A FULLY HOMOMORPHIC ENCRYPTION SCHEME

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Craig Gentry
September 2009

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Dan Boneh) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(John Mitchell)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Serge Plotkin)

Approved for the University Committee on Graduate Studies.

Abstract

We propose the first fully homomorphic encryption scheme, solving a central open problem in cryptography. Such a scheme allows one to compute arbitrary functions over encrypted data without the decryption key – i.e., given encryptions $E(m_1), \ldots, E(m_t)$ of $m_1, \ldots, m_t$, one can efficiently compute a compact ciphertext that encrypts $f(m_1, \ldots, m_t)$ for any efficiently computable function f. This problem was posed by Rivest et al. in 1978.

Fully homomorphic encryption has numerous applications. For example, it enables private queries to a search engine – the user submits an encrypted query and the search engine computes a succinct encrypted answer without ever looking at the query in the clear. It also enables searching on encrypted data – a user stores encrypted files on a remote file server and can later have the server retrieve only files that (when decrypted) satisfy some boolean constraint, even though the server cannot decrypt the files on its own. More broadly, fully homomorphic encryption improves the efficiency of secure multiparty computation.

Our construction begins with a somewhat homomorphic "boostrappable" encryption scheme that works when the function f is *the scheme's own decryption function.* We then show how, through recursive self-embedding, bootstrappable encryption gives fully homomorphic encryption. The construction makes use of hard problems on ideal lattices.

Acknowledgments

This thesis would have been impossible without the support and mentoring of my advisor, Dan Boneh. Even after several years of working with him, I am constantly surprised by his amazing intelligence, infinite energy, boundless optimism, and genuine friendliness. I wish I could incorporate more of his qualities. I have limited optimism about my chances.

In a presentation to my fellow Ph.D. admits four years ago, Dan highlighted fully homomorphic encryption as an interesting open problem and guaranteed an immediate diploma to anyone who solved it. Perhaps I took him too literally. He certainly neglected to mention how much writing would be involved. But I have never gone wrong following his advice.

I have also received a lot of input and support from my friends in the IBM Crypto Group, where I've interned for the past couple of summers, and where I will be working permanently – namely, Ran Canetti (now at Tel Aviv University), Rosario Gennaro, Shai Halevi, Charanjit Jutla, Hugo Krawczyk, Tal Rabin, and Vinod Vaikuntanathan (postdoc). These discussions have led to significant performance optimizations. Also, Tal Rabin has been particularly helpful in terms of optimizing my own performance, so that I could finally finish the thesis.

I have had helpful discussions and received comments and suggestions from many other people, including (non-exhaustively): Boaz Barak, Marten van Dijk, Shafi Goldwasser, Iftach Haitner, Michael Hamburg, Susan Hohenberger, Yuval Ishai, Yael Tauman Kalai, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, Oded Regev, Alon Rosen, Amit Sahai, Adam Smith, Salil Vadhan, and Brent Waters.

Contents

List of Tables

Chapter 1

Introduction

We propose a solution to the old open problem of constructing a *fully homomorphic encryption scheme*. This notion, originally called a *privacy homomorphism*, was introduced by Rivest, Adleman and Dertouzous [120] shortly after the invention of RSA by Rivest, Shamir, and Adleman [121]. Basic RSA is a multiplicatively homomorphic encryption scheme – i.e., given RSA public key $\text{pk} = (N, e)$ and ciphertexts $\{\psi_i \leftarrow \pi_i^e \bmod N\}$, one can efficiently compute $\prod_i \psi_i = (\prod_i \pi_i)^e \bmod N$, a ciphertext that encrypts the product of the original plaintexts. One imagines that it was RSA's multiplicative homomorphism, an accidental but useful property, that led Rivest et al. [120] to ask a natural question: What can one do with an encryption scheme that is *fully* homomorphic: a scheme $\mathcal{E}$ with an efficient algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ that, for any valid public key pk, *any* circuit C (not just a circuit consisting of multiplication gates as in RSA), and any ciphertexts $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\text{pk}, \pi_i)$, outputs

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

a valid encryption of $C(\pi_1, \ldots, \pi_t)$ under pk? Their answer: one can arbitrarily *compute on encrypted data* – i.e., one can process encrypted data (query it, write into it, do anything to it that can be efficiently expressed as a circuit) without the decryption key. As an application, they suggested private data banks. A user can store its data on an untrusted server in encrypted form. Later, it can send a query on the data to the server, whereupon the server can express this query as a circuit to be applied to the data, and use the $\mathsf{Evaluate}_{\mathcal{E}}$ algorithm to construct an encrypted response to the user's query, which the user then decrypts. We obviously want the server's response here to be more concise than the trivial

solution, in which the server just sends all of the encrypted data back to the user to process on its own.

Cryptographers have accumulated a long assortment of "killer" applications for fully homomorphic encryption since then. (See Section 1.8.) However, until now, *we did not have a viable construction.*

1.1 A Very Brief and Informal Overview of Our Construction

Imagine you have an encryption scheme with a "noise parameter" attached to each ciphertext, where encryption outputs a ciphertext with small noise – say, less than n – but decryption works as long as the noise is less than some threshold $N \gg n$. Furthermore, imagine you have algorithms Add and Mult that can take ciphertexts $E(a)$ and $E(b)$ and compute $E(a+b)$ and $E(a*b)$, but at the cost of adding or multiplying the noise parameters. This immediately gives a "somewhat homomorphic" encryption scheme that can handle circuits of depth roughly $\log\log N - \log\log n$.

Now suppose that you have an algorithm Recrypt that takes a ciphertext $E(a)$ with noise $N' < N$ and outputs a "fresh" ciphertext $E(a)$ that also encrypts a, but which has noise parameter smaller than $\sqrt{N}$. This Recrypt algorithm is enough to construct a fully homomorphic scheme out of the somewhat homomorphic one! In particular, before we Add or Mult $E(a)$ and $E(b)$, we can apply Recrypt to $E(a)$ and $E(b)$ to ensure that their noise parameters are small enough so that the noise parameter of $E(a*b)$ is less than N, and so on recursively.

In our construction, we give a somewhat homomorphic encryption scheme. We then show how to modify it so that its decryption circuit has multiplicative depth at most $\log\log N - \log\log n - 1$ – i.e., less depth than what the scheme can handle. It turns out that a somewhat homomorphic encryption scheme that has this self-referential property of being able to handle circuits that are deeper than its own decryption circuit – in which case we say the somewhat homomorphic encryption scheme is "bootstrappable" – is enough to obtain the Recrypt algorithm, and thereby fully homomorphic encryption! In Chapter 1.3 and Chapter 4, we give more details on why bootstrappability is enough.

Our somewhat homomorphic encryption scheme, described in Chapters 5 and 7, uses "ideal lattices". In our exposition, we try to defer the need for technical details about lattices for as long as possible. For now, we mention that we looked to ideal lattices as

a way to construct a bootstrappable encryption scheme for two reasons. First, the circuit complexity of the decryption algorithms in typical lattice based encryption schemes is very low, especially compared to schemes like RSA or ElGamal, which rely on exponentiation, an operation that we do not know how to parallelize well. Second, since ideal lattices correspond to ideals in polynomial rings, they inherit natural Add and Mult operations from the ring. (Additionally, ideal lattices are also appealing since we can base security on standard "hard" problems over ideal lattices, which, as far as we know, are typically just as hard as problems over general lattices.)

However, it takes some work to make our somewhat homomorphic scheme bootstrappable – i.e., to make the depth of decryption circuit shallower than what the scheme can handle. In Chapters 8 and 10, we describe how to modify the scheme to make the decryption circuit sufficiently shallow. Conceptually, our techniques here are similar to those used in server-aided cryptography, where (for example) a user with a slow device wants to delegate most of the decryption work to a server without allowing the server to completely decrypt on its own. In our modification, we place a "hint" about the secret key inside the public key. This hint is not enough to decrypt a ciphertext output by the original scheme, but it can be used to "process" the ciphertext – i.e., construct a new ciphertext (that encrypts the same thing) that can be decrypted by a very shallow circuit. To prove that this hint is not too revealing, we require a second computational hardness assumption, similar to ones that have been studied in the context of server-aided cryptography.

Just to leave you with a flavor of what our somewhat homomorphic encryption scheme looks like, consider the following secret key encryption scheme which merely uses integers. The key is an odd integer $p > 2N$. An encryption of a bit b is simply a random multiple of p, plus a random integer B with the same parity as b – i.e., B is even if $b = 0$ and is odd if $b = 1$. A bit more concretely, the ciphertext is $c = b + 2x + kp$, where x is a random integer in $(-n/2, n/2)$, and k is an integer chosen from some range. You decrypt by setting $b \leftarrow (c \bmod p) \mod 2$, where $(c \bmod p)$ is the number in $(-p/2, p/2)$ that equals c modulo p. Actually, $(c \bmod p)$, which is the "noise parameter" in this scheme, will be in $[-n, n]$, since $b + 2x$ is in that range. However, decryption would have worked correctly as long as $b + 2x \in [-N, N] \subset (-p/2, p/2)$. (As an aside relevant to bootstrapping, we mention that computing $c \bmod p$ can be done by a very shallow circuit, with depth logarithmic in the bit-lengths of c and p.)

Now consider what happens when you add two ciphertexts. You get a ciphertext that

has a similar format to the original ones. Specifically,

$$c \leftarrow c_1 + c_2 = b_1 + b_2 + 2(x_1 + x_2) + (k_1 + k_2)p = b_1 \oplus b_2 + 2x + kp$$

for some integers x and k. Decryption recovers $b_1 \oplus b_2$ as long as $(b_1 + 2x_1) + (b_2 + 2x_2) \in [-N, N]$. Multiplication also gives ciphertexts with a similar format.

$$c \leftarrow c_1 * c_2 = b_1 * b_2 + 2(b_1x_2 + b_2x_1 + 2x_1x_2) + kp = b_1 * b_2 + 2x + kp$$

for some integers x and k. Decryption works whenever $(b_1 + 2x_1) * (b_2 + 2x_2) \in [-N, N]$.

A crucial advantage of replacing integers in the scheme above with ideal lattices is that an ideal lattice has many representations or "bases". Some bases are "good" and can be used as the secret key, while some are "bad" and can be used as the public key – i.e., they are good enough to be used for encryption, but not decryption. So, ideal lattices give us a public key scheme. On the other hand, it is unclear whether the integer p in the toy scheme above can be represented in a way that is useful for encryption but not decryption (nor is security clear even for the secret key version of the scheme).

But, for a moment, imagine that there are good and bad representations of p, such the bad representation can be used in encryption but cannot be used to distinguish whether an integer is close to a multiple of p or is uniform modulo p. How would we prove security? If there is an adversary $\mathcal{A}$ that can break semantic security, $\mathcal{B}$ uses $\mathcal{A}$ to decide which distribution an integer m comes from as follows: give $\mathcal{A}$ the challenge ciphertext $c = b + 2m + kp$ for random k. If m is close to a multiple of p, then so is $2m$, and the closest p-multiple is an even distance away; in particular, $b + 2m \in [-N, N] \bmod p$ and $b + 2m \bmod p = b$, the challenge ciphertext decrypts correctly to b, and $\mathcal{A}$ should guess b with non-negligible advantage. But if m is uniform modulo p, then so is $2m$ (since p is odd), c is independent of b, and $\mathcal{A}$ has no advantage. Basically, $\mathcal{B}$ can distinguish the distribution that m came from by observing whether $\mathcal{A}$ guesses correctly with non-negligible advantage. In Chapter 5, we provide a conceptually similar proof of our ideal lattice scheme based on the ideal coset problem (ICP).

Over the next few Sections, we provide more details about our construction, its security and applications, but still somewhat informally.

1.2 What is Fully Homomorphic Encryption?

Our ultimate goal is to construct a fully homomorphic encryption scheme $\mathcal{E}$. First, let us discuss what it means to be *fully homomorphic*.

At a high-level, the essence of fully homomorphic encryption is simple: given ciphertexts that encrypt $\pi_1, \ldots, \pi_t$, fully homomorphic encryption should allow anyone (not just the key-holder) to output a ciphertext that encrypts $f(\pi_1, \ldots, \pi_t)$ for any desired function f, as long as that function can be efficiently computed. No information about $\pi_1, \ldots, \pi_t$ or $f(\pi_1, \ldots, \pi_t)$, or any intermediate plaintext values, should leak; the inputs, output and intermediate values are always encrypted.

Formally, there are different ways of defining what it means for the final ciphertext to "encrypt" $f(\pi_1, \ldots, \pi_t)$. The minimal requirement is correctness. A fully homomorphic encryption scheme $\mathcal{E}$ should have an efficient algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ that, for any valid $\mathcal{E}$ key pair $(\mathrm{sk}, \mathrm{pk})$, any circuit C, and any ciphertexts $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_i)$, outputs

$$\psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t) \quad \text{such that} \quad \mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi) = C(\pi_1, \ldots, \pi_t)$$

This minimal requirement does not seem to be sufficient, however, since it permits the trivial solution, where ψ simply consists of $(C, \psi_1, \ldots, \psi_t)$ – i.e., where the $\mathsf{Evaluate}_{\mathcal{E}}$ algorithm does not "process" the input ciphertexts at all.

There are a couple of different ways of excluding the trivial solution. One way is to require *circuit privacy* – i.e., (roughly) that the output of $\mathsf{Evaluate}_{\mathcal{E}}$ reveals nothing (at least computationally) about the circuit C that it took as input. If circuit privacy is the only additional requirement, then fully homomorphic encryption (under this definition) can easily be achieved by using a two-flow oblivious transfer (OT) protocol in combination with Yao's garbled circuit [129, 130]. Typically two-flow OT protocols use an additively homomorphic encryption scheme, and the OT query consists of a ciphertext ψ in this encryption scheme. In the fully homomorphic scheme, $\mathsf{Evaluate}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t)$ constructs a Yao garbling $C^\dagger$ of C, uses the OT queries $\psi_1, \ldots, \psi_t$ to construct OT responses $\psi_1^*, \ldots, \psi_t^*$ designed to obliviously transfer Yao keys associated to the t input wires in $C^\dagger$, and outputs $(C^\dagger, \psi_1^*, \ldots, \psi_t^*)$. To decrypt this ciphertext, the key holder "decrypts" the OT responses $\psi_1^*, \ldots, \psi_t^*$ to recover Yao keys for the input wires, and then evaluates the garbled circuit. Sanders, Young and Yung [122] and Beaver [14] show how to achieve *statistical* circuit privacy, but only for limited classes of circuits – namely, NC1 and NLOGSPACE.

The more interesting way of excluding the trivial solution is to require (roughly) that the ciphertext encrypting $C(\pi_1, \ldots, \pi_t)$ should "look like" an "ordinary" ciphertext, as long as $C(\pi_1, \ldots, \pi_t)$ is a single bit (or element of the same plaintext space that contains $\{\pi_i\}$). For example, the size of the ciphertext output by $\mathsf{Evaluate}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t)$ should not depend on C. We focus on this definition. Actually, we use a stronger requirement: that $\mathsf{Decrypt}_{\mathcal{E}}$ be expressible by a circuit $D_{\mathcal{E}}$, which takes a (formatted) secret key and (formatted) ciphertext as input, and *whose size is (a fixed) polynomial in the security parameter*. Of course, this implies that there is an upper bound on the ciphertext size that depends only on the security parameter, and is independent of C. After describing a scheme that meets this definition, we will also describe how to achieve (statistical) circuit privacy (Chapter 20).

To some, it is surprising that such a thing as fully homomorphic encryption is possible even in principle. To see that it is possible, it may be helpful to understand fully homomorphic encryption in terms of a physical analogy – e.g., a photograph developer's darkroom. The developer applies a particular function f to Alice's film when he develops it – i.e., the sequence of steps to develop the film. In principle, he does not need to see anything to apply this procedure, though in practice darkrooms are typically not completely dark. Of course, this analogy is inadequate in that one may ask: why can't the developer walk out of the darkroom and look at the finished product? Imagine that the developer is blind. Then, one may ask: why can't someone else look at the finished product? Imagine that everyone in the world besides Alice is blind. "Sight" is Alice's secret key, and (in this world) it is impossible for anyone else to simulate vision. Although imagining physical analogies should convince you that the notion of fully homomorphic encryption is not a logical fallacy, it seems difficult to construct a perfect physical analogue of fully homomorphic encryption that is not rather far-fetched.

To try another physical analogy, suppose that the owner of a jewelry store (Alice) wants her employees to assemble raw precious materials (diamonds, gold, etc.) into finished products, but she is worried about theft. She addresses the problem by constructing glove boxes for which only she has the key, and she puts the raw materials inside. Using the gloves, an employee can manipulate the items inside the box. Moreover, an employee can put things inside the box – e.g., a soldering iron to use on the raw materials – even though he cannot take anything out. Also, the box is transparent, so that an employee can see what he is doing. (In this analogy, encryption means that the employee is unable to take something out of the box, not that he is unable to see it.) After the employee is done,

Alice can recover the finished product at her leisure by using her key. This analogy is inadequate in the sense that the glove box might become quite cluttered, whereas in the fully homomorphic encryption scheme only the final product need remain. In other words, to improve the analogy, imagine that the employee has some way to make any item in the glove box (of his choosing) disappear (even though he still cannot extract the item).

1.3 Bootstrapping a Scheme that Can Evaluate its Own Decryption Circuit

Now that we have clarified our goal (fully homomorphic encryption), let us try to find a steppingstone. Suppose that, *a priori*, we have a scheme $\mathcal{E}$ that is only guaranteed to be correct for some subset $\mathcal{C}_{\mathcal{E}}$ of circuits – i.e.,

$$\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t)) = C(\pi_1, \ldots, \pi_t)$$

is guaranteed to hold only if $C \in \mathcal{C}_{\mathcal{E}}$ (and, as before, $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_i)$). Can we use $\mathcal{E}$ to construct a scheme $\mathcal{E}^*$ that is fully homomorphic?

In Chapter 4, we show that the answer is yes. Suppose that $\mathcal{C}_{\mathcal{E}}$ contains just two circuits: $D_{\mathcal{E}}$ and the augmentation of $D_{\mathcal{E}}$ by NAND (i.e., a NAND gate connecting two copies of $D_{\mathcal{E}}$), where $D_{\mathcal{E}}$ is the circuit associated to the decryption algorithm.[1] If $\mathcal{E}$ has this self-referential property of being able to evaluate its own (augmented) decryption circuit, we say that $\mathcal{E}$ *bootstrappable*. We show that bootstrappable encryption implies *leveled fully homomorphic encryption* – i.e., that $D_{\mathcal{E}}$ plus the NAND-augmentation of $D_{\mathcal{E}}$ constitute a "complete" set of circuits:

Theorem 1.3.1 (Informal)**.** *If $\mathcal{E}$ is bootstrappable, then, for any integer d, one can construct a scheme $\mathcal{E}^{(d)}$ that can evaluate any circuit (consisting of NAND gates) of depth d. The decryption circuit for $\mathcal{E}^{(d)}$ is the same as for $\mathcal{E}$, and the complexity of encryption is also the same. $\mathcal{E}^{(d)}$'s public key size is $O(d)$ times that of $\mathcal{E}$'s. The complexity of $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$ is polynomial in the security parameter and linear in the circuit size. If $\mathcal{E}$ is semantically secure against chosen plaintext attacks, then so is $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$.*

One drawback of $\mathcal{E}^{(d)}$ is that its public key is $O(d)$ times that of $\mathcal{E}$'s public key. Since

[1] We use NAND because any circuit can be expressed in terms of NAND gates. We could instead augment the decryption circuit by a different set of universal gates.

$\mathcal{E}^{(d)}$ has this unwanted dependence on d, we say that it is merely *leveled* fully homomorphic. Under certain assumptions, we can make the $\mathcal{E}^{(d)}$ public key size be independent of d, in which case we say the derived scheme is *fully homomorphic*.

Why should the fact that $\mathcal{E}$ can evaluate (augmentations of) $D_{\mathcal{E}}$ be so powerful? Suppose that the distributions of $\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t)$ and $\mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, C(\pi_1, \ldots, \pi_t))$ are different. In particular, suppose that there is an "error" associated with each ciphertext, that ciphertexts output by $\mathsf{Encrypt}_{\mathcal{E}}$ have small error, that ciphertexts output by $\mathsf{Evaluate}_{\mathcal{E}}$ have larger error that increases with the depth of the circuit being evaluated, and that eventually (as the depth of the circuit being evaluated increases) the "error" becomes so large that applying $\mathsf{Decrypt}_{\mathcal{E}}$ to the ciphertext results in a decryption error. (In fact, this is the case in our initial ideal lattice construction.) Intuitively, as we are evaluating a circuit and the implicit "error" becomes large, we would like to "refresh" the ciphertext so that the error becomes small again. Obviously, we could refresh a ciphertext if we could completely decrypt it, simply by generating an entirely new and fresh ciphertext that encrypts the same thing, but we want a way to refresh that does not require the secret key. This is the idea behind bootstrapping: we *do* decrypt the ciphertext, but homomorphically!

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

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

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

Above, $\mathsf{Evaluate}$ takes in the bits of sk_1 and ψ_1, each encrypted under pk_2. Then, $\mathcal{E}$ is used to evaluate the decryption circuit homomorphically. The output ψ_2 is thus an encryption under pk_2 of $\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}_1, \psi_1) = \pi$.[2] In other words, $\mathsf{Recrypt}$ decrypts homomorphically using the encrypted secret key, thus obtaining a new ciphertext that encrypts the same thing as the original one.

[2] $\mathsf{Recrypt}$ implies a one-way multi-use proxy re-encryption scheme [19]. We discuss this in more detail in Section 1.8.

Notice how π is doubly encrypted at one point, and we use $\mathsf{Evaluate}_{\mathcal{E}}$ to remove the *inner* encryption. Applying the decryption circuit $D_{\mathcal{E}}$ removes the "error" associated to the first ciphertext under pk_1, but $\mathsf{Evaluate}_{\mathcal{E}}$ simultaneously introduces a new "error" while evaluating the ciphertexts under pk_2. Intuitively, we have made progress as long as the second error is shorter. Note that revealing the encrypted secret key bits $\langle \overline{\mathrm{sk}_{1j}} \rangle$ does not compromise semantic security; these encrypted secret key bits are indistinguishable from encryptions of 0 as long as $\mathcal{E}$ is semantically secure by a standard hybrid argument. This hybrid argument breaks down if $\mathrm{pk}_1 = \mathrm{pk}_2$. However, if $\mathcal{E}$ securely encrypts key-dependent messages (is KDM-secure) [18, 68, 22] – i.e., roughly, if providing a ciphertext that encrypts a function of the secret key does not hurt security – then $\mathsf{Recrypt}$ can have a "self-loop" of encrypted secret keys.

Of course, our goal is to perform nontrivial homomorphic operations on underlying plaintexts, not merely to obtain refreshed encryptions of the same plaintext. If we can also evaluate a NAND augmentation of the decryption circuit, then we can generate an encryption of (π_1 NAND π_2) under pk_2 using the encrypted secret key (sk_1 under pk_2) together with the two ciphertexts encrypting π_1 and π_2, respectively, under pk_1. By recursively performing this type of operation on all ciphertexts at a given level in the circuit, we can evaluate a d-depth circuit of NANDs. If $\mathcal{E}$ is KDM-secure, the derived scheme is fully homomorphic (rather than leveled fully homomorphic). In the random oracle model, we show that a bootstrappable encryption scheme implies a scheme that is both bootstrappable and KDM-secure, and thus implies a fully homomorphic encryption scheme.

Constructing an efficient (leveled) fully homomorphic encryption scheme without using bootstrapping, or using some relaxation of it, remains an interesting open problem.

Again, it may be helpful to view bootstrapping in terms of a physical analogy, although it will, of course, be even more far-fetched. Recall Alice, our jewelry store owner. Imagine that Alice's glove boxes are defective; after an employee uses the gloves for 1 minute, the gloves stiffen and become unusable. Unfortunately for Alice, even her fastest employee cannot assemble some of the more intricate designs in under a minute. But Alice is not only paranoid, but also smart. To an employee that is assembling an intricate design, she gives him (like before) a glove box containing the raw materials, but also several additional glove boxes. Each of these additional glove boxes holds a copy of her master key. To assemble the intricate design, the employee manipulates the materials in box #1 until the gloves stiffen. Then, he places box #1 inside box #2, where the latter box already contains

a master key. Using the gloves for box #2, he opens box #1 with the master key, extracts the partially assembled trinket, and continues the assembly within box #2 until its gloves stiffen. He then places box #2 inside box #3, and so on. When the employee finally finishes his assembly inside box $\#n$, he hands the box to Alice. Of course, this trick will not work unless the employee can open box $\#i$ within box $\#(i+1)$, and have time to make a little bit of progress on the assembly, all before the gloves of box $\#(i+1)$ stiffen. This is analogous to the requirement for a bootstrappable encryption scheme $\mathcal{E}$ – that the complexity of $\mathcal{E}$'s (augmented) decryption circuit is less than what $\mathcal{E}$ can homomorphically evaluate.

We assumed that it was safe to use a single master key that opens all boxes. But maybe it is not safe; maybe an employee could use the gloves for box #2, together with master key inside that box, to open the box from the inside, extract the key, and use it to open box #1 and steal the jewels. However, Alice can avoid this circumstance by using distinct keys for the boxes, and placing the key for box #1 inside box #2, the key for box #2 inside box #3, and so on. This is analogous to the question of whether the encryption scheme is KDM-secure.

As before, the physical analogy only goes so far. In the physical case, box $\#i$ would grow as i increases, and consequently the extraction time would also grow, but our encryption scheme does not have analogous deficiencies. And, again, in our physical analogy, encryption corresponds to being unable to physically access the contents of the box. So, it is not a valid attack for the employee to copy the master key based on what he can *see* through the transparent box. Accordingly, it might be helpful to think of each key as having a certain secret chemical composition which cannot be readily ascertained while the key is inside the box, and that a key opens its associated box through a chemical reaction.

1.4 Ideal Lattices: Ideally Suited to Construct Bootstrappable Encryption

The notion of bootstrappability gives us a new angle on constructing fully homomorphic encryption: it suggests we should look at encryption schemes whose decryption algorithms have low circuit complexity. Within the bootstrappability framework, it does not make much sense to look at exponentiation-based schemes, since exponentiation (as used in RSA, for example) is not even known to be in NC. On the other hand, encryption schemes using lattices or linear codes have very simple decryption algorithms typically dominated by a

matrix-vector multiplication, an operation in NC1. In this paper, we focus on constructing a lattice-based scheme (though we view, say, a code-based construction as an interesting possibility).

Of course, it is not enough to minimize the circuit complexity of decryption; we also must *maximize* the evaluative capacity of the scheme, so that the scheme can evaluate its own (augmented) decryption circuit. While one can easily construct an *additively* homomorphic scheme from ordinary lattices, we need a scheme with both *additive and multiplicative* homomorphisms to evaluate general circuits. This consideration leads us to focus on *ideal lattices*.

In Chapter 7, we describe our initial homomorphic encryption scheme based on ideal lattices. However, one can understand the scheme reasonably well just in terms of rings and ideals (no lattices). Rings and ideals are simple algebraic objects. Examples of rings are $\mathbb{Z}$ (the integers) and the polynomial ring $\mathbb{Z}[x]/(f(x))$, consisting of the residues of integer polynomials modulo a monic polynomial $f(x)$. Rings are closed under addition '+', multiplication '$\times$,' and additive inverse, and have an additive identity '0' and multiplicative identity '1.' An ideal I of a ring R is a subset $I \subseteq R$ such that $\sum_{j=1}^{t} i_j \times r_j \in I$ for any $i_1, \ldots, i_t \in I$ and $r_1, \ldots, r_t \in R$. For example, (2) is an ideal of $\mathbb{Z}$ consisting of the set of even numbers. An example ideal in $\mathbb{Z}[x]/(f(x))$ is $(a(x))$, the set of multiples of $a(x)$ (reduced modulo $f(x)$). However, by these examples, we do not mean to imply that ideals are necessarily *principal*; they may not be generated by a single element. If I is a proper subset of R, we can talk about a *coset* of I within R; e.g., $1 + (2)$ is a coset consisting of the odd numbers. The element $x \in R$ is in the coset $y + I$ if $x - y \in I$. Many of the previous constructions of (partially) homomorphic encryption use rings and ideals, at least implicitly; see Chapter 3.

As a first approximation, here is how a fully homomorphic encryption scheme based on rings and ideals might work. The public key pk contains an ideal I and a plaintext space $\mathcal{P}$, where the latter basically consists of a set of "distinguished representatives" of the cosets of I; the secret key sk consists of some "secret knowledge" concerning I. To encrypt $\pi \in \mathcal{P}$, the encrypter sends $\psi \xleftarrow{R} \pi + I$, a "random" member of the coset $\pi + I$. The decrypter uses its secret knowledge to recover the "distinguished representative" π (distinguished with respect to $\mathcal{P}$) of the coset $\pi + I$. To add and multiply ciphertexts, we simply use the ring

operations '+' and '×':

$$\begin{aligned} \mathsf{Add}(\mathrm{pk}, \psi_1, \psi_2) &= \psi_1 + \psi_2 \in (\pi_1 + \pi_2) + I \\ \mathsf{Mult}(\mathrm{pk}, \psi_1, \psi_2) &= \psi_1 \times \psi_2 \in (\pi_1 \times \pi_2) + I \end{aligned}$$

Ring operations on ciphertexts induce mod-I operations on the underlying plaintexts. In general, for an arithmetized mod-I circuit C, we would have

$$\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t) \in C(\pi_1, \ldots, \pi_1) + I$$

The semantic security of this scheme relies on the hardness of an *ideal membership problem* – i.e., given π' and ψ, is $\psi - \pi' \in I$? This is the approach of the Polly Cracker scheme by Fellows and Koblitz, described in Chapter 3.

The first approximation above does not work for ideal lattices, unfortunately, since the ideal membership problem is not hard. An ideal lattice, as used in this paper, is simply an ideal in $\mathbb{Z}[x]/(f(x))$, $f(x)$ of degree n; each such ideal I can be represented by a lattice generated by the columns of a lattice basis $\mathbf{B}_I$, an $n \times n$ matrix. It so happens that, for any basis $\mathbf{B}_I$ of any lattice (not just an ideal lattice) I and any $\mathbf{v} \in \mathbb{Z}^n$, there is a unique, efficiently-computable distinguished representative $\mathbf{v} \bmod \mathbf{B}_I$. In particular, it holds that $\mathbf{v} \bmod \mathbf{B}_I = \mathbf{v} - \mathbf{B}_I \cdot \lfloor \mathbf{B}_I^{-1} \cdot \mathbf{v} \rceil$, where $\mathbf{B}_I^{-1}$ is the matrix inverse of $\mathbf{B}_I$ and $\lfloor \cdot \rceil$ rounds to the nearest integer vector. To find the distinguished representative for $r \in R$ modulo $\mathbf{B}_I$, one computes $\mathbf{r} \bmod \mathbf{B}_I$ where $\mathbf{r}$ is the coefficient vector of r. To test whether r is a member of I, one simply tests whether $\mathbf{r} \bmod \mathbf{B}_I = 0 \bmod \mathbf{B}_I$. Thus, the ideal membership problem is easy.

