HOMOMORPHIC ENCRYPTION: A MATHEMATICAL SURVEY

CRAIG GENTRY

ABSTRACT

If the first thing that comes to mind when you hear the word “encryption” is the Enigma machine, you might think that encryption is complicated and mathematically uninteresting. In fact, many modern encryption systems are quite simple from a mathematical point of view, especially encryption systems that are *homomorphic*. In these systems, the starting point is a homomorphism that respects some binary operation(s), such as $+$ or $\times$. Depicting this homomorphism with a rectangular commutative diagram, the objects on the top level of the diagram are called ciphertexts, and the objects on the bottom level are called messages or plaintexts. The downward arrows in the diagram are the homomorphism, which we call decryption. The rightward arrows are the operation(s). Decryption commutes with the operations. To the commutative diagram we add one extra ingredient, computational complexity. Specifically, we need for it to be *easy* (in the sense of polynomial-time) for anyone to compute the rightward arrows in the diagram, but *hard* to learn how to compute the downward (decryption) arrows except with some special information that we will call a “secret key.” In short, homomorphic encryption is simply a homomorphism that has been “hardened” in the complexity-theoretic sense.
Homomorphic encryption allows anyone to compute on encrypted data, without needing (or being able) to decrypt, has many exciting applications. *Fully homomorphic encryption* (FHE) systems, which allow a rich (functionally complete) set of operations, were finally discovered in 2009. But all of the FHE systems that we have discovered so far follow the same blueprint, and we still wonder whether there are other ways to build FHE.
This survey presents homomorphic encryption from a mathematical point of view, illustrating with several examples how to start from a homomorphism and harden it to make it suitable for cryptography, pointing out pitfalls and attacks to avoid, laying out the current blueprint for FHE, and (I hope) serving as an inspiration and useful guide in the development of new approaches to FHE.

MATHEMATICS SUBJECT CLASSIFICATION 2020

Primary 68P25; Secondary 68Q17, 14G50

KEYWORDS

Homomorphism, cryptography, encryption, complexity theory

INTERNATIONAL CONGRESS OF MATHEMATICIANS 2022 JULY 6–14

1. INTRODUCTION

Let me sketch *homomorphic encryption* in two different ways, a cryptographic way and a mathematical way.

Cryptographically, homomorphic encryption has the usual 3 algorithms of encryption—namely, key generation K, encryption E, and decryption D. Key generation K generates a random key pair (ek, sk), an encryption key and a secret decryption key. In a "symmetric" encryption system, $ek = sk$; in an "asymmetric" encryption system, ek is public (not secret) and does not equal sk. Encryption E is a randomized algorithm that maps a message $m \in \mathcal{M}$ and some randomness to a ciphertext, $c = \mathsf{E}(ek, m, r)$. Decryption D is a deterministic algorithm that recovers a message from a ciphertext, $m = \mathsf{D}(sk, c)$. It should hold that $m = \mathsf{D}(sk, \mathsf{E}(ek, m, r))$ for all key pairs in the image of K, all $m \in \mathcal{M}$, and all randomness r. For the encryption system to be secure, for any two messages $m_0, m_1 \in \mathcal{M}$ chosen by an adversary, it should be computationally "hard" for the adversary to distinguish encryptions of m_0 from encryptions of m_1, even after seeing many encryptions of these and other values. Homomorphic encryption also has a fourth procedure, V, for evaluation. This procedure uses an additional evaluation key evk, which is public. Evaluation V allows anyone to process encrypted data while it remains encrypted, without using the secret decryption key. For example, one might be able to apply some binary operation $\boxplus$ to two ciphertexts to produce a new ciphertext that encrypts the sum of the original two messages. Formally, for some set of binary operations $\mathcal{F}$ associated to the system, the following is true: for any correctly generated key tuple (ek, sk, evk), and for any ciphertexts c_1, c_2 in the image of E under the key tuple, as well as for any $f \in \mathcal{F}$, we have $\mathsf{D}(\mathsf{V}(evk, f, c_1, c_2)) = f(\mathsf{D}(sk, c_1), \mathsf{D}(sk, c_2))$, that is, running V with the function f on two ciphertexts that happen to decrypt to m_1 and m_2 produces a new ciphertext that decrypts to $f(m_1, m_2)$. We say the system is "unbounded" if operations can be applied repeatedly, indefinitely, not only on ciphertexts in the image of E but also ciphertexts in the image of V. The set of ciphertexts should be finite. (Actually, ciphertexts should be compactly expressible, for efficiency). Thus, homomorphic encryption, via the algorithm V, allows the *processing* of data without giving away *access* to the data. The applications are numerous. For example, if the system is unbounded for operations $+$ and $\times$ (i.e., is *fully homomorphic*), you could give your encrypted financial information to an online service, which could prepare a (encrypted) completed tax form for you (which you could then decrypt), without the online service learning any of your private information.

The cryptographic way also emphasizes an approach called *provable security*. In this approach, one invokes a well-established computational assumption, such as the assumption that it is hard to factor large integers.[1] Then, one constructs a cryptosystem, and *proves* that it is secure if your computational assumption is true. Specifically, one shows that if there is an efficient adversary that violates the security of the cryptosystem, then from that adversary one

1 We will discuss computational complexity and security in more detail later, but as a first approximation one can view the notion of computational "hardness" here as requiring at least that $\mathrm{P} \neq \mathrm{NP}$, i.e., that there exist problems for which one can *verify* a solution efficiently if given a *witness*, but not *find* a solution efficiently.

can build an efficient algorithm to solve the assumedly hard problem. Provable security is an elegant and necessary approach that puts cryptography on a firm and rigorous foundation: anyway, about as firm as possible, given that we are not even certain that P $\neq$ NP. The provable security approach performs the essential function of discouraging and weeding out unproven cryptosystems that might look hard to break at first glance, but are usually broken eventually.

Mathematically, the essence of unbounded homomorphic encryption is captured by a commutative diagram

$$\begin{tikzcd}
\mathcal{C}^2 \arrow[r, "\boxplus{,}\boxtimes"] \arrow[d, dashed, "\mathsf{D}(sk{,}\cdot{,}\cdot)"] & \mathcal{C} \arrow[d, dashed, "\mathsf{D}(sk{,}\cdot)"] \\
\mathcal{M}^2 \arrow[r, "+{,}\times"] & \mathcal{M}.
\end{tikzcd}$$

In the diagram, $\mathcal{M}$ is the set of valid messages and $\mathcal{C}$ is the set of valid ciphertexts. The downward arrows are decryption, which I have drawn as dashed since the arrows should be "hard" to compute without the secret decryption key sk. The rightward arrows are binary operations over $\mathcal{M}$ and $\mathcal{C}$, which anyone can compute easily. To make the diagram cleaner, I simply assumed that the binary operations over $\mathcal{M}$ are $+$ and $\times$ (though other possibilities are interesting), and I used $\boxplus$ and $\boxtimes$, instead of the more ponderous $\mathsf{V}(evk, +, \cdot, \cdot)$ and $\mathsf{V}(evk, \times, \cdot, \cdot)$. The diagram displays how decryption commutes with the binary operations: starting from 2 ciphertexts in the upper-left corner, applying componentwise decryption and then $+$ (for example) produces the same result as first applying $\boxplus$ and then decryption. With the dashed arrows, the diagram depicts homomorphic encryption as a rather straightforward marriage of homomorphism and computational complexity.

The mathematical way does not avoid provable security (nor would we wish it to). Also, the commutative diagram does not refer explicitly to K (key generation) or E (encryption). But the diagram, implicit in the dashed arrow, has a lot to say about provable security, K and E. Ciphertexts are preimages of messages under the decryption map. For the system to be secure, for any two messages $m_0, m_1 \in \mathcal{M}$ chosen by an adversary, it should be computationally "hard" for the adversary to distinguish preimages of m_0 from preimages of m_1, even after seeing many (image, preimage) pairs. In particular, it should be hard to distinguish samples from the *kernel* of the decryption map versus samples from all of $\mathcal{C}$. Typically, one proves the security of a homomorphic encryption system by reducing security to precisely that assumption, namely that $\mathcal{C}$ and $\ker(\mathsf{D}(sk, \cdot))$ are hard to distinguish from samples. Similarly, encryption of m, that is, sampling a random preimage of m, often proceeds by picking *some* preimage c_1 of m and then randomizing it by sampling random $c_2 \leftarrow \ker(\mathsf{D}(sk, \cdot))$ and setting $c = c_1 \boxplus c_2$.

There are already many surveys of homomorphic encryption that follow the cryptographic way [1,2,7,11,22,42,43,48,62,72,74].[2] This survey is aimed at mathematicians. So,

2 Silverberg's survey [72] is aimed at mathematicians, but in a different way than I intend here.

our journey will follow the mathematical way, starting always with a homomorphism (rather than a well-established cryptographic assumption), and then seeking ways to “harden” the homomorphism to make it suitable for cryptography. My ulterior motive for following this way is that I want to encourage mathematicians to be creative, to try to introduce new useful algebraic structures into cryptography’s limited repertoire, and to invent new homomorphic encryption systems (subject, eventually, to the constraints of provable security).

Accordingly, the plan of the survey is to be maximally accessible, useful and inspiring to mathematicians, by presenting:

- Several examples of simple homomorphic encryption systems, starting from their homomorphisms, showing how their homomorphisms are “hardened,” and giving their proofs of security (after defining security);
- General results about homomorphic encryption—including “fully” homomorphic encryption (FHE), which allows arbitrary computations to be performed on data while it remains encrypted—most of which follow directly from the commutative diagram defining the system’s correctness;
- Some discussion of why ring homomorphisms do not seem to give secure FHE systems;
- A clear exposition of an actual FHE system, including how we start with a ring homomorphism, how we harden the ring homomorphism by adding “noise,” and how to base the security of the system on a “hard” problem over integer lattices;
- Some discussion of failed attempts to use different algebraic structures to build fully homomorphic encryption systems in a way that falls outside of the current blueprint;
- A mercifully concise discussion of practical matters, such as the performance characteristics of FHE.

By the end, we will see that the algebraic structures underlying current fully homomorphic encryption (FHE) systems are rather bizarre. In known FHE systems, the set of messages $\mathcal{M}$ is a ring with natural $+$ and $\times$. The set of ciphertexts $\mathcal{C}$ has analogous binary operations $\boxplus$ and $\boxtimes$, but is not a ring, but rather a commutative “double magma”—in particular, the binary operations are not even associative. As an algebraic structure, the ciphertexts are very unstructured. It is an intriguing question whether FHE can be built with a set of ciphertexts that is more structured, e.g., a nonsolvable group.

In the next section, we review some simple early homomorphic encryption systems, their commutative diagrams, and their proofs of security. After these examples, we present some general definitions and results about homomorphic encryption in Section 3, most notably the bootstrapping theorems, which demonstrate that to get a homomorphic encryption system capable of correctly evaluating *any function* on encrypted data (that is, an FHE system), it is enough to get a homomorphic encryption system that can correctly evaluate *a single special function*. In Section 4, we describe in detail the construction of an FHE system. The construction starts with a homomorphism that respects a rich set of operations—such as a ring homomorphism—and hardens it by adding "noise" to it. The noise turns the unbounded homomorphism into a bounded one, but the bounded homomorphism is "bootstrappable," as needed to obtain FHE. We show how to base the security of different versions of the FHE system on different versions of the learning with errors (LWE) problem, whose hardness in turn can be based on hard problems over integer lattices. Finally, in Section 5, we suggest directions for future research.

2. SOME SIMPLE HOMOMORPHIC ENCRYPTION SYSTEMS

Here, as a (safely skippable) warm-up, we present some simple homomorphic encryption systems, starting from their homomorphisms, showing how their homomorphisms are "hardened," and giving their proofs of security (after defining security).

First, some history. Rivest, Adleman, and Dertouzos [67] proposed the notion of homomorphic encryption in 1978—calling it a "privacy homomorphism." They were inspired by a homomorphic property of the RSA encryption system, which Rivest, Shamir, and Adleman [68] had proposed the previous year—namely, that if you multiply two ciphertexts encrypted under the same key, it has the effect of multiplying the messages encrypted inside. They wondered whether it was possible to take this further: to construct a privacy homomorphism capable of general computation on encrypted data, not just multiplications modulo an integer. In [67], they proposed several systems allowing general computation. They knew these systems were insecure against realistic attacks—for example, in some of the systems, if you obtain a few encryptions of 0, it becomes trivial to recover the secret key. These systems were inspiring to later researchers, who eventually found ways to modify them to make them secure—in particular, with "noise"—to construct the fully homomorphic encryption systems that we have today.

Fortunately, for the purposes of this survey, we have some simple homomorphic encryption systems that are also provably secure, based on natural computational assumptions, under the "right" model of security for an encryption system. For these examples, we can start with a homomorphism, show how to "harden" it, and provide a proper proof of security in the "right" model of security. In these examples, the proof of security in this model makes heavy use of the homomorphism. In fact, the assumption used in the proof of security is simply that it is computationally hard to distinguish samples from the kernel of the homomorphism from random samples.

Our first example is the Goldwasser–Micali encryption system, described in 1982 [47]. Goldwasser and Micali were the first to prove an encryption system secure under a natural computational assumption using the "right" model of security. Granted, they had an advantage here, because they also *defined* the model of security. But, to their credit, this model has stood the test of time and is still considered the right one.

2.1. Goldwasser–Micali: HE starting from the Legendre symbol

For a fixed prime p, the Legendre symbol $\left(\frac{\cdot}{p}\right) : (\mathbb{Z}/p\mathbb{Z})^\times \to \{\pm 1\}$ is a group homomorphism, mapping an element of $(\mathbb{Z}/p\mathbb{Z})^\times$ to 1 if it is a quadratic residue (square) modulo p, and to -1 if it is a nonresidue. We have the following commutative diagram:

$$\begin{tikzcd}
(\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/p\mathbb{Z})^\times \arrow[r, "\boxtimes"] \arrow[d, "\left(\frac{\cdot}{p}\right)\text{,}\left(\frac{\cdot}{p}\right)"] & (\mathbb{Z}/p\mathbb{Z})^\times \arrow[d, "\left(\frac{\cdot}{p}\right)"] \\
\{\pm 1\} \times \{\pm 1\} \arrow[r, "\times"] & \{\pm 1\},
\end{tikzcd}$$

where $\times$ denotes multiplication in $\{\pm 1\}$, and $\boxtimes$ denotes multiplication in $(\mathbb{Z}/p\mathbb{Z})^\times$.

How can we "harden" the Legendre symbol homomorphism to build a homomorphic encryption system? The downward arrow, which will eventually become decryption, currently requires only knowledge of p, so we must hide p in some way. A natural way to hide p is to reveal only a composite integer $N = p \cdot q$, where p and q are both large prime integers; N hides p only if it is "hard" to recover p from N via factorization, so we will at least need to assume that factorization is hard. We now have the following commutative diagram:

$$\begin{tikzcd}
(\mathbb{Z}/N\mathbb{Z})^\times \times (\mathbb{Z}/N\mathbb{Z})^\times \arrow[r, "\boxtimes"] \arrow[d, dashed, "\left(\frac{\cdot}{p}\right)\text{,}\left(\frac{\cdot}{p}\right)"] & (\mathbb{Z}/N\mathbb{Z})^\times \arrow[d, dashed, "\left(\frac{\cdot}{p}\right)"] \\
\{\pm 1\} \times \{\pm 1\} \arrow[r, "\times"] & \{\pm 1\},
\end{tikzcd}$$

where now $\boxtimes$ is multiplication modulo N, and the downward arrows are dashed because (we hope) it is hard to learn how to compute the Legendre symbol $\left(\frac{\cdot}{p}\right)$ without the secret p, even after seeing many (image, preimage pairs).

For several reasons, it makes sense to use the subgroup of $(\mathbb{Z}/N\mathbb{Z})^\times$, which we will denote by J_N, of elements with Jacobi symbol 1. First, the fact that the Jacobi symbol $\left(\frac{\cdot}{N}\right)$ is efficiently computable even without the factorization of N makes cryptographers nervous. We can make the Jacobi symbol useless to an attacker by using only elements that have the same Jacobi symbol. Second, restricting to J_N makes the system cleaner by removing unneeded cosets from $(\mathbb{Z}/N\mathbb{Z})^\times$. Third, using J_N will make the computational assumption easier to state. We now have the following commutative diagram:

$$\begin{tikzcd}
J_N \times J_N \arrow[r, "\boxtimes"] \arrow[d, dashed, "\left(\frac{\cdot}{p}\right)\text{,}\left(\frac{\cdot}{p}\right)"] & J_N \arrow[d, dashed, "\left(\frac{\cdot}{p}\right)"] \\
\{\pm 1\} \times \{\pm 1\} \arrow[r, "\times"] & \{\pm 1\}.
\end{tikzcd}$$

The downward arrows are still surjective. Half of the elements of J_N are *squares* in $(\mathbb{Z}/N\mathbb{Z})^*$ with Legendre symbol 1 for both p and q, and half are *nonsquares* with Legendre symbol -1 for both p and q.

Now, let us build a homomorphic encryption system from the commutative diagram. Our diagram indicates that our set of ciphertexts is J_N, and that we decrypt a ciphertext c by computing $\left(\frac{c}{p}\right)$. The diagram also depicts the homomorphism of our system—namely, that by multiplying ciphertexts (modulo N), we implicitly multiply the underlying messages (in $\{\pm 1\}$). So, we have already built the decryption function D (which maps a ciphertext to a message) and the evaluation function V (which uses the binary relation(s) over ciphertexts to implicitly apply the analogous binary relation(s) to the messages that are encrypted).

All what remains is to build key generation K (which generates a random key pair (ek, sk), an encryption key and a secret decryption key) and encryption E (which maps a message from $\mathcal{M}$ and some randomness to a ciphertext using the encryption key ek, i.e., $c = \mathsf{E}(ek, m, r)$.) What do we need to put in the public encryption key to allow a user to generate a random encryption of either $\{\pm 1\}$? Notice that the encryptions of 1, i.e., the subset of J_N that has Legendre symbol 1 for p, are precisely the quadratic residues (squares) in $(\mathbb{Z}/N\mathbb{Z})^\times$. So, given N, anybody can generate a random encryption of 1 easily by taking a random element of $(\mathbb{Z}/N\mathbb{Z})^\times$ and squaring it. To generate a random encryption of -1, the user needs *some* encryption u of -1, namely, a nonsquare in J_N, which it can then randomize via multiplication with a random square. Hence, it suffices to provide (N, u) as the public encryption key.[3]

Below is a cleaner presentation of the Goldwasser–Micali encryption system. Let $\mathsf{CompositeGen}(\lambda, r)$ be a function that takes a security parameter λ and some randomness r as input, and which outputs integer primes p, q of size determined by λ (they have a number of bits polynomial in λ) and their product $N = p \cdot q$.

Goldwasser–Micali encryption system.

- Key Generation: $\mathsf{K}(\lambda, r)$ takes a security parameter λ and some randomness r as input. It outputs $(p, q, N) \leftarrow \mathsf{CompositeGen}(\lambda, r)$. Also, it uses the randomness to generate random $u \in J_N$ that is a nonsquare. The secret key sk is p. The public encryption ek is (N, u). The message set $\mathcal{M}$ is $\{-1, 1\}$. The ciphertext set $\mathcal{C}$ is J_N.

- Encryption: $\mathsf{E}(ek, m, r)$ takes the encryption key ek, a message $m \in \mathcal{M}$ and some randomness r as input. It generates random $t \in (\mathbb{Z}/N\mathbb{Z})^*$. If $m = 1$, it outputs ciphertext $c \leftarrow t^2 \bmod N$, else it outputs $c \leftarrow u \cdot t^2 \bmod N$.

- Decryption: $\mathsf{D}(sk, c)$ takes the secret key sk and a ciphertext $c \in \mathcal{C}$ as input. It outputs $m \leftarrow \left(\frac{c}{p}\right) \in \mathcal{M}$.

- Homomorphic multiplication: it takes two ciphertexts $c_1, c_2 \in \mathcal{C}$ and outputs $c \leftarrow c_1 \boxtimes c_2$.

3 See Section 3.2 for a more generally applicable approach to key generation and encryption, in which key generation involves populating the public key with encryptions (preimages under the decryption map) of several known values, and encryption involves applying the binary relation to the ciphertexts (preimages) in the public key to generate a random encryption (preimage) of the desired value.

Now, let us turn to security. Goldwasser and Micali defined the security of an encryption system using the following game [15, 47].

