. These terms are represented in binary using $\{0,1\} \bmod \mathbf{B}_I$, as in $\mathcal{E}_2$.

$\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi)$. Takes as input the secret key sk and a ciphertext ψ. It performs the following steps:

Step 0: Set the vectors $\mathbf{w}_{ij} \leftarrow M_{ij} \cdot \mathbf{c}_j$
Step 1: Set the vectors $\mathbf{x}_i \leftarrow \sum_{j=1}^{\gamma_{set}(n)} \mathbf{w}_{ij}$
Step 2: From $\gamma_{subset}(n)$ vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, generate integer vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{\gamma_{subset}(n)+1}$ with sum $\lfloor \sum \mathbf{x}_i \rceil$.
Step 3: Compute $\pi \leftarrow \psi - (\sum \mathbf{y}_i) \bmod \mathbf{B}_I$

$\mathsf{SplitKey}_{\mathcal{E}}$ splits $\mathbf{v}_J^{\mathrm{sk}^*}$ into τ, which is a set of vectors with a hidden sparse subset whose sum is $\mathbf{v}_J^{\mathrm{sk}^*}$ modulo I, and sk, which is a set of incidence matrix encoding the subset. To generate τ, one may, for example, just set $\mathbf{t}_1, \ldots, \mathbf{t}_{\gamma_{set}(n)-1}$ to be uniformly random vectors in $J^{-1} \cap \mathcal{P}(\mathbf{B}_I)$. Then, one sets $\mathbf{t}_{\gamma_{set}(n)} \leftarrow \mathbf{v}_J^{\mathrm{sk}^*} - \sum_{i=1}^{\gamma_{subset}(n)-1} \mathbf{t}_i \bmod \mathbf{B}_I$. Then one permutes the vectors.

In $\mathsf{Decrypt}_{\mathcal{E}}$, Steps 2 and 3 are as in $\mathcal{E}_2$ (Section 4), with the crucial difference that Step 2 adds only $\gamma_{subset}(n)$ vectors. In Steps 0 and 1, $\mathsf{Decrypt}_{\mathcal{E}}$ sets $\mathbf{x}_i$ to be the $\gamma_{subset}(n)$ "relevant" vectors in $\{\mathbf{c}_i\}$ – i.e., the ones associated to the subset. Correctness should be clear.

Without Tweak 2, we could have instead used a $\gamma_{set}(n)$-sized set of *matrices* with a hidden $\gamma_{subset}(n)$-sized subset whose sum is related to $(\mathbf{B}_J^{\mathrm{sk}})^{-1}$. This would have resulted in a larger public key.

5.3 Bootstrapping Achieved

We analyzed Steps 2 and 3 in Section 4.2. It is obvious that Step 0 requires only constant depth. We claim that Step 1 requires only constant depth, but why? Computing $\sum_{j=1}^{\gamma_{set}(n)} \mathbf{w}_{ij}$ is very cheap because, in the set $\{\mathbf{w}_{ij} : j \in [1, \gamma_{set}(n)\}$, *there is only one nonzero vector.* Therefore, when we add the vectors, no expensive carry operations are required; we simply "XOR" the vectors together using polynomial-fan-in $\mathsf{Add}_{\mathbf{B}_I}$ operations, using constant depth. At last, we have the following theorem.

THEOREM 11. *The scheme $\mathcal{E}_3$ is bootstrappable when*

$$\gamma_{subset}(n) \leq \left(\frac{\log(r_{\mathsf{Dec}}/m)}{\alpha \cdot 2^c \cdot \log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})} \right)$$

where m arises from the re-definition of $\mathcal{C}_{\mathcal{E}}$ in the Tweaks ($m = 2$ when just Tweak 1 is used), $\alpha > 1$ is chosen so that $(\alpha - 1) \cdot \gamma_{subset}(n)$ is greater than the $\mathrm{polylog}(\gamma_{subset}(n))$ term coming from Lemma 5, and c is a constant representing the depth needed in a circuit having $\mathsf{Add}_{\mathbf{B}_I}$ gates with $\gamma_{\mathsf{Mult}}(R) = n^{\Omega(1)}$ fan-in and $\mathsf{Mult}_{\mathbf{B}_I}$ gates with constant fan-in to sequentially perform $\mathsf{Decrypt}_{\mathcal{E}}$ Steps 0, 1, 3, and 4, and a NAND gate.