So, we use a different approach that involves *two ideals*. Everybody can use a common ideal I, represented by basis $\mathbf{B}_I$. Then, each user generates their own ideal J, with secret and public bases $\mathbf{B}_J^{\mathrm{sk}}$ and $\mathbf{B}_J^{\mathrm{pk}}$, that is relatively prime to I (i.e., $I + J = R$). As before, the plaintext space $\mathcal{P}$ consists of distinguished representatives of the cosets of I. The public key pk also includes the description of a distribution D. To encrypt $\pi \in \mathcal{P}$, the encrypter sets $\pi^* \stackrel{\mathrm{D}}{\leftarrow} \pi + I$, and sends $\psi \leftarrow \pi^* \bmod \mathbf{B}_J^{\mathrm{pk}}$. In other words, the ciphertext has the form $\psi = \pi + i + j$ for $i \in I$ and $j \in J$, where $\pi + i$ comes from the specified distribution D. The decrypter sets

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

For decryption to work, the secret key $\mathbf{B}_J^{\mathrm{sk}}$ should be chosen so as to be compatible with the distribution D, so that $\pi + i$ is always the distinguished representative of $\pi + i + J$ with respect to $\mathbf{B}_J^{\mathrm{sk}}$. In this case, the mod-$\mathbf{B}_J^{\mathrm{sk}}$ operation returns $\pi + i$, after which π is recovered easily. This decryption criterion becomes more complicated as we add and multiply ciphertexts using the basic ring operations. For arithmetized circuit C that uses addition and multiplication modulo I (w.r.t. basis $\mathbf{B}_I$), we have:

$$\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t) = C(\psi_1, \ldots, \psi_t) \in C(\pi_1 + i_1, \ldots, \pi_t + i_t) + J$$

where $i_1, \ldots, i_t \in I$. (The above is an abuse of notation, since on the left C consists of gates that add and multiply the underlying plaintexts modulo I, while in the middle and on the right C uses the ring operations '+' and '×', but we will use this for now.) In this case, for decryption to work, we need $C(\pi_1 + i_1, \ldots, \pi_t + i_t)$ to be the distinguished representative of $C(\pi_1 + i_1, \ldots, \pi_t + i_t) + J$ w.r.t. $\mathbf{B}_J^{\mathrm{sk}}$. We can reverse this statement, and say that the set $\mathcal{C}_{\mathcal{E}}$ of circuits that the scheme $\mathcal{E}$ evaluates correctly consists of those circuits for which $C(\pi_1 + i_1, \ldots, \pi_t + i_t)$ is always the distinguished representative of $C(\pi_1 + i_1, \ldots, \pi_t + i_t) + J$ w.r.t. $\mathbf{B}_J^{\mathrm{sk}}$ when $\mathbf{B}_J^{\mathrm{sk}}$ is generated according to $\mathsf{KeyGen}_{\mathcal{E}}$ and π_k and i_k are chosen according to $\mathsf{Encrypt}_{\mathcal{E}}$. In this case, the mod-$\mathbf{B}_J^{\mathrm{sk}}$ operation recovers $C(\pi_1 + i_1, \ldots, \pi_t + i_t)$, after which the decrypter easily recovers $C(\pi_1, \ldots, \pi_t)$ by reducing modulo $\mathbf{B}_I$.

This characterization of $\mathcal{C}_{\mathcal{E}}$ becomes less nebulous when, in the context of lattices, we give a geometric interpretation to $C(\pi_1 + i_1, \ldots, \pi_t + i_t)$ as a vector indicating the ciphertext vector's "error" or "offset" from the lattice J. In this setting, the distinguished representatives of the cosets of J w.r.t. the basis $\mathbf{B}_J^{\mathrm{sk}}$ are precisely the points in $\mathbb{Z}^n$ that are contained inside the parallelepiped $\mathcal{P}(\mathbf{B}_J^{\mathrm{sk}}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \sum x_i \cdot \mathbf{b}_i, x_i \in [-1/2, 1/2)\}$ associated to the basis $\mathbf{B}_J^{\mathrm{sk}} = \{\mathbf{b}_i\}$. Decryption works as long as the "error vector" is never so long that it falls outside of $\mathcal{P}(\mathbf{B}_J^{\mathrm{sk}})$.[3] Once we specify some radius r_{Dec} such that the parallelepiped $\mathcal{P}(\mathbf{B}_J^{\mathrm{sk}})$ always contains a ball of radius r_{Dec} inside it (for any J generated according to KeyGen), and also specify a radius r_{Enc} such that (in $\mathsf{Encrypt}_{\mathcal{E}}$) the vector $\pi^* \xleftarrow{\mathrm{D}} \pi + I$ always falls within a ball of radius r_{Enc}, the bootstrappability question becomes: is $C(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in \mathcal{B}(r_{\mathsf{Dec}})$ whenever $\mathbf{x}_i \in \mathcal{B}(r_{\mathsf{Enc}})$ for all i and C is an (augmented)

[3] If the error vector does fall outside $\mathcal{P}(\mathbf{B}_J^{\mathrm{sk}})$, the mod-$\mathbf{B}_J^{\mathrm{sk}}$ operation in decryption returns $C(\pi_1 + i_1, \ldots, \pi_t + i_t) + j$ for some nonzero $j \in J$, and the subsequent reduction modulo I is unlikely to return $C(\pi_1, \ldots, \pi_t)$, since J is relatively prime to I. Interestingly, NTRU [69] uses relatively prime ideals in a similar way.

decryption circuit?

We can upper-bound the length of $C(\mathbf{x}_1, \ldots, \mathbf{x}_t)$ for arithmetic circuit C recursively by upper-bounding the "expansion" caused by additions and multiplications. Roughly speaking, we can say that Add operations do not increase the length of the error vector much: if $\psi_1 \in \mathbf{x}_1 + J$ and $\psi_2 \in \mathbf{x}_2 + J$, then $\mathsf{Add}(\mathrm{pk}, \psi_1, \psi_2) \in (\mathbf{x}_1 + \mathbf{x}_2) + J$, where $\|\mathbf{x}_1 + \mathbf{x}_2\| \leq \|\mathbf{x}_1\| + \|\mathbf{x}_2\|$ by the triangle inequality. Mult operations are more expensive; we can show that, for any polynomial ring R, there is a parameter $\gamma_{\mathsf{Mult}}(R)$ such that $\|\mathbf{x}_1 \times \mathbf{x}_2\| \leq \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{x}_1\| \cdot \|\mathbf{x}_2\|$; $\gamma_{\mathsf{Mult}}(R)$ may be, e.g., polynomial in n. (For the Mult operation, vector $\mathbf{x}_i$ is interpreted as the polynomial in R whose coefficient vector is $\mathbf{x}_i$.) Essentially, constant-fan-in Mult gates cause at least as much expansion as polynomial-fan-in Add gates. In the worst case, Mult gates cause the length of the error vector essentially to square with each additional level of the circuit, limiting the circuit depth that the scheme can evaluate to (roughly) $\log \log r_{\mathsf{Dec}}$.

Theorem 1.4.1 (Informal)**.** *Suppose $X \subseteq \mathcal{B}(r_X)$ and $Y \supseteq \mathcal{B}(r_Y)$, $r_X \geq 1$. Then, $C(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in Y$ for all $\mathbf{x}_1, \ldots, \mathbf{x}_t \in X$ and all arithmetic (over R) circuits with multiplicative fan-in of 2, additive fan-in of up to $\gamma_{\mathsf{Mult}}(R)$, and depth up to*

$$\log \log r_Y - \log \log(\gamma_{\mathsf{Mult}}(R) \cdot r_X)$$

I.e., $\mathcal{E}$ correctly evaluates all such circuits of depth up to $\log \log r_{\mathsf{Dec}} - \log \log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$.

So, can we express the (augmented) decryption circuit with depth at most (roughly) $\log \log r_{\mathsf{Dec}}$? Unfortunately, the answer appears to be 'no,' though it is a close call. Specifically, the dominant computation in decryption is $\lfloor (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi \rceil$, which occurs within the computation of $\psi \bmod \mathbf{B}_J^{\mathrm{sk}}$. Roughly speaking, to ensure that the rounding is correct, one must use a sufficient number of bits of precision. Then, the high precision of each number-number multiplication that occurs within the matrix-vector multiplication forces us to use a high-depth circuit. Specifically, two k-bit numbers can be multiplied together using a $O(\log k)$-depth circuit (with constant fan-in). The precision we seem to need is roughly $\log \det(J) > n \cdot \log r_{\mathsf{Dec}}$ bits, and therefore we need about a $O(\log n + \log \log r_{\mathsf{Dec}})$-depth circuit.

Unfortunately, for this initial scheme, it seems that no matter how the parameters are set, the decryption circuit is always slightly too complex for the scheme to evaluate.[4]

[4]However, we do not prove this. It remains possible that the decryption circuit of this initial scheme can

This problem is difficult to fix *post hoc*, in part due to the self-referential nature of the bootstrapability property: intuitively, if one expands the set of circuits that $\mathcal{E}$ can "handle" in an effort to include $D_{\mathcal{E}}$, one seemingly must increase the complexity of $\mathsf{Decrypt}_{\mathcal{E}}$ to accommodate, thereby making the circuit $D_{\mathcal{E}}$ more complex, possibly such that $D_{\mathcal{E}}$ always elusively falls outside of the expanded set. To obtain a bootstrappable encryption scheme, it seems necessary to change the decryption algorithm fundamentally.

1.5 Squashing the Decryption Circuit: The Encrypter Starts Decryption!

To reduce the decryption complexity without affecting the "evaluative capacity" of the scheme at all, our approach, given in Chapter 10, is to enable the *encrypter* to start decryption, thereby easing the burden on the decrypter. Interestingly, the setting is similar to server-aided cryptography, where a user offloads some portion of a computationally intensive cryptographic task, such as decryption, onto an untrusted server; in our case, the encrypter itself plays the server's role.

Abstractly, if $\mathcal{E}^*$ is our original homomorphic encryption scheme, with public and secret keys $(\mathrm{pk}^*, \mathrm{sk}^*)$, the modified scheme $\mathcal{E}$ uses an algorithm that we call $\mathsf{SplitKey}$ to generate a "hint" τ about sk^*, which it puts in the $\mathcal{E}$ public key. Also, $\mathcal{E}$ uses a new algorithm $\mathsf{ExpandCT}$. The encrypter uses this algorithm, in combination with the hint τ, to transform a preliminary ciphertext ψ^* output by $\mathcal{E}^*$ into an "expanded ciphertext" that can be decrypted by a shallower circuit. Here is the abstract transformation in detail; since it is abstract, it is obviously not explained at this point why the expanded ciphertext is easier to decrypt.

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

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

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

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

be expressed in a way that makes the scheme bootstrappable.

We (half facetiously) say that the "encrypter starts decryption" because it uses the secret-key-related value τ to expand the ciphertext in a way that helps reduce the decrypter's circuit complexity. The introduction of τ into the public key provides a "hint" about the secret key sk of the original scheme $\mathcal{E}^*$. However, it is easy to see that $\mathcal{E}$ is semantically secure as long as $\mathcal{E}^*$ is, as long as the following $\mathsf{SplitKey}$ distinguishing problem is hard: given $(\mathrm{pk}^*, \mathrm{sk}^*, \tau)$, distinguish whether τ was generated as the output of $\mathsf{SplitKey}_{\mathcal{E}}(\mathrm{sk}^*, \mathrm{pk}^*)$ (as it should be), or as the output of $\mathsf{SplitKey}_{\mathcal{E}}(\perp, \mathrm{pk}^*)$, where $\perp$ is some distinguished symbol that is independent of sk^*. In the latter case, τ gives no additional information about sk^* that could weaken security.

Theorem 1.5.1 (Informal)**.** *If there is an algorithm $\mathcal{A}$ that breaks the squashed scheme with non-negligible probability, then there is either an algorithm $\mathcal{B}_1$ that breaks the original scheme or an algorithm $\mathcal{B}_2$ that solves the* $\mathsf{SplitKey}$ *distinguishing problem with non-negligible advantage.*

Concretely, we actually apply a couple of technical "tweaks" to our original ideal-lattice-based construction before we apply the above transformation. In one tweak, we show how to simplify the decryption equation in the original scheme from $(\psi^* \bmod \mathbf{B}_J^{\mathrm{sk}}) \bmod \mathbf{B}_I = (\psi^* - \mathbf{B}_J^{\mathrm{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi^* \rceil) \bmod \mathbf{B}_I$ to $(\psi^* - \lfloor \mathbf{v}_J^{\mathrm{sk}} \times \psi^* \rceil) \bmod \mathbf{B}_I$ where '$\times$' is ring multiplication and $\mathbf{v}_J^{\mathrm{sk}} \in \mathbb{Q}^n$. The new secret key $\mathbf{v}_J^{\mathrm{sk}}$ is slightly weaker than the original one, which forces us to reduce r_{Dec} by a polynomial factor (which is insignificant if r_{Dec} is super-polynomial anyway, as it is required to be to obtain our fully homomorphic scheme). Other than that, the modification has no effect on the correctness or security of the scheme. The purpose of the tweak is merely to reduce the size of the tag τ introduced by the above transformation. (We will discuss what τ is in concrete terms momentarily.) The second tweak is to limit the set of "permitted circuits" to those for which the length of the "error" vector never exceeds $r_{\mathsf{Dec}}/2$, rather than r_{Dec}. The purpose of this tweak is to ensure that the coefficients of the vector $\mathbf{v}_J^{\mathrm{sk}} \times \psi^*$ are bounded away from half-integers when ψ^* is a valid ciphertext. In particular, all of the coefficients will be within $1/4$ of an integer; this allows us to simplify the decryption circuit while still ensuring that the rounding operation $\lfloor \mathbf{v}_J^{\mathrm{sk}} \times \psi^* \rceil$ yields the correct answer. Aside from very slightly reducing the evaluative capacity of the scheme, this tweak also has no negative effect.

Now, in our concrete instantiation of $\mathsf{SplitKey}_{\mathcal{E}}$, τ is a random set S (with $\omega(n)$, but $\mathrm{poly}(n)$, members) of vectors $\{\mathbf{u}_i\}$ that has a sparse subset T (with $\omega(1)$, but $o(n)$, members)

whose sum is $\mathbf{v}_J^{\text{sk}}$ modulo I; the new secret key sk is the subset T, encoded as a 0/1-vector in $\{0,1\}^{|S|}$. Distinguishing whether or not the vectors in S are completely uniform and independent of sk^* is a lattice-related problem, whose search version (actually finding the subset) has been studied in the context of server-aided cryptography [91, 114, 106, 96, 105]. We discuss this problem a bit more in the next Section.

In the modified scheme, $\mathsf{ExpandCT}_{\mathcal{E}}$ outputs $\{\mathbf{c}_i \leftarrow \mathbf{u}_i \times \psi^* \bmod \mathbf{B}_I : \mathbf{u}_i \in S\}$. To oversimplify, $\mathsf{Decrypt}_{\mathcal{E}}$ sums up the values $\mathbf{c}_i$ that correspond to elements of T, thereby obtaining $\mathbf{v}_J^{\text{sk}} \times \psi^* \bmod \mathbf{B}_I$, and then rounds to the nearest integer vector. This summation can be performed in depth (roughly) $\log|T|$, regardless of what n is. By choosing $|T|$ small enough, smaller than the depth of the circuits that the scheme can evaluate (which is unaffected by this transformation), the scheme becomes bootstrappable.

The previous paragraph oversimplifies some details. First, the summation of the $|T|$ vectors and the rounding are performed together; the fact that the ultimate result is rounded and taken modulo I allows us to maintain fewer bits of precision in the intermediate computations. The fact that we are promised that the final result is close to an integer vector (due to one of our tweaks), ensures that the rounded result is correct despite the limited precision. Also, we actually still add $|S|$ vectors together, but with the promise that only $|T|$ of them are nonzero. (We have this promise because, after when we multiply in the secret key $\text{sk} \in \{0,1\}^{|S|}$, which has Hamming weight $|T|$, it zeroizes all but $|T|$ of the ciphertext components). Why can we add $|T|$ vectors in only (roughly) $\log|T|$ depth, regardless of the size of $|S|$, when we have the promise that only $|T|$ of the $|S|$ vectors are nonzero (and the other promises, like the fact that we only need the result rounded, and then modulo I)? Essentially, the reason is that summing $|S|$ numbers basically reduces (in terms of circuit depth) to computing the Hamming weight of a vector in $\mathbf{x} \in \{0,1\}^{|S|}$ and expressing the final result in binary – i.e., in $\{0,1\}^{s+1}$ for $s = \lfloor \log|S| \rfloor$. The binary expression of the Hamming weight of $\mathbf{x}$ turns out to be simply $(e_{2^s}(x_1,\ldots,x_{|S|}) \bmod 2, e_{2^{s-1}}(x_1,\ldots,x_{|S|}) \bmod 2, \ldots, e_{2^0}(x_1,\ldots,x_{|S|}) \bmod 2)$, where e_i is the ith elementary symmetric polynomial. If the Hamming weight is guaranteed to be at most $|T|$, we need not bother computing the polynomials of degree higher than $2^{\lfloor \log|T| \rfloor}$, and consequently need less depth.

Theorem 1.5.2 (Informal)**.** *The decryption circuit of $\mathcal{E}$ with the tweaks followed by the above transformation can be expressed as a circuit of depth $c \cdot (\log|T|)^{1+o(1)}$ for some constant c. The scheme becomes bootstrappable when this value is less than* $\log\log(r_{\mathsf{Dec}}/2) -$

$\log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$.

For example, suppose $r_{\mathsf{Dec}} = 2^{n^{c'}}$ for some $c' < 1$ and $\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}} = \mathrm{poly}(n)$. In this case, the scheme becomes bootstrappable when $|T| \leq n^{(c'/c)-o(1)}$.

Devising a physical analogy for our technique for squashing the decryption circuit is rather difficult, but suppose that, in Alice's jewelry store, a key opens a glove box through a chemical reaction. To unlock a box, the employee uses the gloves to rub the key against the inner box until the box dissolves. However, the reaction is too slow; the gloves stiffen before the box dissolves. To address this situation, Alice gives the employee some accelerants, a different one for each box, that the employee can apply to the outside of box $\#i$ right before placing it inside box $\#(i+1)$. The accelerants speed up the chemical reaction between the key and the box, so that the reaction finishes before the gloves stiffen. The chemical composition of the accelerant provides some information about the chemical composition of her key, but not enough information for an employee to construct a key on his own. Notice that the employee should apply the accelerant to box $\#i$ while it is still outside of box $\#(i+1)$; to apply it while box $\#i$ is inside box $\#(i+1)$ would pointlessly waste the usability of the gloves for box $\#(i+1)$.

1.6 Security

The semantic security of our scheme against chosen plaintext attacks relies on the hardness of two problems; the first underlies the original somewhat homomorphic scheme (before the squashing), and the second arises from the addition of the secret key "hint" τ to the public key. CCA1 security for fully homomorphic encryption remains an open problem, while CCA2 security is impossible due to the extreme malleability of ciphertexts.

We prove the security of our somewhat homomorphic construction in two ways. The first way is provided for simplicity. Specifically, in Chapter 5 (and more concretely in Chapter 7), we provide a succinct reduction to a fairly natural problem that may be viewed as a decisional version of the closest vector problem (CVP) or bounded distance decoding problem (BDDP). Roughly, the problem is as follows: given an ideal lattice J and a vector $\mathbf{t}$, decide whether (1) $\mathbf{t}$ is unusually close to the lattice or (2) $\mathbf{t}$ is in a uniformly random coset of the lattice, given the promise that one of these is the case. The idea is that if $\mathbf{t}$ is in the first category, the simulator can use $\mathbf{t}$ to construct a valid ciphertext vector (which is also quite close to the lattice, but a little bit further away than $\mathbf{t}$), but if $\mathbf{t}$ is in the

second category, the ciphertext will be completely independent of the challenge plaintext; the latter case makes use of the fact that I and J are relatively prime.

This reduction, while simple, is not entirely satisfying. First, the problem is not *worst-case*, but rather *average-case*: in particular, J is generated using an algorithm $\mathsf{IdealGen}$ that is part of the scheme's KeyGen algorithm. Second, it would be preferable to base security on a *search* problem rather than a *decision* problem. Finally, although the problem seems natural, it is not as well-established as other problems over lattices.

So, beginning in Chapter 14, we describe a slightly different version of the scheme, along with a chain of security reductions that bases security on a search version of BDDP. Given access to a factoring oracle, we also base security on the worst-case shortest independent vector problem (SIVP) over ideal lattices. Since a factoring oracle can be instantiated efficiently using quantum computation, this result says that if there is an efficient algorithm that breaks the semantic security of scheme with non-negligible advantage, then there is an efficient quantum algorithm that solves ideal-lattice SIVP.

Theorem 1.6.1 (Informal)**.** *If there is an algorithm that breaks the somewhat homomorphic scheme with probability ϵ, then there is a classical algorithm that solves average-case BDDP over ideal lattices for an approximation factor $(r_{\mathsf{Dec}}/r_{\mathsf{Enc}}) \cdot \mathrm{poly}(n, \gamma_{\mathsf{Mult}}(R), 1/\epsilon)$, where the average-case distribution is the same as the distribution of ideals output by $\mathsf{KeyGen}_{\mathcal{E}}$. There is also a quantum algorithm (or a classical algorithm that uses a factoring oracle) that solves worst-case SIVP over ideal lattices for an approximation factor $(r_{\mathsf{Dec}}/r_{\mathsf{Enc}}) \cdot \mathrm{poly}(n, \gamma_{\mathsf{Mult}}(R), 1/\epsilon)$. In both cases, the ring R over which ideals are defined remains fixed.*

The introduction of τ into the public key induces a second problem that we must assume is hard, an instance of the $\mathsf{SplitKey}$ distinguishing problem: roughly, given $\mathbf{v}_J^{\mathrm{sk}}$, distinguish whether S is entirely random, or has a sparse $|T|$-member subset of vectors that sums to $\mathbf{v}_J^{\mathrm{sk}}$. We will refer to this as a sparse vector subset sum problem (SVSSP). If $|T|$ is too small, there are obvious brute force attacks on the SVSSP, along with some more sophisticated time-space tradeoffs [114, 128, 33], that take time essentially exponential in $|T|$. Also, if $|S|$ is so small that the subset sum solution is unique, then one can apply lattice reduction attacks similar to those used against low-density knapsacks [106, 105]. However, if $|T| = \omega(1)$ and $|S|$ is sufficiently large (but still polynomial in n), the brute force attacks take super-polynomial time; also, the lattice reduction attacks break down, since there will be an exponential number of subset sum solutions, and lattice reduction has trouble extracting the sparse solution from the non-sparse ones.

Interestingly, our two assumptions counterbalance each other: basically, if one adjusts the scheme's parameters to make one problem harder, the other problem becomes easier. Using a crude analysis, the breaking time for the second problem using known attacks is roughly $2^{|T|}$. (Here we ignore constants and logarithmic factors in the exponent.) Also, to enable the somewhat homomorphic ideal lattice scheme to evaluate circuits of depth $O(\log |T|)$ as needed to permit bootstrappability, we need the approximation factor for the first problem to be roughly $2^{|T|}$. Using the rule of thumb that a lattice problem for approximation factor 2^k takes time about $2^{n/k}$, the breaking time for the first problem is roughly $2^{n/|T|}$. Setting $|T| \leftarrow \sqrt{n}$ ensures that it takes time at least $2^{\sqrt{n}}$ to break either problem using known attacks. To make this breaking time truly exponential in the security parameter λ, we need the lattice dimension to be $n \approx \lambda^2$. Of course, this analysis does not apply to the somewhat homomorphic scheme, which does not use bootstrapping and relies only on the first assumption, and therefore can use lattices of smaller dimension.

Even this counterbalancing of our assumptions can be viewed through the prism of our physical analogy (Alice's jewelry store) if one squints sufficiently hard. One way that Alice's employees might try to extract a key from a box is simply by cutting through the gloves. To prevent this attack, one would like the gloves to be stiffer. On the other hand, making the gloves stiffer reduces their usability, and so we need a faster chemical reaction between keys and boxes. This forces Alice to give her employees a better accelerant, which provides more precise information about the chemical composition of her keys, and therefore makes it easier for her employees to duplicate a key chemically. By making one attack more difficult, she is forced to make the other easier.

1.7 Performance

When we run $\mathsf{Evaluate}(\mathrm{pk}, C, \Psi)$ over a circuit C and ciphertexts Ψ, the computational complexity of this algorithm is exactly the complexity of computing C *non-homomorphically* times a factor that is polynomial in the security parameter λ. The degree of this polynomial is rather high. If one wants 2^λ security against known attacks on the two problems that underlie the security of our scheme, the required computation per gate is quasi-linear in λ^6. While high, this does not seem entirely unreasonable when one considers that, to get 2^λ security against the number field sieve, one should use an RSA modulus whose bit-length is quasi-linear in λ^3, in which case a full exponentiation takes time quasi-linear in λ^6, even

when one uses fast FFT multiplication. See Chapter 12 for more details.

The story is very different if we only require super-polynomial security: in this case, n can be quasi-linear in the security parameter λ, $|T|$ can be polylogarithmic in n, S quasi-linear in n, and ciphertexts can be represented by a quasi-linear (in n) number of bits. In this case, the complexity of $\mathsf{Recrypt}$ (and hence the computation per gate) can be quasi-linear in λ^3.

Also, for relatively shallow circuits, where bootstrapping (and hence homomorphically evaluating the decryption circuit is unnecessary), the scheme is very practical: one obtains exponential security and, there is a constant c such that one can evaluate circuits of multiplicative depth $c \cdot \log \lambda$ with computation per gate that is quasi-linear in λ^{1+c}. The computation is quasi-linear in λ for constant depth circuits.

1.8 Applications

The most natural applications of fully homomorphic encryption are in the two-party setting. A simple example is making encrypted queries to search engines. To perform an encrypted search, Alice generates a public key pk for the fully homomorphic encryption scheme, and generates ciphertexts $\psi_1, \ldots, \psi_t$ that encrypt her query $\pi_1, \ldots, \pi_t$ under pk. (For example, each π_i could be a bit of her query.) Now, let the circuit C express the server's search function. The server sets $\psi_i^* \leftarrow \mathsf{Evaluate}(\mathrm{pk}, C_i, \psi_1, \ldots, \psi_t)$, where C_i is the sub-circuit of C that computes the ith bit of the output. (We note that, in practice, the evaluation of C_i^* and C_j^* may share intermediate results, in which case it would be needlessly inefficient to run independent instances of the $\mathsf{Evaluate}$ algorithm.) The server sends these ciphertexts to Alice. We know, by the correctness requirement, that $\mathsf{Decrypt}(\mathrm{sk}, \psi_i^*) = C_i(\pi_1, \ldots, \pi_t)$. These latter values constitute precisely the answer to Alice's query, which she recovers through decryption.

Another natural application is searching over encrypted data. In this scenario, Alice stores her files on a server (e.g., on the Internet), so that she can conveniently access her files without needing her own computer. However, she encrypts her files, because otherwise the server could read or leak her private data. Let bits $\pi_1, \ldots, \pi_t$ represent the files, which are encrypted in the ciphertexts $\psi_1, \ldots, \psi_t$. Suppose that later Alice wants to download all of her encrypted files that satisfy a query – e.g., all files containing the 'homomorphic' within 5 words of 'encryption', but not the word 'evoting'. She sends her query to the

server, which expresses it as a circuit C. The server sets $\psi_i^* \leftarrow \mathsf{Evaluate}(\mathrm{pk}, C_i, \psi_1, \ldots, \psi_t)$ and sends these ciphertexts to Alice. Alice decrypts them to recover $C(\pi_1, \ldots, \pi_t)$, the (bits of the) files that satisfy her query. (In this application, as in the encrypted search application, Alice needs to provide an upper bound on the number of bits that the response should have, and the server's encrypted response will be padded or truncated to that upper bound.)

Let us compare fully homomorphic encryption to a previous general solution for secure two-party computation – namely, "Yao's garbled circuit". The problem with Yao's protocol is that the communication complexity is proportional to the size of the circuit C. This makes the protocol rather unattractive in both of the scenarios discussed above (encrypted search and searching encrypted data). In the encrypted search scenario, the search engine would need to send Alice a huge garbled circuit whose size is proportional to the data being searched. In the scenario of searching on encrypted data, Alice would need to send a circuit whose size is proportional to the size of her data; if such communication complexity could be tolerated, then the server might as well just transmit all of Alice's encrypted files to her without "processing" those files at all, and let Alice figure out which files she wants. With fully homomorphic encryption, the communication complexity is much less. In particular, the communication needed, other than pk, is simply the number of bits need to express Alice's (cleartext) query and the server's (cleartext) response, each multiplied by the size of the security parameter, since each cleartext bit becomes a ciphertext. Actually, for the inputs to the circuit – e.g., Alice's query – we can do even better; the scheme's communication overhead here can be only *additive*, rather than multiplicative. Yao's protocol has the advantage of hiding the circuit, but it easy to tweak our fully homomorphic encryption scheme so that it provides unconditional circuit privacy.

Despite nearly minimal communication efficiency, our fully homomorphic encryption scheme does add a fair amount of computational overhead, so asynchronous application scenarios may be more appropriate in practice. An asynchronous example is spam filtering of encrypted emails: given an email encrypted using our scheme under Alice's public key, Alice's email server can homomorphically apply its spam filter to the email to obtain an encryption of '0' (indicating the email is not spam) or '1' (indicating that it is). Later, Alice decrypts this single ciphertext to recover a bit b, and only decrypts the rest of the email if $b = 0$.

Regarding multiparty computation, we already know that we can securely compute any

function. More specifically, one can construct efficient secure protocols for any multiparty computation in which there is an honest majority [56], assuming only the existence of trapdoor permutations. By "efficient," we do not mean that these protocols are necessarily practical. We mean only that the *communication* and *computational* complexity of the secure protocol equals the *computational* complexity of the insecure protocol times some factor that is *polynomial* in the security parameter and number of parties.