Definition 1 (IND-CPA game). The IND-CPA game between a "challenger" and an "adversary" is as follows:

- Key Generation: The challenger uses K and λ to generate a key pair (sk, ek). If the encryption system is asymmetric (public-key), it sends ek to the adversary. It keeps sk secret. The challenger samples a random bit $b \in \{0, 1\}$.
- Training and Challenges: Repeatedly, the adversary selects some messages $m_{i,0}, m_{i,1} \in \mathcal{M}$ that it sends to the challenger. The challenger generates randomness r_i and sends $c_i \leftarrow \mathsf{E}(ek, m_{i,b}, r_i)$ to the adversary. (If the adversary is free to set $m_{i,0} = m_{i,1}$ if it wants an encryption of a known message.)
- Guess: The adversary guesses a bit $b' \in \{0, 1\}$. It wins if $b' = b$.

Definition 2 (Adversary's advantage). In a game against system $\mathcal{E}$ with security parameter λ in which an adversary is trying to guess a random bit $b \in \{0, 1\}$, we define the adversary's advantage $\mathrm{Adv}^{\mathcal{E}}_{\mathcal{A}}(\lambda)$ to be $|\Pr[\mathcal{A}$ guesses b correctly$] - \frac{1}{2}|$.

Definition 3 (IND-CPA security of encryption). We say that an encryption system $\mathcal{E}$ is IND-CPA-secure if, for all probabilistic polynomial time adversaries $\mathcal{A}$ (i.e., that run in time polynomial in λ), the adversary's advantage $\mathrm{Adv}^{\mathcal{E}}_{\mathcal{A}}(\lambda)$ in the IND-CPA game is negligible (i.e., $o(1/\lambda^c)$ for all constants c).

See Appendix A for a discussion of why, for most settings, IND-CPA is a minimal viable notion of security for encryption; here, we just make a few comments about it. In our presentation of the IND-CPA game, we consider only fixed-length messages from the set $\mathcal{M}$; if queries with variable-length messages are allowed, the game requires $m_{i,0}, m_{i,1}$ to be the same length.

In its attack, the adversary is, of course, free to use $\mathsf{E}(ek, \cdot)$ and/or V to produce ciphertexts on its own if the system is asymmetric and/or homomorphic.

An encryption system can be secure under the IND-CPA game only if it is *probabilistic*—that is, there are many ciphertexts for each message. If the system were deterministic, the adversary could easily win by obtaining an encryption of a known message m, and then querying $(m_{i,0} = m, m_{i,1})$ for some $m_{i,1} \neq m$.

The IND-CPA game and our commutative diagram imply that, in a secure encryption system, it is hard to distinguish samples of $\ker(\mathsf{D}(sk, \cdot))$ from samples of $\mathcal{C}$. Making the kernel of the decryption homomorphism indistinguishable from the entire set of ciphertexts is the essence, and hardest part, of hardening a homomorphism to make it suitable for cryptography.

It only remains to define a clear *computational assumption*, and *prove* the security of Goldwasser–Micali based on the assumption. The computational assumption is that it is hard to distinguish whether a randomly sampled element of J_N is a square modulo N. This assumption is formalized as follows:

Definition 4 (Quadratic Residuosity (QR) assumption). For security parameter λ and randomness r, compute $(p, q, N) \leftarrow \mathsf{CompositeGen}(\lambda, r)$. Sample v uniformly from J_N. The QR assumption is that given (N, v) (but not p, q, r), all probabilistic polynomial time (in λ) adversaries $\mathcal{A}$ have negligible advantage in guessing whether v is a quadratic residue modulo N. (The probability in the assumption is taken not just over the sampling of J_N, but also over the choice of N.)

In other words, the QR assumption is that samples from J_N are indistinguishable from samples from the subset of J_N that is in $\ker\big(\big(\frac{\cdot}{p}\big)\big)$.

Note that the assumption (like all of the computational assumptions that we will make) depends on how elements are presented. For example, if our presentation of an element $j \in J_N$ is "too revealing" in that we give j not just as element of $(\mathbb{Z}/N\mathbb{Z})^*$ but also give the value $\big(\frac{j}{p}\big)$, then clearly the assumption becomes false. Generally speaking, it will be clear what "hardened" presentation the assumption is using.

How hard is the QR problem? We do not know of any algorithm for the QR problem that is faster than factoring N. The fastest algorithm for integer factorization is currently the number field sieve [57], which runs in time $\exp(O(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}})$, i.e., subexponential (but superpolynomial) time.

For fixed N, Goldwasser and Micali show how to amplify the success probability of an QR algorithm—given an algorithm that guesses correctly with probability $\frac{1}{2} + \varepsilon$, one can construct an algorithm that uses about $O(\varepsilon^{-1})$ times the computation and guesses with probability very close to 1. This follows from the fact that the QR problem is *random self-reducible*, given a *particular* $j \in J_N$, we can run the initial algorithm on many *random* $j \cdot r$, where r is a random square, and aggregate the results. In other words, random self-reducibility exploits the homomorphism to generate many samples that have the same preimage by multiplying an initial sample with many elements of the kernel.

Now, let us prove the security of the Goldwasser–Micali encryption system. The proof is quite simple. Nonetheless, it is useful because it reduces the security of a complex system (that allows the IND-CPA adversary to "train" and adapt by making interactive dynamically-chosen queries) to a crisp and concise computational assumption.

Theorem 1. *The Goldwasser–Micali encryption system is IND-CPA-secure based on the QR assumption.*

Proof. Suppose that there exists an efficient adversary $\mathcal{A}$ that wins the IND-CPA game with probability $\frac{1}{2} + \varepsilon$. Then, we claim that there exists an efficient algorithm $\mathcal{B}$, running in about the same time as $\mathcal{A}$, that solves the QR problem with probability $\frac{1}{2} + \frac{\varepsilon}{2}$. The theorem follows from this claim.

Here is how algorithm $\mathcal{B}$ works: $\mathcal{B}$ is given an instance of the QR problem, namely (N, v) such that N is a composite number chosen according to the specified distribution and v is sampled uniformly from J_N. Here $\mathcal{B}$'s task is to distinguish whether v is a quadratic residue modulo N. To solve its task, $\mathcal{B}$ assumes the role of the challenger in the IND-CPA game with $\mathcal{A}$. Then $\mathcal{B}$ gives $ek \leftarrow (N, v)$ to $\mathcal{A}$ as the public encryption key, and $\mathcal{B}$ samples a random $b \in \{0,1\}$. When $\mathcal{A}$ sends query $(m_{i,0}, m_{i,1})$, $\mathcal{B}$ samples randomness r_i for encryption, sets $c_i \leftarrow \mathsf{E}(ek, m_{i,b}, r_i)$, and sends c_i to $\mathcal{A}$. Also $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. If $b' = b$, then $\mathcal{B}$ guesses that v is a quadratic nonresidue; otherwise, it guesses that u is quadratic residue.

Now, we have two cases: either v is a nonresidue (as it should be in the real system), or it is a residue. Each case happens with probability $\frac{1}{2}$. In the former case, the public key ek and all of the ciphertexts generated by $\mathcal{B}$ have the same distribution as in the IND-CPA game. Therefore, in this case, the adversary guesses correctly ($b' = b$) with probability $\frac{1}{2} + \varepsilon$ by assumption. In the latter case (if v is a quadratic residue), then v and all of the ciphertexts generated by $\mathcal{B}$ are uniformly random quadratic residues. In particular, the ciphertexts are independent of the messages they are supposed to encrypt, and hence independent of b. In this case, the adversary's guess b' is also independent of b. Thus, $\mathcal{A}$'s success (or lack of it) gives $\mathcal{B}$ a clue about whether or not v is a quadratic nonresidue (or residue). In detail, using QR and QNR to denote the events that v is a quadratic residue or nonresidue, respectively, we have:

$$\begin{aligned}
\Pr[\mathcal{B}\text{ correct}] &= \Pr[\mathcal{B}\text{ correct}|\text{QR and }\mathcal{A}\text{ correct}] \cdot \Pr[\text{QR and }\mathcal{A}\text{ correct}] \\
&\quad + \Pr[\mathcal{B}\text{ correct}|\text{QR and }\mathcal{A}\text{ incorrect}] \cdot \Pr[\text{QR and }\mathcal{A}\text{ incorrect}] \\
&\quad + \Pr[\mathcal{B}\text{ correct}|\text{QNR and }\mathcal{A}\text{ correct}] \cdot \Pr[\text{QNR and }\mathcal{A}\text{ correct}] \\
&\quad + \Pr[\mathcal{B}\text{ correct}|\text{QNR and }\mathcal{A}\text{ incorrect}] \cdot \Pr[\text{QNR and }\mathcal{A}\text{ incorrect}] \\
&= 0 + 1 \cdot \frac{1}{2} \cdot \frac{1}{2} + 1 \cdot \frac{1}{2}\left(\frac{1}{2} + \varepsilon\right) + 0 \\
&= \frac{1}{2} + \frac{\varepsilon}{2}. \qquad \blacksquare
\end{aligned}$$

2.2. ElGamal: HE starting from a linear homomorphism

We provide one more example of a simple homomorphic encryption system. The ElGamal cryptosystem [35] was probably directly inspired by the Diffie–Hellman protocol [31], but it *could* have been invented by starting from a linear homomorphism, and then hardening the homomorphism, as follows.

Let q be a prime integer. For $\vec{s} \in (\mathbb{Z}/q\mathbb{Z})^n$, the inner product $\langle \vec{s}, \cdot \rangle : (\mathbb{Z}/q\mathbb{Z})^n \to \mathbb{Z}/q\mathbb{Z}$ is a linear homomorphism. We have the following commutative diagram:

$$\begin{tikzcd}
(\mathbb{Z}/q\mathbb{Z})^n \times (\mathbb{Z}/q\mathbb{Z})^n \arrow[r, "\boxplus"] \arrow[d, "{\langle \vec{s},\cdot\rangle, \langle \vec{s},\cdot\rangle}"] & (\mathbb{Z}/q\mathbb{Z})^n \arrow[d, "{\langle \vec{s},\cdot\rangle}"] \\
\mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \arrow[r, "+"] & \mathbb{Z}/q\mathbb{Z},
\end{tikzcd}$$

where $\boxplus$ is vector addition.

How can we "harden" the linear homomorphism to build a homomorphic encryption system? The IND-CPA game (see Definition 1) allows an adversary to obtain many (message, ciphertext) pairs. If such pairs have the form $(m, \vec{c}) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$ such that $m = \langle \vec{s}, \vec{c} \rangle$, the adversary can efficiently solve for $\vec{s}$ using linear algebra. One countermeasure to this linear algebra attack is to put the elements of $\mathbb{Z}/q\mathbb{Z}$ "in the exponent." That is, let G be a cyclic group of prime order q whose binary operation we denote multiplicatively. For example, G could be a subgroup of the multiplicative group of a finite field, or of the group of points of an elliptic curve group over a finite field. Let g be a generator of G. In such groups, given G and g, recovering $a \in \mathbb{Z}/q\mathbb{Z}$ from g^a is called the *discrete logarithm* (DL) problem, which is believed to be hard for appropriate groups.[4] We now have the following commutative diagram:

$$\begin{tikzcd}
G^n \times G^n \arrow[r, "\boxtimes"] \arrow[d, dashed, "{\langle \vec{s},\cdot\rangle, \langle \vec{s},\cdot\rangle}"] & G^n \arrow[d, dashed, "{\langle \vec{s},\cdot\rangle}"] \\
G \times G \arrow[r, "\times"] & G,
\end{tikzcd}$$

where now $\langle \vec{s}, \cdot \rangle$ is applied "in the exponent," i.e., for $(g_1, \ldots, g_n) \in G^n$, we have

$$\big\langle \vec{s}, (g_1, \ldots, g_n) \big\rangle = \prod_{i=1}^{n} g_i^{s_i},$$

and $\boxtimes$ is componentwise multiplication.

Now, let us build the Elgamal encryption system from the commutative diagram. Elgamal uses a 2-dimensional secret key $\vec{s}$ with the special form $(-s, 1)$. So, ciphertexts and messages live in G^2 and G, respectively. As depicted by the commutative diagram, decryption involves applying $\langle \vec{s}, \cdot \rangle$ "in the exponent." The diagram also specifies the multiplicative homomorphism. Now, what do we need to put in the public key to allow anyone to generate a random encryption of any element of G? A random encryption of $m \in G$ is simply *some* encryption of m multiplied (via $\boxtimes$) by a random encryption of g^0. Anybody can easily compute *some* encryption of m as $\vec{c} = (g^0, m)$, since $\langle \vec{s}, \vec{c} \rangle = (g^0)^{-s} \cdot m^1 = m$. A random encryption of g^0 has the form $(g^r, g^{r \cdot s})$ for r sampled uniformly from $\mathbb{Z}/q\mathbb{Z}$. To enable generation of a random encryption of g^0, the public encryption key needs only a some nontrivial encryption of g^0, in particular, (g, g^s) suffices.

ElGamal encryption system.

- Key Generation: $\mathsf{K}(\lambda, r)$ takes a security parameter λ and some randomness r as input. It uses λ and the randomness to generate (G, g), a group and generator of order $q = q(\lambda)$. (Alternatively, the group may be preset and common to many users.) It generates a random $s \in \mathbb{Z}/q\mathbb{Z}$ and sets $h \leftarrow g^s$. The secret key sk is s. The public encryption ek is (G, g, h). The message set $\mathcal{M}$ is G. The ciphertext set $\mathcal{C}$ is G^2.

4 The reverse problem of computing g^a from g for $a \in \mathbb{Z}/q\mathbb{Z}$ can be solved efficiently using only $O(\log q)$ multiplications in G using the technique of "repeated squaring."

- Encryption: $\mathsf{E}(ek, m, r)$ takes the encryption key ek, a message $m \in \mathcal{M}$ and some randomness as input. It generates random $t \in (\mathbb{Z}/q\mathbb{Z})$. It outputs $c \leftarrow (g^t, m \times h^t)$.
- Decryption: $\mathsf{D}(sk, c)$ takes the secret key sk and a ciphertext $c = (c_0, c_1) \in \mathcal{C}$ as input. It outputs $m \leftarrow c_0^{-s} \times c_1 \in \mathcal{M}$.
- Homomorphic multiplication: it takes two ciphertexts $c^{(1)}, c^{(2)} \in \mathcal{C}$ and outputs $c \leftarrow c_1 \boxtimes c_2$.

The computational assumption underlying ElGamal is called the Diffie–Hellman assumption.

Definition 5 (Diffie–Hellman (DH) assumption). Let G be a fixed group of order q (determined by security parameter λ) with generator g. Sample a random bit $\beta \in \{0, 1\}$. If $\beta = 0$, sample x and y randomly from $\mathbb{Z}/q\mathbb{Z}$ and set $z = x \cdot y$. If $\beta = 1$, sample x, y, and z randomly from $\mathbb{Z}/q\mathbb{Z}$. Output (G, g, g^x, g^y, g^z). The DH assumption is that all probabilistic polynomial time (in λ) adversaries $\mathcal{A}$ have negligible advantage in guessing the bit β.

Note that is easy to determine whether the discrete logarithms of a tuple satisfy a given linear equation. The DH assumption is basically that it is hard to distinguish whether the discrete logarithms satisfy a degree-2 equation. For some elliptic curve groups over finite fields, the fastest algorithm for distinguishing Diffie–Hellman is to solve the discrete logarithm problem by using the "baby-step giant-step" method, which takes roughly $\sqrt{q}$ computational steps.

Now, we prove the security of ElGamal based on the Diffie–Hellman (DH) assumption.

Theorem 2. *The Elgamal encryption system is IND-CPA secure under the DH assumption.*

Proof. Let $\mathcal{B}$ be an algorithm that is given an instance of the DH problem, namely, (G, g, g^x, g^y, g^z) such that if $\beta = 0$ then $z = x \cdot y$, but if $\beta = 1$ then z is sampled uniformly and independently modulo q. Here $\mathcal{B}$'s task is to distinguish the bit β while $\mathcal{B}$ and $\mathcal{A}$ play the roles of the challenger and adversary in the IND-CPA game. Algorithm $\mathcal{B}$ gives (G, g, g^x) to $\mathcal{A}$ as the public encryption key. Then $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$. When $\mathcal{A}$ queries messages $(m_{i,0}, m_{i,1})$, $\mathcal{B}$ samples randomness $r_i \in \mathbb{Z}/q\mathbb{Z}$, and sends the ciphertext $(g^0, m_{i,b}) \boxtimes ((g^y)^{r_i}, (g^z)^{r_i})$. Adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$, then $\mathcal{B}$ guesses that $\beta = 0$; otherwise it guesses that $\beta = 1$.

One can check that $\mathcal{B}$'s advantage in the DH game is $\frac{\varepsilon}{2}$, where ε is $\mathcal{A}$'s advantage in the IND-CPA game. The idea, as in the security proof for Goldwasser–Micali system, is that everything—namely the public key and ciphertexts—is distributed properly when $\beta = 0$ and $z = x \cdot y$, and so $\mathcal{A}$ should have advantage ε in that case. In particular, $((g^y)^{r_i}, (g^z)^{r_i})$ is a random encryption of g^0, and so the ith ciphertext is indeed a random encryption of $m_{i,b}$. However, when $\beta = 1$, with high probability $((g^y)^{r_i}, (g^z)^{r_i})$ is an encryption of a random value. Therefore the ciphertexts generated by $\mathcal{B}$ encrypt random values independent of b, and $\mathcal{A}$ has no advantage in guessing b. So, $\mathcal{A}$'s success (or lack of it) gives $\mathcal{B}$ a clue about the value of β. ■

3. GENERAL RESULTS ABOUT HOMOMORPHIC ENCRYPTION

Now that we have in mind some simple examples of homomorphic encryption systems, let us provide a general definition of homomorphic encryption and some general results.

3.1. Formal definition of HE

A homomorphic encryption system is, first of all, an encryption system:

Definition 6 (Encryption (syntax)). An encryption system consists of 3 functions: key generation K, encryption E, and decryption D:

- $(sk, ek, \mathsf{params}) \leftarrow \mathsf{K}(\lambda, r)$ takes the security parameter λ and some randomness r and outputs a secret decryption key sk, an encryption key ek, some parameters params of the system, such as the message set $\mathcal{M}$. In a symmetric system, the encryption key ek equals sk and is kept secret. In an asymmetric (or public-key) system, ek is public and does not equal sk. We omit mentioning params as an input to the other functions.
- $c \leftarrow \mathsf{E}(ek, m, r)$ takes the encryption key ek, a message $m \in \mathcal{M}$, and some randomness r and outputs a ciphertext. Encryption is probabilistic: new randomness r is sampled for each encryption.
- $m \leftarrow \mathsf{D}(sk, c)$ takes a secret key and ciphertext and returns a message $m \in \mathcal{M}$.

We write $\vec{c} \leftarrow \mathsf{E}(ek, \vec{m}, \vec{r})$ and $\vec{m} \leftarrow \mathsf{D}(sk, \vec{c})$ for vectors of messages and ciphertexts; K, E, and D all can be computed in time polynomial in the security parameter λ.

Definition 7 (Correctness of encryption). It must hold that $m = \mathsf{D}(sk, \mathsf{E}(ek, m, r))$ for all key tuples (sk, ek) in the image of K, all $m \in \mathcal{M}$, and all randomness r.

The IND-CPA security of encryption system is as described in Definition 3.

A homomorphic encryption system also has a fourth function V (evaluation). The homomorphic property requires some tweaks to K as well. (The functions E and D are as before.)

Definition 8 (Homomorphic encryption (syntax)). A homomorphic encryption system consists of 4 functions: key generation K, encryption E, decryption D, and evaluation V:

- K: As in an encryption system, except that K also outputs a public evaluation key evk, and params includes some description of a set $\mathcal{F}$ of functions, with input and output over $\mathcal{M}$, that the homomorphic encryption system is capable of evaluating correctly (see below).

- E: As in an encryption system.
- D: As in an encryption system.
- $c \leftarrow \mathsf{V}(evk, f, c_1, \ldots, c_t)$ takes evk, a function $f \in \mathcal{F}$, and t ciphertexts $c_1, \ldots, c_t$, where t is the number of inputs to f. It outputs a ciphertext c.

The above K, E, D, and V all can be computed in time polynomial in the security parameter λ, though V's complexity necessarily also depends (polynomially) on the complexity of the function f being evaluated.

The security notion of homomorphic encryption remains IND-CPA security, without reference to V. This is because V is a public function with no secrets. The adversary is, of course, free to try to use V in its attack.

A homomorphic encryption system must satisfy not only the basic correctness of encryption, but also correctness of evaluation. We will define the correctness of evaluation with commutative diagrams. First, note that the images of E and of V need not be the same in general (though they were the same for the simple homomorphic encryption systems we presented in Section 2).

Definition 9 (Fresh and evaluated ciphertexts). We differentiate between two types of ciphertexts:

- "Fresh ciphertexts" (denoted by $\mathcal{C}_{\mathsf{E}}$): ciphertexts in the image of E,
- "Evaluated ciphertexts" (denoted by $\mathcal{C}_{\mathsf{V}}$): a superset of $\mathcal{C}_{\mathsf{E}}$ that also includes ciphertexts in the image of V when evaluated on a function $f \in \mathcal{F}$ and ciphertexts from $\mathcal{C}_{\mathsf{E}}$.