PROOF. As in the proof of Theorem 8, for a c-level circuit, if the inputs to the generalized circuit are in $\mathcal{B}(r)$, the outputs are in $\mathcal{B}((\gamma_{\mathsf{Mult}}(R) \cdot r)^{2^c})$. Combining with Lemma 5, we have that if the inputs to our generalized NAND-augmented decryption circuit are in $\mathcal{B}(r_{\mathsf{Enc}})$, the output is in

$$(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})^{2^c \cdot (\gamma_{subset}(n) + \mathrm{polylog}(\gamma_{subset}(n)))}$$

The result follows when this value is at most r_{Dec}/m. □

For example, suppose $\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}}$ is polynomial in n, and $r_{\mathsf{Dec}} = 2^{n^C}$ for $C < 1$. In this case, $\gamma_{subset}(n)$ can be polynomial in n (but sub-linear). The constant c is not very large, though in practice one would want to optimize it beyond what we have done.

5.4 On the New Hardness Assumption

Concretely, the SplitKey Distinguishing Problem becomes:

DEFINITION 13 (SplitKey DP, CONCRETE VERSION). *Let $\gamma_{set}(n)$ and $\gamma_{subset}(n)$ be functions as above, and $\mathbf{B}_I$ a basis of an ideal I. The challenger sets $(\mathrm{sk}^*, \mathrm{pk}^*) \xleftarrow{R} \mathsf{KeyGen}_{\mathcal{E}^*}$ and $b \xleftarrow{R} \{0,1\}$, where sk^* includes the secret vector $\mathbf{v}_J^{\mathrm{sk}^*} \in R$. If $b = 1$, it sets $\mathbf{v}_J^{\mathrm{sk}^*} \leftarrow \mathbf{0}$. It sets τ to be a set of $\gamma_{set}(n)$ vectors $\mathbf{t}_1, \ldots, \mathbf{t}_{\gamma_{set}(n)}$ that are uniformly random in $J^{-1} \bmod \mathbf{B}_I$ subject to the constraint that there exists a subset $S \subseteq \{1, \ldots, \gamma_{set}(n)\}$ of cardinality $\gamma_{subset}(n)$ such that $\sum_{i \in S} \mathbf{t}_i \in \mathbf{v}_J^{\mathrm{sk}^*} + I$. The problem: guess b given $(\tau, \mathrm{sk}^*, \mathrm{pk}^*)$.*

This problem is related to the following problem, which has been analyzed in connection with server-aided cryptography [34, 51, 44, 38, 43], and which should not be confused with the low-density knapsack problem [45].

DEFINITION 14 (SPARSE SUBSET SUM PROBLEM (SSSP)). *Let $\gamma_{set}(n)$ and $\gamma_{subset}(n)$ be functions as above. Let q be a positive integer. The challenger sets $b \xleftarrow{R} \{0,1\}$. If $b = 0$ it generates τ as a set of $\gamma_{set}(n)$ integers $\{a_1, \ldots, a_{\gamma_{set}(n)}\}$ in $[-q/2, q/2]$ that are uniformly random, except that there exists a subset $S \subseteq \{1, \ldots, \gamma_{set}(n)\}$ of cardinality $\gamma_{subset}(n)$ such that $\sum_{i \in S} a_i = 0 \bmod q$. If $b = 1$, it sets the elements without the constraint. The problem: guess b given τ.*

Known attacks on the SSSP[51, 44, 38, 43] are feasible only for limited choices of parameters. Some of these [51] (and previous related results [58, 17]) essentially amount to a time-space tradeoff whose complexity is $\gamma_{set}(n)^{O(\gamma_{subset}(n))}$; these only require us to take $\gamma_{subset}(n) = \omega(1)$. Nguyen and Shparlinski [43] present a lattice-based cryptanalysis of the SSSP that provably succeeds with advantage at least

$$1 - (\gamma_{subset}(n)^{\gamma_{set}(n)+2} \cdot \alpha)/q$$

where α is a term that is greater than 1. The attack breaks down when $\gamma_{set}(n)$ exceeds $\log q$.