But why should the *communication* complexity of secure multiparty computation depend at all on *computational* complexity of the function being computed? Naor and Nissim [103] showed that, as one would expect, it is possible to construct a secure protocol whose communication complexity is polynomial in the security parameter and the *communication* complexity of the insecure protocol, but their method has a severe shortcoming: the computational complexity of their scheme is *exponential* (in the worst case) in the communication complexity. In eliminating one type of unwanted dependence, it introduces another.

Previous work leaves a fundamental question unanswered: can we make a protocol secure while leaving *both* the communication *and* the computational complexity unchanged, up to a factor polynomial in the security parameter? With fully homomorphic encryption, the answer is essentially 'yes.' More precisely, the answer is 'yes' if we relax the definition of communication complexity to include the bit-lengths of the output functions (which normally would not necessarily be included, since they are not communicated).

Extending our application of fully homomorphic encryption from the two-party setting to the multiparty setting is not entirely trivial, since, in the two-party setting, Bob prevented Alice from seeing any intermediate values encrypted under Alice's key simply by finishing the computation himself, and sending back the final encrypted value to Alice; in the multiparty setting, it is less clear how one prevents Alice from seeing intermediate value encrypted under her key. So, we use an approach initially proposed by Franklin and Haber [45], and further developed by Cramer, Damgard and Nielsen [35] (see also [39]) – namely, basing secure multiparty computation on threshold homomorphic encryption. The idea is simple. The parties must use some (other) scheme for secure computation to set up a public key for the fully homomorphic encryption scheme and distribute shares of the secret key; this introduces additive communication and computational overhead that is independent of the insecure protocol. After setup, they perform exactly the communications and computations that they would in the insecure protocol, except on *encrypted* data; fully homomorphic encryption ensures that, if a party was able to perform computations locally in the insecure

protocol, it is also able to in the secure protocol. Afterwards, they use some scheme for secure computation to perform threshold decryption on the encrypted outputs; again, this overhead is independent of the insecure protocol, except insofar as it depends on the bit-lengths of the function outputs. Cramer et al.'s scheme is dependent on the number of multiplication gates in the circuit because these could not be performed homomorphically. With a fully homomorphic encryption scheme, we avoid this problem, and fully realize their high-level concept of an "arithmetic black box."

To handle malicious parties, we can use Naor and Nissim's [103] transformation from a protocol for multiparty SFE with semi-honest parties to a protocol for malicious ones via a compiler that is *communication-preserving* – i.e., the transformation adds communication polynomial in the security parameter and polylogarithmic in the inputs. (The security parameter should be at least logarithmic in the size of the inputs anyway; otherwise, the work needed to break the scheme would be less than the work needed to process the inputs.) The essential ideas of this transformation come from Kilian's construction of zero-knowledge arguments [78, 79] and Arora et al.'s PCP theorem [8].

The literature mentions numerous other applications where fully homomorphic encryption would be useful. For example, Goldreich and Ostrovsky [57] consider software protection, show that any program can be converted to a pair consisting of an encrypted program and a CPU with λ bits of "shielded" memory, where λ is the security parameter, which defeats "experiments" by an adversary that might either attempt the determine the values that are stored and retrieved from memory, or try to determine the program's "access pattern" – i.e., its attempts to change the values. In their scheme, there is only a logarithmic blow-up in the computation time; however, the shielded CPU needs to be accessed for any nontrivial computation. With a fully homomorphic encryption scheme, the program and values can remain encrypted throughout the computation until the end. The shielded CPU only needs to be accessed to perform the decryption of the final output.

Goldwasser, Kalai and Rothblum [59] introduce the concept of one-time programs, in which they make minimal use of hardware to ensure that a program is used only once. Their approach is essentially to encrypt the program using Yao's garbled circuit, and have a secure device perform the decryption (a toggle bit is used to ensure that this decryption happens only once). One shortcoming of their approach is that the size of the encrypted program is proportional to the maximal running time of the program. With a fully homomorphic encryption scheme, one can construct an (encrypted) one-time program whose

size is proportional to the original program. Essentially, one simply encrypts the program using the fully homomorphic encryption scheme, and runs it homomorphically, using the device to perform the final decryption. The party running the program also needs to generate a NIZK, verifiable by the device, that proves that the final ciphertext was validly constructed by running the encrypted program P on permitted inputs; again, we can use Kilian's communication-efficient zero-knowledge arguments here [78, 79].

Ostrovsky and Skeith [109] propose the notion of public-key obfuscation – i.e., where a sort of obfuscation is achieved simply by encrypting the program; somehow, one then runs the encrypted program, and afterwards decrypts the output. With a fully homomorphic encryption scheme, running the encrypted program is straightforward. Currently, there is a lot of excitement about applications such as web services and cloud computing, where fully homomorphic encryption would permit remote computations on encrypted data with complete privacy.

We have already mentioned the notion of proxy re-encryption in Chapter 1.3. In a proxy re-encryption [19, 29, 71, 70], the idea is that Alice wants to publish a tag τ that will permit anyone to convert a ciphertext encrypted under her public key pk_A into an encryption of the same message under Bob's public key pk_B. Previous proxy re-encryption schemes have some shortcomings. They either are not unidirectional (i.e., Alice's tag can also be used to convert ciphertexts under pk_B to ciphertexts under pk_A, and Alice and Bob must cooperate to produce τ), or they are not multi-use (i.e., it is impossible to construct a sequence of tags $\tau_1, \tau_2, \ldots$ that allows anyone to convert ciphertexts under pk_A to pk_B, pk_B to pk_C, and so on indefinitely, without the ciphertexts growing in size). Recursive application of our $\mathsf{Recrypt}$ algorithm gives the first unidirectional multi-use proxy re-encryption scheme.

With fully homomorphic encryption, one can construct non-interactive zero knowledge proofs (NIZKs) of small size. For example, suppose that Alice wants to prove that $\pi_1, \ldots, \pi_t$ is a satisfying assignment of a boolean circuit C. Alice generates a public key pk for the fully homomorphic encryption scheme, the input ciphertexts $\{\psi_i \leftarrow \mathsf{Encrypt}(\mathrm{pk}, \pi_i)\}$, and the output ciphertext $\psi^* \leftarrow \mathsf{Evaluate}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t)$. The NIZK that her assignment is satisfying consists of NIZK proofs, under any NIZK scheme, that pk, $\{\psi_i\}$ and ψ^* are well-formed, where well-formedness for the ciphertexts means that each ψ_i is a valid encryption of '0' or '1', and ψ^* is a valid encryption of '1'. The verifier checks the NIZKs for well-formedness, and confirms that $\psi^* = \mathsf{Evaluate}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t)$. Intuitively, the NIZK proof works because, if the verifier believes that pk and the input ciphertexts are well-formed, then

the correctness of the encryption scheme implies that the output ciphertext can encrypt '1' only if $C(\pi_1, \ldots, \pi_t) = 1$. The size of this NIZK proof is proportional to the number of inputs to the circuit, but is otherwise independent of the size of the circuit.

For many interesting applications, we do not need the full power of our scheme; rather, a simpler, more efficient version of our scheme that evaluates circuits of logarithmic multiplicative depth suffices. For example, consider private information retrieval from an m-bit database. The querier can simply encrypt the index that it wants using $\log m$ ciphertexts. The database's response corresponds to a $(\log m)$-degree formula evaluated over these ciphertexts, which (essentially) can be computed using a $(\log \log m)$-depth circuit. We can evaluate such shallow circuits using the somewhat homomorphic scheme that we sketched in Chapter 1.4, without requiring either bootstrapping or "squashing the decryption circuit." This basic scheme compares well with the pairing-based scheme of Boneh-Goh-Nissim, which can essentially evaluate quadratic formulas; our basic scheme can also do essentially an arbitrary number of additions, but with greater multiplicative depth. In general, when the function to be evaluated is highly parallel, the bootstrapping step may be unnecessary, permitting better efficiency.

Clearly, several of these applications relate to obfuscation, but the precise relationship between fully homomorphic encryption and obfuscation is unclear. We know that general obfuscation is impossible under a certain definition of obfuscation [12], but obfuscation may be possible under a weaker, but still meaningful, definition. We also know that general obfuscation (under essentially any reasonable definition) would imply fully homomorphic encryption: it would suffice to obfuscate circuits that take ciphertexts encrypting π_1 and π_2 and output appropriately distributed ciphertexts encrypting $\pi_1 + \pi_2$ and $\pi_1 \times \pi_2$. Since general obfuscation would imply fully homomorphic encryption, it seems reasonable to guess that a general obfuscation technique (if one exists) would employ some of the techniques (bootstrapping, etc.) that we use here to construct fully homomorphic encryption. Unlike a fully homomorphic encryption scheme, however, an obfuscated circuit should allow one to compute an unencrypted output. If one is to build a general obfuscation scheme from fully homomorphic encryption, the question becomes: how can one provide, as part of the obfuscated circuit, some sort of decryption key that allows recovery of the final output, in such a way that this decryption key does not permit decryption of interior nodes of the circuit, thereby unraveling the entire obfuscation.

Chapter 2

Definitions related to Homomorphic Encryption

2.1 Basic Definitions

A conventional public-key encryption scheme $\mathcal{E}$ consists of three algorithms: $\mathsf{KeyGen}_{\mathcal{E}}$, $\mathsf{Encrypt}_{\mathcal{E}}$, and $\mathsf{Decrypt}_{\mathcal{E}}$. $\mathsf{KeyGen}_{\mathcal{E}}$ is a randomized algorithm that takes a security parameter λ as input, and outputs a secret key sk and public key pk; pk defines a plaintext space $\mathcal{P}$ and ciphertext space $\mathcal{C}$. $\mathsf{Encrypt}_{\mathcal{E}}$ is a randomized algorithm that takes pk and a plaintext $\pi \in \mathcal{P}$ as input, and outputs a ciphertext $\psi \in \mathcal{C}$. $\mathsf{Decrypt}_{\mathcal{E}}$ takes sk and ψ as input, and outputs the plaintext π. The computational complexity of all of these algorithms must be polynomial in λ. *Correctness* is defined as follows: if $(\mathrm{sk}, \mathrm{pk}) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}}$, $\pi \in \mathcal{P}$, and $\psi \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi)$, then $\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi) \to \pi$.

In addition to the three conventional algorithms, a *homomorphic encryption scheme* $\mathcal{E}$ has a (possibly randomized) efficient algorithm $\mathsf{Evaluate}_{\mathcal{E}}$, which takes as input the public key pk, a circuit C from a permitted set $\mathcal{C}_{\mathcal{E}}$ of circuits, and a tuple of ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$ for the input wires of C; it outputs a ciphertext $\psi \in \mathcal{C}$. Informally, the functionality that we want from $\mathsf{Evaluate}_{\mathcal{E}}$ is that, if ψ_i "encrypts π_i" under pk, then $\psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \Psi)$ "encrypts $C(\pi_1, \ldots, \pi_t)$" under pk, where $C(\pi_1, \ldots, \pi_t)$ is the output of C on inputs $\pi_1, \ldots, \pi_t$.

There are different ways of formalizing the functionality "encrypts $C(\pi_1, \ldots, \pi_t)$." A minimal requirement is *correctness*.

Definition 2.1.1 (Correctness of Homomorphic Encryption)**.** We say that a homomorphic encryption scheme $\mathcal{E}$ is correct for circuits in $\mathcal{C}_{\mathcal{E}}$ if, for any key-pair $(\mathrm{sk}, \mathrm{pk})$ output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, any circuit $C \in \mathcal{C}_{\mathcal{E}}$, any plaintexts $\pi_1, \ldots, \pi_t$, and any ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$ with $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_i)$, it is the case that:

$$\text{if } \ \psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \Psi) \ , \quad \text{then } \ \mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi) \rightarrow C(\pi_1, \ldots, \pi_t)$$

except with negligible probability over the random coins in $\mathsf{Evaluate}_{\mathcal{E}}$.

By itself, mere correctness fails to exclude trivial schemes. In particular, suppose we define $\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \Psi)$ to just output (C, Ψ) without "processing" the circuit or ciphertexts at all, and $\mathsf{Decrypt}_{\mathcal{E}}$ to decrypt the component ciphertexts and apply C to results. This scheme is correct, but uninteresting. We can address this shortcoming by upper-bounding the length of ciphertexts output by $\mathsf{Evaluate}_{\mathcal{E}}$. One way to do this is by placing an upper bound on the size of the decryption circuit $D_{\mathcal{E}}$ for the scheme $\mathcal{E}$ that depends only on the security parameter, as in the following definition.

Definition 2.1.2 (Compact Homomorphic Encryption)**.** We say that a homomorphic encryption scheme $\mathcal{E}$ is compact if there is a polynomial f such that, for every value of the security parameter λ, $\mathcal{E}$'s decryption algorithm can be expressed as a circuit $D_{\mathcal{E}}$ of size at most $f(\lambda)$.

Definition 2.1.3 ("Compactly Evaluates")**.** We say that a homomorphic encryption scheme $\mathcal{E}$ "compactly evaluates" circuits in $\mathcal{C}_{\mathcal{E}}$ if $\mathcal{E}$ is compact and also correct for circuits in $\mathcal{C}_{\mathcal{E}}$.

We can consider various relaxations of compactness, since homomorphic encryption schemes in which the ciphertext size grows sub-linearly with the size of the circuit are still interesting for many applications. For example, we could permit the sizes of the secret key and ciphertexts to grow polynomially with the depth of the circuit. We will informally call such schemes "quasi-compact."

Now, we define *fully homomorphic encryption* as follows.

Definition 2.1.4 (Fully Homomorphic Encryption)**.** We say that a homomorphic encryption scheme $\mathcal{E}$ is fully homomorphic if it compactly evaluates all circuits.

One may consider this definition to be too strong, because, as mentioned above, quasi-compactness could suffice; we avoid using quasi-compactness in our definition both because

it is tedious to formalize, and we will rarely use the notion anyway. A second reason that it is too strong is because it excludes leveled schemes, which only evaluate circuits of depth up to some d, and whose public key length may be $\text{poly}(d)$; hence, the following relaxation.

Definition 2.1.5 (Leveled Fully Homomorphic Encryption)**.** We say that a family of homomorphic encryption schemes $\{\mathcal{E}^{(d)} : d \in \mathbb{Z}^+\}$ is leveled fully homomorphic if, for all $d \in \mathbb{Z}^+$, they all use the same decryption circuit, $\mathcal{E}^{(d)}$ compactly evaluates all circuits of depth at most d (that use some specified set of gates), and the computational complexity of $\mathcal{E}^{(d)}$'s algorithms is polynomial in λ, d, and (in the case of $\mathsf{Evaluate}_{\mathcal{E}}$) the size of the circuit C.

(We assume the set of gates that compose the circuit is understood.)

While fully homomorphic encryption, as we have defined it, seems highly nontrivial to achieve, one still might consider our definition to be too weak, since it does not require *circuit privacy*.

Definition 2.1.6 ((Statistical) Circuit Private Homomorphic Encryption)**.** We say that a homomorphic encryption scheme $\mathcal{E}$ is circuit-private for circuits in $\mathcal{C}_{\mathcal{E}}$ if, for any key-pair (sk, pk) output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, any circuit $C \in \mathcal{C}_{\mathcal{E}}$, and any fixed ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$ that are in the image of $\mathsf{Encrypt}_{\mathcal{E}}$ for plaintexts $\pi_1, \ldots, \pi_t$, the following distributions (over the random coins in $\mathsf{Encrypt}_{\mathcal{E}}$, $\mathsf{Evaluate}_{\mathcal{E}}$) are (statistically) indistinguishable:

$$\mathsf{Encrypt}_{\mathcal{E}}(\text{pk}, C(\pi_1, \ldots, \pi_t)) \quad \approx \quad \mathsf{Evaluate}_{\mathcal{E}}(\text{pk}, C, \Psi)$$

The obvious correctness condition must still hold.

We prefer to consider circuit privacy to be a property that is separate from, and complementary to, full homomorphism. However, we will eventually show how to make our lattice-based fully homomorphic encryption scheme circuit private. Our technique will be to use a public (i.e., not using the secret key) algorithm $\mathsf{RandomizeCT}_{\mathcal{E}}$ that, applied *post hoc*, *induces* the same distribution (statistically) to ciphertexts output by $\mathsf{Encrypt}_{\mathcal{E}}$ and $\mathsf{Evaluate}_{\mathcal{E}}$, while preserving correctness. (See Chapter 20.)

The motivating setting for statistical circuit privacy is two-party computation in the honest-but-curious setting, where Alice holds a circuit, and Bob holds sk. Alice may want her output ciphertext to reveal nothing about her circuit, even though Bob chooses the input ciphertexts. She can hide her circuit by applying $\mathsf{RandomizeCT}_{\mathcal{E}}$ to the ciphertext output by $\mathsf{Evaluate}_{\mathcal{E}}$ before sending result to Bob. When sk is shared, one may also define

a computational version of circuit privacy, but this is covered by the semantic security of encryption scheme, defined in the next Section.

For most applications, it is acceptable to reveal some limited information about the circuit, such as an upper bound on the number of levels. (Since any circuit is a directed acyclic graph, its gates can be topologically sorted and partitioned into *levels*, such that each wire extends from one gate to a gate with a higher level number.) Accordingly, we define the following slight relaxation of circuit privacy.

Definition 2.1.7 (Leveled Circuit Private Homomorphic Encryption)**.** Like circuit private homomorphic encryption, except that there can be a different distribution associated to each level, and the distributions only need to be equivalent if they are associated to the same level (in the circuit).

Unlike circuit privacy, leveled circuit privacy, by itself, does not imply compactness. That is, in a leveled circuit private homomorphic encryption scheme, it is possible for the ciphertext size to grow exponentially with the number of levels. In fact, this is precisely the case in some previous circuit-private schemes, such as SYY [122].

An interesting open question is the extent to which fully homomorphic encryption, as we have defined it, already implies circuit-private fully homomorphic encryption. Intuitively, given a ciphertext ψ that encrypts π, we can "randomize" ψ using the homomorphism – e.g., by repeatedly adding encryptions of '0' – to obtain new encryptions of π. Since the fully homomorphic encryption scheme is compact, this randomization occurs within a non-expanding ciphertext space. One may hope that these randomizations induce a nice, connected, expander-like graph, and that therefore a small number of randomizations results in a statistically random encryption of π. However, the definition of fully homomorphic encryption does not seem even to imply that this graph is connected. It would be nice to state some natural minimal generic property, in addition to full homomorphism, that would imply circuit privacy. (Certainly, the property that adding an encryption of '0,' or multiplying in an encryption of '1,' completely "randomizes" the ciphertext would be sufficient, but in this case circuit privacy is an uninteresting tautology.)

In the definitions above, we have focused on circuits, but one may also consider *programs* that use other *representation models*. For example, one may consider weaker models – e.g., formulas, branching programs, OBDDs, finite automata, decision trees, and truth tables – and consider the efficiency of a homomorphic encryption scheme with respect to one of these

models. For example, although an encryption scheme that is additively homomorphic will not be able to evaluate general circuits efficiently, such a scheme can be used to construct a single-server private information retrieval (PIR) scheme with sub-linear communication; such a PIR scheme, in turn, can be viewed as homomorphic encryption scheme that permits the (efficient) evaluation of a truth table with an output ciphertext that is sub-linear in the size of the table. Ishai and Paskin [73] describe a scheme in which Evaluate takes a branching program (BP) P as input; finite automata, decision trees, and OBDDs can be efficiently represented as BPs. The ciphertext output by their Evaluate algorithm depends polynomially on the number of input ciphertexts and on the depth of the BP, but not on its size. On the other hand, since a program may allow loops, it may permit a more compact representation of the circuit.

2.2 Computational Security Definitions

For an ordinary public key encryption scheme, security against adaptive chosen-ciphertext attacks (CCA2) is captured in the following game.

Setup. The challenger runs $(\mathrm{sk}, \mathrm{pk}) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}}(\lambda)$ and gives pk to the adversary $\mathcal{A}$. It sets $\psi^* \leftarrow \perp$.

Queries. $\mathcal{A}$ issues decryption queries on ciphertexts $\psi_i \neq \psi^*$. The challenger responds with the output of $\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi_i)$. Queries can occur before or after the challenge.

Challenge. $\mathcal{A}$ generates two plaintexts $\pi_0^*, \pi_1^* \in \mathcal{P}$ and sends these to the challenger. The challenger sets $b \xleftarrow{\mathrm{R}} \{0,1\}$ and $\psi^* \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_b^*)$. It sends ψ^* to $\mathcal{A}$.

Guess. $\mathcal{A}$ sends $b' \in \{0,1\}$ to the challenger. $\mathcal{A}$ wins the game if $b' = b$.

Security against "lunchtime attacks" – i.e., CCA1 security – is modeled by a game similar to above, except that $\mathcal{A}$ may make queries only before the challenge. In the game for semantic security, $\mathcal{A}$ is not permitted to make any queries.

In each case, we define $\mathcal{A}$'s advantage in attacking the scheme $\mathcal{E}$ as

$$\mathsf{Adv}(\mathcal{A}, \mathcal{E}, \lambda) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

The probability is over the random bits used by the challenger and the adversary.

Definition 2.2.1 (Semantic Security against (CPA, CCA1, CCA2) attacks)**.** We say $\mathcal{E}$ is semantically secure against (CPA, CCA1, CCA2) attacks if no polynomial time (CPA, CCA1, CCA2)-adversary $\mathcal{A}$ breaks $\mathcal{E}$ with advantage non-negligible in the security parameter λ.

When referring simply to "semantic security," we mean semantic security against chosen plaintext attacks.

We define the CCA2, CCA1, and semantic security games for a *homomorphic* encryption scheme as being identical to the original games, except that now the ciphertext space is potentially larger – i.e., the support of $\mathsf{Evaluate}_{\mathcal{E}}$ rather the support of $\mathsf{Encrypt}_{\mathcal{E}}$; $\mathcal{A}$ can draw its queries from this larger space. Also, $\mathcal{A}$ has more freedom in requesting its challenge. The natural way to define the $\mathsf{Challenge}$ phase is that $\mathcal{A}$ sends the challenger some circuit $C \in \mathcal{C}_{\mathcal{E}}$ with some number $k = \mathrm{poly}(\lambda)$ of inputs, and two sets of plaintexts, $(\pi_{01}, \ldots, \pi_{0k}), (\pi_{11}, \ldots, \pi_{1k}) \in \mathcal{P}^k$; the challenger sets $b \xleftarrow{\mathrm{R}} \{0,1\}$ and outputs $\psi^* \xleftarrow{\mathrm{R}} \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \psi_{b1}, \ldots, \psi_{bk})$ where $\psi_{bi} \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_{bi})$. However, since the adversary can run $\mathsf{Evaluate}_{\mathcal{E}}$ itself, we can simplify the $\mathsf{Challenge}$ phase by having the adversary just request the input ciphertexts $\psi_{b1}, \ldots, \psi_{bk}$.

Clearly, the only difference between the semantic security games for ordinary public key encryption and homomorphic encryption is that, in the latter, the adversary can request more ciphertexts in the $\mathsf{Challenge}$ phase. By a hybrid argument [16], an algorithm $\mathcal{A}$ that that breaks the semantic security in the game above with advantage ϵ can be used to construct an algorithm $\mathcal{B}$ that breaks the semantic security in the original game with advantage ϵ/k; $\mathcal{B}$'s is roughly k times that of $\mathcal{A}$ [52]. Thus, to prove semantic security of a homomorphic encryption scheme, we can just use the semantic security game for ordinary public key encryption.

The same is true for CCA1 and CCA2 security, as long as the scheme is circuit-private. (Circuit privacy ensures that the ciphertext space is the same in both games, thus allowing $\mathcal{B}$ to forward $\mathcal{A}$'s decryption queries to the challenger, and forward the responses back to $\mathcal{A}$.)

Unfortunately, a scheme that has nontrivial homomorphisms cannot be CCA2 secure, because it is malleable. Benign malleability [6] and replayable CCA [30], two relaxed notions of CCA2 security, permit only transformations that preserve the underlying plaintext. Prabhakaran and Rosulek [115] formalize a notion of "homomorphic-CCA security," which permits certain nontrivial operations on a plaintext while remaining nonmalleable with respect to other operations; they present a construction based on pairings. However, their

approach does not extend (in fact, they provide some impossibility results) to schemes that permit certain operations on multiple ciphertexts. Finding meaningful relaxations of CCA2 security in this domain, and particularly for fully homomorphic encryption, is an open area.

There do not seem to be inherent reasons why a homomorphic encryption scheme cannot have semantic or CCA1 security. In particular, "Cramer-Shoup lite" [36] is CCA1 and homomorphic (for one operation). However, we restrict our focus to semantic security, and leave finding a CCA1-secure fully homomorphic encryption scheme as an interesting open problem.

Chapter 3

Previous Homomorphic Encryption Schemes

Basic RSA [121] was the first homomorphic encryption scheme: given ciphertexts $\psi_1 = \pi_1^e \bmod N$ and $\psi_2 = \pi_2^e \bmod N$, one can compute a ciphertext $\psi \leftarrow \psi_1 \cdot \psi_2 = (\pi_1 \cdot \pi_2)^e \bmod N$ that encrypts the product of the original plaintexts. However, basic RSA is deterministic, and therefore not even semantically secure. Despite the lack of semantic security, RSA's multiplicative homomorphism is still useful for many applications.

Rivest, Adleman, and Dertouzos [120] were the first to explore the possibilities of fully homomorphic encryption, which they called a "privacy homomorphism", and they proposed several candidate schemes. However, these early candidates have been broken [25].

Homomorphic encryption schemes that are not semantically secure, like textbook RSA and some proposals in [120], may also have stronger attacks on their one-wayness. In particular, Boneh and Lipton [23] proved that any algebraic privacy homomorphism over a ring $\mathbb{Z}_n$ can be broken in sub-exponential time under a (reasonable) number theoretic assumption, if the scheme is deterministic or offers an equality oracle. See also [92]. In the quantum setting, the situation is even worse. van Dam, Hallgen and Ip [37] proved that, with quantum computation, any deterministic algebraic privacy homomorphism with an equality oracle can be broken in *polynomial* time.

The first scheme with a proof of semantic security based on a well-defined assumption

was proposed by Goldwasser-Micali [61] in the paper that introduced the notion of semantic security. Some other additively homomorphic encryption schemes with proofs of semantically security are Benaloh [17], Naccache-Stern [102], Okamoto-Uchiyama [108], Paillier [110], and Damgard-Jurik [38]. ElGamal [42] is multiplicatively homomorphic. Some semantically secure schemes that allow both addition and multiplication include Boneh-Goh-Nissim [21], which permits computation of quadratic formulas (e.g., 2-DNFs) over ciphertexts, and "Polly Cracker" by Fellows and Koblitz [44], which permits computation of arbitrary circuits over ciphertexts, but where the ciphertext size blows up exponentially with the depth of the circuit. For expository purposes, and since one can easily find other surveys of homomorphic encryption, we characterize these "conventional" homomorphic encryption schemes (although perhaps Polly Cracker is less conventional) as all falling within a certain abstract framework, with security abstractly based on an *ideal membership problem*. We will review these schemes in more detail momentarily. This description will help highlight how our construction is fundamentally different, abstractly relying on an *ideal coset problem* that we define in Chapter 7.

It is also known that one can construct additively homomorphic encryption schemes from lattices or linear codes [60, 113, 77, 94, 95, 7]. The lattice-based scheme [95] and the Reed-Solomon-code-based scheme by Armknecht and Sadeghi [7] also allow multiplications, though with exponential expansion in ciphertext size. Such schemes have a different flavor from the more "conventional" schemes above, because ciphertexts implicitly contain an "error" that grows as ciphertexts are added together. Thus, ciphertexts output by Evaluate do not have the same distribution as ciphertexts output by Encrypt, and at some point the error may become large enough to cause incorrect decryption. For this reason, the homomorphism is sometimes referred to as a "pseudohomomorphism" [77, 94, 95] or a "bounded homomorphism" [113]. (We use different terminology; see Chapter 2.) We will not discuss these schemes in detail here, since the main technical complication – managing the size of the "error" – is also central to our scheme, where it will require an even closer analysis because our multiplicative homomorphism using ideal lattices expands the "error" quite rapidly.

van Dijk [40] describes a technique that they call "interval obfuscation" which can be viewed as a symmetric homomorphic encryption scheme. It uses a secret integer modulus M and a secret integer s that is relatively prime to M. A '0' is encrypted as $s \cdot x \bmod M$ for some $x \in [1, a]$, where a is a "small" integer, while a '1' is encrypted as $s \cdot x \bmod M$ for some

$x \in [b+1, b+a]$, where b is a "large" integer (but still small in comparison to M). One can cryptocompute a homogeneous polynomial of degree d logarithmic in the security parameter by simply adding or multiplying the ciphertexts modulo M. The recipient decrypts c by setting $c' \leftarrow c/s^d \bmod M$ (to remove the blinding factor) and then outputting $\lfloor c'/b^d \rceil$; the idea is that each monomial which is a product of 1's will be represented by some integer that approximately equals b^d after the blinding factor is removed, while the monomials for which the product is 0 will be represented by much smaller integers that can be ignored. One can view their scheme as using a one-dimensional ideal lattice – namely, the ideal (M) in the integers – while our somewhat homomorphic construction in Chapter 7 is conceptually somewhat similar but uses an n-dimensional ideal lattice. At a high level, the reason M must be kept private in their scheme (while we can reveal a basis for the lattice in our scheme) is that lattice problems over one-dimensional lattices are not hard. An initial version of van Dijk's scheme succumbed to attacks that used lattice reduction to recover M. It is an open question as to whether the security of a variant of van Dijk's scheme can be based on a natural hard problem.

Finally, there are schemes that use a singly homomorphic encryption scheme to construct a scheme that can perform more complicated homomorphic operations [122, 73]. Sanders, Young and Yung (SYY) [122] show that one can use a circuit-private additively homomorphic encryption scheme to construct a circuit-private scheme that can handle arbitrary circuits, where the ciphertext size increases exponentially with the depth of the circuit. Their scheme can therefore feasibly evaluate NC1 circuits.

Ishai and Paskin [73] show how to evaluate *branching programs*, and with much smaller ciphertexts than SYY. In their scheme Evaluate outputs a ciphertext whose length is proportional to the *length* of the branching program. This remains true even if the *size* of the branching program is very large – e.g., super-polynomial. However, the computational complexity of their scheme is proportional to the *size*; Barrington [13] tells us that bounded-width polynomial-size branching programs recognize exactly those languages in NC1.