Though the notation suppresses it, these sets depend on the particular encryption key ek and evaluation evk being used.

Definition 10 (Correctness of evaluation (bounded homomorphic encryption)). A homomorphic encryption system correctly evaluates a set of functions $\mathcal{F}$, and is called $\mathcal{F}$-homomorphic, if

$$\mathsf{D}\big(sk, \mathsf{V}(evk, f, c_1, \ldots, c_t)\big) = f\big(\mathsf{D}(sk, c_1), \ldots, \mathsf{D}(sk, c_t)\big)$$

for all (sk, ek, evk) in the image of K, all fresh ciphertexts $\{c_i\}$ in $\mathcal{C}_{\mathsf{E}}$ (for key ek), and all $f \in \mathcal{F}$. This correctness requirement is depicted by the commutative diagram for bounded homomorphic encryption in Figure 1a.

Unless otherwise specified, a homomorphic encryption system is *bounded*, as depicted in Figure 1a, since correctness of evaluation is *a priori* guaranteed to hold only when the inputs are fresh ciphertexts come from $\mathcal{C}_{\mathsf{E}}$, not necessarily from $\mathcal{C}_{\mathsf{V}}$. Bounded homomorphic encryption systems can be trivial and uninteresting—for example, when V does nothing but output f and its input ciphertexts $c_1, \ldots, c_t$, leaving it to the decryption function D to decrypt the c_i's and apply f to the messages.

$$\begin{tikzcd}
\mathcal{C}_{\mathsf{E}}^{*} \arrow[r, "\mathsf{V}(evk,f,\cdot,\ldots,\cdot)"] \arrow[d, dashed, "\mathsf{D}(sk,\cdot,\ldots,\cdot)"] & \mathcal{C}_{\mathsf{V}} \arrow[d, dashed, "\mathsf{D}(sk,\cdot)"] \\
\mathcal{M}^{*} \arrow[r, "f"] & \mathcal{M}
\end{tikzcd}$$

(A) Bounded HE.

$$\begin{tikzcd}
\mathcal{C}_{\mathsf{V}}^{(i)*} \arrow[r, "\mathsf{V}(evk^{(i-1)},f,\cdot,\ldots,\cdot)"] \arrow[d, dashed, "\mathsf{D}(sk^{(i)},\cdot,\ldots,\cdot)"] & \mathcal{C}_{\mathsf{V}}^{(i-1)} \arrow[d, dashed, "\mathsf{D}(sk^{(i-1)},\cdot)"] \\
\mathcal{M}^{*} \arrow[r, "f"] & \mathcal{M}
\end{tikzcd}$$

(B) Leveled HE.

$$\begin{tikzcd}
\mathcal{C}^{*} \arrow[r, "\mathsf{V}(evk,f,\cdot,\ldots,\cdot)"] \arrow[d, dashed, "\mathsf{D}(sk,\cdot,\ldots,\cdot)"] & \mathcal{C} \arrow[d, dashed, "\mathsf{D}(sk,\cdot)"] \\
\mathcal{M}^{*} \arrow[r, "f"] & \mathcal{M}
\end{tikzcd}$$

(C) Unbounded HE.

FIGURE 1
Bounded, leveled, and unbounded systems with $f \in \mathcal{F}$ for function set $\mathcal{F}$. The notation $\mathcal{M}^*$ indicates that the number of copies of $\mathcal{M}$ depends on the number of inputs to f.

One property that can make a homomorphic encryption system nontrivial is if it is *unbounded.*

Definition 11 (Unbounded homomorphic encryption). We say a homomorphic encryption system is unbounded $\mathcal{F}$-homomorphic if, for some set of ciphertexts $\mathcal{C}$, the commutative diagram in Figure 1c holds.

For an unbounded system, the functions in $\mathcal{F}$ can be applied repeatedly, indefinitely. The size of the ciphertexts, and the cost of their decryption, does not depend on how many homomorphic operations were performed. The simple homomorphic encryption systems discussed in Section 2 are unbounded with respect to a single binary operation.

In-between bounded and unbounded, we have a notion of leveled homomorphic encryption.

Definition 12 (Leveled homomorphic encryption). We say a homomorphic encryption system is leveled $\mathcal{F}$-homomorphic if, for any number $n \in \mathbb{N}$ (a parameter to be included in params that indicates the number of levels), the commutative diagram in Figure 1b holds for all $i \in \{1, \ldots, n\}$. The functions K, E, D, V are required to be independent of n, aside from the fact that K generates a key-tuple $(sk^{(i)}, ek^{(i)}, evk^{(i)})$ for each level.

Our definition of leveled homomorphic encryption here is strict; sometimes the definition allows the complexity of the functions to grow polynomially in n. Note that, in the leveled system, V converts ciphertexts under key $ek^{(i)}$ to the next key $ek^{(i-1)}$.

Now what about the set $\mathcal{F}$? The set $\mathcal{F}$ can be limited or powerful. The systems considered in Section 2 are unbounded, but can perform only a single binary operation that is insufficient to perform arbitrary computations. In contrast, we say that $\mathcal{F}$ is *functionally complete* if it contains a functionally complete (or *universal*) set of operations or "gates." Examples of a universal set of gates are $\{\mathsf{AND}, \mathsf{NOT}\}$ and $\{\mathsf{NAND}\}$. Boolean circuits composed of a universal set of Boolean gates are very powerful: any problem that can be computed in polynomial time by a deterministic Turing machine can also be computed by a polynomial-size Boolean circuit family. That is, Boolean circuits can efficiently perform general efficient computation (as classically defined with respect to Turing machines). Our ultimate goal is a homomorphic encryption system that is unbounded for universal gates.

Definition 13 (Fully homomorphic encryption (FHE)). A homomorphic encryption system is called fully homomorphic (resp. leveled fully homomorphic) if it is unbounded (resp. leveled) and its $\mathcal{F}$ contains a functionally complete set of gates.

With a fully homomorphic encryption (FHE) system, we can do arbitrary computations on data while it remains encrypted. Most of the rest of this survey focuses on building FHE.

3.2. General approach to key generation and encryption

Decryption D and evaluation V are the stars of our commutative diagrams, but let us focus on key generation K and encryption E for a moment.

Encryption E is simply the inverse of D. Given $m \in \mathcal{M}$, we encrypt m by sampling from the preimage of m under the decryption map $\mathsf{D}(sk, \cdot)$. In an asymmetric (public-key) system, such sampling must be possible only with the public encryption key ek, without sk. Also, for the system to be IND-CPA secure, this sampling must be probabilistic. Can we devise a simple, natural, secure way to encrypt, that is, to sample randomly from the decryption-map preimage of a message?

For a homomorphic encryption system, a possible answer immediately presents itself. Populate the encryption key ek with (image, preimage) pairs $\{(m_i, c_i)\}$. Then, the encrypter *uses the homomorphism of the system* to generate a random preimage of its m from the given preimages of $\{m_i\}$ in ek. That is, the encrypter finds a random function $f \in \mathcal{F}$ such that $f(m_1, \ldots, m_t) = m$, and then outputs the ciphertext $c = \mathsf{V}(ek, f, c_1, \ldots, c_t)$ as its encryption of m.

But we need to be careful here. The ciphertext c certainly decrypts to the correct message m. But how can we be sure that c does not retain some detectable residue of its history? A homomorphic encryption system may be IND-CPA secure, yet still allow an adversary to distinguish that a ciphertext c was produced by evaluating a particular function f on $c_1, \ldots, c_t$, which in this case would allow the adversary to recover m. In fact, the "trivial" bounded HE system mentioned after Definition 10 (in Section 3.1) has exactly this property. For the encryption procedure to be secure, we need to ensure that the encrypter's ciphertext c "forgets" how it was made.

Suppose that an evaluated ciphertext c is at most ℓ bits, i.e., c only "remembers" ℓ bits. And also suppose that we generate c as (say) a random linear combination of the c_i's in ek, subject to the same random linear combination of the m_i's equaling m. If t (the number of ciphertexts in ek) is very large, such that the entropy of the random linear combination is much greater than ℓ, then indeed (whp) c's precise history will be very ambiguous given only c itself.

Rothblum [69] formalizes this intuition for homomorphic encryption systems that have sufficiently compact ciphertexts and a $\boxplus$ operation that is additively homomorphic over $\mathcal{M} = \mathbb{Z}/2\mathbb{Z}$. Precisely, he proves:

Theorem 3 (Rothblum). *Let $\mathcal{E}_{\text{sym}}$ be an IND-CPA secure symmetric encryption system that is homomorphic with respect to addition modulo 2. Suppose that there is a polynomial t such that homomorphically adding $t(\lambda)$ fresh ciphertexts results in a ciphertext of at most $t(\lambda)/10$ bits. Then from $\mathcal{E}_{\text{sym}}$ one can build an IND-CPA secure asymmetric system $\mathcal{E}_{\text{asym}}$.*

Rothblum presents the result somewhat differently as constructing asymmetric encryption from homomorphic symmetric encryption. But the bottom line is the same: the encryption algorithm practically writes itself as long as there is an efficient procedure (using sk) to sample a handful of random (image, preimage) pairs of the decryption map to put in ek, and we can evaluate homomorphic addition modulo 2 compactly.

In detail, Rothblum builds the asymmetric system as follows:

Rothblum's asymmetric system $\mathcal{E}_{\text{asym}}$.

- $\mathcal{E}_{\text{asym}}.\mathsf{K}$: Compute $\mathcal{E}_{\text{sym}}.\mathsf{K}$ to obtain sk. Use $\mathcal{E}_{\text{sym}}.\mathsf{E}(sk, \cdot, \cdot)$ to generate $X^{(0)} = (X_1^{(0)}, \ldots, X_t^{(0)})$ and $X^{(1)} = (X_1^{(1)}, \ldots, X_t^{(1)})$, which are $t = t(\lambda)$ encryptions of 0 and t encryptions of 1, respectively. The secret decryption key is sk. The public encryption key ek is $(X^{(0)}, X^{(1)})$.
- $\mathcal{E}_{\text{asym}}.\mathsf{E}$: Let $\boxplus$ denote $\mathcal{E}_{\text{sym}}$'s homomorphic addition mod 2 operation. To encrypt $m \in \{0, 1\}$, sample a random vector $r \in \{0, 1\}^t$ such that $r_1 + \cdots + r_t = m \bmod 2$ and output the ciphertext $c = X_1^{(r_1)} \boxplus \cdots \boxplus X_\ell^{(r_t)}$.
- $\mathcal{E}_{\text{asym}}.\mathsf{D}$: Identical to $\mathcal{E}_{\text{sym}}.\mathsf{D}$.

Correctness follows easily from the properties of $\mathcal{E}_{\text{sym}}$.

Rothblum's proof of IND-CPA security comes in two parts. First, he proves that if $\mathcal{E}_{\text{sym}}$ is indeed IND-CPA secure, a polynomial-time adversary will not notice if $X^{(1)}$ is replaced by t more encryptions of 0. This follows immediately from the definition of IND-CPA security (see Definition 3).

Second, assuming now that $X^{(0)}$ and $X^{(1)}$ are now $2t$ i.i.d. encryptions of 0, Rothblum shows that a ciphertext generated as $c = X_1^{(r_1)} \boxplus \cdots \boxplus X_t^{(r_t)}$ "forgets" the value $r_1 + \cdots + r_t \bmod 2$ (the value that is supposed to be encrypted). Specifically, the possible preimages $(r_1, \ldots, r_t)$ for c satisfy $r_1 + \cdots + r_t = 0 \bmod 2$ with probability at most $\frac{1}{2} + 2^{-0.2t+\ell+1}$, where ℓ is the number of bits in c (and similarly for the case of 1 mod 2). As $\ell < t/10$, this probability is negligibly close to $\frac{1}{2}$.

3.3. Getting to functional completeness

Now, let us return to our main goal, which is constructing FHE. Suppose we have an unbounded system $\mathcal{E}$ that is correct for a non-functionally-complete set $\mathcal{F}$. Can we use $\mathcal{E}$ to get an FHE system $\mathcal{E}_{\mathsf{FHE}}$?

In some cases, yes. Here is a silly example. The arithmetic gates $\{+, \times\}$ are not functionally complete over GF(2). In particular, any circuit composed of $\{+, \times\}$ gates can only output 0 when the inputs are all 0 (and so such circuits cannot express functions that output 1 when the inputs are all 0). But this technicality is not a real obstacle to constructing FHE. As long as $\mathcal{E}$ is capable of producing a single encryption of 1 (e.g., via encryption) it can evaluate $\mathsf{NOT}(x)$ as $1 + x$. (And it can emulate $\mathsf{AND}(x, y)$ as $x \times y$.)

Here is a less trivial example. The gates $\{\mathsf{AND}, \mathsf{OR}\}$ are not functionally complete. Circuits composed of $\{\mathsf{AND}, \mathsf{OR}\}$ gates can only compute *monotone* functions (not general functions), where a function f is called monotone if $f(x) \leq f(y)$ whenever $x_i \leq y_i$ for all i. However, via De Morgan's law, we can reexpress any Boolean circuit as a circuit that is monotone except at the input level, which is allowed to have NOT gates. Applying De Morgan's law, given an unbounded $\{\mathsf{AND}, \mathsf{OR}\}$-homomorphic system $\mathcal{E}$, we can construct an FHE system $\mathcal{E}_{\mathsf{FHE}}$ as follows. An $\mathcal{E}_{\mathsf{FHE}}$ ciphertext encrypting "1" consists of an ordered pair of two $\mathcal{E}$ ciphertexts encrypting 1 and 0, respectively. An $\mathcal{E}_{\mathsf{FHE}}$ ciphertext encrypting "0" consists of an ordered pair of two $\mathcal{E}$ ciphertexts encrypting 0 and 1, respectively. Performing a NOT gate in $\mathcal{E}_{\mathsf{FHE}}$ is simple: just swap the $\mathcal{E}$ ciphertexts in the ordered pair. To perform an AND gate in $\mathcal{E}_{\mathsf{FHE}}$, take the $\mathcal{E}$-AND of the first $\mathcal{E}$ ciphertexts in each pair, and the $\mathcal{E}$-OR of the second $\mathcal{E}$ ciphertexts in each pair.

3.4. Homomorphic encryption unbound: Recryption and bootstrapping

Suppose we have a bounded system $\mathcal{E}$ that is $\mathcal{F}$-homomorphic. Can we use $\mathcal{E}$ to get an FHE system $\mathcal{E}_{\mathsf{FHE}}$? Is there some "special" function f such that, if $f \in \mathcal{F}$, we automatically get FHE?

Here is a crazy idea for the "special" function f: *the system's own decryption function* D*!* D is a function, expressible as a circuit, that takes a secret key sk and ciphertext c as input, and outputs a message m. So, can D be in $\mathcal{F}$? Does this sort of self-embedding lead to impossibilities, as in Gödel's Incompleteness Theorem and Turing's Halting Problem? Or, does the self-embedding actually work, and what are the consequences? We will see that, if a homomorphic encryption system can evaluate its own decryption function, plus "a little bit more," we can *bootstrap* the system to obtain a fully homomorphic system.

First, let us work out what happens when $\mathsf{D} \in \mathcal{F}$, and we evaluate D "inside" V. We start with the commutative diagram in Figure 1a for a bounded homomorphic encryption system. In the diagram, $\mathcal{C}_{\mathsf{E}}$ denotes the image of the encryption algorithm E—i.e., "fresh" ciphertexts—and $\mathcal{C}_{\mathsf{V}}$ denotes the superset of $\mathcal{C}_{\mathsf{E}}$ of "evaluated" ciphertexts. The commutative diagram captures the correctness requirement on V with respect to functions from $\mathcal{F}$. Assuming $\mathsf{D} \in \mathcal{F}$, we are interested in what happens when we start with some value in the upper-left corner, and apply D homomorphically (inside V). Since the diagram is commuta-

$$\begin{tikzcd}
\begin{array}{c}\vec{S} = \mathsf{E}(ek, \vec{sk'}) \\ \vec{X} = \mathsf{E}(ek, \vec{c'})\end{array} \arrow[rrr] & & & \begin{array}{c}c = \mathsf{V}(evk, \mathsf{D}(\cdot,\cdot), \vec{S}, \vec{X}) \\ \text{such that } m = \mathsf{D}(sk, c)\end{array} \\
& \mathcal{C}_{\mathsf{E}}^{k+\ell} \arrow[r, "\mathsf{V}(evk, f, \cdot, \ldots, \cdot)"] \arrow[d, "\mathsf{D}(sk, \cdot, \ldots, \cdot)"] & \mathcal{C}_{\mathsf{V}} \arrow[d, "\mathsf{D}(sk, \cdot)"] & f \in \mathcal{F} \\
& \mathcal{M}^{k+\ell} \arrow[r, "f = \mathsf{D}(\cdot,\cdot)"] & \mathcal{M} & \\
\begin{array}{c}\vec{sk'} \in \mathcal{M}^k \\ \vec{c'} \in \mathcal{M}^\ell\end{array} \arrow[uuu] & & & m = \mathsf{D}(sk', c')
\end{tikzcd}$$

FIGURE 2
Recryption: Evaluating decryption homomorphically.

tive, we can gain useful information about what happens by also considering the lower path through the diagram.

Accordingly, now let us assign the bottom rightward arrow f to be $\mathsf{D}(\cdot,\cdot)$ and see what happens as we complete the diagram. Follow along in Figure 2. Though the input to $f = \mathsf{D}(\cdot,\cdot)$ could be arbitrary, the natural input to D is a pair (sk', c') such that sk' is a secret key and c' is a ciphertext with $m = \mathsf{D}(sk', c')$. (Note that sk' might, or might not, equal the key sk that is already in the diagram.) We put those values in the lower-left and lower-right. For this to make sense in the diagram, sk' and c' must be expressible as strings over the message set $\mathcal{M}$. So, suppose that they can be expressed as strings in $\mathcal{M}^k$ and $\mathcal{M}^\ell$, respectively, and let us write them as vectors, $\vec{sk'}$ and $\vec{c'}$. (This reexpression is especially straightforward when $\mathcal{M}$ is simply $\mathbb{Z}/2\mathbb{Z}$, i.e., when messages are bits.) We abuse notation by using D even when the domain is $\mathcal{M}^{k+\ell}$. Continuing to complete the diagram, for the left downward arrow to hold, we need the upper-left to be fresh encryptions of the secret key and ciphertext "bits." So, we will assume for now that ek is public, and we set $\vec{S} = \mathsf{E}(ek, \vec{sk'})$ and $\vec{X} = \mathsf{E}(ek, \vec{c'})$. (We have omitted the randomness of encryption.) Finally, in the top rightward arrow, we apply V with $f = \mathsf{D}(\cdot,\cdot)$ to the freshly encrypted bits of the secret key sk' and ciphertext c', to obtain a ciphertext c in the upper-right corner. Recall that $\mathsf{D} \in \mathcal{F}$ by assumption, so V is guaranteed to satisfy the correctness requirement. So, what does c encrypt? By the commutativity of the diagram, it must encrypt m, just like the original ciphertext c'! If $\mathsf{D} \in \mathcal{F}$, then given an initial ciphertext c' that encrypts m under sk', we can produce a new ciphertext c that encrypts m under sk, by running decryption homomorphically (inside V) using encryptions of bits of the secret key sk' and the ciphertext c'. This process is called *recryption*.

Interestingly, $\vec{X} = \mathsf{E}(ek, \vec{c'})$ is a "double encryption" of m, with the inner encryption under ek', and the outer encryption under ek. Starting with $\vec{X}$ and then taking the down-then-right path through the diagram, we remove the outer encryption first (as you would expect), then the inner encryption, to recover m. Taking the right-then-down path, we remove the inner encryption under ek' first, then the outer encryption!

$$\begin{tikzcd}
\begin{array}{c} \vec{S} = \mathsf{E}(ek, \vec{sk}') \\ \vec{X}^{(1)} = \mathsf{E}(ek, \vec{c}^{(1)}) \\ \vec{X}^{(2)} = \mathsf{E}(ek, \vec{c}^{(2)}) \end{array} \arrow[rrr] & & & \begin{array}{l} c = \mathsf{V}(evk, f, \vec{S}, \vec{X}^{(1)}, \vec{X}^{(2)}) \\ \text{s.t. } \mathsf{NAND}(m^{(1)}, m^{(2)}) = \mathsf{D}(sk, c) \end{array} \\
& \mathcal{C}_{\mathsf{E}}^{k+2\ell} \arrow[r, "\mathsf{V}(evk, f, \cdot, \dots, \cdot)"] \arrow[d, "\mathsf{D}(sk, \cdot, \dots, \cdot)"] & \mathcal{C}_{\mathsf{V}} \arrow[d, "\mathsf{D}(sk, \cdot)"] & f \in \mathcal{F} \\
& \mathcal{M}^{k+2\ell} \arrow[r, "f = \mathsf{NAND}(\mathsf{D}(\cdot,\cdot), \mathsf{D}(\cdot,\cdot))"] & \mathcal{M} & \\
\begin{array}{l} \vec{sk}' \in \mathcal{M}^k \\ \vec{c}^{(1)}, \vec{c}^{(2)} \in \mathcal{M}^\ell \end{array} \arrow[uuu] & & & \begin{array}{l} m^{(1)} = \mathsf{D}(sk', c'^{(1)}) \\ m^{(2)} = \mathsf{D}(sk', c'^{(2)}) \end{array}
\end{tikzcd}$$

FIGURE 3
Recryption-then-NAND and Bootstrappable Homomorphic Encryption.