Though we omit details, current attacks against our concrete version SplitKey DP begin to break down when $\gamma_{set}(n)$ exceeds $\log \det(IJ)$. The intuition is that, once $\gamma_{set}(n)$ is sufficiently large, there will be exponentially many subsets in τ (not necessarily sparse) whose vector sum is congruent to $\mathbf{v}_J^{\mathrm{sk}^*}$; lattice reduction techniques have trouble extracting the sparse subset from among the many subset solutions.

6. PERFORMANCE

Our scheme relies on two assumptions. The first assumption underlies the security $\mathcal{E}_1$ and its approximation factor is mildly impacted by the tweaks of $\mathcal{E}_2$. The second arises from the addition of τ to the public key. Interestingly, the two assumptions counterbalance each other: increasing $\gamma_{subset}(n)$ seems to make the second problem harder, but at the expense of increasing the approximation factor in the first problem, making the first problem easier.

Using a crude analysis, the breaking time for the second problem using known attacks is roughly $2^{\gamma_{subset}(n)}$. (We ignore constants and logarithmic factors in the exponent.) The approximation factor for the first problem is also roughly $2^{\gamma_{subset}(n)}$. Using the rule of thumb that a lattice problem for approximation factor 2^k takes time about $2^{n/k}$, the breaking time for the first problem is roughly $2^{n/\gamma_{subset}(n)}$. Setting $\gamma_{subset}(n) \leftarrow \sqrt{n}$ maximizes the minimal breaking time at about $2^{\sqrt{n}}$. To make this breaking time truly exponential in the security parameter λ, we need $n \approx \lambda^2$.

For each gate, we evaluate the circuit $D_{\mathcal{E}}$ homomorphically. (Actually, there is a tradeoff here, since a "gate" can be taken to be a "normal circuit" of depth greater than 1.) Though it is a shallow circuit, the computational complexity of $D_{\mathcal{E}}$ itself in $\mathcal{E}_3$ (not evaluated homomorphically in the encryption scheme) is quite high (but $\mathrm{poly}(\lambda)$), primarily due to the large unary representation of the secret key in $\mathcal{E}_3$. The fact that decryption is performed homomorphically, with ciphertext elements in $R \bmod \mathbf{B}_J^{\mathrm{pk}}$ being used instead of plaintext elements in $R \bmod \mathbf{B}_I$, multiplies the complexity by another factor that is quasi-linear in $n \cdot \gamma_{subset}(n)$, since roughly $\log \det(IJ) \approx n \cdot \log r_{\mathsf{Dec}}/r_{\mathsf{Enc}} \approx n \cdot \gamma_{subset}(n)$ bits of precision are needed, but again this is polynomial in λ. Two optimizations are: 1) one can reduce the size of the $\mathcal{E}_3$ secret key substantially by allowing a modest increase in the depth of the decryption circuit, and 2) the homomorphic decryption circuit can take the ciphertext bits as input directly [15] (rather than the encrypted ciphertext bits). Making the full scheme practical remains an open problem (though $\mathcal{E}_1$ is quite practical for shallow circuits).

7. CIRCUIT PRIVACY

We omit full details due to lack of space, but mention that one can construct an algorithm $\mathsf{RandomizeCT}_{\mathcal{E}}$ for our $\mathcal{E}_2$ that can be applied to ciphertexts output by $\mathsf{Encrypt}_{\mathcal{E}_2}$ and $\mathsf{Evaluate}_{\mathcal{E}_2}$, respectively, that induces equivalent output distributions. The circuit privacy of $\mathcal{E}_2$ immediately implies the (leveled) circuit privacy of our (leveled) fully homomorphic encryption scheme.