In more detail, Ishai and Paskin use a "leveled" approach to evaluate a branching program, like we will use a leveled approach to evaluate circuits (see Chapter 4), though the details are very different. A (deterministic) *branching program* (BP) P is defined by a DAG from a distinguished initial node in which each nonterminal node has two outgoing edges labeled 0 and 1, and where the terminal nodes also have labels. To compute $P(x)$ where the binary representation of x is $x_1 \cdots x_\ell$, one starts at the distinguished node, and

traverses the DAG in the natural way dictated by $x_1 \cdots x_\ell$ to reach a terminal node, and outputs that node's label as $P(x)$. The *size* of the BP is the number of nodes; the *length* is the length of the longest path. One can topologically arrange the nodes into levels, such that the number of levels is at most one more than the length of the BP, and the edges are all directed downward. BPs are relatively powerful; finite automata, decision trees, and ordered binary decision diagrams all have polynomial-size BPs.

To evaluate a BP, Ishai and Paskin essentially use 1-out-of-2 string OT recursively. Specifically, suppose Alice has a BP with ℓ levels, and Bob has an input $x = x_1 \cdots x_\ell \in \{0,1\}^\ell$ for which he wants to obtain $P(x)$. Bob constructs ℓ 1-out-of-2 string OT queries q_i, which respectively correspond to his bits x_i. Using Bob's queries, Alice evaluates her BP from the bottom-up. In particular, suppose N is a node at level $\ell - 1$ with children $N0$ and $N1$ with labels L_0 and L_1. Alice uses q_ℓ, L_0 and L_1 to construct a string-OT response R that implicitly "encrypts" label L_{x_ℓ}; she then sets R to be the label of N. In this fashion, she gives labels to all of the nodes at level $\ell - 1$, and then (recursively) to nodes at higher levels using Bob's other OT queries. Alice's ultimate response is the label associated to the distinguished node. This final label looks something like a multiple encryption in onion routing, and Bob "decrypts" it as such – using his secret knowledge to recover the label for x_1, then $x_1 x_2$, and so on. The length of Alice's response grows (at least) linearly with ℓ for essentially the same reason that this happens in onion-routing: each layer of "encryption" has additive communication overhead. Using a communication-efficient string-OT scheme – e.g., one built from the length-flexible additively homomorphic Damgard-Jurik encryption scheme [38, 84] – the ciphertext expansion per level is exactly linear. On the other hand, Alice's computation is proportional to the size of the BP, since she must construct OT responses even for "irrelevant" nodes in the BP.

To summarize to current state of affairs as far we know, in terms of schemes that offer more than a single homomorphism and offer a proof of semantic security, we have the schemes by Fellow and Koblitz [44], Melchor et al. [95], Armknecht and Sadeghi [7], and Sanders et al. [122], and related work [14, 83, 85, 86] where ciphertext size grows exponentially with the multiplicative (and sometimes also additive) depth of the circuit. In Boneh-Goh-Nissim [21] and Ishai-Paskin [73], Evaluate outputs small ciphertexts but handles a limited class of circuits – quadratic formulas, or circuits which correspond to branching programs of manageable size.

Now, we review the more "conventional" homomorphic encryption schemes whose semantic security can be based on a natural problem, like Goldwasser-Micali and Paillier. Since our scheme will rely heavily on properties of algebraic *rings* and *ideals*, we explain how these previous schemes implicitly use these objects. By describing previous schemes using these abstractions, we will see how the semantic security of most of these schemes relies on the hardness of an *ideal membership problem* – i.e., determining whether a member of the ring is also a member of the ideal.

Basically, a *ring* is a mathematical object like a field, except that not every element has a multiplicative inverse. Examples include the integers $\mathbb{Z}$, or the integers modulo a composite integer N: $\mathbb{Z}/N\mathbb{Z}$. Rings have an additive identity '0', a multiplicative identity '1', allow additive inverses, and are closed under addition and multiplication. An *ideal* I of a ring R is a subset of R that is closed under addition, and is also closed under multiplication *with elements of* R. An example is the ideal (2) of $\mathbb{Z}$, the set of even numbers; multiplying an element of (2) with an element of $\mathbb{Z}$ gives an element in (2). For ideal $I \subset R$, R/I is the ring of *cosets* of I in R; e.g., if $R = \mathbb{Z}$ and $I = (2)$, R/I consists of the cosets $0 + (2)$ (the even integers, the additive identity of R/I) and $1 + (2)$ (the odd integers, the multiplicative identity of R/I).

With these abstractions, we can say that many previous homomorphic encryption schemes fall within the following framework. (Essentially, this abstract framework is explicit in Fellows' and Koblitz's description of Polly Cracker [44].)

$\mathsf{KeyGen}(\lambda)$. Generates some representation of a finite ring R with an efficient '+' operation, and possibly an efficient '$\times$' operation. It also fixes an ideal I of R. The plaintext space $\mathcal{P}$ is a set of distinguished representatives of R/I. The secret key is a function $f : R \to \mathcal{P}$ such that $f(r)$ is the distinguished representative of $r + I$. The public key pk includes the encoding of R and an algorithm Samp_I to sample (efficiently) from I.

$\mathsf{Encrypt}(\mathrm{pk}, \pi)$. Set $i \xleftarrow{\mathrm{R}} \mathsf{Samp}_I(R)$ and $\psi \leftarrow \pi + i$.

$\mathsf{Decrypt}(\mathrm{sk}, \psi)$. Output $f(\psi)$.

$\mathsf{Add}(\psi_1, \psi_2)$. Output $\psi_1 + \psi_2$.

$\mathsf{Mult}(\psi_1, \psi_2)$. Output $\psi_1 \times \psi_2$.

For example, in Goldwasser-Micali, KeyGen generates a modulus $N = pq$ for $p = 2p' + 1$ and $q = 2q' + 1$, and a number $x \in (\mathbb{Z}/N\mathbb{Z})^*$ whose Legendre symbols are $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$. In terms of the abstract framework, the underlying ring R is $\mathbb{Z}/(2p'q')$, which corresponds to

the powers of x modulo N. The underlying ideal I is (2), the set of quadratic residues, even powers of x. The plaintext space is $\{0,1\}$, represented as $\{x^0, x^1\}$. The function $f : R \to \mathcal{P}$ on input $r \in R$ (i.e., x^r) is given by outputting the distinguished representative of $r + (2)$. Sampling from I is efficient. Also, the '+' operation is efficient, though the '$\times$' operation is not; hence, the scheme is only additively homomorphic.

Remark 3.0.2. The abstract framework hides some issues regarding how plaintexts are represented. For example, as applied to Goldwasser-Micali, the framework would say plaintext space is $\{x^0, x^1\}$, versus the usual $\{0,1\}$. For Goldwasser-Micali, this is fine since the encrypter can easily map the latter representation to the former. This is the case with the other schemes as well.

Remark 3.0.3. Of course, a lot of complexity is hidden in the function f. Goldwasser-Micali uses Legendre symbols. Paillier uses a more elaborate approach. Some schemes, such as Boneh-Goh-Nissim (described below) can use only a small (polynomial-sized) subset of the potential plaintext space because the function f involves an otherwise infeasible computation – e.g., discrete logarithm.

It is easy to see that the abstract scheme is semantically secure assuming the following *ideal membership problem* is hard.

Definition 3.0.4 (Ideal Membership Problem (IMP))**.** According to a prescribed distribution, the challenger generates an encoding of R, an ideal I, and an algorithm Samp_I that samples from I. It sets a bit $b \xleftarrow{R} \{0,1\}$. If $b = 0$, it sets $x \xleftarrow{R} \mathsf{Samp}_I(R)$. If $b = 1$, it sets $x \xleftarrow{R} R$. The problem is to guess b given (x, R, Samp_I) – i.e., essentially to decide whether or not x is a member of I.

Theorem 3.0.5. *If there is an algorithm $\mathcal{A}$ that breaks the semantically security of the abstract scheme with advantage ϵ, then there is an algorithm $\mathcal{B}$ that solves the IMP with advantage $\epsilon/2$.*

Proof. Given instance (x, R, Samp_I) of IMP, $\mathcal{B}$ includes (R, Samp_I) in pk, which it sends to $\mathcal{A}$. $\mathcal{A}$ requests a challenge ciphertext on one of $\pi_0, \pi_1 \in \mathcal{P}$. $\mathcal{B}$ sets $\beta \xleftarrow{R} \{0,1\}$, and sends the challenge $\psi \leftarrow \pi_\beta + x$ to $\mathcal{A}$. $\mathcal{A}$ sends guess β', and $\mathcal{B}$ sends guess $b' \leftarrow \beta \oplus \beta'$ to the challenger.

If $b = 0$, then $\mathcal{B}$'s simulation is perfect; in particular, the challenge is a valid encryption of π_β. In this case, $\mathcal{A}$ should guess β with advantage ϵ, and thus $b' = b$ with advantage ϵ.

If $b = 1$, x is random in R, and thus the challenge ciphertext is a random element of R, independent of β. In this case, β' is independent of β, and so b' is independent of b, so that $\mathcal{B}$'s advantage is 0. Overall, $\mathcal{B}$'s advantage is $\epsilon/2$. □

Obviously, Goldwasser-Micali uses quadratic residuosity as its version of the IMP. Benaloh is similar to Goldwasser-Micali, but uses ideals of the form (m) for $m \neq 2$ where m divides $\phi(N)$. In Paillier, the ring is $\mathbb{Z}/(p'q'N)$, the ideal is (N), and it is based on the N-th residuosity problem. Damgard-Jurik extends Paillier to the ring is $\mathbb{Z}/(p'q'N^k)$ and uses the ideal is (N^k). Okamoto-Uchiyama uses a modulus of the form $N = p^2q$, and uses the ring $\mathbb{Z}/(pp'q')$ and the ideal $\mathbb{Z}/(p)$.

The above schemes can all be said to be based on a subgroup (or subset) membership problem [**?**], since only one operation (namely addition, which is instantiated as group multiplication) is actually being used. Two schemes that make more use of the ring structure are Polly Cracker [44] and Boneh-Goh-Nissim (BGN) [21].

The Polly Cracker scheme was proposed by Fellows and Koblitz [44]. They state essentially the abstract framework above and propose an instantiation of it using the polynomial ring $R = \mathbb{F}_q[x_1, \ldots, x_n]$. The ideal I is presented as a set of generating polynomials $\mathcal{P} = \{p_i(x_1, \ldots, x_n)\}$ having a common (secret) root $(a_1, \ldots, a_n)$; the ideal I is the set of all polynomials of the form $\sum p_i(\mathbf{x}) \cdot r_i(\mathbf{x})$ for $r_i(\mathbf{x}) \in R$. To sample from I, one uses the generators, though there is plenty of freedom here in setting the sampling distribution since R and I are infinite. The plaintext space is $\mathbb{F}_q$. The abstract function f is instantiated as evaluation of the ciphertext polynomial at $(a_1, \ldots, a_n)$, a homomorphism whose kernel contains I.

The security of Polly Cracker in practice still seems to be an open question. Various efficient attacks have been proposed for various sets of parameters [43, 46] – roughly speaking, parameters for which the underlying IMP is not hard because it is possible to recover the common root using Groebner bases. Modified versions of Polly Cracker have been proposed [83, 85, 86], also with attacks [126]. But there does not seem to be an efficient general attack. See [82] for a survey of Polly Cracker cryptanalysis.

Ciphertext expansion in Polly Cracker is a serious problem. Add simply adds two ciphertext polynomials, and Mult multiplies them. In the worst-case, Mult operations are extremely expensive: the ciphertext length grows doubly-exponentially in the multiplicative depth of the circuit, since each Mult operation can square the number of monomials. Even with respect to the additive depth, the ciphertext size can grow exponentially. It is

certainly conceivable that some incarnation of Polly Cracker could escape this deficiency and still be secure, but so far no such scheme is known.

BGN is a practical scheme that permits homomorphic evaluation of quadratic formulas – i.e., it allows one level of multiplication and an arbitrary number of additions. It is an interesting case because it uses multiple different representations of its underlying ring R. Specifically, KeyGen generates a modulus $N = pq$, two groups $\mathbb{G}, \mathbb{G}_1$ of order N with an efficiently computable non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ (where typically $\mathbb{G}$ is an elliptic curve group and $\mathbb{G}_1$ is a multiplicative subgroup of a finite field), a generator g of $\mathbb{G}$, and an element $h = g^p$. In terms of the abstract framework, the underlying ring R is $\mathbb{Z}/(N)$, which is represented in the public key both by $(\mathbb{G}, g)$ and implicitly $(\mathbb{G}_1, e(g, g))$; we will call these the $\mathbb{G}$-representation and the $\mathbb{G}_1$-representation. The ideal I is (p), the p-residues; it can be sampled efficiently using h. Essentially, the underlying IMP is, given the representations of R, the map e, and the generator of I, to decide whether an element $x \in R$, given in $\mathbb{G}$-representation, is in I. (The BGN paper states its underlying hard problem in a different way, without the generator of I, that is equivalent up to a factor of 2 in the adversary's advantage.)

Adding two ciphertexts in BGN is done in the usual way, but the Mult operation is more interesting. Mult uses the pairing operation, meaning that it can only be applied to two ciphertexts in $\mathbb{G}$-representation, and the output has a $\mathbb{G}_1$-representation: i.e., for $\psi_1 \in \pi_1 + (p)$ and $\psi_2 \in \pi_2 + (p)$, $\mathsf{Mult}(\psi_1, \psi_2) = \psi_1 \times \psi_2 \in \pi_1 \times \pi_2 + (p)$, but the latter ciphertext represents the ring element differently (in $\mathbb{G}_1$-representation); concretely, this multiplication in the exponent occurs by computing $e(g^x, g^x) \to e(g, g)^{xy}$. Since there is no known way to efficiently map from the $\mathbb{G}_1$-representation back to the $\mathbb{G}$-representation, the scheme is limited to one level of multiplication.

To decrypt a ciphertext $g^{\pi+tp}$ in $\mathbb{G}$ in BGN, the decrypter computes $(g^{\pi+tp})^q = g^{\pi q}$, and then $\mathrm{DL}_{g^q}(g^{\pi q}) = \pi$; it decrypts ciphertexts in $\mathbb{G}_1$ similarly. For the discrete logarithm computation to be feasible, π must be from a set of polynomial size – say, a polynomial-sized interval centered at 0. However, subject this constraint on the input ciphertexts $\{\psi_i\}$ from $\mathbb{G}$, the scheme can homomorphically compute arbitrary polynomial-sized quadratic formulas on $\{\psi_i\}$, and still be able to decrypt the result in polynomial time.

In principle, one can also squeeze ElGamal into the above framework. One can view R as $\mathrm{GL}(2, \mathbb{F}_p)$, the general linear group of 2×2 matrices over $\mathbb{F}_p$, and an ideal $I_b \subset R$ as the subset of matrices whose second row is b times the first column. Basically, I_b corresponds

to the set of valid DDH tuples (g, g^a, g^b, g^{ab}) involving b. We can define addition in R as simply adding the matrices together entry-wise; I_b is closed under addition. This operation is efficient even if matrix is represented "in the exponent," as in ElGamal, permitting the additive homomorphishm. Multiplication in R is right-multiplication; one can see that right-multiplying a term in I_b with a term in R gives a term in I_b. However, obviously right-multiplication cannot be efficient if the Diffie-Hellman problem is hard.

Strictly speaking, however, since none of these schemes aside from Polly Cracker actually makes full use of the ring homomorphism, their dependence on an IMP may be more coincidental than essential. For example, one can modify BGN in a way that preserves the ability to evaluate quadratic formulas, while dispensing with the need to use a composite modulus N, and without using an ideal membership problem; instead, it is based on a "rank" problem similar to the linear assumption. On the other hand, this modification would become exponentially inefficient if extended to handle n-degree polynomials over ciphertexts with a hypothetical n-linear map; for this more robust homomorphism, it would seem more efficient to use BGN's original ideal-membership approach.

Chapter 4

Bootstrappable Encryption

4.1 Leveled Fully Homomorphic Encryption from Bootstrappable Encryption, Generically

Assume we have an encryption scheme $\mathcal{E}$ that compactly evaluates some set of circuits $\mathcal{C}_{\mathcal{E}}$. We want to use $\mathcal{E}$ to construct a homomorphic encryption scheme that can handle arbitrary circuits. In this Chapter we prove a fundamental result: that if $\mathcal{C}_{\mathcal{E}}$ contains (slight augmentations of) $\mathcal{E}$'s *own decryption circuit* $D_{\mathcal{E}}$ – i.e., if $\mathcal{E}$ "compactly evaluates" its (augmented) decryption circuit – then we can use $\mathcal{E}$ to construct an efficient scheme that handles circuits of arbitrary depth.

A bit more specifically, for any integer d, we use $\mathcal{E}$ to construct a scheme $\mathcal{E}^{(d)}$ that can compactly evaluate circuits of depth up to d. The decryption circuit for $\mathcal{E}^{(d)}$ is still $D_{\mathcal{E}}$; the secret key and ciphertexts are the same size as in $\mathcal{E}$. The public key in $\mathcal{E}^{(d)}$ consists of $d+1$ public keys from $\mathcal{E}$, together with a chain of encrypted $\mathcal{E}$ secret keys – the first $\mathcal{E}$ secret key encrypted under the second $\mathcal{E}$ public key, and so on. In short, the family of schemes $\{\mathcal{E}^{(d)}\}$ is leveled fully homomorphic. We base the semantic security of $\mathcal{E}^{(d)}$ on that of $\mathcal{E}$ using a hybrid argument; as usual with hybrid arguments, the reduction loses a factor linear in d. In Chapter 4.3, we describe how one can obtain a fully homomorphic encryption scheme (where the public key size does not depend on the maximum number of levels we want to evaluate) by assuming key-dependent-message (KDM) security, specifically *circular-security* – i.e., that one can safely encrypt a $\mathcal{E}$ secret key under its associated public key.

Since this critical property of $\mathcal{E}$ – that it can compactly evaluate (slight augmentations of) its own decryption circuit – is self-referential and universal, we give it the obvious name: bootstrappability. Why should bootstrappability be such a powerful feature? At a high level, the reason is that bootstrappability allows us periodically to "refresh" ciphertexts associated to interior nodes in a circuit; we can refresh for an arbitrary number of levels in the circuit, and thus can evaluate circuits of arbitrary depth. To "refresh" a ciphertext that encrypts a plaintext π under $\mathcal{E}$ public key pk_i, we *re-encrypt* it under pk_{i+1} and then *homomorphically apply the decryption circuit* to the result, using the secret key sk_i that is encrypted under pk_{i+1}, thereby obtaining an encryption of π under pk_{i+1}. Homomorphically evaluating the decryption circuit decrypts the *inner* ciphertext under pk_i, but within homomorphic encryption under pk_{i+1}. The implicit decryption "refreshes" the ciphertext, but the plaintext is never revealed; the plaintext is always covered by at least one layer of encryption. Now that the ciphertext is refreshed, we can "continue" correctly evaluating the circuit.

To see how this works mathematically, begin by considering the following algorithm, called $\mathsf{Recrypt}$. For simplicity, suppose the plaintext space $\mathcal{P}$ is $\{0,1\}$ and $D_{\mathcal{E}}$ is a boolean circuit in $\mathcal{C}_{\mathcal{E}}$. Let $(\mathrm{sk}_1, \mathrm{pk}_1)$ and $(\mathrm{sk}_2, \mathrm{pk}_2)$ be two $\mathcal{E}$ key-pairs. Let ψ_1 be an encryption of $\pi \in \mathcal{P}$ under pk_1. Let $\overline{\mathrm{sk}_{1j}}$ be an encryption of the j-th bit of the *first secret key* sk_1 under the *second public key* pk_2. $\mathsf{Recrypt}$ takes as these things as input, and outputs an encryption of π under pk_2.

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

Set $\overline{\psi_{1j}} \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}_2, \psi_{1j})$ where ψ_{1j} is the j-th bit of ψ_1

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

Output ψ_2

Above, the $\mathsf{Evaluate}$ algorithm takes in all of the bits of sk_1 and all of the bits of ψ_1, each encrypted under pk_2. Then, $\mathcal{E}$ is used to evaluate the decryption circuit homomorphically. The output ψ_2 is thus an encryption under pk_2 of $\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}_1, \psi_1) \to \pi$.

Remark 4.1.1. The $\mathsf{Recrypt}$ algorithm implies a proxy one-way re-encryption scheme [19]. Roughly speaking, a one-way proxy re-encryption scheme allows the owner of sk_1 to generate a tag that enables an untrusted proxy to convert an encryption of π under pk_1 to an encryption of π under pk_2, but not the reverse. In our case, the tag is $\langle\overline{\mathrm{sk}_{1j}}\rangle$, the encrypted

secret key. Strictly speaking, the security model for proxy re-encryption typically requires the security of the delegator's secret key, even against a collusion of delegatee's who also get to see the delegating tags. However, this requirement seems unnecessary, since a delegatee will be able to decrypt ciphertexts directed to the delegator anyway.

In the $\mathsf{Recrypt}$ algorithm above, the plaintext π is doubly encrypted at one point – under both pk_1 and pk_2. Depending on the encryption scheme $\mathcal{E}$, however, this double encryption might be overkill. Suppose $\mathsf{WeakEncrypt}_{\mathcal{E}}$ is an algorithm such that the image of $\mathsf{WeakEncrypt}_{\mathcal{E}}(\mathrm{pk}, \pi)$ is always a subset of the image of $\mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi)$. Then we can replace the first step of $\mathsf{Recrypt}_{\mathcal{E}}$ with:

$$\text{Set } \overline{\psi_{1j}} \xleftarrow{\mathrm{R}} \mathsf{WeakEncrypt}_{\mathcal{E}}(\mathrm{pk}_2, \psi_{1j}) \quad \text{where} \quad \psi_{1j} \text{ is the } j\text{-th bit of } \psi_1$$

Let us call this relaxation $\mathsf{Recrypt}'_{\mathcal{E}}$. The main point of this relaxation is that $\mathsf{WeakEncrypt}$ does not need to be semantically secure for $\mathsf{Recrypt}'_{\mathcal{E}}$ to be a secure one-way proxy re-encryption scheme, or for $\mathsf{Recrypt}'_{\mathcal{E}}$ to be useful toward bootstrapping (as we will see below). Thus, depending on $\mathcal{E}$, $\mathsf{WeakEncrypt}_{\mathcal{E}}$ can be very simple – e.g., for some schemes, and in particular for the ideal-lattice-based scheme that we describe later, $\mathsf{WeakEncrypt}_{\mathcal{E}}$ might leave the input "bits" entirely unmodified. This will unfortunately not help us much in terms of making the encryption scheme bootstrappable; essentially, it will add one circuit level to what $\mathcal{E}$ can evaluate. However, it will affect the eventual *computational complexity* of our scheme, since it will require less computation to apply the decryption circuit homomorphically to ciphertexts in which the outer encryption is weak. Another way of viewing this relaxation is that we only need to be able to evaluate non-uniform decryption circuits, where the ciphertext is "hard-wired" into the circuit (making this circuit simpler than the "normal" decryption circuit that takes the ciphertext (and secret key) as input.

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

Definition 4.1.2 (Augmented Decryption Circuit)**.** Let $D_{\mathcal{E}}$ be $\mathcal{E}$'s decryption circuit, which takes a secret key and ciphertext as input, each formatted as an element of $\mathcal{P}^{\ell(\lambda)}$, where $\mathcal{P}$ is the plaintext space. Let Γ be a set of gates with inputs and output in $\mathcal{P}$, which includes the trivial gate (input and output are the same). We call a circuit composed of multiple copies

of $D_{\mathcal{E}}$ connected by a single g gate (the number of copies equals the number of inputs to g) a "g-augmented decryption circuit." We denote the set of g-augmented decryption circuits, $g \in \Gamma$, by $D_{\mathcal{E}}(\Gamma)$.

Definition 4.1.3 (Bootstrappable Encryption Scheme)**.** As before, let $\mathcal{C}_{\mathcal{E}}$ denote the set of circuits that $\mathcal{E}$ can compactly evaluate. We say that $\mathcal{E}$ is *bootstrappable* with respect to Γ if

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

For example, if Γ includes the trivial gate and NAND, $\mathcal{E}$ is bootstrappable with respect to Γ if $\mathcal{C}_{\mathcal{E}}$ contains $D_{\mathcal{E}}$ and the circuit formed by joining two copies of $D_{\mathcal{E}}$ with a NAND gate. Remarkably, as we will show, if there is a scheme $\mathcal{E}$ that can compactly evaluate only these two circuits, then there is a scheme that is leveled fully homomorphic.

Remark 4.1.4. We could relax the bootstrappability definition slightly to say that $\mathcal{E}$ only needs to be able to homomorphically evaluate its (augmented) decryption circuit when the input ciphertext is weakly encrypted, similar to the relaxation $\mathsf{Recrypt}'_{\mathcal{E}}$ above. But, this makes the definition of bootstrappable more cumbersome; we will continue with the definition above, and remind the reader occasionally that the relaxation can be used.

From the informal description above, it should already be somewhat clear how to use a bootstrappable encryption scheme to construct a leveled fully homomorphic one; below, we give a more formal description. Let $\mathcal{E}$ be bootstrappable with respect to a set of gates Γ. For any integer $d \geq 1$, we use $\mathcal{E}$ to construct a scheme $\mathcal{E}^{(d)} = (\mathsf{KeyGen}_{\mathcal{E}^{(d)}}, \mathsf{Encrypt}_{\mathcal{E}^{(d)}}, \mathsf{Evaluate}_{\mathcal{E}^{(d)}}, \mathsf{Decrypt}_{\mathcal{E}^{(d)}})$ that can handle all circuits of depth d with gates in Γ. Note that in the description below we encrypt the secret keys in reverse order; the only reason is that this ordering simplifies our description of the recursion in $\mathsf{Evaluate}$. When we refer to the level of a wire in C, we mean the level of the gate for which the wire is an input. We use the notation $D_{\mathcal{E}}(\Gamma, \delta)$ to refer to the set of circuits that equal a δ-depth circuit with gates in Γ augmented by $D_{\mathcal{E}}$ (copies of $D_{\mathcal{E}}$ become inputs to the δ-depth circuit).

$\mathsf{KeyGen}_{\mathcal{E}^{(d)}}(\lambda, d)$. Takes as input a security parameter λ and a positive integer d. For $\ell = \ell(\lambda)$ as in Definition 4.1.2, it sets

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

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

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

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

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

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

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

$\mathsf{Reduce}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C^\dagger_\delta, \Psi^\dagger_\delta)$. Takes as input a public key $\mathrm{pk}^{(\delta)}$, a tuple of input ciphertexts $\Psi^\dagger_\delta$ (where each input ciphertext should be in the image of $\mathsf{Encrypt}_{\mathcal{E}^{(\delta)}}$), and a circuit $C^\dagger_\delta \in D_{\mathcal{E}}(\Gamma, \delta + 1)$. It sets C_δ to be the sub-circuit of $C^\dagger_\delta$ consisting of the first δ levels. It sets Ψ_δ to be the induced input ciphertexts of C_δ, where the ciphertext $\psi^{(w)}_\delta$ associated to wire w at level δ is set to $\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}_\delta, C^{(w)}_\delta, \Psi^{(w)}_\delta)$, where $C^{(w)}_\delta$ is the sub-circuit of $C^\dagger_\delta$ with output wire w, and $\Psi^{(w)}_\delta$ are the input ciphertexts for $C^{(w)}_\delta$. It outputs (C_δ, Ψ_δ).

High-level review of the $\mathsf{Evaluate}$ **algorithm.** We are given a circuit C_d of, say, d levels with gates in Γ. For each input wire w of C_d, there is an associated input ciphertext ψ_w

encrypted under pk_d. We are also given an encryption scheme $\mathcal{E}$ that compactly evaluates circuits in $D_{\mathcal{E}}(\Gamma)$.

Note that we have not assumed that $\mathcal{E}$ can evaluate gates in Γ; we have only assumed it can evaluate gates in Γ that are augmented by the decryption circuit. So, our first step is to augment C_d by placing copies of $D_{\mathcal{E}}$ at the leaves of C_d (as in $\mathsf{Augment}$), thereby constructing $C^{\dagger}_{d-1}$. Now, what are the input ciphertexts for our new circuit $C^{\dagger}_{d-1}$?

Reconsider the algorithm $\mathsf{Recrypt}'_{\mathcal{E}}$. In $\mathsf{Recrypt}'_{\mathcal{E}}$, we begin with a ciphertext ψ_1 encrypting π under pk_1 for the single wire w, and an "empty" circuit C_1 (since $\mathsf{Recrypt}'_{\mathcal{E}}$ doesn't actually evaluate any gates, it just generates a new encryption of the same plaintext). Our next step was to augment C_1 with the decryption circuit $D_{\mathcal{E}}$ to obtain $C_2 \leftarrow D_{\mathcal{E}}$. The input ciphertexts Ψ_2 to C_2 included the encrypted secret key bits, and the weakly encrypted bits of ψ_1. We then showed that the ciphertext generated by $\psi_2 \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}_2, C_2, \Psi_2)$, which is also associated to wire w, also encrypts π, but now under pk_2.

In the full scheme above, the ciphertexts that we associate to the decryption circuit that was attached to wire w are analogous to the ones we used in $\mathsf{Recrypt}'_{\mathcal{E}}$: the encrypted secret key (sk_d under pk_{d-1}), and the re-encryption ciphertexts of ψ_w under pk_{d-1}. By the correctness of $\mathsf{Recrypt}$, the ciphertext *now* associated to w (after performing $\mathsf{Evaluate}_{\mathcal{E}}$) should encrypt the same plaintext as ψ_w, but now under pk_{d-1}.

The Reduce step simply performs this $\mathsf{Evaluate}$ up to the wire w, and one level beyond. We know that $\mathsf{Evaluate}$ can correctly continue one level beyond the wire w, because (by assumption) $\mathcal{E}$ can evaluate not just the decryption circuit attached to w, but can evaluate a circuit containing one Γ-gate above w. Reduce outputs C_{d-1} and ciphertexts associated to C_{d-1}'s input wires. We have made progress, since C_{d-1} is one level shallower than C_d. We perform this entire process $d-1$ more times to obtain the final output ciphertexts.

Remark 4.1.5. Previously, we said that $\mathsf{Evaluate}$ takes as input ciphertexts that are "fresh" outputs of $\mathsf{Encrypt}$. However, we note $\mathsf{Evaluate}_{\mathcal{E}^{(\delta)}}$ also operates correctly on ciphertexts output by $\mathsf{Evaluate}$. (For $\delta < d$ above, this is precisely what $\mathsf{Evaluate}_{\mathcal{E}^{(\delta)}}$ does.)

4.2 Correctness, Computational Complexity and Security of the Generic Construction

Here we state and prove some theorems regarding the generic construction. Regarding correctness, we have the following theorem.