You may be underwhelmed. What have we gained by recryption? The homomorphic encryption system probably already provides us with much simpler ways to obtain an encryption of the same message, such as applying homomorphic addition $\boxplus$ with an encryption of 0, or homomorphic multiplication $\boxtimes$ with an encryption of 1. Why bother with the more complex process?

First, notice that we do not have any guarantee that c' can participate correctly in even simple operations such as $\boxplus$ and $\boxtimes$. The ciphertext c' is not necessarily a fresh encryption. On the other hand, recryption does not perform additional operations on c'. Instead, it performs operations on *fresh* encryptions from $\mathcal{C}_\mathsf{E}$ that encrypt the bits of sk' and c'. Recryption is guaranteed to work correctly assuming $\mathsf{D} \in \mathcal{F}$.

Second, as hinted above, a good motivation for getting a new encryption of the same message would be if the new ciphertext is "refreshed," i.e., it can participate correctly in more operations, guaranteed. Trivially, we could refresh c' by decrypting it then applying E to obtain a fresh encryption of the same message. But this process requires sk', which we want to keep secret. So we arrive at the real motivation for considering recryption: maybe it can refresh a ciphertext by decrypting it *homomorphically*, requiring only an *encrypted* secret key.

If a recrypted ciphertext c is indeed refreshed, it means we should be able to apply some additional operation after recryption, such as a NAND gate (if the messages are bits). So, assume that the messages are bits and that the function $f = \mathsf{NAND}(\mathsf{D}(\cdot,\cdot), \mathsf{D}(\cdot,\cdot))$ is in $\mathcal{F}$. This function f takes as input a secret key sk' and two ciphertexts $c'^{(1)}, c'^{(2)}$, decrypts the two ciphertexts with the secret key, and applies the NAND gate to the two messages. Since this f is in $\mathcal{F}$ by assumption, we have the guarantee that if $m^{(1)} = \mathsf{D}(sk', c'^{(1)})$ and $m^{(2)} = \mathsf{D}(sk', c'^{(2)})$, then going clockwise through the commutative diagram in Figure 3 gives us a ciphertext c such that $\mathsf{NAND}(m^{(1)}, m^{(2)}) = \mathsf{D}(sk, c)$. By using this process for every NAND gate, we can evaluate an arbitrary circuit of NAND gates. Recall that NAND is, by itself, a functionally complete gate, enabling general computation. Therefore, we obtain a fully homomorphic encryption system.

$$\begin{tikzcd}
\vec{S} = \mathsf{E}(ek, \vec{sk'}) \arrow[rr] & & \begin{array}{c} c = \mathsf{V}(evk, \mathsf{D}_{c'}(\cdot), \vec{S}) \\ \text{such that } m = \mathsf{D}(sk, c) \end{array} & \\
& \mathcal{C}_{\mathsf{E}}^k \arrow[r, "\mathsf{V}(evk, f, \cdot, \dots, \cdot)"] \arrow[d, "\mathsf{D}(sk, \cdot, \dots, \cdot)"] & \mathcal{C}_{\mathsf{V}} \arrow[d, "\mathsf{D}(sk, \cdot)"] & f \in \mathcal{F} \\
& \mathcal{M}^k \arrow[r, "f = \mathsf{D}_{c'}(\cdot)"] & \mathcal{M} & \\
\vec{sk'} \in \mathcal{M}^k \arrow[uuu] & & m = \mathsf{D}(sk', c') &
\end{tikzcd}$$

FIGURE 4

Recryption variant: Evaluating decryption homomorphically with ciphertext prewired.

To get FHE, all we need is a bounded homomorphic encryption system such that this weird function $\mathsf{NAND}(\mathsf{D}(\cdot,\cdot), \mathsf{D}(\cdot,\cdot))$ is in $\mathcal{F}$! We call such a bounded homomorphic encryption system *bootstrappable*, and call this process *bootstrapping*.

In retrospect, bootstrapping seems like the almost inevitable answer for how to refresh a ciphertext generically. Generically, if all we are given is the commutative diagram of a bounded homomorphic encryption system (with its function set $\mathcal{F}$) and a ciphertext c in $\mathcal{C}_{\mathsf{V}}$ but not $\mathcal{C}_{\mathsf{E}}$ that we want to refresh, the only possible way to refresh it is to somehow use V. We cannot input c as a ciphertext directly into V, since it is not in $\mathcal{C}_{\mathsf{E}}$. Yet, we must give V some inputs that retain the information in c, and that V can operate on correctly. We have only two choices: either c must be embedded in *fresh* ciphertexts input to V (e.g., encrypted bitwise, as above) or c must be embedded in the function f that we give to V to evaluate. In either case, since the only useful thing we know about c is that it is in $\mathcal{C}_{\mathsf{V}}$ and encrypts some m, it seems that the only meaningful functions we can evaluate on c are functions that first decrypt c (and thereafter perform some operations on m). And so we arrive at bootstrapping. (This reasoning does not preclude nongeneric techniques for refreshing ciphertexts.)

Embedding the ciphertext(s) to be refreshed in the function to be evaluated—as opposed to encrypting the bits of these ciphertexts—is actually preferable. If we do not encrypt the bits of ciphertext(s), we do not need ek to be public (the encryption system can be symmetric). Also, we do not need to reexpress ciphertexts in terms of message "bits." In detail, in this approach, instead of evaluating $\mathsf{D}(\cdot,\cdot)$, we can evaluate the function $\mathsf{D}_{c'} = \mathsf{D}(\cdot, c')$, where c' comes "prewired." Similarly, we can replace $\mathsf{NAND}(\mathsf{D}(\cdot,\cdot), \mathsf{D}(\cdot,\cdot))$ with $\mathsf{NAND} \circ \mathsf{D}_{c'^{(1)}, c'^{(2)}}(\cdot)$, a function that when given sk' as input, decrypts ciphertexts $c'^{(1)}$ and $c'^{(2)}$ and then NANDs their respective messages. We provide revised versions of recryption and recrypt-then-NAND in Figures 4 and 5.

Now, let us state our FHE result a bit more formally.

Definition 14 (Bootstrappable homomorphic encryption). A (possibly bounded) homomorphic encryption system $\mathcal{E}$ with function set $\mathcal{F}$ is *bootstrappable* if there is a functionally complete set of binary gates Γ, such that for all $g \in \Gamma$, and all ciphertexts $c^{(1)}, c^{(2)} \in \mathcal{C}_{\mathsf{V}}$, $g \circ \mathsf{D}_{c^{(1)}, c^{(2)}}(\cdot) \in \mathcal{F}$.

$$\begin{tikzcd}
\vec{S} = \mathsf{E}(ek, \vec{sk'}) \arrow[rrr] & & & {\begin{array}{l} c = \mathsf{V}(evk, f, \vec{S}) \text{ such that} \\ \mathsf{NAND}(m^{(1)}, m^{(2)}) = \mathsf{D}(sk, c) \end{array}} \\
& \mathcal{C}_{\mathsf{E}}^{k} \arrow[r, "{\mathsf{V}(evk, f, \cdot, \ldots, \cdot)}"] \arrow[d, "{\mathsf{D}(sk, \cdot, \ldots, \cdot)}"] & \mathcal{C}_{\mathsf{V}} \arrow[d, "{\mathsf{D}(sk, \cdot)}"] & f \in \mathcal{F} \\
& \mathcal{M}^{k} \arrow[r, "{f = \mathsf{NAND} \circ \mathsf{D}_{c'^{(1)}, c'^{(2)}}(\cdot)}"] & \mathcal{M} & \\
\vec{sk'} \in \mathcal{M}^{k} \arrow[uuu] & & {\begin{array}{l} m^{(1)} = \mathsf{D}(sk', c'^{(1)}) \\ m^{(2)} = \mathsf{D}(sk', c'^{(2)}) \end{array}} &
\end{tikzcd}$$

FIGURE 5
Recryption-then-NAND and Bootstrappable Homomorphic Encryption, with two ciphertexts prewired; $\mathsf{NAND} \circ \mathsf{D}_{c'^{(1)}, c'^{(2)}}(\cdot)$ is a function that, when given sk' as input, decrypts ciphertexts $c'^{(1)}$ and $c'^{(2)}$ and then NANDs their respective messages.

We limited the above definition to binary gates for convenience, and because it usually seems to suffice in practice. Figure 5 depicts a bootstrappable homomorphic encryption system where Γ includes NAND, which is functionally complete by itself.

Theorem 4 (Bootstrapping to leveled FHE [40, 41]). *If $\mathcal{E}$ is a bootstrappable system that is IND-CPA secure, then it can be transformed into a leveled FHE system $\mathcal{E}_{LFHE}$ that is IND-CPA secure.*

Theorem 4 gives us leveled FHE, not unbounded FHE. For unbounded FHE, there is an issue in the proof of IND-CPA security. Specifically, to obtain unbounded FHE, we need for it to be secure to encrypt $\mathcal{E}$'s secret key sk under its companion encryption key ek. We call this property *circular security*. (For leveled FHE, we can avoid the circular security issue by encrypting each $sk^{(i)}$ under the *next* encryption key $ek^{(i-1)}$, so that the encrypted secret keys form an acyclic chain.)

Definition 15 (Circular security). A encryption system $\mathcal{E}$ is circular secure if it is IND-CPA secure even when $\{\vec{S} = \mathsf{E}(ek, \vec{sk})\}$ is public, where $\vec{sk} \in \mathcal{M}^k$ are the "bits" of the secret key sk.

Theorem 5 (Bootstrapping to unbounded FHE [40, 41]). *If $\mathcal{E}$ is a bootstrappable system that is circular secure, then it can be transformed into an unbounded FHE system $\mathcal{E}_{FHE}$ that is circular secure.*

Let us prove Theorem 5 first, because it is simpler.

Proof of Theorem 5. The construction of $\mathcal{E}_{\text{FHE}}$ is given below; $\mathcal{E}_{\text{FHE}}$ is unbounded, since $\mathcal{E}_{\text{FHE}}.\mathsf{V}$ outputs a ciphertext in $\mathcal{C}_{\mathsf{V}}$ (the set of evaluated ciphertexts of $\mathcal{E}$) whenever the input ciphertexts are in $\mathcal{C}_{\mathsf{V}}$. It correctly evaluates any gate g in the functionally complete set Γ (in time polynomial in the security parameter). Therefore, it is fully homomorphic.

The circular security of $\mathcal{E}_{\text{FHE}}$ follows directly from that of $\mathcal{E}$, since the K, E, and D functions of the two systems are the same, the IND-CPA game makes no reference to the V function, and the encrypted secret key $\vec{S}$ is the same in the two systems. ■

Unbounded FHE construction. Let $\mathcal{E} = (\mathcal{E}.\mathsf{K}, \mathcal{E}.\mathsf{E}, \mathcal{E}.\mathsf{D}, \mathcal{E}.\mathsf{V})$ be a circular-secure bootstrappable homomorphic encryption system. We construct a circular-secure FHE system $\mathcal{E}_{\text{FHE}} = (\mathcal{E}_{\text{FHE}}.\mathsf{K}, \mathcal{E}_{\text{FHE}}.\mathsf{E}, \mathcal{E}_{\text{FHE}}.\mathsf{D}, \mathcal{E}_{\text{FHE}}.\mathsf{V})$ as follows. (Below, we will omit the randomness of the encryptions.)

- $\mathcal{E}_{\text{FHE}}.\mathsf{K}$: Identical to $\mathcal{E}.\mathsf{K}$. Note that $\mathcal{E}$ publishes $\vec{S} = \mathsf{E}(ek, \vec{sk})$, which we call part of the evaluation key evk.
- $\mathcal{E}_{\text{FHE}}.\mathsf{E}$: Identical to $\mathcal{E}.\mathsf{E}$.
- $\mathcal{E}_{\text{FHE}}.\mathsf{D}$: Identical to $\mathcal{E}.\mathsf{D}$.
- $\mathcal{E}_{\text{FHE}}.\mathsf{V}(evk, g, c^{(1)}, c^{(2)})$: output $\mathcal{E}.\mathsf{V}(evk, g \circ \mathsf{D}_{c^{(1)},c^{(2)}}(\cdot), \vec{S})$.

So, we have a very clean construction of unbounded circular-secure FHE from circular-secure bootstrappable encryption. Unfortunately, we do not understand circular security very well. Indeed, we can construct encryption systems that are IND-CPA secure (under natural assumptions), but which break completely when an encryption of the secret key is published. These systems tend to be contrived (specifically designed to break), but still these counterexamples are sobering. Also, encryption systems that are *provably* circular secure (based on natural assumptions) are rare compared to provably IND-CPA secure systems. More to the point, and looking ahead, we know how to build IND-CPA secure bootstrappable encryption based on well-studied assumptions about the hardness of computational problems over integer lattices, while the assumptions underlying current unbounded FHE systems are less well-understood.

With that motivation, let us now prove Theorem 4, which says that we can get IND-CPA secure *leveled* FHE from IND-CPA secure bootstrappable encryption.

The proof uses a common technique in cryptographic security proofs called a *hybrid argument*. In a hybrid argument, in each step we change what one ciphertext encrypts—e.g., from encrypting a secret key to encrypting 0—and prove that if $\mathcal{E}$ is IND-CPA secure then an adversary should not notice the difference. By the end of the hybrid argument, all purported encrypted secret keys are in fact encryptions of 0, and are therefore useless to the adversary. The hybrid argument works for a leveled system because the encrypted secret keys $\vec{S}^{(i)}$ form an acyclic chain: the secret key at level i is encrypted under the encryption key at level $i - 1$. If the encrypted secret keys form a loop, the hybrid argument does not go through. In particular, the first time we replace an encrypted secret key with an encryption of 0, we break the key loop, and this change may be efficiently distinguishable if the system is not circular secure.

Proof of Theorem 4. The construction of $\mathcal{E}_{\text{LFHE}}$ is given below. By Lemma 1, $\mathcal{E}_{\text{LFHE}}$ is a leveled FHE system. The IND-CPA security of $\mathcal{E}_{\text{LFHE}}$ follows from Lemma 7, which says that if there is an adversary $\mathcal{A}$ in the IND-CPA game against $\mathcal{E}_{\mathrm{LFHE}}$ with n levels that has advantage ε, then there is an adversary $\mathcal{B}$ in the IND-CPA game against $\mathcal{E}$ that has advantage at least $\varepsilon/2(n+1)$, and runs in about the same time as $\mathcal{A}$. ■

Leveled FHE construction. Let $\mathcal{E} = (\mathcal{E}.\mathsf{K}, \mathcal{E}.\mathsf{E}, \mathcal{E}.\mathsf{D}, \mathcal{E}.\mathsf{V})$ be the bootstrappable system with function set $\mathcal{F}$ including functionally complete gates Γ. We construct a leveled FHE system

$$\mathcal{E}_{\mathrm{LFHE}} = (\mathcal{E}_{\mathrm{LFHE}}.\mathsf{K}, \mathcal{E}_{\mathrm{LFHE}}.\mathsf{E}, \mathcal{E}_{\mathrm{LFHE}}.\mathsf{D}, \mathcal{E}_{\mathrm{LFHE}}.\mathsf{V})$$

as follows. (Below, we will omit the randomness of the key generations and encryptions.)

- $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{K}$: Let $n \in \mathbb{N}$ be the number of levels specified in params. For $i \in \{0, \ldots, n\}$, compute key tuples $(sk^{(i)}, ek^{(i)}, evk^{(i)}) \leftarrow \mathcal{E}.\mathsf{K}(\lambda, i)$. For $i \in \{1, \ldots, n\}$, set $\vec{S}^{(i)} = \mathsf{E}(ek^{(i-1)}, \vec{sk}^{(i)})$. For $i \in \{1, \ldots, n\}$, set $evk'^{(i-1)} = (evk^{(i-1)}, \vec{S}^{(i)})$.

- $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{E}$: Identical to $\mathcal{E}.\mathsf{E}$ using $ek^{(n)}$ and attaching the label n to the ciphertext.

- $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{D}$: The ciphertext c comes with a label in $i \in \{0, \ldots, n\}$. Output $\mathcal{E}.\mathsf{D}(sk^{(i)}, c)$.

- $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{V}$: Given gate g and two ciphertexts $c^{(1)}, c^{(2)}$ with label i, output

$$\mathcal{E}_{\mathrm{LFHE}}.\mathsf{V}(evk'^{(i-1)}, g, c^{(1)}, c^{(2)}) = \mathcal{E}.\mathsf{V}\big(evk^{(i-1)}, g \circ \mathsf{D}_{c^{(1)}, c^{(2)}}(\cdot), \vec{S}^{(i)}\big),$$

and set the label of the ciphertext to $i-1$.

Lemma 1. *If $\mathcal{E}$ is bootstrappable, the system $\mathcal{E}_{LFHE}$ described above is leveled fully homomorphic.*

Proof of Lemma 1. Recall that $\mathcal{C}_{\mathsf{E}}^{(i)}$ is the image of $\mathcal{E}.\mathsf{E}$ with encryption key $ek^{(i)}$ and $\mathcal{C}_{\mathsf{V}}^{(i)}$ is the superset of $\mathcal{C}_{\mathsf{E}}^{(i)}$ that also includes ciphertexts in the image of $\mathcal{E}.\mathsf{V}$ under $evk^{(i)}$ with functions from $\mathcal{F}$ and ciphertexts from $\mathcal{C}_{\mathsf{E}}^{(i)}$. $\mathcal{E}_{\mathrm{LFHE}}$ is leveled, as depicted in Figure 1b, since $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{V}$ outputs a ciphertext in $\mathcal{C}_{\mathsf{V}}^{(i-1)}$ whenever the input ciphertexts are in $\mathcal{C}_{\mathsf{V}}^{(i)}$.

Moreover, $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{V}$ correctly evaluates any gate g in the functionally complete set Γ (in time polynomial in the security parameter). So, it is a leveled FHE system. (Note that the system must perform some bookkeeping relating to the labels of the ciphertexts, but this is not part of the actual functions $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{K}$, $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{E}$, $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{D}$, and $\mathcal{E}_{\mathrm{LFHE}}.\mathsf{V}$, and does not contribute to their complexity.) ■

We provide Lemma 7 and its proof in Appendix B.

3.5. Computational hardness, cryptanalysis, and learning

The aspect of homomorphic encryption (and cryptography in general) that is probably hardest to understand is *computational hardness*.

Computational hardness is often described in terms of the P vs. NP question. Roughly speaking, P consists of problems that can be solved (on a Turing machine) in time polynomial in the size of the problem instance. For example, two λ bit numbers can be multiplied together in time $O(\lambda^2)$ using grade-school multiplication, or even time $O(\lambda \cdot \log \lambda)$ using more sophisticated techniques [49]. NP consists of problems for which a solution (together with a "witness" or proof) can be verified in polynomial time. For example, integer factorization is in NP, since given a nontrivial factorization (p, q) of integer N, one can verify in polynomial time that $N = p \cdot q$ via multiplication (which is polynomial time). On the other hand, integer factorization is not known to be in P, since there are no known polynomial-time algorithms for *finding* p and q.

We have no proof that P $\neq$ NP, and therefore nobody knows whether computational hardness (of the type needed for the security of public-key encryption systems) even exists. Certainly if P $=$ NP, public-key encryption systems are insecure. In this case, one could efficiently *find* the randomness r used in key generation, since one can efficiently *verify* running K with randomness r indeed generates the targeted key pair. Even if we assume P $\neq$ NP, this assumption provides little support for public-key encryption systems, which rely on the hardness of problems that are unlikely to be NP-complete.

In the absence of helpful lower bounds, we are forced to turn to upper bounds. That is, we consider well-studied problems—such as integer factorization, the discrete logarithm problem, and finding short vectors in integer lattices—for which the fastest known algorithms run in time superpolynomial (preferably exponential) in the size of the problem instance. Then, we assume that the best known algorithms are not much worse than the best possible algorithms, and base the security of our cryptosystem on the assumed hardness of the well-studied problem. Even for well-studied problems, this approach can be precarious. For example, although the integer factorization problem has been considered for centuries, a dramatic algorithmic improvement came in 1990 with the invention of the number field sieve [57], which factors λ-bit integers in subexponential time $\exp(O(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}})$, considerably faster than the previous quadratic sieve algorithm, which takes time $\exp(O(\log N)^{\frac{1}{2}}(\log\log N)^{\frac{1}{2}})$.