The idea is simple: to construct a *random* encryption ψ' of π from a *particular* encryption ψ of π, we simply add an encryption of 0 that has a *much* larger random "error" vector than ψ – super-polynomially larger, so that the new error vector statistically obliterates all information about ψ's error vector. This entails another re-definition of $\mathcal{C}_{\mathcal{E}}$.

8. ACKNOWLEDGMENTS

We thank Boaz Barak, Dan Boneh, Ran Canetti, Shafi Goldwasser, Iftach Haitner, Yuval Ishai, Tal Rabin, Yael Tauman Kalai, Salil Vadhan, and Brent Waters for very helpful discussions.

9. REFERENCES

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). *STOC '96*, pp. 99–108.
[2] M. Ajtai and C. Dwork. A public key cryptosystem with worst-case / average-case equivalence. *STOC '97*, pp. 284–293.
[3] J.H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. *Eurocrypt '02*, pp. 83–107.
[4] F. Armknecht and A.-R. Sadeghi. A new approach for algebraically homomorphic encryption. Eprint 2008/422.
[5] L. Babai. On Lovász's lattice reduction and the nearest lattice point problem. Combinatorica 6 (1986), 1–14.
[6] D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. *STOC '86*, pp. 1–5.
[7] D. Beaver. Minimal-latency secure function evaluation. *Eurocrypt '00*, pp. 335–350.
[8] J. Benaloh. Verifiable secret-ballot elections. Ph.D. thesis, Yale Univ., Dept. of Comp. Sci., 1988.
[9] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. *SAC '02*, pp. 62–75.
[10] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. *Eurocrypt '98*, pp. 127–144.
[11] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. *TCC '05*, pp. 325–341.
[12] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-Secure Encryption from Decision Diffie-Hellman. *Crypto '08*, pp. 108–125.
[13] D. Boneh and R. Lipton. Searching for Elements in Black-Box Fields and Applications. *Crypto '96*, pp. 283–297.
[14] J. Boyar, R. Peralta, and D. Pochuev. On the Multiplicative Complexity of Boolean Functions over the Basis $(\wedge, \oplus, 1)$. Theor. Comput. Sci. 235(1), pp. 43–57, 2000.
[15] R. Canetti. Personal communication, 2008.
[16] R. Canetti, H. Krawczyk, and J.B. Nielsen. Relaxing chosen-ciphertext security. *Crypto '03*, pp. 565–582.
[17] D. Coppersmith and G. Seroussi. On the minimum distance of some quadratic residue codes. IEEE Trans. Inform. Theory 30 (1984), 407–411.
[18] W. van Dam, S. Hallgren, and L. Ip. Quantum Algorithms for Some Hidden Shift Problems. *SIAM J. Comput.*, v. 36., no. 3, pp. 763–778, 2006.
[19] I. Damgard and M. Jurik. A Length-Flexible Threshold Cryptosystem with Applications. *ACISP '03*, pp. 350–356.
[20] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *Crypto '84*, pp. 469–472.
[21] M. Fellows and N. Koblitz. Combinatorial cryptosystems galore! *Contemporary Mathematics*, v. 168 of *Finite Fields: Theory, Applications, and Algorithms*, FQ2, pp. 51–61, 1993.
[22] S. Goldwasser and D. Kharchenko. Proof of plaintext knowledge for the Ajtai-Dwork cryptosystem. *TCC 2005*, pp. 529–555.
[23] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. *STOC '82*, pp. 365–377.
[24] S. Halevi and H. Krawczyk. Security under key-dependent inputs. *ACM CCS '07*.
[25] J. Hoffstein, J. Silverman, and J. Pipher. NTRU: A Ring Based Public Key Cryptosystem. In *Proc. of ANTS '98*, LNCS 1423, pages 267–288.
[26] Y. Ishai and A. Paskin. Evaluating Branching Programs on Encrypted Data. *TCC '07*.
[27] A. Kawachi, K. Tanaka, K. Xagawa. Multi-bit cryptosystems based on lattice problems. *PKC '07*, pp. 315–329.