Theorem 4.2.1. *Let $\mathcal{E}$ be bootstrappable with respect to a set of gates Γ. Then $\mathcal{E}^{(d)}$ compactly evaluates all circuits of depth d with gates in Γ – i.e., if Γ is a universal set of gates, the family $\mathcal{E}^{(d)}$ is leveled fully homomorphic.*

Proof. (Theorem 4.2.1) First, we define a convenient notation: let $D(\delta, w, C, \Psi)$ denote the plaintext value for wire w in circuit C induced by the decryptions (under sk_δ) of the ciphertexts Ψ associated to C's input wires. If C is empty (has no gates), then the input wires are the same as the output wires, and $D(\delta, w, C, \Psi)$ just denotes the decryption of the single ciphertext $\psi \in \Psi$ associated to w. To prove correctness, it suffices to show that

$$D(d, w_{out}, C_d, \Psi_d) = D(0, w_{out}, C_0, \Psi_0) \tag{4.1}$$

for every output wire w_{out} of C_0 (at level 0).

First, when $(C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1}) \leftarrow \mathsf{Augment}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C_\delta, \Psi_\delta)$, we claim that $D(\delta, w, C_\delta, \Psi_\delta) = D(\delta-1, w, C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1})$ for any wire w at level at most $\delta-1$. This follows from the correctness of $\mathsf{Recrypt}$ (generalized beyond a boolean plaintext space and boolean circuits), and the fact that circuits C_δ and $C^\dagger_{\delta-1}$ are identical up to level $\delta - 1$.

Next, when $(C_\delta, \Psi_\delta) \leftarrow \mathsf{Reduce}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C^\dagger_\delta, \Psi^\dagger_\delta)$, we have $D(\delta, w, C^\dagger_\delta, \Psi^\dagger_\delta) = D(\delta, w, C_\delta, \Psi_\delta)$ for any wire at level at most δ. This follows from the correctness of $\mathsf{Evaluate}_{\mathcal{E}}$ over circuits in $D_{\mathcal{E}}(\Gamma)$, and the fact that circuits $C^\dagger_\delta$ and C_δ are identical up to level δ.

From these two claims, Equation 4.1 follows.

□

Note that Γ is arbitrary. For example, each gate in Γ could be a circuit of (ANDs, ORs, NOTs) of depth m and fan-in 2; for this value of Γ, Theorem 4.2.1 implies the scheme correctly evaluates boolean circuits up to depth $d \cdot m$.

We need to check that the computational complexity of $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$ is reasonable – e.g., that recursive applications of $\mathsf{Augment}$ do not increase the effective circuit size exponentially.

Theorem 4.2.2. *For a circuit C of depth at most d and size s (defined here as the number of wires), the computational complexity of applying $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$ to C is dominated by at most $s \cdot \ell \cdot d$ applications of $\mathsf{WeakEncrypt}_{\mathcal{E}}$ and at most $s \cdot d$ applications of $\mathsf{Evaluate}_{\mathcal{E}}$ to circuits in $D_{\mathcal{E}}(\Gamma)$, where ℓ is as in Definition 4.1.2.*

Proof. (Theorem 4.2.2) There is a pre-processing step to ensure that all wires in the circuit connect gates at consecutive levels; clearly, this step increases the number of wires in the

circuit by at most a multiplicative factor of d. It remains to prove that, for the pre-processed circuit, the computational complexity is dominated by at most $s' \cdot \ell$ applications of $\mathsf{Encrypt}$ and at most s' applications of $\mathsf{Evaluate}_{\mathcal{E}}$ to circuits in $D_{\mathcal{E}}(\Gamma)$, where s' is the size of the pre-processed circuit.

The complexity of $\mathsf{Augment}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C_\delta, \Psi_\delta)$ is dominated by replacing each ciphertext $\psi \in \Psi_\delta$ by the ciphertexts $\langle\langle\overline{\mathrm{sk}_{\delta j}}\rangle, \langle\overline{\psi_j}\rangle\rangle$; generating the $\langle\overline{\psi_j}\rangle$'s involves $|W_\delta| \cdot \ell$ applications of $\mathsf{WeakEncrypt}_{\mathcal{E}}$, where W_δ is the set of wires at level δ. Summing over all δ, there are at most $s' \cdot \ell$ applications of $\mathsf{WeakEncrypt}_{\mathcal{E}}$.

The complexity of $\mathsf{Reduce}_{\mathcal{E}^{(\delta)}}(\mathrm{pk}^{(\delta)}, C^\dagger_\delta, \Psi^\dagger_\delta)$ is dominated by the evaluation of $C^{(w)}_\delta$ for each $w \in W_\delta$, which involves $|W_\delta|$ applications of $\mathsf{Evaluate}_{\mathcal{E}}$ to circuits in $D_{\mathcal{E}}(\Gamma)$. Summing over all δ, there are at most s' such applications. The theorem follows.

□

Finally, assuming the semantic security of $\mathcal{E}$, we prove the semantic security of $\mathcal{E}^{(d)}$.

Theorem 4.2.3. *Let $\mathcal{A}$ be an algorithm that (t, ϵ)-breaks the semantic security of $\mathcal{E}^{(d)}$. Then, there is an algorithm $\mathcal{B}$ that (t', ϵ')-breaks the semantic security of $\mathcal{E}$ for $t' \approx t \cdot \ell$ and $\epsilon' \geq \epsilon/\ell(d+1)$, for ℓ as in Definition 4.1.2.*

Proof. (Theorem 4.2.3) Let $(\mathcal{E})^\ell$ be equivalent to $\mathcal{E}$, but with plaintext space $\mathcal{P}^{\leq \ell}$, where $\mathsf{Encrypt}_{(\mathcal{E})^\ell}$ involves up to ℓ invocations of $\mathcal{E}$ and a concatenation of the results. We use a hybrid argument to show that $\mathcal{B}$ (t'', ϵ'')-breaks the semantic security of $(\mathcal{E})^\ell$ for $t'' \approx t$ and $\epsilon'' \geq \epsilon/(d+1)$, from which the result follows.

For $k \in [0, d]$, let Game k denote a game against $\mathcal{E}^{(d)}$ in which everything is exactly as in the real-world game, except that for all $i \in [1, k]$ the challenger sets

$$(\mathrm{sk}'_i, \mathrm{pk}'_i) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}}(\lambda) \quad \text{and} \quad \overline{\mathrm{sk}_{ij}} \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}_{i-1}, \mathrm{sk}'_{ij})$$

In other words, for $i \in [1, k]$, $\overline{\mathrm{sk}_{ij}}$ is the encryption (under pk_{i-1}) of the j-th bit of a *random* secret key sk'_i unrelated to sk_i. Game $d+1$ is identical to Game d, except that the challenger ignores b and (π_0, π_1), generates a random plaintext π of the appropriate length, and encrypts π to construct the challenge ciphertext. Let ϵ_k denote the adversary's advantage in Game k.

Since Game 0 is identical to the real world attack, the adversary's advantage is ϵ by assumption. Also, $\epsilon_{d+1} = 0$, since the challenge is independent of b. Consequently, for some

$k \in [0, d]$, it must hold that $|\epsilon_k - \epsilon_{k+1}| \geq \epsilon/(d+1)$; fix this value of k.

$\mathcal{B}$ uses $\mathcal{A}$ to break $(\mathcal{E})^\ell$ as follows. $\mathcal{B}$ receives from the challenger a public key pk. $\mathcal{B}$ generates the secret and public values exactly as in Game k, except that it replaces its original value of pk_k with pk. Also, if $k < d$, it generates a dummy key pair $(\mathrm{sk}'_{k+1}, \mathrm{pk}'_{k+1}) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, sets $\pi_0 \leftarrow \mathrm{sk}_{k+1}$ and $\pi_1 \leftarrow \mathrm{sk}'_{k+1}$, and requests a challenge ciphertext (under pk) encrypting either $\pi_0, \pi_1 \in \mathcal{P}^\ell$. The challenger generates $\beta \xleftarrow{\mathrm{R}} \{0,1\}$ and sends a tuple of ciphertexts $\langle \psi_j \rangle$ encrypting the bits $\langle \pi_{\beta j} \rangle$. $\mathcal{B}$ replaces its original tuple $\langle \overline{\mathrm{sk}_{(k+1)j}} \rangle$ with the tuple $\langle \psi_j \rangle$. One can verify that the public values are generated exactly as in Game $k + \beta$. $\mathcal{B}$ sends the public values to $\mathcal{A}$.

Eventually, $\mathcal{A}$ requests a challenge ciphertext on π_0 or π_1. $\mathcal{B}$ sets $b \xleftarrow{\mathrm{R}} \{0,1\}$. If $k < d$, $\mathcal{B}$ sends the values $\psi_j \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}_d, \pi_{bj})$. If $k = d$, $\mathcal{B}$ generates random $\pi \xleftarrow{\mathrm{R}} \mathcal{P}$ and asks the challenger for a challenge ciphertext on π_b or π. The challenger generates $\beta \xleftarrow{\mathrm{R}} \{0,1\}$ and encrypts π_b or π accordingly, and $\mathcal{B}$ forwards the challenge to $\mathcal{A}$. $\mathcal{A}$ sends a bit b'. $\mathcal{B}$ sends bit $\beta' \leftarrow b \oplus b'$ to the challenger. One can verify that the challenge is generated as in Game $k + \beta$.

Since $\mathcal{B}$'s simulation has the same distribution as Game $k + \beta$, and the probability that $\mathcal{B}$ outputs 0 is $\epsilon_{k+\beta}$. The result follows.

□

4.3 Fully Homomorphic Encryption from KDM-Secure Bootstrappable Encryption

The length of the public key in $\mathcal{E}^{(d)}$ is proportional to d (the depth of the circuits that can be evaluated). It would be preferable to have a construction $\mathcal{E}^*$ where the public key size is completely independent of the circuit depth, a construction that is fully homomorphic rather than merely leveled fully homomorphic. Here is the obvious way to make the public key pk^* of $\mathcal{E}^*$ short: for $\mathcal{E}$ key pair $(\mathrm{sk}, \mathrm{pk})$, pk^* includes only pk and (the "bits" of) sk encrypted under pk. In other words, we have a *cycle* (in fact, a *self-loop* in this example) of encrypted secret keys rather than an acyclic chain. It is clear that $\mathcal{E}^*$ is correct: the recursive algorithm $\mathsf{Evaluate}_{\mathcal{E}^*}$ works as before, except that the implicit recryptions generate "refreshed" ciphertexts under the same public key.

Why didn't we present this construction in the first place? Using an *acyclic* chain of encrypted secret keys allowed us to base the security of $\mathcal{E}^{(d)}$ on $\mathcal{E}$ using a hybrid argument;

this hybrid argument breaks down when there is a cycle. In general, a semantically secure encryption scheme is not guaranteed to be *KDM-secure* – i.e., secure when the adversary can request the encryptions of *key-dependent messages*, such as the secret key itself. Typically, when we prove an encryption scheme semantically secure, there is not an obvious *attack* when the adversary is given the encryption of a key-dependent message. However, KDM-security is highly nontrivial to prove. The problem is precisely that the usual hybrid argument breaks down.

Remark 4.3.1. Canetti [27] proposed the acyclic, leveled approach as a way to remove the need for KDM-security. Our initial approach had actually been to use $\mathcal{E}^*$ (with the self-loop), and assume, or try to prove, KDM-security.

Let us review (a restriction of) the definition of KDM-security. We will say a scheme $\mathcal{E}$ is KDM-secure if all polynomial-time adversaries $\mathcal{A}$ have negligible advantage in the following KDM-security game.

KDM-Security Game.

$\mathsf{Setup}(\lambda, n)$. The challenger sets $(\mathrm{sk}_i, \mathrm{pk}_i) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}(\lambda)$ for $i \in [0, n-1]$ for integer $n = \mathrm{poly}(\lambda)$. It chooses a random bit $b \xleftarrow{\mathrm{R}} \{0,1\}$. If $b = 0$, then for $i \in [0, n-1]$ and $j \in [1, \ell]$, it sets $\overline{\mathrm{sk}_{ij}} \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}_{(i-1) \bmod n}, \mathrm{sk}_{ij})$, where sk_{ij} is the jth "bit" of sk_i. If $b = 1$, it generates the $\overline{\mathrm{sk}_{ij}}$ values as encryptions of random secret keys, unrelated to $\mathrm{pk}_0, \ldots, \mathrm{pk}_{n-1}$. It sends the public keys and encrypted secret keys to $\mathcal{A}$.

Challenge and Guess. Basically as in the semantic security game.

This definition of KDM-security is a restriction of the general setting [18, 68, 22], where $\mathcal{A}$ can select multiple functions f, and request the encryption of $f(\mathrm{sk}_0, \ldots, \mathrm{sk}_{n-1})$. However, when $\mathcal{E}$ is a bootstrappable encryption scheme, $\mathcal{A}$ can use the cycle of encrypted secret keys in our game to generate the encryption of $f(\mathrm{sk}_0, \ldots, \mathrm{sk}_{n-1})$ under any pk_i, as long as f can be computed in polynomial time. Hence, we only need to consider our restricted setting [65]. We have the following theorem.

Theorem 4.3.2. *Suppose $\mathcal{E}$ is KDM-secure and also bootstrappable with respect to a universal set of gates Γ. Then, $\mathcal{E}^*$, obtained from $\mathcal{E}$ as described above (with the self-loop), is semantically secure (and fully homomorphic).*

The theorem is a straightforward consequence of the fact that, from *any* loop of public keys and encrypted secret keys that includes $(\mathrm{pk}_0, \mathrm{sk}_0)$, one can compute an encryption of

sk_0 under pk_0. There does not seem to be any advantage in having pk^* contain any cycle of encrypted secret keys other than a self-loop.

Absent proof of KDM-security in the plain model, one way to obtain fully homomorphic encryption from bootstrappable encryption is simply to *assume* that the underlying bootstrappable encryption scheme is also KDM-secure. This assumption, though unsatisfying, does not seem completely outlandish. While an encrypted secret key is very useful in a bootstrappable encryption scheme – indeed, one may view this as the essence of bootstrappability – we do not see any actual attack on a bootstrappable encryption scheme that provides a self-encrypted key.

4.4 Fully Homomorphic Encryption from Bootstrappable Encryption in the Random Oracle Model

Above, we constructed a fully homomorphic encryption $\mathcal{E}^*$ from a bootstrappable encryption scheme $\mathcal{E}$ basically by adding a "self-loop" – a $\mathcal{E}$ secret key sk encrypted under its corresponding public key pk – to the $\mathcal{E}^*$ public key pk^*. We showed that $\mathcal{E}^*$ should inherit the semantic security of $\mathcal{E}$, *under the assumption* that $\mathcal{E}$ is KDM-secure – in particular, under the assumption that it is "safe" to reveal a direct encryption of a secret key under its own public key (as opposed to some possibly-less-revealing non-identity function of the secret key). Can we provide any evidence that $\mathcal{E}^*$ is semantically secure without this assumption?

Here we provide some evidence in the random oracle model. First, given a *leveled* fully homomorphic scheme $\mathcal{E}^{(d)}$ and a hash function, we define an intermediate scheme $\mathcal{E}^{(d)\dagger}$. $\mathcal{E}^{(d)\dagger}$ is the same as $\mathcal{E}^{(d)}$, except for the following. The public key includes a hash function $H : \mathcal{P}^{\ell'} \to \mathcal{P}^{\ell}$. Also, in KeyGen, one generates $r \xleftarrow{\mathrm{R}} \mathcal{P}^{\ell'}$, sets $\overline{r_j} \xleftarrow{\mathrm{R}} \mathsf{Encrypt}_{\mathcal{E}^{(d)}}(\mathrm{pk}^{(d)}, r_j)$ for $j \in [1, \ell']$, sets $\sigma \leftarrow H(r) \star \mathrm{sk}_0$, and includes $(\langle \overline{r_j} \rangle, \sigma)$ in the public key. (Assume $\star$ is some invertible operation such that $a \star b$ would completely hide $b \in \mathcal{P}^{\ell}$ if $a \in \mathcal{P}^{\ell}$ were a one-time pad.) In other words, the $\mathcal{E}^{(d)\dagger}$ public key includes some additional information: an encryption of the the secret key sk_0, where the encryption uses a hash function that will be treated as a random oracle in the security analysis.

Next, we prove the following theorems.

Theorem 4.4.1. *If $\mathcal{E}^{(d)}$ is semantically secure, then $\mathcal{E}^{(d)\dagger}$ is semantically secure in the random oracle model.*

Theorem 4.4.2. *Suppose $\mathcal{E}$ is leveled circuit-private (in addition to being bootstrappable) and let $\mathcal{E}^{(d)\dagger}$ and $\mathcal{E}^*$ be constructed from $\mathcal{E}$ as described above. Then, if $\mathcal{E}^{(d)\dagger}$ is semantically secure (in the plain model), and the circuit required to compute the hash function H and invert the $\star$ operation is at most d levels, then $\mathcal{E}^*$ is semantically secure.*

The result here should be quite surprising. The scheme $\mathcal{E}^*$ does not even contain a hash function, and yet we are basically claiming that it is secure in the random oracle model! This is the first instance that we are aware of where one scheme is proven secure in the random oracle model, and then a second scheme's security is based on the first scheme, even though the second scheme *does not use a hash function.*

How is this possible? First, let us consider Theorem 4.4.1. This theorem basically just states the previously known result [18] that it is easy to construct a KDM-secure encryption scheme in the random oracle model. This is because the random oracle allows the reduction to construct a "fake" ciphertext purportedly encrypting the secret key, such that the adversary finds out that it was fake only after it has queried the random oracle; this query gives the reduction all of the information that it needs to solve the underlying problem. In our particular case, $\mathcal{E}^{(d)\dagger}$ has a loop among $(\mathrm{sk}_0, \mathrm{pk}_0), \ldots, (\mathrm{sk}_d, \mathrm{pk}_d)$, because $\mathcal{E}^{(d)}$ reveals direct encryptions of sk_i under pk_{i-1} for $i \in [1, d]$, and $\mathcal{E}^{(d)\dagger}$ also reveals an *indirect* encryption $(\langle \overline{r_j} \rangle, \sigma)$ of sk_0 under pk_d ("indirect," because encryption in $\mathcal{E}$ does not normally use a hash function). However, the reduction algorithm in the proof of Theorem 4.4.1 will construct σ simply as a random string – i.e., it does not actually need to know anything about sk_0.

Theorem 4.4.2 is perhaps the more surprising result. But the result is actually a simple consequence of the fact that: given a correctly constructed $\mathcal{E}^{(d)\dagger}$ public key, the reduction algorithm can generate an $\mathcal{E}$-encryption of sk_0 under pk_0, as needed for the $\mathcal{E}^*$ public key. How do we generate the latter ciphertext? The reduction algorithm is given $\langle \overline{r_j} \rangle$, an encryption of r under pk_d. It simply uses the leveled homomorphism and the circuit corresponding to the hash function H to compute a ciphertext that encrypts $H(r)$ from the ciphertext that encrypts r. Then, given that ciphertext and the value of $\sigma = H(r) \star \mathrm{sk}_0$, it computes a ciphertext that encrypts sk_0 *in the natural way* – i.e., directly, rather than with the hash function. We assumed that the hash function H and the $\star$ operation can be computed with a circuit of depth at most d; therefore, our leveled homomorphic scheme $\mathcal{E}^{(d)}$ has enough levels to evaluate this circuit. Consequently, we obtain a "natural" encryption of sk_0 (i.e., under $\mathcal{E}$) under some public key pk_i for $i \geq 0$, and we can use $\mathsf{Recrypt}$ operations

to obtain a natural encryption of sk_0 under pk_0. This ciphertext is an output of $\mathsf{Evaluate}_{\mathcal{E}}$, but circuit privacy guarantees that the ciphertext is distributed as if it were output directly by $\mathsf{Encrypt}_{\mathcal{E}}$.

Remark 4.4.3. Although one can view $(\langle \overline{r_j} \rangle, \sigma)$ as an "encryption" of sk_0, this "encryption" function is not the usual encryption function and it might have a very complex decryption circuit, much more complex than $D_{\mathcal{E}}$. In particular, we cannot assume that its decryption circuit is in $\mathcal{C}_{\mathcal{E}}$. This why we needed many (d) levels in the leveled scheme to recover sk_0, and could not immediately use a self-loop from the outset.

So, if $\mathcal{E}^*$ is secure in the random oracle model despite not using a hash function, does that imply that it is secure in the plain model? Certainly not. The obstacle to this conclusion is obviously that random oracles cannot be instantiated in general [28]. A bit more specifically, however, the obstacle is that the proof of Theorem 4.4.2 depends crucially on the correctness of the ciphertext $(\langle \overline{r_j} \rangle, \sigma)$ in ${\mathcal{E}^{(d)}}^{\dagger}$ to construct (homomorphically) an encryption of sk_0 under pk_0 as needed for the $\mathcal{E}^*$ public key; however, in the proof of Theorem 4.4.1 the ciphertext is not correct (except with negligible probability): the adversary finds out that it was fake only after it has queried r to the random oracle, giving the reduction all the information it needs.

Proof. (Theorem 4.4.1) Let $\mathcal{A}$ be an algorithm that attacks the semantic security of ${\mathcal{E}^{(d)}}^{\dagger}$; from $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ that attacks the semantic security of $\mathcal{E}^{(d)}$. $\mathcal{B}$ will actually request $\ell' + 1$ challenge ciphertexts; thus, the reduction loses a factor of $\ell' + 1$ under the usual hybrid argument.

The challenger gives $\mathcal{B}$ a $\mathcal{E}^{(d)}$ public key. It also sets a bit $b \xleftarrow{\mathrm{R}} \{0, 1\}$. $\mathcal{B}$ selects two messages $r^{(0)}, r^{(1)} \in \mathcal{P}^{\ell'}$ and sends them to the challenger. The challenger sets $\Psi \xleftarrow{\mathrm{R}} \{\mathsf{Encrypt}(\mathrm{pk}_d, r_j^{(b)}) : j \in [1, \ell']\}$ and sends back Ψ. The following is included in the public key that $\mathcal{B}$ sends to $\mathcal{A}$: the public key for $\mathcal{E}^{(d)}$ sent by the challenger, the set of ciphertexts Ψ, and $\sigma \xleftarrow{\mathrm{R}} \mathcal{P}^{\ell}$.

$\mathcal{A}$ requests a challenge ciphertext on one $\pi_0, \pi_1 \in \mathcal{P}$. $\mathcal{B}$ forwards the query to the challenger, who responds with a ciphertext encrypting π_b, which $\mathcal{B}$ forwards to $\mathcal{A}$.

Eventually, either $\mathcal{A}$ queries some $r' \in \{r^{(0)}, r^{(1)}\}$ to the random oracle, or $\mathcal{A}$ finishes with a guess b'. In the former case, $\mathcal{B}$ sets b' so that $r' = r^{(b')}$. In either case, $\mathcal{B}$ sends b' as its guess to the challenger.

Let p be the probability that $\mathcal{A}$ queries some $r' \in \{r^{(0)}, r^{(1)}\}$ to the random oracle. $\mathcal{B}$'s simulation appears perfect to $\mathcal{A}$ if it does not query some $r' \in \{r^{(0)}, r^{(1)}\}$; in this case, which occurs with probability $1 - p$, $\mathcal{A}$'s advantage is at least ϵ. Since $\mathcal{A}$'s view is independent of $r^{(1-b)}$, the probability that it queries $r^{(b)}$ to the random oracle is at least $p - q_H/|\mathcal{P}|^{\ell'}$, where q_H is the number of random oracle queries make by $\mathcal{A}$. Overall $\mathcal{B}$'s advantage in guessing b' is at least $(1-p)\epsilon + p - q_H/|\mathcal{P}|^{\ell'} \geq \epsilon - q_H/|\mathcal{P}|^{\ell'}$.

□

Proof. (Theorem 4.4.2) The proof is essentially a simple consequence of the fact that, given a public key for ${\mathcal{E}^{(d)}}^\dagger$, it is easy to generate the public key for $\mathcal{E}^*$ homomorphically.

Let $\mathcal{A}$ be an algorithm that breaks the semantic security of $\mathcal{E}^*$. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks the semantic security of ${\mathcal{E}^{(d)}}^\dagger$.

$\mathcal{B}$ receives a ${\mathcal{E}^{(d)}}^\dagger$ public key from the challenger. This public key consists of $\langle \mathrm{pk}_i \rangle_{i \in [0,\delta]}$, $\langle \overline{\mathrm{sk}_{ij}} \rangle_{i \in [1,\delta]}$, $\langle \overline{r_j} \rangle_{j \in [1,\ell']}$, and $\sigma = H(r) \star \mathrm{sk}_0$. From $\langle \overline{r_j} \rangle$, $\mathcal{B}$ uses the homomorphism of $\mathcal{E}^{(d)}$ to compute ciphertexts Ψ that encrypt $H(r)$. It encrypts σ, and then uses the homomorphism to recover to obtain an encryption of sk_0 from the encryptions of $H(r)$ and σ (inverting the $\star$ operation). By assumption, these homomorphic operations take at most d levels. If it takes only $\delta < d$ levels, and we obtain an encryption of sk_0 under $\mathrm{pk}_{d-\delta}$, then we can perform $\mathsf{Recrypt}$ operations until we have the desired encryption of sk_0 under pk_0. By circuit privacy, this ciphertext is distributed properly. $\mathcal{B}$ includes the encryption of sk_0 under pk_0 as the encrypted secret key contained in the public key for $\mathcal{E}^*$ that it provides to $\mathcal{A}$.

$\mathcal{A}$ requests a challenge ciphertext on one $\pi_0, \pi_1 \in \mathcal{P}$. $\mathcal{B}$ forwards the query to the challenger, who responds with a ciphertext encrypting π_b. $\mathcal{B}$ uses $\mathsf{Recrypt}$ operations to obtain an encryption of π_b under pk_0 and forwards the result to $\mathcal{A}$. $\mathcal{A}$ sends a guess b', which $\mathcal{B}$ forwards to the challenger.

Clearly, $\mathcal{B}$'s advantage is the same as $\mathcal{A}$'s.

□

Chapter 5

An Abstract Scheme Based on the Ideal Coset Problem

Our goal now is to construct a bootstrappable encryption scheme, a scheme that can homomorphically evaluate a rich set of circuits that includes its own decryption circuit, "plus some." In the past, attempts to construct fully homomorphic encryption have focused solely on *maximizing* the complexity of the circuits that the scheme can evaluate. Our notion of bootstrapability gives us a different way of attacking the problem – by *minimizing* the complexity of the scheme's decryption circuit.

Our strategy for minimizing the circuit complexity of decryption is to construct our scheme using *ideal lattices*, since decryption in lattice-based cryptosystems is typically dominated by a simple operation, such as an easily parallelizable matrix-vector multiplication (in contrast to, say, RSA, where decryption involves exponentiation, an operation not even known to be in NC). We begin describing the ideal-lattice-based scheme in Chapter 7, after providing some basic background on ideal lattices in Chapter 6.

In this Chapter, we describe our strategy for maximizing the "evaluative capacity" of the scheme abstractly, without reference to lattices. Generally speaking, our exposition strategy throughout the paper is to defer technical lattice details for as long as possible. One reason is to make the presentation more modular, and therefore easier to understand. Another reason is that some of our techniques – e.g., bootstrapping, and using techniques from server-aided cryptography to "squash the decryption circuit" – maybe applicable to schemes that use different underlying mathematics – e.g., linear codes, or something less similar to lattices.

5.1 The Ideal Coset Problem

We saw in Chapter 3 that many previous homomorphic encryption schemes base security on some *ideal membership problem (IMP)*. For example, in the "Polly Cracker" scheme by Fellows and Koblitz [44], the public key consists of some multivariate polynomials that generate the ideal I of polynomials having a common root $\mathbf{x}$, and π is encrypted by outputting a sample $\psi \xleftarrow{\mathrm{R}} \pi + I$. One can easily see that this is semantically secure if it is hard to distinguish membership in I – in particular, deciding whether $\psi - \pi \in I$. Unfortunately, one can also see that homomorphic operations, especially multiplication, expand the ciphertext size potentially exponentially in the depth.

Since we will ultimately use lattices, we apparently need a different abstract approach, since it is easy to distinguish membership in a lattice L: given a basis $\mathbf{B}$ of L and $\mathbf{t} \in \mathbb{R}^n$, one simply determines whether $\mathbf{t} \bmod \mathbf{B} = \mathbf{0} \bmod B$. Instead, we base security on an *ideal coset problem (ICP)*, which we will state abstractly in terms of rings and ideals. Recall that a *ring* R is an algebraic object that is closed under addition '+' and multiplication '$\times$' and additive inverse, with an additive identity '0' and multiplicative identity '1'. An *ideal* I of a ring R is a subset satisfying $a + b \in I$ and $r \times a \in I$ for all $a, b \in I$ and $r \in R$. The sum and product of two ideals I and J are, respectively, $\{i + j : i \in I, j \in J\}$ and the additive closure of $\{i \times j : i \in I, j \in J\}$. Two ideals I and J are *relatively prime* if $I + J = R$. For example, if $R = \mathbb{Z}$, the ideals (2) (the even integers) and (5) (the integers divisible by 5) are relatively prime: $(2) + (5) = (1)$.

Now, the ideal coset problem (ICP) is as follows.

Definition 5.1.1 (Ideal Coset Problem (ICP))**.** Fix R, $\mathbf{B}_I$, algorithm $\mathsf{IdealGen}$, and an algorithm Samp_1 that efficiently samples R. The challenger sets $b \xleftarrow{\mathrm{R}} \{0, 1\}$ and $(\mathbf{B}_J^{\mathrm{sk}}, \mathbf{B}_J^{\mathrm{pk}}) \xleftarrow{\mathrm{R}} \mathsf{IdealGen}(R, \mathbf{B}_I)$. If $b = 0$, it sets $\mathbf{r} \xleftarrow{\mathrm{R}} \mathsf{Samp}_1(R)$ and $\mathbf{t} \leftarrow \mathbf{r} \bmod \mathbf{B}_J^{\mathrm{pk}}$. If $b = 1$, it samples $\mathbf{t}$ uniformly from $R \bmod \mathbf{B}_J^{\mathrm{pk}}$. The problem: guess b given $(\mathbf{t}, \mathbf{B}_J^{\mathrm{pk}})$.

Basically the ICP asks one to decide whether $\mathbf{t}$ is uniform modulo J, or whether it was chosen according to a known "clumpier" distribution induced by Samp_1. Of course, the ICP will be impossible if Samp_1 also samples uniformly modulo J, but the security of our encryption scheme will rely on the ICP being hard for a "clumpier" instantiation of Samp_1; the hardness of the problem depends on the particular instantiation of Samp_1. Note that it is possible for the ICP to be hard even when the IMP is easy.