Another surprise has been quantum computation. In 1994, Shor [71] made quantum computation famous by discovering an efficient quantum algorithm for problems including integer factorization and discrete logarithm. More generally, we now have the following result by Watrous [78]:

Theorem 6 (Watrous). *Let G be a solvable (e.g., abelian) group, given by generators. There is a polynomial-time quantum algorithm to compute $|G|$ (with small error probability).*

Corollary 1 (Armknecht et al. [10]). *Group homomorphic encryption systems in which the ciphertext set is a finite solvable group and decryption is a group homomorphism cannot be IND-CPA secure against a quantum adversary.*

The corollary follows because Watrous' algorithm allows one to distinguish between the entire group G of ciphertexts and the proper subgroup H of encryptions of 1 (the kernel of the decryption map). The attack also applies when the ciphertext set is a finite ring and decryption is a ring homomorphism, since the encryptions of 0 form an ideal that is an abelian additive subgroup of the ciphertext set.

There are also efficient quantum attacks [77] (and subexponential classical attacks [18]) on FHE systems that have a zero oracle—that is, an efficient method to distinguish when a ciphertext encrypts 0. But we are anyway only interested in IND-CPA secure systems, for which no such oracle can exist.

Currently there are no general attacks on homomorphic encryption systems based on nonsolvable groups, but there are also no known plausibly-secure constructions. Again, for the construction to be secure, one must ensure that the ciphertext group G and the proper subgroup H of encryptions of 1 are hard to distinguish. A homomorphic encryption system using nonsolvable groups must avoid at least the following attacks:

- Solvability of H: If H is solvable and G is not, they can be distinguished easily by computing their respective derived series.
- Watrous' order finding algorithm: Even if G and H are both nonsolvable, Watrous' algorithm can distinguish them if the cyclic subgroups generated by randomly sampled elements $g \leftarrow G$ and $h \leftarrow H$ have distinguishably different distributions of orders.
- Linear representations: If one can efficiently compute (or one is given) linear representations of G and H, one may be able to distinguish G and H using linear algebra.

It may seem obvious, but it is an essential point: in a secure encryption system, the decryption function cannot be linear, since linear decryption leads to a trivial linear algebra attack (e.g., Gaussian elimination).[5]

More generally, the decryption function cannot be *learnable*, in the sense of Valiant's "probabilistically approximately correct" (PAC) learning model [75].[6] In the PAC learning model, one is given samples $(x, f(x))$ with x coming from a training set X, and the goal is to learn f well-enough to output $f(x)$ with high probability on a new sample x. This model is nearly identical to the IND-CPA game, where we use $f = \mathsf{D}(sk, \cdot)$.

Since the models are so similar, we can look to learning theory to help us design a decryption function for an IND-CPA secure system [16]. For example, Linial, Mansour, and Nisan [58] give an algorithm to learn a function expressible as an AC circuit of size s, depth d, and n inputs with accuracy parameter ε in time $n^{O(\log s/\varepsilon)^d}$. (AC circuits are Boolean circuits that have AND and OR gates with arbitrarily many inputs, as well as NOT gates.) So, we cannot have decryption be a constant depth AC circuit if we want (as we usually do) it to take time exponential (or at least subexponential) in the security parameter λ for an adversary to break our system.

5 There are a surprising number of FHE proposals without proofs of security, and they are almost always insecure for the simple reason that decryption is linear.

6 Interestingly, the decryption function of a secure bootstrappable encryption system must satisfy an interesting dichotomy: it must be simultaneously *unlearnable* (complex) and *evaluatable* (not too complex).

On the other hand, learning theory suggests that it can be difficult to learn functions from samples that are *noisy*, i.e., from samples $(x, f(x) + e)$, where e is some *error* or *noise*. Accordingly, two learning problems that have become useful to cryptography are the *learning with errors (LWE)* problem [65], and the *learning parity with noise (LPN)* problem, where one is tasked with distinguishing whether given samples are completely random or have the form $(\vec{a}_i, \langle\vec{a}_i, \vec{s}\rangle + e_i)$, where $\vec{s}$ is a secret vector and the e_i's are random noise values. In LWE the vectors are over $\mathbb{Z}/q\mathbb{Z}$ and $|e_i| \ll q$, while in LPN the vectors are over $\mathbb{Z}/2\mathbb{Z}$ and the e_i's are Bernoulli random variables that are usually 0 but sometimes 1. While the inner product is linear, the noise introduces nonlinearity, in particular, it defeats linear algebra. As far as we know, these learning problems are hard even for quantum adversaries. The security of all current FHE constructions relies on the hardness of such learning with noise problems.

4. NOISY CONSTRUCTIONS OF FULLY HOMOMORPHIC ENCRYPTION

The simple FHE systems in Section 2 are unbounded, but only for a single operation that is not functionally complete. How can we construct an unbounded system capable of evaluating, say, $\{+, \times\}$?

For evaluating $\{+, \times\}$, a natural approach to try is a ring homomorphism. However, as we saw in Section 3.5, a system in which decryption is a ring homomorphism can be broken efficiently by a quantum adversary. Moreover, ring homomorphisms have a linearity that may be exploited even by classical adversaries.

How can we defeat linear algebra attacks? As we saw in Section 2.2, one answer is to put values "in the exponent." Unfortunately, in groups for which the Diffie–Hellman assumption holds, values in the exponent can be added efficiently but not multiplied.[7]

As we saw in Section 3.5, another way to defeat linear algebra is to add "noise" or "errors" to the linear equations. Linear algebra is notoriously brittle against noise. As we will see, while adding noise "hardens" the homomorphism, this security comes at a cost: the noise turns our unbounded ring homomorphism into a bounded one. Fortunately, by calibrating the noise level, we can make the bounded homomorphism bootstrappable to achieve FHE while basing security on reasonable hardness assumptions.

4.1. Overview of the noisy approach

Virtually all known constructions of FHE follow essentially the same blueprint:[8] (1) construct bootstrappable encryption by (perhaps implicitly) starting from an insecure

7 Some groups for which the discrete logarithm is hard feature a bilinear map—such as a Weil or Tate pairing—that effectively allows one multiplication "in the exponent" (see [17, 54]). Cryptographically-secure multilinear maps are an ongoing area of research [19, 38].

8 Some constructions avoid this blueprint by constructing FHE from *cryptographic program obfuscation* [14, 26, 31, 39, 53]. While much progress has been made on basing obfuscation systems on well-established computational assumptions [53], all current constructions of obfuscation still rely on the hardness of learning with noise problems, and are less efficient and more complicated than more "direct" constructions of FHE.

unbounded homomorphism that respects some functionally complete set of gates and then "hardening" the homomorphism with noise, and (2) invoke the bootstrapping theorems (Theorems 4 and 5) to get FHE from bootstrappable encryption. Here we sketch a fairly general technique for hardening a homomorphism with noise.

For convenience, let us fix our message space to $\mathcal{M} = \mathbb{Z}/2\mathbb{Z}$ and our gates to $\{+, \times\}$. We start with an unbounded homomorphism that respects $\{+, \times\}$:

$$\begin{tikzcd}
\mathcal{C} \times \mathcal{C} \arrow[r, "\boxplus{,}\boxtimes"] \arrow[d, "\delta_{sk}(\cdot){,}\delta_{sk}(\cdot)"] & \mathcal{C} \arrow[d, "\delta_{sk}(\cdot)"] \\
\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \arrow[r, "+{,}\times"] & \mathbb{Z}/2\mathbb{Z}.
\end{tikzcd}$$

For example, $\mathcal{C}$ might be $\mathbb{Z}[x]$, and $\delta_{sk}(\cdot)$ could be evaluation of the ciphertext polynomial at the secret key $sk \in \{0,1\}^n$ modulo 2. As another example, $\mathcal{C}$ could be a set of integer matrices that all have sk as an eigenvector with integer eigenvalue, $\boxplus$ and $\boxtimes$ could be matrix addition and multiplication, and decryption could be recovering the eigenvalue modulo 2.

In these examples, the homomorphism is unbounded but unsuitable for cryptography. For the polynomial evaluation homomorphism, one problem is that as we apply $\boxtimes$, the degree increases and the number of monomials can increase exponentially.[9] To control the number of monomials, one approach is to publish a Gröbner basis G for the ideal $I(sk) = \{p(x) : p(sk) = 0\}$ of polynomials that evaluate to 0 at sk. But while keeping the monomial basis small helps efficiency, it opens up a trivial linear algebra attack to recover sk. The matrix-based system is also breakable via linear algebra.

To harden these homomorphisms, we add *noise*. Here is one way to do it. First, we make a trivial observation: if we replace δ_{sk} with $\lfloor \delta_{sk} \rceil$ in the above diagram, nothing changes since δ_{sk} is already integral over $\mathcal{C}$. But now let us expand the set of ciphertexts so that $\delta_{sk}(c)$ is *not necessarily integral*. Rather we let ciphertexts be *noisy*. We refer to the value $\delta_{sk}(c) - \lfloor \delta_{sk}(c) \rceil$ as the *noise* of the ciphertext c. Now decryption involves applying $\delta_{sk}(c)$, removing the noise to obtain $\lfloor \delta_{sk}(c) \rceil$, and then reducing modulo 2. But now we must ask: do the $\boxplus$, $\boxtimes$ operations "play well" with the noise? Starting from fresh ciphertexts in $\mathcal{C}_{\mathsf{E}}$, which presumably have a small amount of noise, how many (possibly modified) $\boxplus$, $\boxtimes$ operations can we apply with the guarantee that the following diagram commutes?

$$\begin{tikzcd}
\mathcal{C}_{\mathsf{E}}^{*} \arrow[r, "\text{bounded number of tweaked } \boxplus{,}\boxtimes"] \arrow[d, "\lfloor\delta_{sk}(\cdot)\rceil{,}\lfloor\delta_{sk}(\cdot)\rceil"] & \mathcal{C}_{\mathsf{V}} \arrow[d, "\lfloor\delta_{sk}(\cdot)\rceil"] \\
\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \arrow[r, "\text{bounded number of } +{,}\times"] & \mathbb{Z}/2\mathbb{Z}
\end{tikzcd}$$

9 Fellows and Koblitz [37] described an encryption system where decryption involves evaluating a ciphertext polynomial at a secret point $\vec{s}$. However, it is not practically usable as a homomorphic encryption system due to this explosion of monomials.

The diagram will commute for function f if evaluating f on fresh ciphertexts—say, ciphertexts of noise at most some ε_0—always results in ciphertexts with noise bounded comfortably below 1/2 that is, the noise is guaranteed not to wrap modulo 1 and result in a possible decryption error. The hope is that $\boxplus$, $\boxtimes$, or tweaked versions of them, play well with the noise and do not amplify it too much, so that if we choose ε_0 well, we can get the diagram to commute for (say) the recrypt-then-NAND function while achieving IND-CPA security based on a reasonable hardness assumption.

Below, we will discuss an instantiation of this framework in detail. Before we describe this construction, we introduce some hardness assumptions related to learning with noise, and show how to construct a symmetric encryption system from one of these assumptions.

4.2. Learning with noise problems

Suppose that p is a large integer. Whenever you press a button, you get an approximate multiple of p, that is, a random integer of the form $x_i = q_i \cdot p + r_i$. Can you recover p? If r_i is always 0, then you can recover p efficiently using the Euclidean algorithm (as soon as you have samples for which the q_i's are relatively prime). But if the "noise" r_i (and other values) are sampled from well-chosen distributions, this problem, called the approximate GCD problem, appears hard.

Definition 16 (Approximate GCD problem [52,76]). Let λ be a security parameter. Let $\alpha = \alpha(\lambda)$, $\beta = \beta(\lambda)$, and $\gamma = \gamma(\lambda)$ be parameters. Fix integer p, sampled as a random integer of β bits. Given arbitrarily many samples $x_i = q_i \cdot p + r_i$, sampled as random γ-bit integers subject to the constraint that $x_i - p\lfloor x_i/p \rceil$ is at most α bits, output p.

Another learning with noise problem is learning parity with noise.

Definition 17 (Learning Parity with Noise (LPN) problem). Let λ be a security parameter. Let $n = n(\lambda)$ be an integer, and $\chi = \chi(\lambda)$ a Bernoulli distribution. Fix a vector $\vec{s} \in (\mathbb{Z}/2\mathbb{Z})^n$ sampled according to χ^n. Given arbitrarily many samples $(\vec{a}_i, b_i)$ where $\vec{a}_i$ is sampled uniformly from $(\mathbb{Z}/2\mathbb{Z})^n$, e_i is sampled from χ, and $b_i \leftarrow \langle \vec{a}_i, \vec{s} \rangle + e_i \bmod 2$, output $\vec{s}$.

(There is some abuse of terms in this description—with mixing of $\mathbb{Z}$, $\mathbb{Z}/2\mathbb{Z}$, and mod 2—but it will be understood that we are really working over $\mathbb{Z}$ and then reducing modulo 2 to representatives of $\mathbb{Z}/2\mathbb{Z}$.)

The LPN problem is easy if the noise e_i is always 0, in which case we can solve for $\vec{s}$ using linear algebra, but the problem appears to be hard for an appropriate choice of parameters. In the normal formulation of LPN, $\vec{s}$ is sampled uniformly from $\{0, 1\}^n$, but Applebaum et al. [9] showed that the problem is just as hard when $\vec{s}$ is sampled from the noise distribution χ.

Finally, we come to the learning with errors problem.

Definition 18 (Learning with Errors (LWE) problem [65]). Let λ be a security parameter. Let $n = n(\lambda)$ and $q = q(\lambda)$ be integers, and $\chi = \chi(\lambda)$ a distribution over $\mathbb{Z}$. Fix a vector

$\vec{s} \in \mathbb{Z}^n$ sampled according to χ^n. Given arbitrarily many samples $(\vec{a}_i, b_i)$ where $\vec{a}_i$ is sampled uniformly from $(\mathbb{Z}/q\mathbb{Z})^n$, e_i is sampled from χ, and $b_i \leftarrow \langle \vec{a}_i, \vec{s} \rangle + e_i \bmod q$, output $\vec{s}$.

Again, if the noise e_i is always 0, we can solve for $\vec{s}$ using linear algebra given enough samples, but the presence of appropriate noise seems to make the problem hard. Typically, χ is taken to be a discrete Gaussian distribution over $\mathbb{Z}$, with deviation $\sigma \ll q$. Note that $\vec{s}$ is chosen according to χ, as Applebaum et al.'s result [9] (mentioned above) applies in this context as well. Rather than referring explicitly to the noise distribution χ, sometimes it is convenient to refer to a bound ε on the size of the noise.

Definition 19 (ε-bounded distributions). A distribution ensemble $\{\chi_n\}_{n\in\mathbb{N}}$, supported over the integers, is called ε-bounded if $\Pr_{e\leftarrow\chi_n}[|e| > \varepsilon]$ is negligible in n.

All of these learning with noise problems are useful for cryptography. As far as we know, they are hard even for quantum adversaries. We have constructions of leveled FHE based on approximate GCD [28,46,76] and LWE [23,24,46], while constructing FHE based on LPN appears to be more difficult (see [21,53]). In this paper, we will focus mainly on LWE, and describe a construction of leveled FHE based on LWE.

Since IND-CPA security is about distinguishing between two distributions, it is helpful to define a decision version of LWE which is also about distinguishing between two distributions.

Definition 20 (Decision LWE). As in Definition 18, except that the challenger sets a random bit $\beta \in \{0, 1\}$, and outputs samples according to one of two distributions:

(1) If $\beta = 0$, it outputs $(\vec{a}_i, b_i)$ as uniform samples from $(\mathbb{Z}/q\mathbb{Z})^{n+1}$.

(2) If $\beta = 1$, it samples them according to the distribution in Definition 18.

The problem is to guess β (with nonnegligible advantage). The $\mathrm{LWE}_{n,q,\chi}$ assumption is that this decision $\mathrm{LWE}_{n,q,\chi}$ problem is hard.

What do we know about the hardness of LWE? For n and q that are polynomial in λ, Regev gave a polynomial-time reduction from search LWE to decision LWE. When the noise is extremely small or has some structure, there are subexponential algorithms to solve LWE [12]. For example, when $e_i \in \{0, 1\}$ for all i, solving LWE is easy given $O(n^2)$ samples, since one can compute $\vec{s} \times \vec{s}$ as the solution to the $O(n^2)$-dimensional system of linear equations given by the equalities $\langle \vec{a}_i, \vec{s} \rangle \cdot (\langle \vec{a}_i, \vec{s} \rangle - 1)$. However, for discrete Gaussian error distributions with σ polynomial in n, the hardness of LWE appears to depend solely on the ratio q/ε.

In particular, the LWE problem has been shown to be as hard on average (for random instances) as certain lattice problems in the worst-case (the hardest instances) [3,4,65]. An n-dimensional lattice is a (full-rank) additive subgroup of $\mathbb{R}_n$. For lattice dimension parameter n and number d, the shortest vector problem GapSVP_γ is the problem of distinguishing whether an n-dimensional lattice has a nonzero vector of Euclidean norm less than d or no nonzero vector shorter than $\gamma(n) \cdot d$. The gist of the theorem below is that if there is a quantum algorithm for average-case n-dimensional LWE for ratio q/ε, then there is a quantum algorithm for worst-case n-dimensional GAPSVP_γ for γ just a little larger than q/ε.

Theorem 7 (Regev [65]). *Let n, q be integers and $\alpha \in (0, q)$ be such that $\alpha > 2\sqrt{n}$. Let χ be a discrete Gaussian distribution over $\mathbb{Z}$ with deviation α. If there exists an efficient algorithm that solves $\mathrm{LWE}_{n,q,\chi}$, then there exists an efficient quantum algorithm that approximates the decision version of the shortest vector problem (GAPSVP) and the shortest independent vectors problem (SIVP) to within $\tilde{O}(n \cdot \frac{q}{\alpha})$ in the worst case.*

A discrete Gaussian with deviation α will be ε-bounded for $\varepsilon = \alpha \cdot \tilde{O}(\sqrt{n})$. Again, as long as the deviation of discrete Gaussian noise exceeds a certain lower bound, the hardness of LWE appears to depend only on the ratio between q and the size of the noise.

GAPSVP_γ is NP-hard for any constant γ, but, unfortunately, in cryptography we need γ to be larger (at least n in the theorem above). For $\gamma = \mathrm{poly}(n)$, the fastest algorithm to solve GAPSVP_γ takes time $2^{O(n)}$. As a crude rule of thumb, the fastest known algorithm to solve GAPSVP_{2^k} takes roughly $2^{n/k}$ time [70]. Interestingly, there are no quantum algorithms for GAPSVP that perform significantly better than classical algorithms.

Looking ahead to the construction of FHE, some reformulations and variants of LWE will be useful. Sometimes, we prefer to view LWE samples as polynomials.

Fact 1. *A sample $(\vec{a}, b)$ such that $b = \langle \vec{a}, \vec{s} \rangle + e$ can be viewed as a degree-1 polynomial $c(\vec{x}) = b - \sum_j a_j \cdot x_j$ such that $c(\vec{s}) = e$.*

Homomorphic multiplication of ciphertexts related to LWE samples becomes more natural when the ciphertexts are viewed as polynomials. The reformulation in Fact 1 suggests a generalization of LWE to higher-degree polynomials.

Definition 21 (LWE with higher degree polynomials). The values $\lambda, n, \chi, \vec{s}$ are as in Definition 18. There is also a $\mathrm{poly}(\lambda)$-size set of polynomials $P \subset \mathbb{Z}[x_1, \ldots, x_n]$ that is fixed independently of $\vec{s}$. As in LWE (as reformulated in Fact 1), we are given an arbitrary number of degree-1 polynomials $c(\vec{x})$ such that $c(\vec{s}) \bmod q$ has distribution χ. In addition, for each $p(\vec{x}) \in P$, we are given a degree-1 polynomial $c(\vec{x})$ such that $c(\vec{s}) - p(\vec{s}) \bmod q$ has distribution χ.

That is, instead of getting just degree-1 polynomials $c(\vec{x})$ that always evaluate to something small at $\vec{s}$, we can now also obtain potentially higher-degree polynomials $c(\vec{x}) - p(\vec{x})$ that always evaluate to something small at $\vec{s}$. This problem is related to the circular security of some LWE-based systems.

4.3. Encryption based on LWE

Regev [65] built an IND-CPA asymmetric encryption system based on decision LWE. Below, we describe a symmetric version of Regev's encryption system. Since the symmetric system is additively homomorphic modulo 2 (and satisfies other conditions), we can apply Rothblum's theorem (Theorem 3) to get an asymmetric system that looks similar to Regev's.