[28] A.K. Lenstra, H.W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Math. Ann. 261(4) (1982) 515–534.
[29] F. Levy-dit-Vehel and L. Perret. A Polly Cracker system based on satisfiability. In *Coding, Crypt. and Comb., Prog. in Comp. Sci. and App. Logic*, v. 23, pp. 177–192.
[30] L. Ly. A public-key cryptosystem based on Polly Cracker, Ph.D. thesis, Ruhr-Universität Bochum, Germany, 2002.
[31] L. Ly. Polly two – a new algebraic polynomial-based public-key scheme. AAECC, 17(3-4), 2006.
[32] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. *ICALP '06*.
[33] V. Lyubashevky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. *TCC '08*.
[34] T. Matsumoto, K. Kato, and H. Imai. Speeding up secret computations with insecure auxiliary devices. *Crypto '88*, pp. 497–506.
[35] U. Maurer and D. Raub. Black-Box Extension Fields and the Inexistence of Field-Homomorphic One-Way Permutations. *Asiacrypt '07*, pp. 427–443.
[36] C.A. Melchor, G. Castagnos, and P. Gaborit. Lattice-based homomorphic encryption of vector spaces. ISIT '08, pp. 1858–1862.
[37] C.A. Melchor, P. Gaborit, and J. Herranz. Additive Homomorphic Encryption with t-Operand Multiplications. Eprint 2008/378.
[38] J. Merkle. Multi-round passive attacks on server-aided RSA protocols. *ACM CCS '00*, pp. 102–107.
[39] D. Micciancio. Improving Lattice Based Cryptosystems Using the Hermite Normal Form. *CaLC '01*, pp. 126–145.
[40] D. Micciancio. Improved cryptographic hash functions with worst-case / average-case connection. *STOC '02*, pp. 609–618.
[41] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. *FOCS '02*, pp. 356–365.
[42] D. Naccache and J. Stern. A New Public-Key Cryptosystem Based on Higher Residues. *ACM CCS '98*.
[43] P.Q. Nguyen and I. Shparlinski. On the Insecurity of Some Server-Aided RSA Protocol. *Asiacrypt '01*, pp. 21–35.
[44] P.Q. Nguyen and J. Stern. The Beguin-Quisquater server-aided RSA protocol from Crypto '95 is not secure. *Asiacrypt '98*, pp. 372–379.
[45] A.M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Crypt. and Comp. Num. Th.*, Proc. Sympos. Appl. Math., vol. 42, AMS, 1990, pp. 75–88.
[46] T. Okamoto and Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. *Eurocrypt '98*, pp. 308–318.
[47] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *Eurocrypt '99*, pp. 223–238.
[48] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. *TCC '06*, pp. 145–166.
[49] C. Peikert and A. Rosen. Lattices that Admit Logarithmic Worst-Case to Average-Case Connection Factors. *STOC '07*, pp. 478–487.
[50] C. Peikert and B. Waters. Lossy Trapdoor Functions and Their Applications. *STOC '08*, pp. 187–196.
[51] B. Pfitzmann and M. Waidner. Attacks on protocols for server-aided RSA computation. *Eurocrypt '92*, pp. 153–162.
[52] M. Prabhakaran and M. Rosulek. Homomorphic Encryption with CCA Security. *ICALP '08*.
[53] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *STOC '05*, pp. 84–93.
[54] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pp. 169–180, 1978.
[55] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Comm. of the ACM*, 21:2, pages 120–126, 1978.
[56] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC1. *FOCS '99*, pp. 554–567, 1999.
[57] C.P. Schnorr. A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms. *Theoretical Computer Science*, 53(2-3):201–224, 1987.
[58] D.R. Stinson. Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. Mathematics of Computation, vol. 71, no. 237, pages 379–391, 2001.