5.2 An Abstract Scheme

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

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

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

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

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

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

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

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

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

Remark 5.2.1. Concerning $\mathsf{IdealGen}$, it is fine if the secret basis $\mathbf{B}_J^{\mathrm{sk}}$ defines a lattice $\mathcal{L}(\mathbf{B}_J^{\mathrm{sk}})$ for a (possibly fractional) ideal that *contains* J, rather than being exactly J.

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

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

Definition 5.2.2 (Generalized Circuit)**.** Let C be a mod-$\mathbf{B}_I$ circuit. We say generalized circuit $g(C)$ of C is the circuit formed by replacing C's $\mathsf{Add}_{\mathbf{B}_I}$ and $\mathsf{Mult}_{\mathbf{B}_I}$ operations with addition '+' and multiplication '$\times$' in the ring R.

Here are a few more definitions relevant to Theorem 5.2.6 below, which concerns correctness.

Definition 5.2.3 (X_{Enc} and X_{Dec})**.** Let X_{Enc} be the image of Samp. Notice that all ciphertexts output by $\mathsf{Encrypt}$ are in $X_{\mathsf{Enc}}+J$. Let X_{Dec} equal $R \bmod \mathbf{B}_J^{\mathrm{sk}}$, the set of distinguished representatives of cosets of J wrt the secret basis $\mathbf{B}_J^{\mathrm{sk}}$.

Definition 5.2.4 (Permitted Circuits)**.** Let

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

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

Definition 5.2.5 (Valid Ciphertext)**.** ψ is a *valid ciphertext* wrt $\mathcal{E}$ public key pk and permitted circuits $\mathcal{C}_{\mathcal{E}}$ if it equals $\mathsf{Evaluate}(\mathrm{pk}, C, \Psi)$ for some $C \in \mathcal{C}_{\mathcal{E}}$, where each $\psi \in \Psi$ is in the image of $\mathsf{Encrypt}$. The circuit C may be the identity circuit, in which case the output of $\mathsf{Evaluate}$ is simply an output of $\mathsf{Encrypt}$.

Finally, we prove correctness with respect to $\mathcal{C}_{\mathcal{E}}$.

Theorem 5.2.6. *Assume $\mathcal{C}_{\mathcal{E}}$ is a set of permitted circuits containing the identity circuit. $\mathcal{E}$ is correct for $\mathcal{C}_{\mathcal{E}}$ – i.e.,* $\mathsf{Decrypt}$ *correctly decrypts valid ciphertexts.*

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

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

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

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

as required.

□

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

Again, we note the rather confusing fact that C "automatically" reduces the result modulo $\mathbf{B}_I$, since it uses mod-$\mathbf{B}_I$ gates. It does not particularly matter how these mod-$\mathbf{B}_I$ gates are implemented; in particular, it is more confusing than helpful to imagine a boolean implementation of these gates. Instead, one should just observe that the generalized circuit manages to lazily emulate these gates, reducing its output modulo $\mathbf{B}_I$ at the end of the computation. C's mod-$\mathbf{B}_I$ operations are never actually "implemented;" they only occur implicitly. Later, when we consider whether our scheme is bootstrappable, and analyze the depth of the decryption circuit in terms of mod-$\mathbf{B}_I$ gates, it will again be tempting to consider how these gates are "implemented." But in fact these gates are "given" in the sense that they are emulated (without any intermediate reduction steps) by the usual ring operations.

5.3 Security of the Abstract Scheme

For the following abstract "instantiation" of Samp, and where I is a principle ideal generated by some $\mathbf{s} \in R$ (and $\mathbf{s}$ is encoded in $\mathbf{B}_I$), we provide a simple proof of semantic security based on the ICP.

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

Obviously, the output is in $\mathbf{x} + I$ since $\mathbf{s} \in I$.

Theorem 5.3.1. *Suppose that there is an algorithm $\mathcal{A}$ that breaks the semantic security of $\mathcal{E}$ with advantage ϵ when it uses Samp. Then, there is an algorithm $\mathcal{B}$, running in about the same time as $\mathcal{A}$, that solves the ICP with advantage $\epsilon/2$.*

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

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

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

□

Chapter 6

Background on Ideal Lattices I: The Basics

From the abstract construction in Chapter 5, among the objects that we need to make concrete are: the ring R, the ideals I and J, how to compute $\mathbf{t} \bmod \mathbf{B}_M$, the algorithms Samp and $\mathsf{IdealGen}$, and a concrete version of the ICP. In this Chapter, we provide some basic background material needed to instantiate these things while using ideal lattices. Later, we will provide more background on ideal lattices as needed.

6.1 Basic Background on Lattices

Let $\mathbb{R}$ denote the real numbers, and $\mathbb{Z}$ the integers. We write vectors in column form using bold lower-case letters, e.g. $\mathbf{v}$; We write matrices as bold capital letters, e.g., $\mathbf{B}$; $\mathbf{b}_i$ is the ith column. We use $\|\mathbf{v}\|$ to denote the Euclidean length of a vector $\mathbf{v}$. For matrix $\mathbf{B}$, we use $\|\mathbf{B}\|$ to denote the length of the longest column vector in $\mathbf{B}$.

An n-dimensional *lattice* of rank $k \leq n$ is

$$L = \mathcal{L}(\mathbf{B}) = \left\{ \mathbf{Bc} \ : \ \mathbf{c} \in \mathbb{Z}^k \right\}, \mathbf{B} \in \mathbb{R}^{n \times k}$$

where the k columns $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$ of the *basis* are linearly independent. All lattices in this paper are full rank – i.e., $k = n$. Usually lattices in this paper are sub-lattices of $\mathbb{Z}^n$ – i.e., the lattice vectors have integer coefficients.

Every lattice has an infinite number of lattice bases. If $\mathbf{B}_1$ and $\mathbf{B}_2$ are two lattice

bases of L, then there is some matrix $\mathbf{U}$ that is unimodular (i.e., $\mathbf{U}$ has integer entries and $\det(\mathbf{U}) = \pm 1$) satisfying $\mathbf{B}_1 \cdot \mathbf{U} = \mathbf{B}_2$. Since $\mathbf{U}$ is unimodular, $|\det(\mathbf{B}_i)|$ is invariant for different bases of L. Since it is invariant, we may refer to $\det(L)$. This value is precisely the size of the quotient group $\mathbb{Z}^n/L$ if L is an integer lattice.

To basis $\mathbf{B}$ of lattice L we associate the half-open parallelepiped $\mathcal{P}(\mathbf{B}) \leftarrow \{\sum_{i=1}^n x_i \mathbf{b}_i : x_i \in [-1/2, 1/2)\}$. The volume of $\mathcal{P}(\mathbf{B})$ is precisely $\det(L)$.

Informally, we say that some bases of L are "good" and some are "bad;" a basis $\mathbf{B}$ of L is "good," roughly speaking, if the vectors of $\mathbf{B}$ are reasonably short and nearly orthogonal. Of course, for any basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, it must hold that $\prod_{i=1}^n \|\mathbf{b}_i\| \geq \det(L)$; roughly speaking, good bases come closer to reaching equality than bad ones.

For $\mathbf{t} \in \mathbb{R}^n$, we use $\mathbf{t} \bmod \mathbf{B}$ to denote the unique vector $\mathbf{t}' \in \mathcal{P}(\mathbf{B})$ such that $\mathbf{t} - \mathbf{t}' \in L$. Given $\mathbf{t}$ and $\mathbf{B}$, $\mathbf{t} \bmod \mathbf{B}$ can be computed efficiently as $\mathbf{t} - \mathbf{B} \cdot \lfloor \mathbf{B}^{-1} \cdot \mathbf{t} \rceil$, where $\lfloor \cdot \rceil$ rounds the coefficients of a vector to the nearest integer. Let $\mathrm{dist}(L, \mathbf{t})$ denote $\min_{\mathbf{v} \in L}\{\|\mathbf{t} - \mathbf{v}\|\}$. Clearly, for any basis $\mathbf{B}$, $\|\mathbf{t} \bmod \mathbf{B}\| \geq \mathrm{dist}(L, \mathbf{t})$, though again (roughly speaking) "good" bases come closer to equality.

In some sense, the *worst* basis of a lattice L is its unique upper-triangular Hermite normal form $\mathrm{HNF}(L)$. Given any basis $\mathbf{B}$ of L, one can compute $\mathrm{HNF}(L)$ efficiently – i.e., in time $\mathrm{poly}(n, \log \|\mathbf{B}\|)$. Thus, $\mathrm{HNF}(L)$ does not "reveal" more about L's structure than any other basis, making $\mathrm{HNF}(L)$ a good choice for the public lattice basis to be included in a public key [97].

The *dual lattice* of L, denoted L^*, is defined as $L^* = \{\mathbf{x} \in \mathrm{span}(L) : \forall \mathbf{v} \in L, \langle \mathbf{x}, \mathbf{v} \rangle \in \mathbb{Z}\}$. It holds that $\det(L) \cdot \det(L^*) = 1$. If $\mathbf{B}$ is a basis for the full-rank lattice L, then $(\mathbf{B}^{-1})^T$ (the inverse transpose of $\mathbf{B}$) is a basis of L^*.

The ith *successive minimum* $\lambda_i(L)$ is the smallest radius r such that L contains at least i linearly independent vectors of norm at most r. In particular $\lambda_1(L)$ is the length of the shortest nonzero vector in L. A very good basis may have some of these very short vectors.

The two most well-known lattices problems are the shortest vector problem (SVP) and closest vector problem (CVP). Here are their approximate versions.

Definition 6.1.1 ($\gamma(n)$-Shortest Vector Problem (SVP))**.** Given a basis for a lattice L of dimension n, output a nonzero vector $\mathbf{v} \in L$ of length at most $\gamma(n) \cdot \lambda_1(L)$.

Definition 6.1.2 ($\gamma(n)$-Closest Vector Problem (CVP))**.** Given a basis for a lattice L of dimension n and a vector $\mathbf{t} \in \mathbb{R}^n$, output a nonzero vector $\mathbf{v} \in L$ such that $\|\mathbf{t} - \mathbf{v}\| \leq \gamma(n) \cdot \mathrm{dist}(L, \mathbf{t})$.

A close variant of the SVP is the shortest independent vector problem (SIVP), defined as follows.

Definition 6.1.3 ($\gamma(n)$-Shortest Independent Vector Problem (SIVP))**.** Like the SVP, except one outputs linearly independent $\mathbf{v}_1, \ldots, \mathbf{v}_n \in L$, all of length at most $\gamma(n) \cdot \lambda_n(L)$.

In a variant of the CVP, one is given the promise that the closest L-vector to $\mathbf{t}$ is *much* closer than any other – e.g., by a factor of $\gamma(n)$.

Definition 6.1.4 ($\gamma(n)$-Bounded Distance Decoding Problem (BDDP))**.** Same as $\gamma(n)$-CVP, but with the promise that there is a unique solution – i.e., $(\gamma(n)+1)\cdot\text{dist}(L, \mathbf{t}) < \lambda_1(L)$.

In other words, the BDDP is the CVP under the promise that $\mathbf{t}$ is very close to the lattice L, and that in fact the solution $\mathbf{v}$ is unique. The solution is unique, since if $\|\mathbf{t}-\mathbf{v}\| < \lambda_1(L)/(\gamma(n)+1)$, then $\|\mathbf{t}-\mathbf{w}\| \geq \|\mathbf{v}-\mathbf{w}\|-\|\mathbf{t}-\mathbf{v}\| > \lambda_1(L)\cdot\gamma(n)/(\gamma(n)+1) > \gamma(n)\cdot\text{dist}(L, \mathbf{t})$ for all $\mathbf{w} \in L \setminus \{\mathbf{v}\}$. This definition of the BDDP is non-standard, in the sense that in $\gamma(n)$-BDDP, $\gamma(n)$ is typically defined to be an upper bound on the ratio $\text{dist}(L, \mathbf{t})/\lambda_1(L)$, whereas we prefer (essentially) to define it to be a lower-bound on $\lambda_1(L)/\text{dist}(L, \mathbf{t})$, since (in our formulation) the problem becomes easier as $\gamma(n)$ becomes larger (as in $\gamma(n)$-SVP, $\gamma(n)$-CVP, and $\gamma(n)$-SIVP).

Aside from BDDP, the above problems are known to be NP-hard for very small approximation factors. For all of these problems, the best polynomial-time approximation algorithms are variants of the lattice reduction algorithm LLL by Lenstra et al. [81] or Babai's nearest plane algorithm [13]; these algorithms only work for essentially-exponential (e.g., $2^{\mathcal{O}(n(\log\log n)/\log n)}$ [5]) approximation factors. As a rough rule of thumb, approximating these lattice problems to within a factor of 2^k takes time about $2^{n/k}$, using known algorithms [123].

6.2 Basic Background on Ideal Lattices

To our knowledge, the first use of ideal lattices in cryptography was the NTRU cryptosystem by Hoffstein et al. [69],[1] though the connection to lattices was made explicit later in cryptanalysis [34, 93, 47]. None of this cryptanalysis has broken the core average-case problem underlying the scheme. NTRU's main selling point is efficiency; encryption and

[1]Strictly speaking, NTRU's lattice has a $2n \times 2n$ basis, where each $n \times n$ quadrant generates an ideal lattice.

decryption very fast – much faster than RSA, for example – since the operations involved are simple (multiplications in the ring $\mathbb{Z}_q[x]/(x^n - 1)$ for small integer q), and since n can be reasonably small (several hundreds) since the best known lattice attacks on NTRU take time essentially exponential in n.

Recent cryptography involving ideal lattices [98, 111, 112, 88, 99] is typically framed immediately with reference to Ajtai's worst-case / average-case connection. In these works, they have been used to construct, for example, hash functions and signature schemes.

Our construction will use the polynomial ring $R = \mathbb{Z}[x]/(f(x))$, where $f(x)$ is a monic polynomial of degree n. We view an element $\mathbf{v} \in R$ both as a ring element and *as a vector* – specifically, the coefficient vector $\mathbf{v} \in \mathbb{Z}^n$. The ideal $(\mathbf{v})$ generated by $\mathbf{v}$ directly corresponds to the lattice generated by the column vectors $\{\mathbf{v}_i \leftarrow \mathbf{v} \times x^i \bmod f(x) : i \in [0, n-1]\}$; we call this the *rotation basis* of the *ideal lattice* $(\mathbf{v})$. Specifically, any $\mathbf{w} \in (\mathbf{v})$ is in the lattice generated by the rotation basis $\{\mathbf{v}_i\}$, since there must be some $\mathbf{a}$ for which $\mathbf{w} = \mathbf{v} \times \mathbf{a}$, and then $\mathbf{w} = \sum_i a_i \mathbf{v}_i$. Conversely, if $\mathbf{w}$ is in the lattice generated by $\{\mathbf{v}_i\}$, then $\mathbf{w} = \sum_i a_i \mathbf{v}_i$ for some integers $\{a_i\}$, which implies that $\mathbf{w} = \mathbf{v} \times \mathbf{a}$ in the ring R, where $\mathbf{a} = \sum_i a_i \cdot x^i$. In general, the rotation basis for the product of two elements $\mathbf{a}, \mathbf{b} \in \mathbb{Q}[x]/(f(x))$ is the rotation basis of $\mathbf{a} \times \mathbf{b}$. Also the matrix-vector product of a rotation basis $\mathbf{a}$ with the vector $\mathbf{b}$ is the vector $\mathbf{a} \times \mathbf{b}$.

Generally speaking, an ideal $I \subset R$ need not be *principal* – i.e., have a single generator – and a basis $\mathbf{B}_I$ of I need not be a rotation basis. Suppose it is generated by $\mathbf{v}$ and $\mathbf{w}$. In this case, the ideal is represented by the lattice generated by the columns $\{\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}, \mathbf{w}_0, \ldots, \mathbf{w}_{n-1}\}$, where $\mathbf{w}_i$ is the vector associated to $\mathbf{w} \times x^i$. Of course, the vectors in this set will be linearly dependent. A lattice reduction algorithm, such as LLL, will find these dependencies and output a basis for the lattice associated to I that contains only linearly independent vectors.

Sometimes we will use inverses in the ring $\mathbb{Q}[x]/(f(x))$. In this case, to avoid complications, we assume $f(x)$ is irreducible and therefore all nonzero terms have inverses. If I is an ideal in R, I^{-1} is a *fractional ideal*. I^{-1} is defined in a somewhat similar way as a dual lattice; it is the set $\{\mathbf{x} \in \mathbb{Q}[x]/(f(x)) : \forall \mathbf{y} \in I, \mathbf{x} \times \mathbf{y} \in R\}$. Aside from the fact that I^{-1} is not necessarily a subset of R, it is exactly like a normal ideal – in particular, it is closed under addition and under multiplication with R-elements. We say that (possibly fractional) ideals I and J are relatively prime if $I + J \supseteq R$. For example, ideal $(2/5)$ and $(3/7)$ are relatively prime (contain (1)), but $(3/5)$ and $(3/7)$ are not, since (1) is not in $(3/35)$.

For principal ideal $(\mathbf{v})$, the fractional ideal $(\mathbf{v})^{-1}$ is generated by $1/\mathbf{v}$, where the inverse is in $\mathbb{Q}[x]/(f(x))$. The determinant associated to the ideal lattice for $(\mathbf{v})$ (we may occasionally refer to this determinant as the *norm* of the ideal, denoted $\mathrm{Nm}(I)$) is the inverse of the determinant of $(1/\mathbf{v})$. For an ideal I that has multiple generators $\mathbf{v}_1, \mathbf{v}_2, \ldots$, the fractional ideal I^{-1} is the intersection of $(1/\mathbf{v}_1), (1/\mathbf{v}_2), \ldots$.

In our constructions, we will use a polynomial ring as defined above. Such rings are called *monogenic* number rings, or *simple algebraic extensions*, because they are isomorphic to $\mathbb{Z}[\alpha]$ where α is a root of $f(x)$. *Algorithmically*, such rings are easy to work with, which will be important later for minimizing the complexity of our decryption circuit.

Algebraically, however, a more natural ring would be the ring of integers associated to a number field. A number field is a finite extension $K = \mathbb{Q}(\alpha)$ of the rationals $\mathbb{Q}$, isomorphic to $\mathbb{Q}[x]/(f(x))$ for some polynomial $f(x)$ irreducible over $\mathbb{Q}$ for which $f(\alpha) = 0$. The *ring of integers* of a number field K is:

$$\mathcal{O}_K = \{x \in K : f^x_{\mathbb{Q}} \in \mathbb{Z}[x]\} \text{ , where } f^x_{\mathbb{Q}} \text{ is the (monic) minimal polynomial of } x \text{ in } \mathbb{Q}[x]$$

While it may not be immediately obvious that $\mathcal{O}_K$ is even a ring, $\mathcal{O}_{\mathbb{Q}(\alpha)}$ generally has better algebraic properties than $\mathbb{Z}[\alpha]$, most notably that every ideal I of the ring of integers factors uniquely as a product of prime ideals in the ring. Also, all ideals I of $\mathcal{O}_K$ are "invertible" – i.e., $I^{-1} \cdot I = \mathcal{O}_K$ when the inverse I^{-1} is taken in $\mathcal{O}_K$; this is not necessary true in $\mathbb{Z}[\alpha]$, where $I^{-1} \cdot I$ may be a subset of R if $\mathrm{Nm}(I)$ is divisible by one of a small number of *singular* primes whose squares divide the discriminant $\Delta(f)$ of $f(x)$ [127]. Peikert and Rosen [112] show that ideal lattices associated to the ring of integers in fields with very small root discriminant have very small worst-case / average-case connection factors, only logarithmic (versus polynomial) in n. While their approach is appealing, and most likely can be used in connection with our scheme, we choose instead to use $\mathbb{Z}[\alpha]$ because using integer vectors permits us to push complicated details away from the decryption circuit, which is already quite complicated. Also, it is straightforward, though tedious, to simply avoid the singular primes when working with $\mathbb{Z}[\alpha]$.

Since all of the hardness assumptions are with respect to a fixed ring R, one must choose it wisely. For example, a seemingly attractive choice for R is the ring $\mathbb{Z}[x]/(x^n - 1)$. Aside from efficiency, this choice in some sense maximizes the multiplicative depth of circuits that our scheme can evaluate, since one can bound the Euclidean length $\|\mathbf{u} \times \mathbf{v}\|$ by $\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| \cdot$

$\|\mathbf{v}\|$ for $\gamma_{\mathsf{Mult}}(R) = \sqrt{n}$; other rings have larger values of $\gamma_{\mathsf{Mult}}(R)$. We note that the NTRU encryption scheme, whose core hard problem has never been broken, uses this ring (though it uses a lattice basis that consists of 4 quadrants, where each quadrant is a basis of an ideal lattice in R). On the other hand, although there is no known attack against ideal lattice problems in this ring that is completely fatal, there are some attacks that suggest that this ring may be weaker than others. One fairly obvious attack by Gentry [47] works when n is composite; essentially, it reduces a lattice problem over $\mathbb{Z}[x]/(x^{cm} - 1)$ to a much more tractable m-dimensional lattice problem over $\mathbb{Z}[x]/(x^m - 1)$ for small constant c. Generally, one would prefer $f(x)$ to be irreducible. Even when n is prime, Gentry and Szydlo [50] gave an algorithm that can be adapted to take an n-dimensional basis of a *principal* ideal lattice I of $R = \mathbb{Z}[x]/(x^n - 1)$, and construct a $(n+1)/2$-dimensional lattice basis that contains at least one nonzero I-vector of length at most $\sqrt{2} \cdot \lambda_1(I)$; if I has an *orthonormal* basis, their algorithm can find it in polynomial time. But again we mention that these attacks are not fatal for $\mathbb{Z}[x]/(x^n - 1)$. If one simply takes n prime and (easily) avoids ideals with orthonormal bases, the Gentry-Szydlo attack only gives an attack whose running time is at best square root of the original time of attack, which is fine (in principle) if the original time of attack is super-polynomial.

6.3 Probability Background

A family $\mathcal{H}$ of hash functions from X to Y, both finite sets, is said to be 2-universal if $\Pr_{h \xleftarrow{R} \mathcal{H}}[h(x) = h(x')] = 1/|Y|$ for all distinct $x, x' \in X$. A distribution D is ϵ-uniform if its statistical distance from the uniform distribution is at most ϵ, where the statistical difference between two distributions D_1, D_2 is $\frac{1}{2}\sum_{x \in X} |D_1(x) - D_2(x)|$.

Lemma 6.3.1 (Leftover Hash Lemma [72])**.** *Let $\mathcal{H}$ be a family of 2-universal hash functions from X to Y. Suppose that $h \xleftarrow{R} \mathcal{H}$ and $x \xleftarrow{R} X$ are chosen uniformly and independently. Then, $(h, h(x))$ is $\frac{1}{2}\sqrt{|Y|/|X|}$-uniform over $\mathcal{H} \times Y$.*

Chapter 7

A Somewhat Homomorphic Encryption Scheme

7.1 Why Lattices?

To bootstrap our new notion of bootstrappability, we ask a natural question: where do we find encryption schemes that have decryption algorithms with low circuit complexity?

We note that this is not an *essential* question. Conceivably, $\mathcal{E}$ could be tailored so that it evaluates *only* its (augmented) decryption circuits $D_{\mathcal{E}}(\Gamma)$, or very few gates outside of this small set, even though its decryption circuit is "complex" [58]. However, our approach will be to look for a scheme that evaluates circuits at least *as complex as* (e.g., in terms of depth) its (augmented) decryption circuit.

Under this approach, it does not make much sense to look at schemes based on factoring or variants of Diffie-Hellman, even though there are several homomorphic schemes here – RSA [121], Goldwasser-Micali [61], ElGamal [42], Paillier [110], Boneh-Goh-Nissim [21], etc. In all of these schemes, decryption uses some operation – exponentiation, Legendre symbol computation, pairing – that is not even known to have circuit complexity in NC. For these schemes, we can reduce the depth of the decryption circuit somewhat by using techniques like those described in Section 10, where we offload some decryption work onto the encrypter, who outputs a longer ciphertext that can be decrypted by a shallower circuit, but we do not see how to reduce the decryption depth enough to make these schemes bootstrappable.

On the other hand, for encryption schemes based on *lattices* or *linear codes*, the dominant decryption operation is typically an inner product or matrix-vector multiplication, which is

in NC1 (assuming the bit-length of the coefficients are polynomial in the vector dimension).

7.2 Why *Ideal* Lattices?

To be bootstrappable, it is not enough that the scheme has a decryption circuit of low complexity; the scheme needs to be able to *evaluate* that circuit. We already have schemes that can evaluate circuits in NC1. In fact, unless one wants circuit privacy (as in Sanders-Young-Yung [122]), "evaluating" circuits of logarithmic depth is completely trivial: one simply outputs the circuit and the "unprocessed" input ciphertexts. So, why is it not trivial to construct a bootstrappable encryption scheme from a lattice-based scheme that has a decryption circuit in NC1?

The problem with the trivial construction, and with SYY, is that they achieve logarithmic depth by permitting the ciphertext size to grow exponentially with the circuit depth. As the ciphertext grows, the decryption circuit must also grow to handle the larger ciphertexts. In short, as one allows larger and larger ciphertexts, the evaluation depth will never "catch up" to the depth of the decryption circuit. To obtain a bootstrappable encryption scheme, it seems necessary to consider encryption schemes that have more complex *inherent* homomorphisms.

As we will see, while general lattices offer an additive structure, ideal lattices also have a multiplicative structure that will enable us to evaluate deep arithmetic circuits (though we will need more tricks before we ultimately obtain a bootstrappable scheme).

7.3 A Geometric Approach to Maximizing the Circuit Depth that Can Be Evaluated

In Section 5, where we described the abstract scheme, we saw that $\mathcal{E}$ correctly evaluates circuit C if the generalized circuit $g(C)$ satisfies $g(C)(x_1, \ldots, x_t) \in X_{\mathsf{Dec}}$ for all $(x_1, \ldots, x_t) \in {X_{\mathsf{Enc}}}^t$. For example, it correctly evaluates the gate $\mathsf{Add}_{\mathbf{B}_I}$ if $X_{\mathsf{Enc}} + X_{\mathsf{Enc}} \subseteq X_{\mathsf{Dec}}$, and the gate $\mathsf{Mult}_{\mathbf{B}_I}$ if $X_{\mathsf{Enc}} \times X_{\mathsf{Enc}} \subseteq X_{\mathsf{Dec}}$. Our hope is that applying these gates – indeed, even applying high-depth circuits – does not cause to much "expansion," so that the output of the generalized circuit remains within X_{Dec}.

An important reason that we use ideal *lattices*, versus ideals over general rings, is that lattices permit a clean analysis of X_{Enc} and X_{Dec} in terms of *Euclidean length*. When we

implement the abstract scheme using a polynomial ring $\mathbb{Z}[x]/(f(x))$ and ideal lattices as summarized above, the sets X_{Enc} and X_{Dec} become subsets of $\mathbb{Z}^n$. We re-characterize these sets geometrically as follows.

Definition 7.3.1 (r_{Enc} and r_{Dec})**.** Let r_{Enc} be the smallest value such that $X_{\mathsf{Enc}} \subseteq \mathcal{B}(r_{\mathsf{Enc}})$, where $\mathcal{B}(r)$ is the ball of radius r. Let r_{Dec} be the largest such that $X_{\mathsf{Dec}} \supseteq \mathcal{B}(r_{\mathsf{Dec}})$.

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

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

$\mathcal{C}_{\mathcal{E}}$ is defined like the maximal set $\mathcal{C}_{\mathcal{E}}{}'$ of permitted circuits in Definition 5.2.4, but we have replaced X_{Enc} and X_{Dec} with $\mathcal{B}(r_{\mathsf{Enc}})$ and $\mathcal{B}(r_{\mathsf{Dec}})$. Clearly, $\mathcal{C}_{\mathcal{E}} \subseteq \mathcal{C}_{\mathcal{E}}{}'$. (At several points later in the paper, we narrow our set of permitted circuits again so as to enable a less complex decryption algorithm.)

For fixed values of r_{Enc} and r_{Dec}, what is $\mathcal{C}_{\mathcal{E}}$? This is a geometric problem, and we can bound the Euclidean length $\|g(C)(\mathbf{x}_1, \ldots, \mathbf{x}_t)\|$ by bounding the lengths of $\|\mathbf{u} + \mathbf{v}\|$ and $\|\mathbf{u} \times \mathbf{v}\|$ in terms of $\|\mathbf{u}\|$ and $\|\mathbf{v}\|$. For addition, this is easy: using the triangle inequality, we have $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ for $\mathbf{u}, \mathbf{v} \in R$. For multiplication, we can prove that $\|\mathbf{u} \times \mathbf{v}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| \cdot \|\mathbf{v}\|$, where $\gamma_{\mathsf{Mult}}(R)$ is some factor that is dependent only on the ring R. (See [89] for a different definition of the expansion factor for multiplication.)

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

Theorem 7.3.2. *Suppose $r_{\mathsf{E}} \geq 1$ and that circuit C's additive fan-in is $\gamma_{\mathsf{Mult}}(R)$, multiplicative fan-in is 2, and depth is at most*

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

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

In particular, $\mathcal{E}$ correctly evaluates circuits of depth up to $\log \log r_{\mathsf{Dec}} - \log \log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$.

Proof. For a d-depth circuit, let r_i be an upper-bound on the Euclidean norm of the values at level i, given that $r_d = r_{\mathsf{E}}$. By the triangle inequality, an addition (or subtraction) gate at level i outputs some $\mathbf{v} \in R$ such that $\|\mathbf{v}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i$. A multiplication gate at level i

outputs some $\mathbf{v} \in R$ such that $\|\mathbf{v}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i^2$. In either case, $r_{i-1} \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i^2$, and thus $r_0 \leq (\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{E}})^{2^d}$. The result follows.

□

An (oversimplified) bottom line from Theorem 7.3.2 is that, to maximize the depth of circuits that $\mathcal{E}$ can correctly evaluate (see Theorem 5.2.6), we should minimize $\gamma_{\mathsf{Mult}}(R)$ and r_{Enc}, and maximize r_{Dec}. Most of the remainder of this section consists of proposals toward this goal.

7.4 Instantiating the Ring: The Geometry of Polynomial Rings

From Theorem 7.3.2, it seems important to set $f(x)$ so that $R = \mathbb{Z}[x]/(f(x))$ has a reasonably small value of $\gamma_{\mathsf{Mult}}(R)$. (Recall that $\gamma_{\mathsf{Mult}}(R)$ is a value such that $\|\mathbf{u} \times \mathbf{v}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| \times \|\mathbf{v}\|$.) The following results show that there are many $f(x)$ for which the associated $\gamma_{\mathsf{Mult}}(R)$ is only polynomial in n. Lyubashevsky and Micciancio [89] actually already have results of a similar flavor to those in this Section in a full version of a paper using ideal lattices for hash functions [89], for a definition of "expansion factor" (analogous to our $\gamma_{\mathsf{Mult}}(R)$) that is a bit more cumbersome to generalize to high-degree products.