The idea of the system is simple. Suppose that we want to encrypt $m \in \{0,1\}$ under secret $\vec{s} \in (\mathbb{Z}/q\mathbb{Z})^n$. Generate an LWE sample $(\vec{a}, b)$ such that $b = \langle \vec{a}, \vec{s} \rangle + e$. The LWE assumption says that it is hard, without $\vec{s}$, to tell whether $b - \langle \vec{a}, \vec{s} \rangle$ is actually distributed according to χ (with small deviation) or uniformly. In the latter case, b is like a one-time pad, even given $\vec{a}$. Accordingly, to encrypt m, we mask it with b—specifically, we encrypt m as $(\vec{a}, b + m \cdot \lfloor q/2 \rceil)$. The key-holder knows a good approximation of b—namely, $\langle \vec{a}, \vec{s} \rangle$—and therefore can remove b, up to small "noise." Thereafter, it can recover m, whose value is preserved (despite the small noise) in the most significant bit by multiplying it $\lfloor q/2 \rceil$.

More formally, the LWE-based encryption system is as follows.

Symmetric encryption system $\mathcal{E}_{\mathrm{LWE}}$.

- K: takes security parameter λ and randomness r and generates parameters $n = n(\lambda)$, $q = q(\lambda)$, and $\chi = \chi(\lambda)$. It generates secret key $sk = ek = \vec{s} \leftarrow \chi^n$.
- E: takes ek, a message $m \in \{0,1\}$, and randomness r. It generates a random LWE sample $(\vec{a}, b)$. (That is, it generates random $\vec{a} \in (\mathbb{Z}/q\mathbb{Z})^n$, $e \leftarrow \chi$ and sets $b = \langle \vec{a}, \vec{s} \rangle + e$.) It outputs ciphertext $c = (\vec{a}, u)$, where $u = b + m \cdot \lfloor q/2 \rceil \bmod q$.
- D: takes sk and a ciphertext c. It computes $m' \leftarrow u - \langle \vec{a}, \vec{s} \rangle \bmod q$. Depending on whether m' is close to 0 or $\lfloor q/2 \rceil$, output $m = 0$ or $m = 1$.

Regarding correctness, for a well-formed ciphertext we have $m' = b + m \cdot \lfloor q/2 \rceil - \langle \vec{a}, \vec{s} \rangle = e + m \cdot \lfloor q/2 \rceil$, which is close to 0 when $m = 0$, and to $\lfloor q/2 \rceil$ when $m = 1$.

Theorem 8. *$\mathcal{E}_{\mathrm{LWE}}$ is IND-CPA secure based on LWE.*

The proof of security follows the usual format of having an IND-CPA adversary $\mathcal{A}$, an LWE adversary $\mathcal{B}$ who plays the role of the challenger in the IND-CPA game, and an LWE challenger, with $\mathcal{B}$ mostly forwarding slightly modified transmissions between $\mathcal{A}$ and the challenger. If the LWE samples are uniform, the ciphertexts that $\mathcal{B}$ sends to $\mathcal{A}$ will also be uniform, and $\mathcal{A}$ can have no advantage guessing which bit is encrypted. If the LWE samples are well formed, the ciphertexts that $\mathcal{B}$ sends to $\mathcal{A}$ are well formed and $\mathcal{A}$ should have its assumed advantage ε. $\mathcal{B}$ guesses that the LWE samples are well formed if $\mathcal{A}$ guesses correctly, and $\mathcal{B}$ wins with advantage at least $\varepsilon/2$. (The calculation is as in the proof of Theorem 1.)

4.4. Bootstrappable encryption construction

Here we present a bootstrappable encryption system. Historically, the first bootstrappable system was rather complex [41]. But after a sequence of works [6,20,23,24], bootstrappable encryption became simple enough to describe in a blog post [13]. We mostly follow Barak and Brakerski's excellent exposition [13] here.

We start with the LWE-based encryption system $\mathcal{E}_{\mathrm{LWE}}$ of Section 4.3, but we make some cosmetic changes. First, we view ciphertexts as degree-1 polynomials (see Fact 1). This viewpoint will make multiplication of ciphertexts somewhat more natural than if we viewed them as vectors. Second, we basically divide ciphertexts by $q/2$, allowing them to be fractional. Recall that in $\mathcal{E}_{\text{LWE}}$, a ciphertext c (which we will view now as a polynomial) has the property that $c(\vec{s})$ is close to $m \cdot \lfloor q/2 \rfloor$ modulo q. If we divide the ciphertext by $q/2$, we get something more natural—namely, $c(\vec{s})$ is close to m modulo 2, or in other words $\lfloor c(\vec{s}) \rceil = m \bmod 2$. This change allows for a simple description of decryption. It also allows for a simple definition of the "noise" of a ciphertext—namely $c(\vec{s}) - \lfloor c(\vec{s}) \rceil$, the distance of $c(\vec{s})$ to the nearest integer. We write

$$c(\vec{s}) =_{\varepsilon} m \bmod 2$$

to indicate that ciphertext c has noise of magnitude at most $\varepsilon \geq 0$ and $\lfloor c(\vec{s}) \rceil = m \bmod 2$. This notation will simplify the tracking and bounding of ciphertext noise as we apply homomorphic operations $\boxplus, \boxtimes$. We will use ε_0 to denote the noise bound on fresh ciphertexts output by E. We write the system $\mathcal{E}_{\text{boot}}$ below.

Bootstrappable encryption system $\mathcal{E}_{\text{boot}}$.

- K: As in $\mathcal{E}_{\text{LWE}}$, except that we calibrate the parameters $q = q(\lambda)$ and $\chi = \chi(\lambda)$ to achieve bootstrapping, and we set $q = 2^{\kappa+1}$ for some κ. Also, we set evaluation key evk as described below.
- E: Generate c as in $\mathcal{E}_{\text{LWE}}$. Write c as a polynomial $c(\vec{x}) \in \mathbb{Z}[x_1, \ldots, x_n]$ (as described in Fact 1). Divide c by 2^{κ}, i.e., $c(\vec{x}) \leftarrow c(\vec{x})/2^{\kappa}$, a polynomial in $\mathbb{R}[x_1, \ldots, x_n]$. Output $c(\vec{x})$.
- D: Remove as much precision as possible in the coefficients of $c(\vec{x})$ while maintaining correctness before "actual" decryption. Then output $\lfloor c(\vec{s}) \rceil \bmod 2$.
- V: Apply homomorphic operations $\{\boxplus, \boxtimes\}$ for $\{+, \times\}$ modulo 2, as described below.

Notice that we have split decryption into two phases, namely a preprocessing phase where we drop unneeded precision, and a second phase where we do the "actual" decryption. The purpose of the preprocessing is to facilitate bootstrapping: it is important to minimize the complexity of "actual" decryption as much as possible to get recrypt-then-NAND function inside the function set $\mathcal{F}$ that the system can correctly evaluate.

Before we consider $\boxplus, \boxtimes$, it will be useful to specify the *proper form* of ciphertext polynomials in this system, an invariant that we will maintain while performing homomorphic operations.

Definition 22 (Proper form of ciphertexts in $\mathcal{E}_{boot}$). A ciphertext $c(\vec{x})$ is in *proper form* if it is a degree-1 polynomial with coefficients that are in $(-1, 1]$ with κ bits of precision.

Toward maintaining the proper form invariant, observe that reducing a ciphertext polynomial c modulo 2, that is, adjusting c's coefficients by even integers into the range $(-1, 1]$, does not change the ciphertext's noise or the bit that it decrypts to, since $\vec{s}$ is integral.

So, after we add or multiply ciphertext polynomials, we will always reduce the coefficients modulo 2 back into the range $(-1, 1]$, perhaps without mentioning this explicitly.

Now, let us add and multiply ciphertext polynomials! Suppose we add ciphertexts:

$$c = \sum_i c^{(i)}.$$

Then,

$$c^{(i)}(\vec{s}) =_{\varepsilon_i} m^{(i)} \bmod 2 \Longrightarrow c(\vec{s}) =_{\sum_i \varepsilon_i} m \bmod 2,$$

where $m = \sum_i m^{(i)}$. If the original ciphertexts are in the proper form, then so is c. The ciphertext c has noise bounded by the sum of the noises of the original ciphertexts. It will decrypt to the "right" value—in particular, the noise will not "wrap" and overwhelm the "signal"—as long as $\sum_i \varepsilon_i < 1/2$.

Lemma 2 (Addition $\boxplus$). *Let $c^{(i)}(\vec{s}) =_{\varepsilon_i} m^{(i)} \bmod 2$ for all i. Let $c = \sum_i c^{(i)} \bmod 2$ and $m = \sum_i m^{(i)} \bmod 2$. Then $c(\vec{s}) =_{\sum_i \varepsilon_i} m \bmod 2$.*

Suppose we multiply two ciphertexts over $\mathbb{R}[\vec{x}]$:

$$c = c^{(1)} \cdot c^{(2)}.$$

Then, c is a degree-2 polynomial. Regarding the noise, we have:

$$\begin{aligned} c^{(i)}(\vec{s}) &=_{\varepsilon_i} m^{(i)} \bmod 2 \\ &= m^{(i)} + 2k^{(i)} + e^{(i)}, \quad m^{(i)} \in \{0,1\},\ k^{(i)} \in \mathbb{Z},\ \left|e^{(i)}\right| \leq \varepsilon_i. \end{aligned}$$

And so

$$\begin{aligned} &c^{(1)}(\vec{s}) \cdot c^{(2)}(\vec{s}) \\ &\quad = (m^{(1)} + 2k^{(1)} + e^{(1)}) \cdot (m^{(2)} + 2k^{(2)} + e^{(2)}) \\ &\quad = m^{(1)} \cdot m^{(2)} + 2k + e^{(1)}(m^{(2)} + 2k^{(2)}) + e^{(2)}(m^{(1)} + 2k^{(1)}) + e^{(1)} \cdot e^{(2)}, \quad k \in \mathbb{Z} \\ &\quad =_{\varepsilon} m^{(1)} \cdot m^{(2)} \bmod 2, \quad \varepsilon = (\varepsilon_1 + \varepsilon_2) \cdot (|\vec{s}|_1 + 1), \end{aligned}$$

where $|\vec{s}|_1$ is the ℓ_1 norm of $\vec{s}$. The new noise is bounded by ε, because if we let B be an upper bound on $|c^{(i)}(\vec{s})|$, then ε is at most $(\varepsilon_1 + \varepsilon_2) \cdot B$, and B is at most $|\vec{s}|_1 + 1$ since the coefficients of $c^{(i)}$ are in $(-1, 1]$.

But $c = c^{(1)} \cdot c^{(2)}$ is not in the proper form. We can easily reduce its coefficients modulo 2 into the range $(-1, 1]$ and drop precision beyond κ bits. Dropping precision costs us an addition noise term of at most $2^{-\kappa} \cdot |\vec{s}|_1^2$. But the biggest issue is that c is degree-2. We need to somehow *relinearize* c so that it is degree-1, as required.

To relinearize, we publish some polynomials in the evaluation key evk that will help us reduce the degree. You can think of these relinearization polynomials as a "noisy Gröbner basis" (using the term loosely): they allow us to reduce degree-2 polynomials to degree-1, but this reduction introduces additional noise.

Definition 23 (Evaluation key for $\mathcal{E}_{\text{boot}}$ (version 1)). As evk, publish polynomials $P = \{p_{i,j,k}(\vec{x})\}$ that are:

- "Pseudoencryptions" of 0: We have $p_{i,j,k}(\vec{s}) =_{\varepsilon_0} 0 \bmod 2$, where ε_0 is the noise of fresh ciphertexts,
- Slightly quadratic: We have $p_{i,j,k}(\vec{x}) = 2^{-k} \cdot x_i \cdot x_j + \ell_{i,j,k}(\vec{x})$ where $k \in \{0, \ldots, \kappa\}$ for precision parameter κ, and $\ell_{i,j,k}(\vec{x})$ is a degree-1 polynomial.

With this evaluation key evk in hand to facilitate relinearization, here is the entire $\boxtimes$ procedure.

$\boxtimes$ for $\mathcal{E}_{\mathbf{boot}}$.

- Compute $c = c^{(1)} \cdot c^{(2)}$ over $\mathbb{R}[x]$.
- Reduce c modulo 2 into the range $(-1, 1]$.
- Drop precision beyond κ bits.
- Relinearize using polynomials P: Call the polynomial so far $c(\vec{x})$. Write each coefficient $c_{i,j}$ (of monomial $x_i x_j$) in terms of its binary decomposition: $c_{i,j} = \sum_{k=0}^{\kappa} c_{i,j,k} 2^{-k}$ with each $c_{i,j,k} \in \{0, 1\}$. Next, subtract off a subset sum of the $p_{i,j,k}$'s to obtain a linear polynomial
$$\text{relinearize}_P\left(c(\vec{x})\right) = c(\vec{x}) - \sum_{i \le j,k} c_{i,j,k} \cdot p_{i,j,k}(\vec{x}).$$
- Reduce the resulting polynomial modulo 2 into the range $(-1, 1]$.

The ciphertext polynomial output by $\boxtimes$ is in the proper form. Relinearization introduces noise of magnitude at most $n^2 \cdot (\kappa + 1) \cdot \varepsilon_0$. Now, let us bound how $\boxtimes$ affects the noise overall.

Lemma 3 (Multiplication $\boxtimes$). *Let $c^{(1)}(\vec{s}) =_{\varepsilon_1} m^{(1)} \bmod 2$ for $i \in \{1, 2\}$. Let $c = c^{(1)} \boxtimes c^{(2)}$ and $m = m^{(1)} \times m^{(2)}$. Then $c(\vec{s}) =_{\varepsilon_\boxtimes} m \bmod 2$, for $\varepsilon_\boxtimes = (\varepsilon_1 + \varepsilon_2) \cdot (|\vec{s}|_1 + 1) + 2^{-\kappa} \cdot |\vec{s}|_1^2 + n^2 \cdot (\kappa + 1) \cdot \varepsilon_0$. For reasonable parameter settings, $\boxtimes$ multiplies the noise by an $O(poly(n))$ factor.*

Proof. The exact expression for the noise comes from the bounds above on the noise added by individual steps of $\boxtimes$. Now, take ε to be the maximum of $\varepsilon_1, \varepsilon_2$, both of which are at least ε_0, the latter being the noise of fresh ciphertexts. Recall that we can choose $\vec{s}$ from the noise distribution (except that, unlike the noise and ciphertexts, we have not divided $\vec{s}$ by 2^κ). So, the coefficients of $\vec{s}$ are bounded by $2^\kappa \cdot \varepsilon_0$, and the middle term $2^{-\kappa} \cdot |\vec{s}|_1^2$ is at most $n \cdot |\vec{s}|_1 \cdot \varepsilon_0$. We satisfy the conditions of Theorem 7 as long as the noise distribution (according to which the coefficients of $\vec{s}$ are also chosen) has deviation $2\sqrt{n}$, so we can take $|\vec{s}|_1 = \text{poly}(n)$. We will also take $\kappa = \text{poly}(n)$. Then, we have that the new noise is bounded by $\varepsilon \cdot \text{poly}(n)$. ■

We have established that (for reasonable parameter settings), a single $\boxplus$ or $\boxtimes$ operation increases the noise by at most a factor of $p(n)$ for some polynomial p. (See Lemmas 2 and 3.) This result gives us the following commutative diagram:

$$\begin{tikzcd}
\mathcal{C}_\varepsilon \times \mathcal{C}_\varepsilon \arrow[r, "\boxplus{,}\boxtimes"] \arrow[d, "\lfloor \mathrm{ev}_{\vec{s}}(\cdot) \rceil \bmod 2"] & \mathcal{C}_{\varepsilon \cdot p(n)} \arrow[d, "\lfloor \mathrm{ev}_{\vec{s}}(\cdot) \rceil \bmod 2"] \\
\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \arrow[r, "+{,}\times"] & \mathbb{Z}/2\mathbb{Z},
\end{tikzcd}$$

where $\mathcal{C}_\varepsilon$ denotes ciphertexts with noise bounded by ε, and $\mathrm{ev}_{\vec{s}}(\cdot)$ denotes evaluation of a polynomial at $\vec{s}$.

Now, let us extend this observation to evaluation of an arithmetic circuit C of $\{+, \times\}$ gates. Recall that the depth d of the circuit C is the length of the longest path from an input gate to the output gate, considering the circuit as a directed acyclic graph.

Lemma 4. *Let $p(x)$ be a polynomial such that $\boxplus$ and $\boxtimes$ multiply the noise of input ciphertexts by at most $p(n)$. Given input ciphertexts of noise at most ε_0, the above system can evaluate any arithmetic circuit of depth d with noise at most $\varepsilon_0 \cdot p(n)^d$.*

For example, if we want to evaluate a d-depth circuit so that the noise of the final ciphertext is at most (say) $1/4$, it suffices to take $\varepsilon_0 \leq (1/4) \cdot p(n)^{-d}$.

As we are aiming for a bootstrappable encryption system, we are especially interested in the depth of the decryption circuit (and the recrypt-then-NAND circuit, which has depth 2 more than the decryption circuit). Recall that we drop unneeded precision before "actual" decryption. Importantly, after dropping precision, the complexity of decryption does *not* depend on ε_0 at all. Indeed, ε_0 becomes a free parameter that we can eventually set as small as needed to allow us to evaluate the decryption circuit.

Lemma 5. *Let c be a ciphertext such that $c(\vec{s}) =_{1/4} m \bmod 2$. For any κ' such that $2^{\kappa'} > 4 \cdot (|\vec{s}|_1 + 1)$, we can drop to κ' bits of precision in c while preserving correctness of decryption (to the message m).*

Proof. Dropping to κ' bits of precision adds at most $2^{-\kappa'} \cdot (|\vec{s}|_1 + 1)$ to the noise. By assumption, $2^{-\kappa'} \cdot (|\vec{s}|_1 + 1) < 1/4$, so that the total noise remains $< 1/2$, and correctness of decryption is preserved. ■

Now we bound the arithmetic circuit depth of "actual" decryption (after dropping to $\lceil \log(4 \cdot (|\vec{s}|_1 + 1)) \rceil$ bits of precision in c).

Lemma 6. *For reasonable parameters, the decryption circuit has depth*

$$O(\log n + \log\log |\vec{s}|_1) = O(\log n).$$

Proof. The computation of the inner product of two vectors—represented in the natural way, with coefficients in binary representation—is in NC^1 (see Section 4 of [56]), meaning that it can be computed in circuit depth proportional to the logarithm of the description length of the vectors. Decryption is an inner product of vectors of dimension n with coefficients of $O(\log \|\vec{s}_1\|)$ bits. Therefore, it can be computed in depth $O(\log n + \log\log |\vec{s}|_1)$. Regarding $\|\vec{s}_1\|$, we can satisfy the conditions of Theorem 7 as long as the noise distribution (according to which the coefficients of $\vec{s}$ are also chosen) has deviation $2\sqrt{n}$, and so we can take $|\vec{s}|_1 = \mathrm{poly}(n)$. The result follows. ■

Theorem 9. *$\mathcal{E}_{\text{boot}}$ is bootstrappable for some ε_0 (the noise bound of fresh ciphertexts) that is $n^{-O(\log n)}$.*

Proof. By Lemma 6, the "actual" decryption circuit (hence recrypt-then-NAND) applied to ciphertexts with dropped precision has depth $O(\log n)$. By Lemmas 4 and 5, for some $\varepsilon_0 = n^{-O(\log n)}$, we can apply recrypt-then-NAND and drop precision while keeping the noise below $1/2$. ■

Theorem 10. *$\mathcal{E}_{\text{boot}}$ is IND-CPA secure based on the LWE with higher degree polynomials assumption (Definition 21) for some $q = n^{O(\log n)}$ and some poly(n)-bounded distribution χ.*

Proof. The proof is similar to the proof of IND-CPA security for the basic symmetric encryption system based on LWE (see Theorem 8), except that we use LWE with higher degree polynomials (versus regular LWE) to sample the "slightly quadratic" polynomials P used for relinearization in $\boxtimes$. ■

By Theorem 4, we get leveled FHE based on this assumption. In fact, we can get circular-secure FHE based on variants of the LWE with higher degree polynomials assumption, e.g., where we still only use linear and "slightly quadratic" polynomials, but allow the secret $\vec{s}$ to come from $\{0,1\}^n$ rather than χ^n. We elaborate on this observation in Section 4.5.

But our underlying encryption system $\mathcal{E}_{\text{LWE}}$ is based on the LWE assumption. How can we get a bootstrappable encryption system also based on LWE assumption, rather than the less well-established LWE with higher degree polynomials assumption? Clearly, the issue originates with the current version of $\mathcal{E}_{\text{boot}}$, which requires publication of "slightly quadratic" polynomials that are used in relinearization. Can we publish a different set of polynomials to facilitate relinearization, and somehow base security on LWE?