Theorem 7.4.1. *Let $f(x)$ be a monic polynomial of degree n. Let $F(x) = x^n \cdot f(1/x)$ and $g(x) = F(x)^{-1} \bmod x^{n-1}$. Then, $\|\mathbf{u} \times \mathbf{v}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| \cdot \|\mathbf{v}\|$ for some*

$$\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot \|f\| \cdot \|g\|)$$

Proof. (Theorem 7.4.1) Let $t(x) \leftarrow \mathbf{u}(x) \cdot \mathbf{v}(x)$ be the (unreduced) degree $2n-2$ product of $\mathbf{u}$ and $\mathbf{v}$. Let $t(x) = q(x)f(x) + r(x)$, where $r(x) = t(x) \bmod f(x)$ is a polynomial of degree $n-1$, and $q(x)$ is a polynomial of degree $n-2$. We have $\|\mathbf{u} \times \mathbf{v}\| = \|r\|$, the latter term denoting the Euclidean norm of the vector formed by the coefficients of $r(x)$.

Note that each coefficient of $t(x)$, being an inner product of some subset of coefficients of $\mathbf{u}$ and $\mathbf{v}$, must have norm less than $\|\mathbf{u}\| \cdot \|\mathbf{v}\|$; overall, $\|t\| \leq \sqrt{2n} \cdot \|\mathbf{u}\| \cdot \|\mathbf{v}\|$.

Let $T(x) = x^{2n-2}t(1/x)$, $Q(x) = x^{n-2}q(1/x)$, and $R(x) = x^{2n-2}r(1/x)$. Then, $T(x) = Q(x)F(x) + R(x)$, where T, Q, F are all integer polynomials with the same degrees and norms as t, q, f. R, which has the same norm as r, is divisible by x^{n-1}, implying that

$Q(x) = T(x)g(x) \bmod x^{n-1}$. Since $Q(x)$ has degree $n-2$, this equation implies $\|Q\| \leq \sqrt{2n} \cdot \|T\| \cdot \|g\|$. We have

$$\begin{aligned} \|\mathbf{u} \times \mathbf{v}\| &= \|r\| = \|R\| \leq \|T\| + \|Q \cdot F\| \\ &\leq \|T\| + \sqrt{2n} \cdot \|Q\| \cdot \|F\| \\ &\leq \|T\| + 2n \cdot \|T\| \cdot \|g\| \cdot \|F\| \\ &= \|t\| \cdot (1 + 2n \cdot \|f\| \cdot \|g\|) \\ &\leq \|\mathbf{u}\| \cdot \|\mathbf{v}\| \cdot \sqrt{2n} \cdot (1 + 2n \cdot \|f\| \cdot \|g\|) \end{aligned}$$

as required.

□

To find a suitable ring $R = \mathbb{Z}[x]/(f(x))$ for which $\gamma_{\mathsf{Mult}}(R)$ is small, it suffices to find an $f(x)$ for which both $F(x)$ and $F(x)^{-1} \bmod x^{n-1}$ have small norms, where $F(x) = x^n \cdot f(1/x)$. This gives us a lot of freedom in choosing $f(x)$.

For example, we can sample $f(x)$ from the large class of polynomials such that $f(x)$ has small norm and $f(x) = x^n - h(x)$ where $h(x)$ is a polynomial of degree at most $(n+1)/2$. In this case, for $R = \mathbb{Z}[x]/(f(x))$, one can prove that $\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot \|f\|^2)$. One can generalize this to the case that $h(x)$ has degree at most $n - (n-1)/k$ for $k > 2$.

Theorem 7.4.2. *Suppose $f(x) = x^n - h(x)$ where $h(x)$ has degree at most $n-(n-1)/k$ for $k \geq 2$. Then, for $R = \mathbb{Z}[x]/(f(x))$, it holds that $\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot (\sqrt{(k-1)n}\|f\|)^k)$.*

Proof. Let $F(x) = x^n \cdot f(1/x) = 1 - x^n \cdot h(1/x)$. Let $H(x) = x^n \cdot h(1/x)$. Note that $H(x)$ is divisible by x^m for integer $m \geq (n-1)/k$, since $h(x)$ has degree at most $n - (n-1)/k$. This fact implies that $1 - H(x)^k = 1 \bmod x^{n-1}$. So, $g(x) \leftarrow F(x)^{-1} = 1/(1 - H(x)) = (1 - (H(x))^k)/(1 - H(x)) \bmod x^{n-1}$, and we have:

$$\begin{aligned} \|g(x)\| &\leq 1 + \|H\| + \cdots + \|H^{k-1}\| \\ &\leq 1 + \|H\| + \cdots + ((k-1)n)^{(k-1)/2}\|H\|^{k-1} \\ &\leq 1 + \|f\| + \cdots + ((k-1)n)^{(k-1)/2}\|f\|^{k-1} \\ &\leq \left((\sqrt{(k-1)n}\|f\|)^k - 1\right) / \left((\sqrt{(k-1)n}\|f\|) - 1\right) \end{aligned}$$

Since $\|f\| < (\sqrt{(k-1)n}\|f\|) - 1$, we have $\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot (\sqrt{(k-1)n}\|f\|)^k)$.

□

Undoubtedly there are suitable $f(x)$ that do not fall into the class of polynomials above. For example, let $a_1, \ldots, a_k, b_1, \ldots, b_k$ be polynomials, such that for each i, $a_i = x^{r_i} - 1$ and $b_i = (1-x^{r_i s_i})/(1-x^{r_i})$ for some $\{r_i\}, \{s_i\}$ where $r_i \cdot s_i \geq n-1$ and $r_i < n-1$. Then, for each i, $a_i b_i = 1 \bmod x^{n-1}$ (nontrivially) and $\|a_i\|$ and $\|b_i\|$ are both quite small. We could set $F(x)$ and $g(x)$ by picking a random subset $S \subseteq \{1, \ldots, k\}$ and setting $F(x) \leftarrow \prod_{i \in S} a_i \bmod x^{n-1}$ and $g(x) \leftarrow \prod_{i \in S} b_i \bmod x^{n-1}$. The Euclidean norms of F and g would be rather small, since the Euclidean norms of the a_i's and b_i's were very small. This technique seems messier than the approach above; the point here is that the approach above is not the only approach.

A simple case is to set $f(x) \leftarrow x^n - 1$. For the ring $R = \mathbb{Z}[x]/(x^n - 1)$, it is easy to show that $\gamma_{\mathsf{Mult}}(R) \leq \sqrt{n}$.

Lemma 7.4.3. *Suppose* $\mathbf{x}, \mathbf{y} \in R = \mathbb{Z}[x]/(x^n - 1)$*, and let* $\mathbf{z} \leftarrow \mathbf{x} \times \mathbf{y}$*. Then* $\|\mathbf{z}\| \leq \sqrt{n} \cdot \|\mathbf{x}\| \cdot \|\mathbf{y}\|$.

Proof. Consider the i-th coefficient z_i of $\mathbf{z}$; we have

$$z_i = \sum_j x_j \cdot y_{i-j \bmod n}$$

In particular, since z_i is an inner product of (rotated versions of) $\mathbf{x}$ and $\mathbf{y}$, we have that $|z_i| \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\|$ (for all i). The result follows.

□

However, such *circulant* ideal lattices come with the disclaimer, mentioned in Section 6.2, that there are non-fatal but somewhat disconcerting attacks on hard problems over this particular ring.

We also prefer $f(x)$ to be irreducible, so that $K = \mathbb{Q}(x)/(f(x))$ is a field. In this case, $\mathbb{Z}[x]/(f(x))$ inherits the nice properties of its overlying ring of integers $\mathcal{O}_K$, with some qualifications. (See Section 6.) Using irreducible $f(x)$ also seems to make R less vulnerable to cryptanalytic attacks, such as that in [47]. If desired, we can get many of the benefits of using $\mathbb{Z}[x]/(x^n - 1)$ by instead using $\mathbb{Z}[x]/(f(x))$ for $f(x) = (x^n - 1)/(x - 1)$, which is irreducible when n is prime.

7.5 Instantiating Encrypt and Minimizing r_{Enc}

From Theorem 7.3.2, we would like to set r_{Enc} to be as small as possible, consistent with security. Recall that $X_{\mathsf{Enc}} \subseteq \mathcal{B}(r_{\mathsf{Enc}})$ is the image of the Samp algorithm used in $\mathsf{Encrypt}$, where our security proof (Theorem 5.3.1) holds when $\mathsf{Samp}(\mathbf{B}_I, \mathbf{x})$ runs $\mathbf{r} \leftarrow \mathsf{Samp}_1(R)$ and outputs $\mathbf{x} + \mathbf{r} \times \mathbf{s}$, where $\mathbf{s}$ is a generator of the ideal I. Let ℓ_{Samp_1} be an upper bound on the length of $\mathbf{r}$, drawn according to Samp_1. We have

$$r_{\mathsf{Enc}} = \max\{\|\mathbf{x} + \mathbf{r} \times \mathbf{s}\|\} \leq n \cdot \|\mathbf{B}_I\| + \sqrt{n} \cdot \ell_{\mathsf{Samp}_1} \cdot \|\mathbf{B}_I\|$$

Toward minimizing r_{Enc}, we can choose $\mathbf{s}$ to be short – e.g., use $\mathbf{s} = 2 \cdot \mathbf{e}_1$.

The size of ℓ_{Samp_1} is a security issue. We need it to be large enough so that the min-entropy of $\mathbf{t}$ mod $\mathbf{B}_J^{\mathrm{pk}}$ in the ICP is large. As a concrete example, one could set $\ell_{\mathsf{Samp}_1} = n$, and have Samp_1 sample a uniformly random integer vector in $\mathcal{B}(\ell_{\mathsf{Samp}_1})$.

Overall, we can take r_{Enc} to be polynomial in n. We note that, even in this case, the plaintext space may be as large as $[R : I] = \det(I)$, which can be exponential in n.

There are certainly alternative ways of generating I and instantiating Samp. For example, one may set $\mathbf{s}$ in such a way that the Hermite normal form of $(\mathbf{s})$ has all 1's along the diagonal, except for the upper-left corner, which equals $\det(I)$. (This property of the Hermite normal form will always hold when $\det(I)$ is prime.) This gives a plaintext space isomorphic to $\mathbb{Z}_{\det(I)}$, which may be more useful than the space $\mathbb{Z}_2^n$ for some application. Also, the image of Samp is not necessarily very "nice" – e.g., it may not be "spherical," but may rather be distorted in a way that depends on the ring R. In Section 14.1, we discuss a different way to instantiate Samp is using Gaussian distributions over lattices.

7.6 Instantiating Decrypt and Maximizing r_{Dec}

From Theorem 7.3.2, we would like to set r_{Dec} to be as large as possible, consistent with security. Recall that r_{Dec} is the radius of the largest sphere centered at $\mathbf{0}$ that is circumscribed by $\mathbf{B}_J^{\mathrm{sk}}$. Also, recall our decryption equation.

$$\pi = \psi - \mathbf{B}_J^{\mathrm{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi \rceil \bmod \mathbf{B}_I$$

To maximize r_{Dec}, one strategy is simply to scale up the parallelepiped $\mathbf{B}_J^{\mathrm{sk}}$. But this does

not really buy us anything. For a fixed ratio $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$, one can verify that our maximum depth (per Theorem 7.3.2) of $\log\log r_{\mathsf{Dec}} - \log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$ *decreases* as we scale up r_{Dec} and r_{Enc} simultaneously. (If we scale up r_{Dec} without scaling up r_{Enc}, this increases the approximation factor of the associated bounded distance decoding lattice problem, which hurts security. See Section 7.7.) The important property of $\mathbf{B}_J^{\mathrm{sk}}$ is its shape – i.e., we want the parallelepiped $\mathcal{P}(\mathbf{B}_J^{\mathrm{sk}})$ to be "fat" enough to contain a large sphere. This property is easier to formalize in terms of the inverse matrix $(\mathbf{B}_J^{\mathrm{sk}})^{-1}$, whose transpose is a basis (or independent set) of the dual lattice $\mathcal{L}(\mathbf{B}_J^{\mathrm{sk}})$.

Lemma 7.6.1. *Let $\mathbf{B}$ be a lattice basis and $\mathbf{B}^* = (\mathbf{B}^{-1})^T$. Let r be the radius of the largest sphere, centered at $\mathbf{0}$, circumscribed by $\mathcal{P}(\mathbf{B})$ (permitting tangential overlap). Then, $r = 1/(2 \cdot \|\mathbf{B}^*\|)$. In particular,*

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

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

Proof. Suppose $\|\mathbf{x}\| < 1/(2 \cdot \|\mathbf{B}^*\|)$. Each coefficient of $\mathbf{B}^{-1} \cdot \mathbf{x}$ is an inner product of $\mathbf{x}$ with a column vector of $\mathbf{B}^*$, and therefore has magnitude at most $\|\mathbf{x}\| \cdot \|\mathbf{B}^*\| < 1/2$. This implies that $\lfloor \mathbf{B}^{-1} \cdot \mathbf{x} \rceil = 0$, that $\mathbf{x} = (\mathbf{x} \bmod \mathbf{B})$, and that $\mathbf{x} \in \mathcal{P}(\mathbf{B})$. Now, suppose $\|\mathbf{x}\| > 1/(2 \cdot \|\mathbf{B}^*\|)$ and is parallel to the longest vector $\mathbf{b}_i$ in $\mathbf{B}^*$. Then, $|\langle \mathbf{b}_i, \mathbf{x} \rangle| > 1/2$, implying that $\lfloor \mathbf{B}^{-1} \cdot \mathbf{x} \rceil \neq 0$, and that $\mathbf{x} \neq (\mathbf{x} \bmod \mathbf{B})$, and that $\mathbf{x} \notin \mathcal{P}(\mathbf{B})$.

□

The relevance of Lemma 7.6.1 is that the decryption equation above is correct when ψ is at most $r_{\mathsf{Dec}} = 1/(2 \cdot \|((\mathbf{B}_J^{\mathrm{sk}})^{-1})^T\|)$ away from a lattice point in J.

It is easy to imagine ad hoc ways of instantiating $\mathsf{IdealGen}$ so that the parallelepiped $\mathcal{P}(\mathbf{B}_J^{\mathrm{sk}})$ is "fat" – i.e., contains a sphere whose radius is only polynomially shorter than the parallelepiped's diameter. For example, one could generate a random vector $\mathbf{v}$ and simply set $\mathbf{B}_J^{\mathrm{sk}}$ to be the rotation basis of $\mathbf{v}$, and set $\mathbf{B}_J^{\mathrm{pk}}$ to be the HNF of $(\mathbf{v})$. Very roughly speaking, if $\mathbf{v}$ is generated as a vector that is very "nearly parallel" to $\mathbf{e}_1$ (i.e., the vector $(1, 0, \ldots, 0)$), then the rotational basis will have r_{Dec} within a small (polynomial) factor of $\lambda_1(J)$. More formally, we have the following lemma.

Lemma 7.6.2. *Let $t \geq 4 \cdot n \cdot \gamma_{\mathsf{Mult}}(R) \cdot s$. Suppose $\mathbf{v} \in t \cdot \mathbf{e}_1 + \mathcal{B}(s)$ – i.e., $\mathbf{v}$ is in the ball of radius s centered at $t \cdot \mathbf{e}_1$. Let $\mathbf{B}$ be the rotation basis of $\mathbf{v}$. Then, $\mathcal{P}(\mathbf{B})$ circumscribes a*

ball of radius at least $t/4$.

Proof. For $i \in [0, n-1]$, let $\mathbf{v}_i = \mathbf{v} \times x^i$, and $\mathbf{z}_i = \mathbf{v}_i - t \cdot \mathbf{e}_i$. We have that $\|\mathbf{z}_i\| = \|\mathbf{z}_0 \times x^i\| \leq \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{z}_0\| \leq \gamma_{\mathsf{Mult}}(R) \cdot s$. (In other words, we have that $\mathbf{v}_i = t \cdot \mathbf{e}_1 + \mathbf{z}_i$ is nearly parallel to $\mathbf{e}_i$ when $\gamma_{\mathsf{Mult}}(R) \cdot s$ is much smaller than t.)

For every point $\mathbf{a}$ on the surface of $\mathcal{P}(\mathbf{B})$, there is an i such that

$$\mathbf{a} = (\pm 1/2) \cdot \mathbf{v}_i + \sum_{j \neq i} x_j \cdot \mathbf{v}_j$$

for $x_j \in [-1/2, 1/2]$. So,

$$|\langle \mathbf{a}, \mathbf{e}_i \rangle| \geq t/2 - n \cdot \gamma_{\mathsf{Mult}}(R) \cdot s$$

In particular, $\|\mathbf{a}\| \geq t/2 - n \cdot \gamma_{\mathsf{Mult}}(R) \cdot s$ and the lemma follows.

□

Perhaps lattice problems over principal ideal lattices generated in the above ad hoc fashion are easy, though currently no efficient attacks are known. A "better" instantiation of $\mathsf{IdealGen}$, which permits a security reduction from worst-case SIVP, is given in Section 18.

7.7 Security of the Concrete Scheme

When instantiated with ideal lattices, the ideal coset problem (ICP) becomes the following problem.

Definition 7.7.1 ((Decision) Bounded Distance Decoding Problem (Decision BDDP) for Ideal Lattices)**.** Fix a polynomial ring $R = \mathbb{Z}[x]/(f(x))$, algorithm $\mathsf{IdealGen}$ that samples a basis of an ideal in R, and an algorithm Samp_1 that efficiently samples $\mathbb{Z}^n$. The challenger sets $b \xleftarrow{R} \{0,1\}$ and $(\mathbf{B}_J^{\mathrm{sk}}, \mathbf{B}_J^{\mathrm{pk}}) \xleftarrow{R} \mathsf{IdealGen}(R, \mathbf{B}_I)$. If $b = 0$, it sets $\mathbf{r} \xleftarrow{R} \mathsf{Samp}_1(R)$ and $\mathbf{t} \leftarrow \mathbf{r} \bmod \mathbf{B}_J^{\mathrm{pk}}$. If $b = 1$, it samples $\mathbf{t}$ uniformly from $R \bmod \mathbf{B}_J^{\mathrm{pk}}$. The problem: guess b given $(\mathbf{t}, \mathbf{B}_J^{\mathrm{pk}})$.

In short, the problem is to decide whether $\mathbf{t}$ is uniform modulo the ideal lattice J, or whether $\mathbf{t}$ was sampled according to a known "clumpier" distribution induced by Samp_1.

Obviously, the hardness of decision BDDP depends crucially on Samp_1 – i.e., decision BDDP is an average-case problem whose hardness depends on the (average-case) distribution of Samp_1. For example, if $\mathsf{Samp}_1(R)$ always output the zero vector $\mathbf{0}$, or sampled according to some other distribution with very low min-entropy, the problem would be easy. However, based on current knowledge, it seems reasonable to believe the problem can be hard when Samp_1's min-entropy is high – e.g., when $\mathbf{r}$ is sampled from a sphere of radius n, or when $\mathbf{r}$ is sampled according to a discrete n-dimensional Gaussian distribution with a standard deviation parameter $s = \omega(\sqrt{\log n})$. We defer details regarding discrete Gaussian distributions until Section 13; for now, as a concrete example, let's suppose that $\mathbf{r}$ is sampled uniformly from a sphere of radius $\ell_{\mathsf{Samp}_1} = n$.

The hardness of decision BDDP also depends on how J is generated – in particular, on the value $\lambda_1(J)$, and whether $\lambda_1(J)$ is much larger than ℓ_{Samp_1}. In particular, if $\lambda_1(J)/\ell_{\mathsf{Samp}_1} \geq 2^n$ (and we could replace the rhs with a *slightly* sub-exponential value), then Babai's nearest plane algorithm [13] or variants of the lattice reduction algorithm LLL [81] can be used to recover the closest J-vector to $\mathbf{t}$ in polynomial time. This attack breaks decision BDDP for these parameters, since it is a very safe bet that $\mathbf{t}$ was generated using Samp_1 when $\mathrm{dist}(J, \mathbf{t}) < \ell_{\mathsf{Samp}_1}$; if $\mathrm{dist}(J, \mathbf{t}) > \ell_{\mathsf{Samp}_1}$, it is a certain bet that $\mathbf{t}$ was generated uniformly. However, there are no known attacks when, for example, $\lambda_1(J) = 2^{O(\sqrt{n})}$ (and ℓ_{Samp_1} is as before).

Above, we suggested ways of instantiating the ring R, the algorithm Samp used in $\mathsf{Encrypt}$, and the algorithm $\mathsf{IdealGen}$ used in KeyGen. Let's reconsider these suggestions, and revisit the sizes of r_{Enc} and r_{Dec}, with a view to how they impact the hardness of the induced decision BDDP.

In Section 7.5, we observed that r_{Enc} is at most $n \cdot \|\mathbf{B}_I\| + \sqrt{n} \cdot \ell_{\mathsf{Samp}_1} \cdot \|\mathbf{B}_I\|$, where $\mathbf{B}_I$ can be chosen so that $\|\mathbf{B}_I\|$ is polynomial in n (or even constant). In short, we can have r_{Enc} only polynomially larger than ℓ_{Samp_1}. In Section 7.6, we observed that one can instantiate $\mathsf{IdealGen}$ so that it outputs a secret basis $\mathbf{B}_J^{\mathrm{sk}}$ for J such that, if r_{Dec} is the radius of the largest ball circumscribed by $\mathcal{P}(\mathbf{B}_J^{\mathrm{sk}})$, then r_{Dec} is only polynomially smaller than $\lambda_1(J)$. Overall, we can make $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$ be within a polynomial factor of $\lambda_1(J)/\ell_{\mathsf{Samp}_1}$, where the latter is essentially the approximation factor of our decision BDDP problem. As a rule of thumb, solving 2^k-approximate decision BDDP takes time roughly $2^{n/k}$ using known attacks; so, $r_{\mathsf{Dec}} = 2^{O(\sqrt{n})}$ and $r_{\mathsf{Enc}} = \mathrm{poly}(n)$ seems to be a reasonable setting of parameters. When $r_{\mathsf{Dec}} = 2^{n^{c_1}}$ and $\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}} = 2^{n^{c_2}}$, then Theorems 5.2.6 and 7.3.2

imply that the scheme can correctly evaluate circuits of depth $(c_1 - c_2) \log n$.

Remark 7.7.2. Setting r_{Dec} to be small permits a weaker assumption, but leads to a scheme that can evaluate only very shallow circuits. Let us suppose that $r_{\mathsf{Dec}} = n^{\alpha(n)}$ and $\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}} = n^{\beta(n)}$, for some functions $\alpha(n), \beta(n)$. As far as we know, for irreducible $f(x)$, $\gamma_{\mathsf{Mult}}(R)$ must be at least polynomial in n, so $\beta(n)$ must be at least constant. In this case, the scheme can evaluate depth $\log \alpha(n) - \log \beta(n)$. This implies that we can only evaluate constant depth circuits, unless $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$ is super-polynomial. Though we omit details here, constant depth will be insufficient to make our eventual scheme bootstrappable; bootstrappability will require the BDDP approximation factor to be super-polynomial.

Again, one may question how hard the decision BDDP actually is for our ad hoc instantiation of $\mathsf{IdealGen}$. In Section 6, we mentioned that Gentry and Szydlo [50] have a polynomial-time attack on *circulant* ideal lattices that have orthonormal bases. This attack suggests that we may want to avoid principal ideal lattices with "nearly orthonormal" bases even in non-cyclotomic polynomial rings. We provide an alternative $\mathsf{IdealGen}$ algorithm in Section 18, and provide a worst-case / average-case connection for $\mathsf{IdealGen}$'s distribution in Section 17.

We stress that our analysis below regarding the decryption circuit does not rely on the ad hoc concrete suggestions in this section – e.g., the analysis does not require I or J to be principal ideals.

7.8 How Useful is the Somewhat Homomorphic Scheme By Itself?

The momentum of our paper is directed toward obtaining a bootstrappable, and hence a (leveled) fully homomorphic, encryption scheme. However, we pause briefly to consider how we can use our somewhat homomorphic scheme even if we do not try to bootstrap.

Theorem 7.3.2 tells us that we can evaluate circuits of depth

$$\log \log r_{\mathsf{Dec}} - \log \log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$$

even if the $\mathsf{Add}_{\mathbf{B}_I}$ gates have high fan-in (i.e., $\gamma_{\mathsf{Mult}}(R)$ fan-in). We have seen above that we can take r_{Dec} to be of the form 2^{n^c} for some constant $c < 1$, and $\gamma_{\mathsf{Mult}}(R)$ and r_{Enc} to be polynomial in n. In this case, we can evaluate logarithmic depth.

Already this is a significant improvement on prior work. For example, the Boneh-Goh-Nissim (BGN) pairing-based cryptosystem [21] was the first to permit efficient evaluation of 2-DNF formulas, quadratic formulas that may have a polynomial number of monomials. Being able to compute quadratic formulas is extremely useful – e.g., Groth, Ostrovsky, and Sahai [63] used their system to construct a perfectly NIZK system for general circuits (with length proportion to the size of the circuit). However, one shortcoming of the BGN system is its small plaintext space – $\log \lambda$ bits for security parameter λ. Our somewhat homomorphic scheme, without the bootstrapping, already improves upon this, allowing both greater multiplicative depth in the circuit and a larger plaintext space.

As an example, we obtain the first single-database private information retrieval scheme with communication complexity $O(\lambda \cdot \log m)$, where λ is the security parameter and m is bit-length of the database $s_1, \ldots, s_m$. The querier encrypts the binary representation $\pi_1, \ldots, \pi_M$ of the index that it wants, constructing the ciphertexts $\psi_1, \ldots, \psi_M$, where $M = \lfloor \log m \rfloor + 1$. Homomorphically, the server homomorphically evaluates the formula

$$f(\pi_1, \ldots, \pi_M, s_1, \ldots, s_m) = \sum_{t \in \{0,1\}^M} s_t \cdot \prod_{j=1}^{M} (t_i - \pi_i + 1) \bmod 2$$

where, in s_t, t is interpreted as a number in $[1, m]$. Notice that this formula encrypts the correct entry in the database. Also, observe that if the ciphertexts $\psi_1, \ldots, \psi_M$ have offsets in $\mathcal{B}(r_{\mathsf{Enc}})$, then the offset of the output is in $\mathcal{B}(r)$ for $r = O(m \cdot (\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})^M) = O((2 \cdot \gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})^M)$. If one permits $r_{\mathsf{Dec}} = 2^{\theta(\sqrt{n})}$, then one can permit $M = \theta(\sqrt{n}/\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}}))$, which is polynomial in n. In other words, our scheme correctly evaluates the PIR formula even when the database is sub-exponential (super-polynomial) in size, though of course the computation would be very high in that case.

In general, when the function to be evaluated is highly parallel, the bootstrapping step may be unnecessary, permitting better efficiency.

Chapter 8

Tweaks to the Somewhat Homomorphic Scheme

At this point, we have described our somewhat homomorphic scheme in enough detail to begin considering whether the scheme is bootstrappable. First, however, we describe two "tweaks" to the scheme. The purpose of these tweaks is to lower the eventual circuit complexity of decryption without substantially reducing the depth that the scheme can evaluate.

As the first tweak, we modify the secret key of our scheme so that the decryption equation simplifies from

$$\pi = \psi - \mathbf{B}_J^{\mathrm{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi \rceil \bmod \mathbf{B}_I$$

to

$$\pi = \psi - \lfloor \mathbf{v}_J^{\mathrm{sk}} \times \psi \rceil \bmod \mathbf{B}_I$$

where $\mathbf{v}_J^{\mathrm{sk}} \in J^{-1}$.

Before describing the tweak, it is helpful to understand the relationship between the dual of a lattice (a good basis for which was originally used as the decryption key) and the inverse of an ideal lattice (a vector from which is used as the decryption key in our revised decryption equation).

8.1 On the Relationship between the Dual and the Inverse of an Ideal Lattice

Recall the definition of the dual of an ideal lattice J: $J^* = \{\mathbf{x} \in \mathbb{R}^n : \forall \mathbf{v} \in J,\ \langle \mathbf{x}, \mathbf{v} \rangle \in \mathbb{Z}\}$. The inverse in $R = \mathbb{Z}[x]/(f(x))$ of an ideal has a superficially similar definition: $J^{-1} = \{\mathbf{x} \in \mathbb{Q}[x]/(f(x)) : \forall \mathbf{v} \in J,\ \mathbf{x} \times \mathbf{v} \in \mathbb{R}\}$.

If $\mathbf{B}_J$ happens to be a rotation basis of $J = (\mathbf{v})$, then the inverse $J^{-1} = (1/\mathbf{v})$ is generated by the rotation basis of $1/\mathbf{v}$, the columns of $\mathbf{B}_J^{-1}$. However, the dual of J is generated by the inverse *transpose* of $\mathbf{B}_J$. So it is certainly not true in general that the ideal lattice associated to J^{-1} is generated by the dual of the ideal lattice associated to J.[1] However, for rotation bases, since the bases of the dual and the inverse are just transposes of each other, we have the following easy lemma, which is analogous to Lemma 7.6.1.

Lemma 8.1.1. *Let* $\mathbf{B}$ *be a rotation basis and* $\mathbf{B}^*$ *be its inverse transpose. Then,* $\|\mathbf{B}^*\| \cdot \sqrt{n} \geq \|\mathbf{B}^{-1}\| \geq \|\mathbf{B}^*\|/\sqrt{n}$. *In particular, if* $\mathbf{B}_J^{\text{sk}}$ *is a rotation basis, we have* $1/(2\sqrt{n} \cdot \|(\mathbf{B}_J^{\text{sk}})^{-1}\|) \leq r_{\mathsf{Dec}} \leq \sqrt{n}/(2 \cdot \|(\mathbf{B}_J^{\text{sk}})^{-1}\|)$.

Proof. Let b_{ij} be the highest-magnitude coefficient in the matrix $\mathbf{B}^{-1}$. Then,

$$\|\mathbf{B}^{-1}\| \geq b_{ij} \geq \|\mathbf{B}^*\|/\sqrt{n} \quad \text{and} \quad \|\mathbf{B}^*\| \geq b_{ij} \geq \|\mathbf{B}^{-1}\|/\sqrt{n}$$

Using Lemma 7.6.1, we have

$$1/(2\sqrt{n} \cdot \|(\mathbf{B}_J^{\text{sk}})^{-1}\|) \leq r_{\mathsf{Dec}} \leq \sqrt{n}/(2 \cdot \|(\mathbf{B}_J^{\text{sk}})^{-1}\|)$$

□

Can we provide a more precise characterization of this relationship between the dual and the inverse for general (non-principal) ideal lattices? For example, given a short vector in J^{-1}, can we find a short basis of J^*? Or, given a short vector in J^*, can we output a short basis of J^{-1}. The answer to both of these questions is yes.