The trick to get security based on LWE is similar to the trick that we used to get IND-CPA secure leveled FHE from IND-CPA secure bootstrappable encryption—namely, to avoid a circular security issue, we use an acyclic chain of encrypted secret keys. The slightly quadratic relinearization polynomials used in the above version of $\mathcal{E}_{\text{boot}}$ can be viewed as an encryption of the secret key under itself. Specifically, we have:

$$\begin{aligned} p^{(\ell)}_{i,j,k}(\vec{s}) &= 2^{-k} \cdot x_i \cdot x_j + \ell_{i,j,k}(\vec{s}) =_{\varepsilon_0} 0 \bmod 2 \\ \Longrightarrow \ell_{i,j,k}(\vec{s}) &=_{\varepsilon_0} -2^{-k} \cdot s_i \cdot s_j . \end{aligned}$$

That is, the linear polynomial $\ell_{i,j,k}(\vec{x})$, which has the proper form of a ciphertext, encrypts (in some sense) the value $-2^{-k} \cdot s_i \cdot s_j$, a quadratic monomial of the secret key itself. In version 2 of $\mathcal{E}_{\text{boot}}$, we modify the relinearization polynomials so that each one is, in effect, an encryption of a quadratic monomial over some $\vec{s}^{(\ell)}$ under the *next* secret $\vec{s}^{(\ell-1)}$, so that we have an acyclic chain of encrypted secret keys. In detail:

Definition 24 (Evaluation key for $\mathcal{E}_{boot}$ (version 2)). Publish polynomials $P = \{p^{(\ell)}_{i,j,k}(\vec{x}, \vec{y})\}$ that are:

- "Pseudoencryptions" of 0: We have $p^{(\ell)}_{i,j,k}(\vec{s}^{(\ell)}, \vec{s}^{(\ell-1)}) =_{\varepsilon_0} 0 \bmod 2$, where ε_0 is the noise of fresh ciphertexts,
- Slightly quadratic (and some linear): We have

$$p^{(\ell)}_{i,j,k}(\vec{x}, \vec{y}) = 2^{-k} \cdot x_i \cdot x_j + \ell_{i,j,k}(\vec{y})$$

  where $k \in \{0, \dots, \kappa\}$ for precision parameter κ, and $\ell_{i,j,k}(\vec{y})$ is a degree-1 polynomial. For $i = 0$, we have linear polynomials $p^{(\ell)}_{i,j,k}(\vec{x}, \vec{y}) = 2^{-k} \cdot x_j + \ell_{i,j,k}(\vec{y})$.

Unlike version 1, we need linear polynomials in addition to the slightly quadratic ones, because we are using the polynomials not only to relinearize but simultaneously to transfer the ciphertexts from one key ($\vec{s}^{(\ell)}$) to the next ($\vec{s}^{(\ell-1)}$). In the system, we can apply $\boxplus$ or $\boxtimes$ to two ciphertexts under $\vec{s}^{(\ell)}$, with the result under $\boxtimes$ being under the next key $\vec{s}^{(\ell-1)}$. The noise analysis is identical to version 1.

Theorem 11. *$\mathcal{E}_{\text{boot}}$ (version 2) is IND-CPA secure based on the LWE assumption (Definition 18) for some $q = n^{O(\log n)}$ and some poly(n)-bounded distribution χ.*

Proof. (Sketch) The proof is similar to that of Lemma 7, where we proved the IND-CPA security of a leveled FHE system that uses an acyclic chain of encrypted secret keys, using a so-called hybrid argument where in a sequence of steps we replace encrypted secret keys with encryptions of 0. The main idea in this proof is that each relinearization polynomial $p^{(\ell)}_{i,j,k}(\vec{x}, \vec{y})$, whose special property is that it evaluates to a small value at $(\vec{s}^{(\ell)}, \vec{s}^{(\ell-1)})$, looks indistinguishable from random to an adversary that does not know $\vec{s}^{(\ell-1)}$, even if it knows $\vec{s}^{(\ell)}$. ■

Recall from Section 4.2 that, as far as we know, LWE is hard even if the ratio of q to the noise is subexponential in n, so Theorem 11 provides a strong security guarantee. It is possible to base the security of leveled FHE on LWE even for factors that are polynomial in n [25].

4.5. Reflections on the overall FHE system

Now that we have completed our modular description of the FHE system, it is interesting to demodularize the system to see what is happening overall. To simplify the overall picture for the moment, let us imagine that the secret $\vec{s}$ is in $\{0,1\}^n$, so that the "bits" of $\vec{s}$ are in fact the coefficients of $\vec{s}$. We operate on linear ciphertext polynomials $c(\vec{x})$ that have small noise when evaluated at $\vec{s}$. When we multiply polynomials, we use a noisy partial Gröbner basis to reduce the resulting polynomial back to linear while changing the evaluation at $\vec{s}$ by only small noise. As we apply $\boxplus$ and $\boxtimes$, the noisiness increases until we have a $c(\vec{x})$

whose noise at $\vec{s}$ is nearly 1/2, so that it can no longer participate safely in operations. But now imagine that the underlying polynomial that we are evaluating with these $\boxplus$'s and $\boxtimes$'s is $f(\vec{x}) = \mathsf{D}_c(\vec{x})$. It holds that $f(\vec{s}) = m$, the message encrypted by c, but we do not know $\vec{s}$. Instead, we start evaluating $f(\vec{x})$ as a formal polynomial, except that we reduce the degree using our noisy Gröbner basis. This noisy basis allows us to reduce the degree to linear, while preserving (up to small noise) the evaluation at $\vec{s}$ (which is what we care about). At the end, we get a linear polynomial whose evaluation at $\vec{s}$ equals (up to small noise) the value $f(\vec{s}) = m$. Hence, this linear polynomial is a new encryption of m. This new encryption has noise that depends only on the complexity of D_c and the noisiness of our noisy Gröbner basis, not on the noisiness of c as a ciphertext, and therefore it can (if we calibrate our noise appropriately) participate in more operations.

It is also interesting to consider what our ciphertext set looks like as an algebraic structure. Our unbounded system has a clean commutative diagram

$$\begin{tikzcd}
\mathcal{C} \times \mathcal{C} \arrow[r, "\boxplus{,}\boxtimes"] \arrow[d, "\lfloor \mathrm{ev}_{\vec{s}}(\cdot) \rceil \bmod 2"] & \mathcal{C} \arrow[d, "\lfloor \mathrm{ev}_{\vec{s}}(\cdot) \rceil \bmod 2"] \\
\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \arrow[r, "+{,}\times"] & \mathbb{Z}/2\mathbb{Z},
\end{tikzcd}$$

but now $\boxplus$ and $\boxtimes$ hide a lot of complexity, e.g., $\boxtimes$ is in fact $\times \circ \mathsf{D}_{c^{(1)},c^{(2)}}(\cdot)$. The operations $\boxplus, \boxtimes$ are not even associative: $\mathcal{C}$ is a magma under both operations. In fact, these operations are so unstructured that it is almost as if they select a pseudorandom ciphertext that encrypts the appropriate value. While this intuition may not be entirely accurate, Ducas and Stehlé [34] show that starting with any c that encrypts m, successive alternations of recryption and small injections of noise converge to a *canonical* distribution over ciphertexts that encrypt m, effectively erasing the ciphertext's history.

5. NEW DIRECTIONS IN HOMOMORPHIC ENCRYPTION

Many questions about FHE remain unresolved. Below, we discuss a few of these questions—in particular, questions about improving efficiency, handling the circular security issue, avoiding bootstrapping, building noise-free FHE, and exploring quantum FHE.

Some of these questions can be answered (not entirely satisfactorily) with the ultimate cryptographic hammer: cryptographic program obfuscation [14, 31, 39, 53]. Informally, program obfuscation takes a program P and produces an obfuscated program $O(P)$ that has the same input/output functionality, but where $O(P)$ is otherwise "unintelligible." Program obfuscation was proven impossible to achieve for a certain "virtual black box" notion of unintelligibility [14]. However, there are now constructions of program obfuscation [39], even based on well-established computational assumptions [53], for a notion of unintelligibility based on indistinguishability.[10] Namely, an indistinguishability obfuscator iO offers the fol-

10 Say it 10 times fast!

lowing guarantee: if there are two circuits C_0, C_1 of the same size and equivalent input/output functionality and iO obfuscates one of them (C_b for random $b \in \{0, 1\}$) to produce $O(C_b)$, it is computationally hard to distinguish b. (Notice the similarity to IND-CPA security for encryption.) Obfuscation is even more powerful than FHE—in particular, you can build FHE from versions of it [26]—but it is currently also computationally much more expensive than FHE. A crucial difference between an obfuscated program and an FHE-encrypted program is that the obfuscated program has unencrypted output.

How practical can we make the current noisy FHE systems? Since the first construction of FHE in 2009 [40,41] there have been four generations of FHE systems, with the second, third, and fourth generations each offering significant performance gains. Currently, efforts are underway to standardize homomorphic encryption systems [51].

The second generation of FHE systems simultaneously improved security and efficiency by basing security on LWE [24], and then using variants of LWE over polynomial rings, such as ring-LWE [60] and the NTRU problem [50,59]. Polynomial rings naturally facilitate *batching* or *SIMD* operations on encrypted data [73]—that is, via the Chinese Remainder Theorem, they allow many individual messages to be packed in single ciphertexts, and for many messages to be (implicitly) operated on through a single ciphertext operation. Automorphisms of the polynomial rings allow message slots to be permuted within a ciphertext. These optimizations, together with better techniques to control the growth of the noise in ciphertexts [23], reduced the overhead of FHE—that is, multiplicative factor of how much processing it takes to operated on FHE-encrypted data versus unencrypted data—to only polylogarithmic in the security parameter λ [45].

The third generation [8,25,29,30,33,46] introduced techniques for fast bootstrapping, reducing bootstrapping time to tens of milliseconds. However, so far, these techniques are incompatible with the batching techniques of the second generation.

The fourth generation systems [27] are optimized for operating on floating-point data, making it more friendly for real-world applications such as logistic regression and neural nets. Current estimates are that, depending on the type of computation, the overhead of fourth generation systems is as low as $10000\times$, and this overhead can be further reduced with customized hardware [36,55,64].

An inherent limitation of FHE systems is that they do not support certain types of computation, such as RAM (random access model) computations. FHE computations are inherently input-oblivious, i.e., the structure of the computation cannot depend on the input, since IND-CPA security implies no information about the input is disclosed. This limitation can be overcome by using heavier machinery—in particular, cryptographic program obfuscation [44]—since obfuscated programs can disclose unencrypted information.

Can we base unbounded FHE on a well-established computational hardness assumption? Current unbounded (versus leveled) FHE systems require a circular security assumption—namely, that it is secure to encrypt the secret decryption key under its companion encryption key. For the noisy FHE system presented in Section 4, the encryption of the secret key manifests as a collection of nonlinear polynomials that evaluate to some small (noise) value at the secret key $\vec{s}$. Can we reduce the LWE with higher degree polynomials assumption to a more well-established assumption, such as the hardness of problems over integer lattices? Can we somehow avoid nonlinear polynomials altogether, and build unbounded FHE based on LWE?

Can we get unbounded FHE without bootstrapping or recryption? In Section 3.4, we saw that bootstrapping seems unavoidable as a generic technique to convert a bounded homomorphic encryption system to an unbounded one. The one FHE system based on well-established assumptions that does not use bootstrapping [53] instead uses program obfuscation, which currently is less practical than more direct constructions of FHE. Moreover, the technique to build FHE [26] from obfuscation uses obfuscation to decrypt ciphertexts, perform an operation on them, and then encrypt the result—a process that (like bootstrapping) still involves computing the decryption function. Can we get unbounded FHE while avoiding expensive repeated computation of the decryption function? Note that we can get leveled FHE without bootstrapping [23] for a relaxed definition of leveled that allows the parameters to grow with the number of levels. (Our Definition 12 for leveled systems is not relaxed.)

Can we build "noise-free" FHE?. So far, all constructions of FHE use "noise," even the obfuscation-based solution [53]. For building bootstrappable encryption, noise has the nice feature that we can calibrate the noise level, decreasing it until the system becomes bootstrappable. For the system in Section 4.4, this calibration was especially easy because adjusting the noise level did not affect decryption complexity at all (though it affects the computational assumption). While noise has these nice features, it also introduces complexities, and one wonders whether it is possible to construct noise-free FHE.

Mathematically, perhaps the cleanest approach to noise-free FHE would be to construct a multilinear map with suitable cryptographic properties [19]. We have cryptographic *bi*linear maps from Weil and Tate pairings over abelian varieties, which have proven to be enormously useful. We can obtain FHE (and obfuscation) from cryptographic multilinear maps of higher degree, but so far we have no viable noise-free constructions.

Nuida [63] proposed a construction of noise-free FHE using nonsolvable groups with certain properties, but groups with these properties are not known to exist. As discussed in Section 3.5, there are many obstacles to constructing a secure homomorphic encryption based on nonsolvable groups.

Are fundamentally new constructions possible in the quantum setting? The FHE system presented here works for a computation expressible as a polynomial-size circuit with classical gates, like $\{+, \times\}$. What if we want to privately delegate a computation not known to be in P, but which is easy for a quantum computer, like factoring an encrypted integer? For that, we need an FHE system capable of evaluating quantum gates. Mahadev [61] resolved the question of quantum (leveled) FHE, showing that a classical client can privately outsource a quantum computation to a quantum server, under the surprisingly minimal assumption that LWE is hard for quantum computers. One wonders whether this result, and the suggestive similarities between quantum error correction and managing ciphertext noise in FHE systems, will lead to new techniques even for classical FHE systems. The question of quantum obfuscation is not resolved. While preliminary results [5] indicate that virtual black box obfuscation of quantum circuits is impossible, they leave open the question of indistinguishability obfuscation for quantum circuits.

ACKNOWLEDGMENTS

I would like to give special thanks to Dan Boneh for his encouragement to work on this problem, and to Shai Halevi for such fruitful collaboration. I would also like to thank the many others who have made this area so exciting to work in.

A. WHAT DOES IT MEAN FOR AN ENCRYPTION SYSTEM TO BE SECURE?

This section provides informal philosophical discussion about what it means for an encryption system to be secure—in particular, about why IND-CPA security (see Definitions 1 through 3) is the "right" minimal notion of security for an encryption system.

To see why, let us try to reinvent the security model ourselves. We have an "adversary" that is trying to "break" the encryption system. In general, we can model the security of encryption as a game between a "challenger" and the adversary that the adversary is trying to "win." To specify the game, there are 3 aspects to consider:

(1) *Adversary's power inside the system*: How can the adversary interact with the encryption system? Does the adversary know how the algorithms (key generation, encryption, decryption) work, is it allowed to see many ciphertexts, on messages of its choice, can it ask for ciphertexts to be decrypted, can it see how transmitted and decrypted messages affect peoples' "behavior," can it ask for bits of the secret key or functions of the secret key?

(2) *Adversary's power outside the system*: Is the adversary limited to running polynomial time algorithms, polynomial time quantum computation, polynomial time nondeterministic computation, is its computational power unbounded?

(3) *Object of the game*: Is the object to recover the secret key, to recover the message encrypted by a ciphertext, to merely distinguish which of two messages a ciphertext encrypts, to produce a new ciphertext that encrypts a message related to a message encrypted by a given ciphertext?

Now, let us start to prune the numerous possibilities given above.

First, as a theoretical matter, we can assume without loss of generality that adversary knows how the system works. We simply label what the adversary does not know as the secret key. The secret key may include hidden aspects of how key generation, encryption, and decryption operate, but really this is just a matter of semantics, and these algorithms can always be redefined so that secret information is localized to the secret key, and the algorithms themselves are public. Also, as a practical matter, we have Kerckhoff's principle, which (as reformulated by Claude Shannon) simply states that "the enemy knows the system." The practical justification for this principle is that "security by obscurity" rarely works. Rather, empirically, one is more likely to obtain a secure system by making it comprehensible to friendly cryptanalysts. (Provable security, à la Goldwasser and Micali, is an extreme version of Kerckhoff's principle, in which we proudly display a concise clearly-specified mathematical problem on which the cryptosystem's security is based.) So, we take the algorithms of encryption—key generation K, encryption E, decryption D, and evaluation V (if applicable)—as known.

Object of the game. The purpose of an encryption system is to hide a message. Clearly, we should not require the adversary to recover the secret key, since it might be able to recover information about an encrypted message without it. From a ciphertext, the adversary will trivially know an upper bound on the message length, but it should not be able to determine *anything* else. (Lacking a general way to characterize what is *important* in a message, we should require that the adversary cannot distinguish *anything* nontrivial.) Goldwasser and Micali capture this intuition with their definition of semantic security: "Informally, a system is semantically secure if whatever an eavesdropper can compute about the cleartext given the cyphertext, he can also compute without the cyphertext" [47]. In particular, given a ciphertext encrypting a message, the *a priori* and *a posteriori* distributions of the message should be identical (up to a negligible factor) from the perspective of the adversary. Goldwasser and Micali proved that this notion of semantic security is modeled well by the IND-CPA game, which allows the adversary to choose message pairs $\{m_{i,0}, m_{i,1}\}$ whose encryptions it thinks it is most able to distinguish.

To make things even easier on the adversary, we could require only that it produce a new ciphertext (not given to it by the challenger) that encrypts a message related to (e.g., the same as) in the challenge ciphertext. A system that prevents this attack is called *nonmalleable* [32]. We do not consider nonmalleability to be part of a *minimal* viable notion of security for encryption for two reasons. First, we are considering homomorphic encryption, which is inherently malleable; the whole point is to produce new ciphertexts that encrypt values meaningfully related to those of previous ciphertexts. Second, there are general techniques (that we will not review here) that prevent malleability. To a large extent, nonmalleability can be "added on" to an IND-CPA secure encryption system after the fact.

Adversary's power outside the system. Aside from interacting with the system, the adversary's power (outside of the system) comes down to its computational power. Claude Shannon resolved the case of a computationally unbounded adversary. He showed that one can perfectly hide a message (except for an upper bound on its length) by encrypting it with a one-time pad (a perfectly random key as long as the message). In some settings, such as military or diplomatic settings that demand absolute eternal secrecy, a one-time pad might be the right solution. However, we are also interested in many other (most) settings, where distributing a one-time pad is not practical. Accordingly, to allow more practical systems, we permit computational assumptions—that is, we only require our encryption systems to be secure against probabilistic polynomial-time adversaries, and assume that some problems in NP are hard to solve in probabilistic polynomial time. (Of course, even this seemingly minimal assumption may turn out to be false, as currently we are not even certain that $\mathrm{P} \neq \mathrm{NP}$.)

Adversary's power inside the system. As mentioned before, we must allow the adversary to "know the system." Moreover, as we are not in the setting of the one-time-pad, the adversary should be able to see many ciphertexts encrypted under the same key. Furthermore, since the adversary in real life might be able to influence what the encrypter encrypts, the game should allow the adversary to choose what messages are encrypted, or even have all of the messages depend on some bit that it is trying to guess. The IND-CPA game gives the adversary this power.

But why not give the adversary even more power in our minimal notion of security? For example, why not allow the adversary to choose ciphertexts to be decrypted by the challenger, or to receive some (not-completely-revealing) function of the secret key? Indeed, these forms of security are important. The IND-CCA (indistinguishability of ciphertexts against a chosen ciphertext attack) game allows the adversary to query the decryption of ciphertexts, a model that is actually quite realistic in real life, because adversaries can potentially break cryptosystems by observing how keyholders react after decrypting their ciphertexts, even (or perhaps especially) if those ciphertexts are malformed. Key "leakage" and "side channel" attacks, in which the adversary obtains some limited information about the secret key, are also quite realistic, because (unless special care is taken) even the amount of time the keyholder takes to decrypt can leak information about the secret key.

The reason why we consider IND-CPA still to be acceptable minimal notion of security is that there are techniques for achieving IND-CCA and security against side channel attacks that are mostly orthogonal to IND-CPA security—that is, they can, to a large extent, be applied to an IND-CPA secure system after the fact. Even for homomorphic encryption, which is inherently malleable, one can use so-called noninteractive zero-knowledge arguments to ensure that the keyholder decrypts only after verifying a cryptographic proof that the ciphertext is well formed and resulted from a "permitted" evaluation over valid ciphertexts. In the real world, combining homomorphic encryption systems with proof systems in this way is actually important for preventing devastating attacks. But, again, these considerations are largely orthogonal to the security of the underlying homomorphic system, and we therefore do not consider them to be part of the minimal notion of security.

Conclusion. Out of many possible security notions, we pruned the possibilities to land on IND-CPA security as the "right" minimal notion of security for a homomorphic encryption system. Weaker notions may not provide much security at all in realistic contexts, and stronger notions typically can be achieved by using an IND-CPA-secure system in combination with orthogonal techniques.

B. HYBRID ARGUMENT FOR LEVELED FHE

Lemma 7. *Let $\mathcal{E}$ be an IND-CPA secure encryption system such that any secret key sk can be expressed as a vector $\vec{sk} \in \mathcal{M}^k$, where $\mathcal{M}$ is the message set of the system. Let $\mathcal{E}_{LFHE}$ be a system in which we publish encrypted secret keys $\vec{S}^{(i)} = \mathcal{E}.\mathsf{E}(ek^{(i-1)}, \vec{sk}^{(i)})$, where $(sk^{(i)}, ek^{(i)})$ for $i \in \{0, \dots, n\}$ is an $\mathcal{E}$ key pair. Suppose that $\mathcal{E}_{LFHE}.\mathsf{E}$ is the same as $\mathcal{E}.\mathsf{E}$, using encryption key $ek^{(n)}$. Then $\mathcal{E}_{LFHE}$ is also IND-CPA secure in the following sense. Suppose that there is an adversary $\mathcal{A}$ in the IND-CPA game against $\mathcal{E}_{LFHE}$ that has advantage ε. Then there is an adversary $\mathcal{B}$ in the IND-CPA game against $\mathcal{E}$ that has advantage at least $\varepsilon/2(n+1)$, and that runs in about the same time as $\mathcal{A}$.*

Proof of Lemma 7. We consider $\mathcal{A}$'s behavior in a sequence of games: Game 0, Game 1, ..., Game n. Game 0 is identical to the IND-CPA game for $\mathcal{E}_{\text{LFHE}}$. Game i is identical, except that the values $\vec{S}^{(i+1)}, \dots, \vec{S}^{(n)}$ are constructed correctly (as in the system), but the values $\vec{S}^{(1)}, \dots, \vec{S}^{(i)}$ are all encryptions of 0. Whatever game we are in, the $\mathcal{E}_{\text{LFHE}}$-IND-CPA-challenger samples a random bit $b \in \{0, 1\}$. When $\mathcal{A}$ queries messages $(m_{j,0}, m_{j,1})$, the challenger encrypts $m_{j,b}$ under $ek^{(n)}$.

Let ε_i be $\mathcal{A}$'s advantage in guessing b in Game i. Since Game 0 is the true IND-CPA game for $\mathcal{E}_{\text{LFHE}}$ as given in Definition 3, we have $\varepsilon_0 = \varepsilon$. Therefore either ε_n or $\varepsilon_i - \varepsilon_{i+1}$ for some $i \in \{0, \dots, n-1\}$ must exceed $\varepsilon/(n+1)$ in magnitude. Set i^* so that the magnitude of $\varepsilon_{i^*} - \varepsilon_{i^*+1}$ is maximized (or set $i^* = n$ if ε_n it is the biggest contributor).

Then $\mathcal{B}$ attacks $\mathcal{E}$ by using $\mathcal{A}$ as follows: $\mathcal{B}$ participates in an IND-CPA game with an $\mathcal{E}$-challenger who flips a bit $\beta \in \{0, 1\}$. This game is associated to some key pair, which $\mathcal{B}$ will label formally as $(sk^{(i^*)}, ek^{(i^*)})$. If the system is asymmetric, it will receive $ek^{(i^*)}$ from the challenger. Also $\mathcal{B}$ assumes the role of the challenger in the $\mathcal{E}_{\text{LFHE}}$ game and flips a bit $b \in \{0, 1\}$.

Then $\mathcal{B}$ uses $\mathcal{E}.\mathsf{K}$ to generate key tuples $(sk^{(i)}, ek^{(i)})$ for all $i \neq i^*$. Here is how $\mathcal{B}$ generates the $\vec{S}^{(i)}$ values for $i \in \{1, \dots, n\}$. For all $i \geq i^* + 2$, it generates each $\vec{S}^{(i)}$ correctly (as in the system) as an encryption of $\vec{sk}^{(i)}$ under $ek^{(i-1)}$. For $i \leq i^*$, it generates each $\vec{S}^{(i)}$ as an encryption of 0 under $ek^{(i-1)}$.

If $i^* = n$, then this is a complete set of $\vec{S}^{(i)}$'s, and $\mathcal{B}$ sends the complete $\mathcal{E}_{\text{LFHE}}$ public key to $\mathcal{A}$. When $\mathcal{A}$ queries messages $(m_{j,0}, m_{j,1})$, $\mathcal{B}$ forwards these messages to the $\mathcal{E}$-challenger as queries. The $\mathcal{E}$-challenger encrypts $m_{j,\beta}$ under $ek^{(n)}$, and sends the ciphertext to $\mathcal{B}$, which forwards the ciphertext to $\mathcal{A}$. Then $\mathcal{A}$ submits a guess and $\mathcal{B}$ forwards that guess to the $\mathcal{E}$-challenger. Now $\mathcal{B}$'s advantage is the same as $\mathcal{A}$'s. Since the distribution seen by $\mathcal{A}$ is precisely as in Game n, $\mathcal{A}$'s advantage is ε_n.

If $i^* \neq n$, then $\mathcal{B}$ generates $\vec{S}^{(i^*+1)}$ as follows. It submits $\vec{0}$ and $\vec{sk}^{(i^*+1)}$ to the $\mathcal{E}$-challenger. If $\beta = 0$, the challenger sends to $\mathcal{B}$ the ciphertexts $\mathcal{E}.\mathsf{E}(ek^{(i^*)}, \vec{0})$, else it sends the ciphertexts $\mathcal{E}.\mathsf{E}(ek^{(i^*)}, \vec{sk}^{(i^*+1)})$. Then $\mathcal{B}$ labels the ciphertext from the $\mathcal{E}$-challenger as $\vec{S}^{(i^*+1)}$ and sends the complete $\mathcal{E}_{\text{LFHE}}$ public key to $\mathcal{A}$. When $\mathcal{A}$ queries messages $(m_{j,0}, m_{j,1})$, $\mathcal{B}$ encrypts $m_{j,b}$ under $ek^{(n)}$. Notice that from $\mathcal{A}$'s perspective, if $\beta = 0$ then its view is as in Game $i^* + 1$, and if $\beta = 1$ its view is as in Game i^*. Therefore, $\mathcal{A}$'s advantage is ε_{i^*+1} if $\beta = 0$ and ε_{i^*} if $\beta = 1$. Also $\mathcal{B}$ guesses that $\beta = 1$ if $\mathcal{A}$ guesses b

correctly, otherwise it guesses that $\beta = 0$. Now $\mathcal{B}$'s success probability is

$$\begin{aligned}
\Pr[\mathcal{B}\text{ correct}] = {} & \Pr[\mathcal{B}\text{ correct}|\beta = 0 \text{ and } \mathcal{A}\text{ correct}] \cdot \Pr[\beta = 0 \text{ and } \mathcal{A}\text{ correct}] \\
& + \Pr[\mathcal{B}\text{ correct}|\beta = 0 \text{ and } \mathcal{A}\text{ incorrect}] \cdot \Pr[\beta = 0 \text{ and } \mathcal{A}\text{ incorrect}] \\
& + \Pr[\mathcal{B}\text{ correct}|\beta = 1 \text{ and } \mathcal{A}\text{ correct}] \cdot \Pr[\beta = 1 \text{ and } \mathcal{A}\text{ correct}] \\
& + \Pr[\mathcal{B}\text{ correct}|\beta = 1 \text{ and } \mathcal{A}\text{ incorrect}] \cdot \Pr[\beta = 1 \text{ and } \mathcal{A}\text{ incorrect}] \\
= {} & 0 \cdot \big[(1/2)(1/2 + \varepsilon_{i^*+1})\big] + 1 \cdot \big[(1/2)(1/2 - \varepsilon_{i^*+1})\big] \\
& + 1 \cdot \big[(1/2)(1/2 + \varepsilon_{i^*})\big] + 0 \cdot \big[(1/2)(1/2 - \varepsilon_{i^*})\big] \\
= {} & 1/2 + (\varepsilon_{i^*} - \varepsilon_{i^*+1})/2.
\end{aligned}$$

■

REFERENCES

[1] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.* **51** (2018), no. 4, 1–35.

[2] C. Aguilar-Melchor, S. Fau, C. Fontaine, G. Gogniat, and R. Sirdey, Recent advances in homomorphic encryption: a possible future for signal processing in the encrypted domain. *IEEE Signal Process. Mag.* **30** (2013), no. 2, 108–117.

[3] M. Ajtai, Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on theory of computing*, pp. 99–108, ACM, 1996.

[4] M. Ajtai and C. Dwork, A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the twenty-ninth annual ACM symposium on theory of computing*, pp. 284–293, ACM, 1997.

[5] G. Alagic, Z. Brakerski, Y. Dulek, and C. Schaffner, Impossibility of quantum virtual black-box obfuscation of classical circuits. 2020, arXiv:2005.06432.

[6] M. R. Albrecht, P. Farshim, J.-C. Faugere, and L. Perret, Polly cracker, revisited. In *International conference on the theory and application of cryptology and information security*, pp. 179–196, Springer, 2011.

[7] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. M. T. Morrison, A. Sahai, and V. Vaikuntanathan, Homomorphic encryption standard. Available at http://homomorphicencryption.org/, accessed February 2019, November 2018.

[8] J. Alperin-Sheriff and C. Peikert, Faster bootstrapping with polynomial error. In *Advances in cryptology—CRYPTO 2014, Part I*, edited by J. A. Garay and R. Gennaro, pp. 297–314, Springer, 2014.

[9] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in cryptology—CRYPTO 2009, 29th annual international cryptology conference, Santa Barbara, CA, USA, August 16–20, 2009. Proceedings*, pp. 595–618, Springer, 2009.

[10] F. Armknecht, T. Gagliardoni, S. Katzenbeisser, and A. Peter, General impossibility of group homomorphic encryption in the quantum world. In *International workshop on public key cryptography*, pp. 556–573, Springer, 2014.

[11] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand, A guide to fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* **2015** (2015), 1192.

[12] S. Arora and R. Ge, New algorithms for learning in presence of errors. In *ICALP (1)*, pp. 403–415, Lecture Notes in Comput. Sci. 6755, Springer, 2011.

[13] B. Barak and Z. Brakerski, Building the Swiss Army Knife. https://windowsontheory.org/2012/05/02/building-the-swiss-army-knife/, May 2, 2012.

[14] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*, pp. 1–18, Springer, 2001.

[15] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, A concrete security treatment of symmetric encryption. In *Proceedings 38th annual symposium on foundations of computer science*, pp. 394–403, IEEE, 1997.

[16] A. Blum, M. Furst, M. Kearns, and R. J. Lipton, Cryptographic primitives based on hard learning problems. In *Annual international cryptology conference*, pp. 278–291, Springer, 1993.

[17] D. Boneh and M. Franklin, Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pp. 213–229, Springer, 2001.

[18] D. Boneh and R. J. Lipton, Algorithms for black-box fields and their application to cryptography. In *Annual international cryptology conference*, pp. 283–297, Springer, 1996.

[19] D. Boneh and A. Silverberg, Applications of multilinear forms to cryptography. *Contemp. Math.* **324** (2003), no. 1, 71–90.

[20] Z. Brakerski, Fully homomorphic encryption without modulus switching from classical GapSVP. In *Annual cryptology conference*, pp. 868–886, Springer, 2012.

[21] Z. Brakerski, When homomorphism becomes a liability. In *Theory of cryptography conference*, pp. 143–161, Springer, 2013.

[22] Z. Brakerski, Fundamentals of fully homomorphic encryption. In *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*, pp. 543–563, ACM, 2019.

[23] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory* **6** (2014), no. 3, 13.

[24] Z. Brakerski and V. Vaikuntanathan, Efficient fully homomorphic encryption from (standard) lwe. *SIAM J. Comput.* **43** (2014), no. 2, 831–871.

[25] Z. Brakerski and V. Vaikuntanathan, Lattice-based FHE as secure as PKE. In *Innovations in theoretical computer science, ITCS'14*, edited by M. Naor, pp. 1–12, ACM, 2014.

[26] R. Canetti, H. Lin, S. Tessaro, and V. Vaikuntanathan, Obfuscation of probabilistic circuits and applications. In *Theory of cryptography conference*, pp. 468–497, Springer, 2015.

[27] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT (1)*, pp. 409–437, Lecture Notes in Comput. Sci. 10624, Springer, 2017.

[28] J. H. Cheon and D. Stehlé, Fully homomophic encryption over the integers revisited. In *Annual international conference on the theory and applications of cryptographic techniques*, pp. 513–536, Springer, 2015.

[29] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In *Advances in cryptology–ASIACRYPT 2016. ASIACRYPT 2016*, pp. 3–33, Lecture Notes in Comput. Sci. 10031, Springer, 2016.

[30] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In *ASIACRYPT (1)*, pp. 377–408, Lecture Notes in Comput. Sci. 10624, Springer, 2017.

[31] W. Diffie and M. Hellman, New directions in cryptography. *IEEE Trans. Inf. Theory* **22** (1976), no. 6, 644–654.

[32] D. Dolev, C. Dwork, and M. Naor, Nonmalleable cryptography. *SIAM Rev.* **45** (2003), no. 4, 727–784.

[33] L. Ducas and D. M. FHEW, bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT (1)*, pp. 617–640, Lecture Notes in Comput. Sci. 9056, Springer, 2015.

[34] L. Ducas and D. Stehlé, Sanitization of fhe ciphertexts. In *Proceedings, Part I, of the 35th annual international conference on advances in cryptology—EUROCRYPT 2016*, pp. 294–310, Lecture Notes in Comput. Sci. 9665, Springer, 2016.

[35] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **31** (1985), no. 4, 469–472.

[36] A. Feldmann, N. Samardzic, A. Krastev, S. Devadas, R. Dreslinski, K. Eldefrawy, N. Genise, C. Peikert, and D. Sanchez, F1: A fast and programmable accelerator for fully homomorphic encryption (extended version). 2021, arXiv:2109.05371.

[37] M. Fellows and N. Koblitz, Combinatorial cryptosystems galore! *Contemp. Math.* **168** (1994), 51–51.

[38] S. Garg, C. Gentry, and S. Halevi, Candidate multilinear maps from ideal lattices. In *Annual international conference on the theory and applications of cryptographic techniques*, pp. 1–17, Springer, 2013.

[39] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.* **45** (2016), no. 3, 882–929.

[40] C. Gentry, *A fully homomorphic encryption scheme*. Stanford university, 2009.

[41] C. Gentry, Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st ACM symposium on theory of computing—STOC 2009*, pp. 169–178, ACM, 2009.

[42] C. Gentry, Computing arbitrary functions of encrypted data. *Commun. ACM* **53** (2010), no. 3, 97–105.

[43] C. Gentry, Computing on the edge of chaos: Structure and randomness in encrypted computation. In *Proceedings of the international congress of mathematicians (ICM)*, pp. 609–632, Kyung Moon SA, Seoul, 2014.

[44] C. Gentry, S. Halevi, M. Raykova, and D. Wichs, Outsourcing private RAM computation. In *2014 IEEE 55th annual symposium on foundations of computer science*, pp. 404–413, IEEE, 2014.

[45] C. Gentry, S. Halevi, and N. Smart, Fully homomorphic encryption with polylog overhead. In *Advances in cryptology—EUROCRYPT 2012*, pp. 465–482, Lecture Notes in Comput. Sci. 7237, Springer, 2012. Full version at http://eprint.iacr.org/2011/566.

[46] C. Gentry, A. Sahai, and B. Waters, Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster. In *Advances in cryptology—CRYPTO 2013, Part I*, edited by R. Canetti, and J. A. Garay, pp. 75–92, Springer, 2013.

[47] S. Goldwasser and S. Micali, Probabilistic encryption. *J. Comput. System Sci.* **28** (1984), no. 2, 270–299.

[48] S. Halevi, Homomorphic encryption. In *Tutorials on the foundations of cryptography*, pp. 219–276, Springer, 2017.

[49] D. Harvey and J. Van Der Hoeven, Integer multiplication in time $o(n \log n)$. *Ann. of Math.* **193** (2021), no. 2, 563–617.

[50] J. Hoffstein, J. Pipher, and J. H. Silverman NTRU, A ring-based public key cryptosystem. In *ANTS*, edited by J. Buhler, pp. 267–288, Lecture Notes in Comput. Sci. 1423, Springer, 1998.

[51] Homomorphic Encryption Standardization. 2021, https://homomorphicencryption.org.

[52] N. Howgrave-Graham, Approximate integer common divisors. In *International cryptography and lattices conference*, pp. 51–66, Springer, 2001.

[53] A. Jain, H. Lin, and A. Sahai, Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing*, pp. 60–73, ACM, 2021.

[54] A. Joux, A one round protocol for tripartite Diffie–Hellman. In *International algorithmic number theory symposium*, pp. 385–393, Springer, 2000.

[55] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, Over 100× faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus. *IACR Cryptol. ePrint Arch.* **2021** (2021), 508.

[56] R. M. Karp and V. Ramachandran, *A survey of parallel algorithms for shared-memory machines*. United States: N. p., 1989.

[57] A. K. Lenstra, H. W. Jr. Lenstra, M. S. Manasse, and J. M. Pollard, The number field sieve. In *Proceedings of the twenty-second annual ACM symposium on theory of computing*, pp. 564–572, ACM, 1990.

[58] N. Linial, Y. Mansour, and N. Nisan, Constant depth circuits, Fourier transform, and learnability. *J. ACM* **40** (1993), no. 3, 607–620.

[59] A. López-Alt, E. Tromer, and V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pp. 1219–1234, ACM, 2012.

[60] V. Lyubashevsky, C. Peikert, and O. Regev, On ideal lattices and learning with errors over rings. In *Advances in cryptology—EUROCRYPT'10*, edited by H. Gilbert, pp. 1–23, Lecture Notes in Comput. Sci. 6110, Springer, 2010.

[61] U. Mahadev, Classical homomorphic encryption for quantum circuits. *SIAM J. Comput.* **(0):FOCS18–189** (2020).

[62] P. Martins, L. Sousa, and A. Mariano, A survey on fully homomorphic encryption: an engineering perspective. *ACM Comput. Surv.* **50** (2017), no. 6, 1–33.

[63] K. Nuida, Towards constructing fully homomorphic encryption without ciphertext noise from group theory. In *International symposium on mathematics, quantum theory, and cryptography*, pp. 57–78, Springer, Singapore, 2021.

[64] B. Reagen, W.-S. Choi, Y. Ko, V. T. Lee, H.-H. S. Lee, G.-Y. Wei, and D. B. Cheetah, Optimizing and accelerating homomorphic encryption for private inference. In *IEEE international symposium on high-performance computer architecture (HPCA)*, pp. 26–39, IEEE, 2021.

[65] O. Regev, On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on theory of computing*, pp. 84–93, ACM, 2005. Full version in [66].

[66] O. Regev, On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **56** (2009), no. 6, 34:1–34:40.

[67] R. Rivest, L. Adleman, and M. Dertouzos, On data banks and privacy homomorphisms. In *Foundations of secure computation*, pp. 169–177, Academic Press, 1978.

[68] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21** (1978), no. 2, 120–126.

[69] R. Rothblum, Homomorphic encryption: from private-key to public-key. In *Theory of cryptography conference*, pp. 219–234, Springer, 2011.

[70] C.-P. Schnorr, A hierarchy of polynomial time lattice basis reduction algorithms. *Theoret. Comput. Sci.* **53** (1987), no. 2–3, 201–224.

[71] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **41** (1999), no. 2, 303–332.

[72] A. Silverberg, Fully homomorphic encryption for mathematicians. In *Women in numbers 2: research directions in number theory*, p. 111, Contemp. Math. 606, AMS, 2013.

[73] N. P. Smart and F. Vercauteren, Fully homomorphic SIMD operations. *Des. Codes Cryptogr.* **71** (2014), no. 1, 57–81. Early version at http://eprint.iacr.org/2011/133.

[74] V. Vaikuntanathan, Computing blindfolded: new developments in fully homomorphic encryption. In *2011 IEEE 52nd annual symposium on foundations of computer science*, pp. 5–16, IEEE, 2011.

[75] L. G. Valiant, A theory of the learnable. *Commun. ACM* **27** (1984), no. 11, 1134–1142.

[76] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, Fully homomorphic encryption over the integers. In *Advances in cryptology—EUROCRYPT 2010, 29th annual international conference on the theory and applications of cryptographic techniques, French Riviera, May 30—June 3, 2010. Proceedings*, pp. 24–43, Springer, 2010.

[77] W. Van Dam, S. Hallgren, and L. Ip, Quantum algorithms for some hidden shift problems. *SIAM J. Comput.* **36** (2006), no. 3, 763–778.

[78] J. Watrous, Quantum algorithms for solvable groups. In *Proceedings of the thirty-third annual ACM symposium on theory of computing*, pp. 60–67, ACM, 2001.

CRAIG GENTRY

Algorand Foundation, New York, NY, USA, craigbgentry@gmail.com