Lemma 8.1.1 already answers the first question. Let $\mathbf{B}_J$ be a basis of J, with column vectors $\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}$. If $\mathbf{v}$ is a short vector in J^{-1} and $\mathbf{B}_\mathbf{v}$ is its rotation basis, then $\mathbf{v} \times \mathbf{u}_i \in R$ for all i, and therefore $\mathbf{B}_\mathbf{v} \cdot \mathbf{B}_J$ is an integer matrix. This implies that the rows of $\mathbf{B}_\mathbf{v}$ form

[1] Contrary to what we stated in [48]. Lyubashevsky [87] indicated this error.

an independent set in J^*. The longest row of $\mathbf{B_v}$ cannot be much longer than the longest column, as in the proof of Lemma 7.6.1.

The second question – i.e., whether one can generate a short basis of J^{-1} from a short vector in J^* is more challenging, but we have the following lemma.

Lemma 8.1.2. *Let $\mathbf{w} \in J^*$, where J^* is the dual of the ideal lattice J. Let*

$$\mathbf{v} = \sum_{i=0}^{n-1} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1}$$

Then, $\mathbf{v} \in J^{-1}$. Let $\mathbf{B_v}$ be the rotation basis of $\mathbf{v}$. Then, $\|\mathbf{B_v}\| \leq \sqrt{n} \cdot \|f\| \cdot \|\mathbf{w}\|$. This applies even when J is a fractional ideal.

The idea of the proof is to take $\mathbf{w} \in J^*$, place it as the bottom row in an $n \times n$ matrix, and then to try to fill out the rest of the matrix so that we end up with the rotation basis of a vector in J^{-1}. Together, the vector $\mathbf{w}$ and the polynomial $f(x)$ dictate what the rest of the matrix must be.

Proof. We claim that the bottom row of $\mathbf{B_v}$ is $(w_0, w_1, \ldots, w_{n-1})$. In other words, in some sense, one can view $\mathbf{B_v}$ as an "extension" of the single row $(w_0, w_1, \ldots, w_{n-1})$ into an entire matrix that happens to be a rotation basis.

Denote the columns of $\mathbf{B_v}$ by $\mathbf{v}^{(k)} = \mathbf{v} \cdot x^i \bmod f(x)$. We claim that

$$\mathbf{v}^{(k)} = \sum_{i=k}^{n-1} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1+k} - \sum_{i=0}^{k-1} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k}$$

from which it follows that the coefficient of x^{n-1} in $\mathbf{v}^{(k)}$ is indeed w_k. This is clearly true

for $k = 0$; assume it is true for $k' - 1$. We have that

$$\begin{aligned}
\mathbf{v}^{(k')} &= x\left(\sum_{i=k'-1}^{n-1} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1+k'-1} - \sum_{i=0}^{k'-2} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k'-1}\right) \bmod f(x) \\
&= \sum_{i=k'}^{n} x^i \sum_{j=i}^{n} f_j \cdot w_{j-i-1+k'} - \sum_{i=1}^{k'-1} x^i \sum_{j=0}^{i-1} f_j \cdot w_{j-i-1+k'} \bmod f(x) \\
&= \sum_{i=k'}^{n} x^i \sum_{j=i}^{n} f_j \cdot w_{j-i-1+k'} - \sum_{i=1}^{k'-1} x^i \sum_{j=0}^{i-1} f_j \cdot w_{j-i-1+k'} - (f_n \cdot w_{k'-1}) \cdot f(x) \\
&= \sum_{i=k'}^{n} x^i \sum_{j=i}^{n} f_j \cdot w_{j-i-1+k'} - \sum_{i=1}^{k'-1} x^i \sum_{j=0}^{i-1} f_j \cdot w_{j-i-1+k'} - \sum_{i=0}^{n} x^i \cdot w_{k'-1} \cdot f_i \\
&= \sum_{i=k'}^{n} x^i \left(-f_i \cdot w_{k'-1} + \sum_{j=i}^{n} f_j \cdot w_{j-i-1+k'}\right) \\
&\qquad - \sum_{i=1}^{k'-1} x^i \left(f_i \cdot w_{k'-1} + \sum_{j=0}^{i-1} f_j \cdot w_{j-i-1+k'}\right) - w_{k'-1} \cdot f_i \\
&= \sum_{i=k'}^{n} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1+k'} - \sum_{i=1}^{k'-1} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k'} - w_{k'-1} \cdot f_i
\end{aligned}$$

as required.

To show that $\mathbf{v} \in J^{-1}$, it suffices to prove that $\mathbf{z} \leftarrow \mathbf{v} \times \mathbf{x} \in R$ for any $\mathbf{x} \in J$. Let $\mathbf{B_x}$ and $\mathbf{B_z}$ be the rotation bases of $\mathbf{x}$ and $\mathbf{z}$. We know that $\mathbf{B_z} = \mathbf{B_v} \cdot \mathbf{B_x}$. We also know that the bottom row of $\mathbf{B_z}$ is an integer vector, since this row is $\mathbf{w} \cdot \mathbf{B_x}$ and $\mathbf{w}$ has an integer inner product with all vectors in J (which includes the column vectors of $\mathbf{B_x}$).

Assume, toward a contradiction that $\mathbf{z}$ is not an integer vector – in particular, that i^* is the largest such that the coefficient z_{i^*} is not an integer. Consider $\mathbf{z}^{(n-i^*-1)} \leftarrow x^{n-i^*-1} \cdot \mathbf{z} \bmod f(x)$, which is a column vector in $\mathbf{B_z}$. In $x^{n-i^*-1} \cdot \mathbf{z}$, the coefficients of x^n through x^{2n-i^*-2} – all of the highest coefficients – are integers. Therefore, since $f(x)$ is monic, $\mathbf{z}^{(n-i^*-1)} = x^{n-i^*-1} \cdot \mathbf{z} - a(x) \cdot f(x)$, where $a(x)$ is an integer polynomial. On the other hand, the coefficient of x^{n-1} in $x^{n-i^*-1} \cdot \mathbf{z}$ is not an integer, since z_{i^*} is not an integer. Consequently, since $\mathbf{z}^{(n-i^*-1)}$ differs from $x^{n-i^*-1} \cdot \mathbf{z}$ by an integer polynomial, the coefficient of x^{n-1} in $\mathbf{z}^{(n-i^*-1)}$ is also not an integer. But we have established that the bottom row of $\mathbf{B_z}$ is integral, a contradiction. Therefore, $\mathbf{z}$ is in R and $\mathbf{v} \in J^{-1}$.

Regarding $\|\mathbf{B_v}\|$, we have established that each entry of this matrix is an inner product of two vectors – one vector with coefficients in $\{f_0, \ldots, f_n\}$, the other with coefficients in $\{w_0, \ldots, w_{n-1}\}$ (up to sign). The magnitude of each coefficient in $\mathbf{B_v}$ is therefore at most $\|f\| \cdot \|\mathbf{w}\|$, implying that $\|\mathbf{B_v}\| \leq \sqrt{n} \cdot \|f\| \cdot \|\mathbf{w}\|$.

□

8.2 Transference Lemmas for Ideal Lattices

As an easily corollary, we can obtain a bound on the determinant of J^{-1} in terms of $\det(J)$, and also place a bound on $\lambda_n(J^{-1})$ in terms of $\lambda_n(J)$. Not all ideals are "invertible" in the sense that it is not always the case that $J^{-1} \cdot J = R$. (See Section 13.4 for more details on this.) But we bound the discrepancy in the following lemma.

Lemma 8.2.1. *Let J be a (possibly fractional) ideal of $R = \mathbb{Z}[x]/(f(x))$. Then, $\lambda_n(J^{-1}) \leq \sqrt{n} \cdot \|f\| \cdot \lambda_1(J^*) \leq n \cdot \|f\|/\lambda_n(J)$. Also, $\det(J^{-1}) < n^n \cdot \|f\|^n / \det(J)$.*

Proof. Let $\mathbf{w}$ be a vector in J^* of length $\lambda_1(J^*)$. Generate $\mathbf{v} \in J^{-1}$ from $\mathbf{w} \in J^*$ as in Lemma 8.1.2, and let $\mathbf{B_v}$ be its rotation basis. By Lemma 8.1.2, $\|\mathbf{B_v}\| \leq \sqrt{n} \cdot \|f\| \cdot \|\mathbf{w}\|$. By the transference theorem $\lambda_1(L) \cdot \lambda_n(L^*) \leq \sqrt{n}$ for general lattices, we have that $\|\mathbf{w}\| \leq \sqrt{n}/\lambda_n(J)$, which implies the first statement. Since $\det(J^*) = 1/\det(J)$, $\|\mathbf{w}\| \leq \sqrt{n}/\det(J)^{1/n}$ by Minkowski, we have $\det(\mathbf{B_v}) \leq n^n \cdot \|f\|^n / \det(J)$. □

Using Lemma 8.2.1, we can upper bound $\lambda_n(J)$ in terms of n, $|f|$ and $\det(J)$.

Lemma 8.2.2. *Let J be an ideal of $R = \mathbb{Z}[x]/(f(x))$. Then, $\lambda_n(J) < n \cdot \|f\| \cdot \det(J)^{1/n}$.*

Proof. We have

$$\begin{aligned} \lambda_n(J) &\leq n \cdot \|f\|/\lambda_n(J^{-1}) \quad \text{(by Lemma 8.2.1)} \\ &\leq n \cdot \|f\|/\det(J^{-1})^{1/n} \\ &\leq n \cdot \|f\| \cdot \det(J)^{1/n} \end{aligned}$$

□

We have a similar result regarding the product of two general ideals (not necessarily inverses of each other).

Lemma 8.2.3. *Let J and K be two (possibly fractional) ideals of R. Then, $\lambda_n(JK) < n \cdot \|f\|(\det(J) \cdot \det(K))^{1/n}$. Also, $\det(JK) \leq n^n \cdot \|f\|^n \cdot \det(J) \cdot \det(K)$.*

Proof. This would follow trivially from Lemma 8.2.2, except that it is possible that $\det(J \cdot K) > \det(J) \cdot \det(K)$ when J and K are divisible by singular primes (see Chapter 13.4).

By Lemma 8.2.1, we have that

$$\lambda_n(JK) \leq \sqrt{n} \cdot \|f\| \cdot \lambda_1(((JK)^{-1})^*)$$

The determinant of the latter ideal is at most $\det(J) \cdot \det(K)$, since, in general, $\det(I_1 \cdot I_2) \geq \det(I_1) \cdot \det(I_2)$ and $\det(I) \cdot \det(I^{-1}) \geq 1$ (see Chapter 13.4). So, by Minkowski, $\lambda_n(JK) < n \cdot \|f\|(\det(J) \cdot \det(K))^{1/n}$.

By Lemma 8.2.1, we have that $\det(JK) \cdot \det((JK)^{-1}) < n^n \cdot \|f\|^n$. So, we have

$$\begin{aligned} n^n \cdot \|f\|^n &\geq \det(JK) \cdot \det((JK)^{-1}) \\ &\geq \det(JK) \cdot \det(J^{-1}) \cdot \det(K^{-1}) \\ &\geq \det(JK) \cdot \det(J^*) \cdot \det(K^*) \end{aligned}$$

from which the result follows. □

8.3 Tweaking the Decryption Equation

Having characterized the relationship between the inverse and the dual, we return to our first tweak.

Tweak 1: From $\mathbf{B}_I$ and secret key $\mathbf{B}_J^{\text{sk}}$, compute a certain short $\mathbf{v}_J^{\text{sk}} \in J^{-1}$ and redefine decryption to output $\pi = \psi - \lfloor \mathbf{v}_J^{\text{sk}} \times \psi \rceil \bmod \mathbf{B}_I$. Also, redefine $\mathcal{C}_{\mathcal{E}}$, so that it instead uses $\mathcal{B}(r_{\mathsf{Dec}}/(n^{2.5} \cdot \|f\| \cdot \|\mathbf{B}_I\|)$ instead of $\mathcal{B}(r_{\mathsf{Dec}})$.

Purpose: To simplify the decryption equation and improve computational efficiency.

This tweak is not actually essential, since matrix-vector multiplication is just as parallelizable as ring multiplication – i.e., the circuits have essentially the same depth. However, the tweak reduces the size of our secret key. This will help reduce the computational complexity of decryption (and, thus, the computational complexity of the homomorphic decryption step in bootstrapping). Essentially, it makes the already shallow decryption circuit less wide.

Tweak 1 requires us to reduce the permitted distance of ciphertexts from the J-lattice. But it does not affect our maximum evaluation depth very much when $|f|$ and $\|\mathbf{B}_I\|$ are only polynomial in n, and $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$ is super-polynomial (as it will need to be to make our scheme bootstrappable).

Toward understanding how this simplification works, suppose that it is the case that $\mathbf{B}_J^{\mathrm{sk}}$ is the rotation basis for some vector $\mathbf{w}_J^{\mathrm{sk}} \in \mathbb{Z}[x]/(f(x))$. Let $\mathbf{x}_J^{\mathrm{sk}} = 1/\mathbf{w}_J^{\mathrm{sk}} \in \mathbb{Q}[x]/(f(x))$. Then, since the rotation basis of $\mathbf{x}_J^{\mathrm{sk}}$ is precisely $(\mathbf{B}_J^{\mathrm{sk}})^{-1}$, and by properties of rotation bases (see Chapter 6.2) we have that

$$\pi = \psi - \mathbf{B}_J^{\mathrm{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi \rceil \bmod \mathbf{B}_I = \psi - \mathbf{w}_J^{\mathrm{sk}} \times \lfloor \mathbf{x}_J^{\mathrm{sk}} \times \psi \rceil \bmod \mathbf{B}_I$$

As for generating the initial $\mathbf{B}_J^{\mathrm{sk}}$ as a rotation basis, for now we just mention that the ad hoc instantiation of $\mathsf{IdealGen}$ given in Chapter 7.6 suffices. However, as the lemmas below establish, Tweak 1 works even when $\mathbf{B}_J^{\mathrm{sk}}$ is not a rotation basis.

Lemma 8.3.1. *Let $\mathbf{B}_J^{\mathrm{sk}}$ be an initial secret basis that decrypts correctly for parameter r_{Dec}. From $\mathbf{B}_J^{\mathrm{sk}}$ and $\mathbf{B}_I$, we can compute in polynomial time a vector $\mathbf{v}_J^{\mathrm{sk}} \in J^{-1}$ such that the rotation basis of $1/\mathbf{v}_J^{\mathrm{sk}}$ circumscribes a ball of radius at least $r_{\mathsf{Dec}}/(n^{2.5} \cdot \|f\| \cdot \|\mathbf{B}_I\|)$. In particular, if ψ is a valid ciphertext according to Tweak 1, in the sense that it equals $\pi + i + j$ for plaintext π, $i \in I$, $j \in J$, and $\pi + i \in \mathcal{B}(r_{\mathsf{Dec}}/(n^{2.5} \cdot \|f\| \cdot \|\mathbf{B}_I\|))$, then $\pi = \psi - (\mathbf{v}_J^{\mathrm{sk}})^{-1} \times \lfloor \mathbf{v}_J^{\mathrm{sk}} \times \psi \rceil \bmod \mathbf{B}_I$. For our particular value of $\mathbf{v}_J^{\mathrm{sk}} \in J^{-1}$, it will also hold that $\pi = \psi - \lfloor \mathbf{v}_J^{\mathrm{sk}} \times \psi \rceil \bmod \mathbf{B}_I$.*

Proof. Since $\mathbf{B}_J^{\mathrm{sk}}$ be an initial secret basis that decrypts correctly for parameter r_{Dec}, Lemma 7.6.1 tells us that $\|((\mathbf{B}_J^{\mathrm{sk}})^{-1})^T\| \leq 1/2r_{\mathsf{Dec}}$. Let $\mathbf{w} \in J^*$ be a vector in this basis. By Lemma 8.1.2, we can use $\mathbf{w}$ to generate a vector $\mathbf{x} \in J^{-1}$ whose rotation basis $\mathbf{B}_{\mathbf{x}}$ has length at most $\sqrt{n} \cdot \|f\| \cdot \|\mathbf{w}\| \leq \sqrt{n} \cdot \|f\|/2r_{\mathsf{Dec}}$. From $\mathbf{B}_{\mathbf{x}}$ and a vector in I of length at most $\|\mathbf{B}_I\|$, we can generate an independent set $\mathbf{B}_{J^{-1}I}$ of $(\mathbf{x}) \cdot I \subset J^{-1}I$ of length at most $\sqrt{n} \cdot \|\mathbf{B}_{\mathbf{x}}\| \cdot \|\mathbf{B}_I\| \leq n \cdot \|f\| \cdot \|\mathbf{B}_I\|/2r_{\mathsf{Dec}}$. We set $\mathbf{v}_J^{\mathrm{sk}} \leftarrow \mathbf{e}_1 \bmod \mathbf{B}_{J^{-1}I}$. It has length at most $n^2 \cdot \|f\| \cdot \|\mathbf{B}_I\|/2r_{\mathsf{Dec}}$.

Let $\mathbf{B}_J^{\dagger}$ be the rotation basis of $(\mathbf{v}_J^{\mathrm{sk}})^{-1}$; we want to prove that this basis can be used as the secret key for ciphertexts that are valid according to Tweak 1. Certainly $\mathbf{B}_J^{\dagger}$ fulfills the requirement of generating a super-lattice of J, since $\mathbf{v}_J^{\mathrm{sk}}$ generates a sub-lattice of J^{-1}. It remains to show that a large enough sphere is circumscribed by $\mathbf{B}_J^{\dagger}$. Let r'_{Dec} be the radius

of the largest such sphere. We have

$$r'_{\mathsf{Dec}} \geq 1/(2\sqrt{n} \cdot \|(\mathbf{B}_J^{\dagger})^{-1}\|) \geq r_{\mathsf{Dec}}/(n^{2.5} \cdot \|f\| \cdot \|\mathbf{B}_I\|)$$

where the first inequality follows from Lemma 8.1.1, and the second substitutes in the upper bound on the length of the rotation basis for $\mathbf{v}_J^{\mathrm{sk}}$. The correctness of decryption with the new key follows.

However, now we need to establish that we can simply drop the $(\mathbf{v}_J^{\mathrm{sk}})^{-1}$ term in the decryption equation. Since I and J are relatively prime, there is a vector $\mathbf{j} \in J \cap (1+I)$. Such a $\mathbf{j}$ can be found efficiently using the Chinese remainder theorem and bases for I and J. Let $\mathbf{r} = \mathbf{j} \times \mathbf{v}_J^{\mathrm{sk}}$. Since $\mathbf{v}_J^{\mathrm{sk}} \in J^{-1}$, we have $\mathbf{r} \in R$. In fact, since $\mathbf{v}_J^{\mathrm{sk}} \in 1 + J^{-1}I$, we have $\mathbf{r} \in 1 + I$. Since, by the correctness of decryption, we know that $(\mathbf{v}_J^{\mathrm{sk}})^{-1} \times \lfloor \mathbf{v}_J^{\mathrm{sk}} \cdot \psi \rceil \in R$ (even though $\mathbf{v}_J^{\mathrm{sk}})^{-1}$ may not be in R, we have the following congruences modulo I:

$$\begin{aligned}(\mathbf{v}_J^{\mathrm{sk}})^{-1} \times \lfloor \mathbf{v}_J^{\mathrm{sk}} \cdot \psi \rceil \quad &= \mathbf{r} \times (\mathbf{v}_J^{\mathrm{sk}})^{-1} \times \lfloor \mathbf{v}_J^{\mathrm{sk}} \cdot \psi \rceil \\ &= \mathbf{j} \times \lfloor \mathbf{v}_J^{\mathrm{sk}} \cdot \psi \rceil \\ &= \lfloor \mathbf{v}_J^{\mathrm{sk}} \cdot \psi \rceil\end{aligned}$$

□

8.4 A Tweak to Reduce the Circuit Complexity of the Rounding Step in Decryption

Tweak 2 will actually be more critical than Tweak 1 for reducing the depth of our decryption circuit and enabling bootstrapping.

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

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

Remark 8.4.1. If using Tweak 1 and Tweak 2, then use $\mathcal{B}(r_{\mathsf{Dec}}/(2n^{2.5} \cdot \|f\| \cdot \|\mathbf{B}_I\|)$ in the redefinition of permitted circuits – i.e., a radius half as small as the one used in Tweak 1. For simplicity, in this Section, we will abuse notation and use r_{Dec} to refer to the value of the permitted radius before Tweak 2.

The purpose of the tweak will become clearer as we delve into the details of the decryption circuit. But, briefly, recall that $\mathsf{Decrypt}$ computes $\mathbf{B}_J^{\mathrm{sk1}} \cdot \lfloor (\mathbf{B}_J^{\mathrm{sk2}})^{-1} \cdot \psi \rceil$. (If Tweak 1 is used, then $\mathbf{B}_J^{\mathrm{sk1}}$ is just the identity matrix and $(\mathbf{B}_J^{\mathrm{sk2}})^{-1}$ is the rotation basis of $\mathbf{v}_J^{\mathrm{sk}}$.) If we permitted the coefficients of $(\mathbf{B}_J^{\mathrm{sk2}})^{-1} \cdot \psi$ to be very close to half-integers, we would need high precision to ensure correct rounding. However, after Tweak 2, we have the following lemma:

Lemma 8.4.2. *If ψ is a valid ciphertext after Tweak 2, then each coefficient of $(\mathbf{B}_J^{\mathrm{sk2}})^{-1} \cdot \psi$ is within $1/4$ of an integer.*

Proof. Observe that $\psi \in \mathcal{B}(r_{\mathsf{Dec}}/2) + J$. Let $\psi = \mathbf{x} + \mathbf{j}$ for $\mathbf{x} \in \mathcal{B}(r_{\mathsf{Dec}}/2)$ and $\mathbf{j} \in J$. We have $(\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi = (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \mathbf{x} + (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \mathbf{j}$, where the former term has coefficients of magnitude at most $1/4$ by Lemma 7.6.1 and the latter is an integer vector.

□

This fact will help us simplify our decryption circuit, and does not substantially impair the evaluative capacity of our scheme. The new maximum evaluation depth, per Theorem 7.3.2, is $\log\log(r_{\mathsf{Dec}}/2) - \log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$, which is less than the original amount by only a sub-constant additive factor.

Again, to use Tweaks 1 and 2 simultaneously, use $\mathcal{B}(r_{\mathsf{Dec}}/(2n^{2.5} \cdot \|f\| \cdot \|\mathbf{B}_I\|)$.

Chapter 9

Decryption Complexity of the Tweaked Scheme

To decrypt, we compute

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

where $\psi \in \mathbb{Z}^n$, $\mathbf{B}_J^{\mathrm{sk1}} \in \mathbb{Z}^{n \times n}$, $\mathbf{B}_J^{\mathrm{sk2}} \in \mathbb{Q}^{n \times n}$, and $\mathbf{B}_I$ is a basis of an ideal I of $R = \mathbb{Z}[x]/(f(x))$. From Tweak 2, we have the promise that the coefficients of $\mathbf{B}_J^{\mathrm{sk2}} \cdot \psi$ are all within $1/4$ of an integer. Optionally, Tweak 1 ensures that $\mathbf{B}_J^{\mathrm{sk1}}$ is the identity matrix and $\mathbf{B}_J^{\mathrm{sk2}}$ is a rotation matrix. How do we optimally express this computation as a circuit?

Let us split the computation into pieces – in particular, the following steps:

Step 1: Generate n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with sum $\mathbf{B}_J^{\mathrm{sk2}} \cdot \psi$.

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

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

We do not claim that this way of splitting up the computation leads to an optimal decryption circuit. But, we will eventually see that, thanks to Tweak 2, Step 3 can be done in constant depth using a circuit with polynomial fan-in addition gates. (In Theorem 7.3.2, we saw that constant fan-in multiplication gates were as bad as, or worse than, polynomial fan-in addition gates.) We will see that Step 2 requires a deep circuit, but that there is a way to squash this aspect of the computation. (See Chapter 10.) Step 1 could be done by multiplying the n columns of $\mathbf{B}_J^{\mathrm{sk2}}$ by the n coefficients of ψ. But our method for

squashing the decryption circuit will eliminate Step 1. So, we will concentrate on analyzing the complexity of Steps 2 and 3 in this Chapter.

To better understand the circuit complexity issues here, consider the problem of adding n numbers in $[0, 1)$, each one a fraction in $[0, 1)$ represented in binary with k bits of precision. As far as we know, this requires a constant fan-in boolean circuit of depth $\Omega(\log n + \log k)$. Here is a concrete example of such a circuit. First, we use the "3-for-2" trick (see Karp's [76] survey of parallel algorithms): given 3 numbers in binary representation, there is a constant-depth (say, depth c) boolean circuit that replaces these 3 numbers with 2 numbers having the same sum. (Essentially, one of the two numbers receives the XOR of the 3 addends, and the other number receives the carry bits.) Using this trick recursively, one can replace n numbers with 2 numbers having the same sum in depth approximately $c \cdot \log_{3/2} n$. As for adding the final two numbers, there is certainly no general guarantee that this can be done in constant depth. The problem is that the least significant bit of the addends could affect the most significant bit of the sum. One needs $\Omega(\log k)$ depth to ensure the final sum is computed correctly.

But suppose one is given the *promise* that the sum of the numbers is very close to an integer, and that one is only interested in computing this integer. In this case, we can eliminate all but $O(\log n)$ bits of precision in each of the n addends, and still obtain the correct result. This integer can be computed in $c \log_{3/2} n + O(\log \log n)$ depth; there is no longer any dependence on k. Indeed, this was the purpose of Tweak 2 – to obtain exactly this promise.

However, the $c \log_{3/2} n$ term is still problematic for us. We have seen that our somewhat homomorphic scheme can evaluate $O(\log n)$ depth, but where the hidden constant is less than 1, whereas the c induced by the 3-for-2 trick (combined with the constant $\log_{3/2} 2$) is certainly greater than 1, and thus prevents bootstrapping. Also, even after we apply our "squashing the decryption circuit" technique to make our scheme bootstrappable, a constant factor in the depth of the decryption circuit makes a huge difference in the performance and security of the scheme. Can we make this constant smaller?

Toward this goal, we compute the rounded sum using elementary symmetric polynomials. Roughly speaking, using symmetric polynomials eliminates some of the inefficiencies of the 3-for-2 technique. Also, although we have been saying (as shorthand) that we want to minimize the "depth" of the decryption circuit $D_{\mathcal{E}}$, this is an oversimplification; we are actually trying to minimize $\|D_{\mathcal{E}}(\mathbf{x}_1, \ldots, \mathbf{x}_t)\|$ where the inputs $\mathbf{x}_i$ are in $\mathcal{B}(r_{\mathsf{Enc}})$. The value

$\|D_{\mathcal{E}}(\mathbf{x}_1, \ldots, \mathbf{x}_t)\|$ is actually more tightly related to the *degree* of the multivariate polynomial $D_{\mathcal{E}}(\mathbf{x}_1, \ldots, \mathbf{x}_t)$ than to the depth of the circuit that computes this polynomial. Elementary symmetric polynomials are the lowest-degree multivariate polynomials (that we know of) that compute certain Hamming weights that arise when computing the sum of numbers.

What do elementary symmetric polynomials have to do with adding up n numbers, represented in binary? Let $\{a_i\}$ be the n numbers, where a_i has bits $(a_{i,-1}, \ldots, a_{i,-T})$. We can add up these numbers by separately adding up the least significant bits of the numbers, the penultimate bits, etc., and thereafter combining the partial results. That is, for $j \in [-1, -T]$, we compute the Hamming weight b_j, represented in binary, of $(a_{1,j}, \ldots, a_{n,j})$, and then we add up the T numbers b_j. (We established above that the precision T only needs to be logarithmic in n, so this final step should take up much less depth than computing the binary representations b_j of the Hamming weights.) Now, it turns out, through the magic of binomial coefficients, that the binary representation of the Hamming weight of $(x_1, \ldots, x_n)$ is given by

$$(e_{2^{\lfloor \log n \rfloor}}(x_1, \ldots, x_n) \bmod 2, \ldots, e_{2^0}(x_1, \ldots, x_n) \bmod 2)$$

where $e_i(x_1, \ldots, x_n)$ is the ith elementary symmetric polynomial over $x_1, \ldots, x_n$. (See Lemma 4 of [24].) The highest degree among these polynomials is at most n, versus the multivariate polynomial we would obtain from the 3-for-2 trick, which has degree n^c for some $c > 1$. Also, we know how to efficiently evaluate the elementary symmetric polynomials. They are simply coefficients of the polynomial $p(z) = \prod_{i=1}^{n}(z - x_i)$.

We have been talking about the decryption circuit as if it is boolean. However, for bootstrapping to work – i.e., to be able to perform decryption homomorphically – we know that we need to express decryption as a mod-$\mathbf{B}_I$ circuit. Of course, one option is simply to take $I = (2)$. This is fine, except that our stronger security results, beginning in Chapter 14, require $\det(I)$ to be only polynomial in n, whereas the ideal (2) has 2^n cosets. (We hasten to add that the reduction given in Chapter 5 applies even for $I = (2)$.) In any case, it is easy to emulate boolean circuits using mod-$\mathbf{B}_I$ circuits for any I. In particular, for $\mathbf{x}, \mathbf{y} \in \{0, 1\}$, the value $1 - \mathbf{x} \times \mathbf{y}$ equals $\text{NAND}(\mathbf{x}, \mathbf{y}) \in \{0, 1\}$, regardless of the ring of cosets in which the computation is performed. We restrict the plaintext space $\mathcal{P}$ to be $\{0, 1\} \bmod \mathbf{B}_I$, and represent the inputs and output as elements of this restricted plaintext space, regardless of the underlying ideal I. Of course, this plaintext space restriction is

unnecessary if we use the somewhat homomorphic scheme without bootstrapping.

Restricting the plaintext space to $\{0,1\}$ rather than using all $\det(I)$ cosets of I, just so that we can emulate boolean circuits, seems rather wasteful and inefficient. Is this waste necessary? We leave this as an open problem to which we have not found a satisfactory solution. As far as we can tell, adding terms represented in general "base-I", where $\det(I)$ is large, results in "carries" that are represented by multivariate polynomials of degree too high for our purposes.

Now, we have the following lemma regarding Step 2.

Lemma 9.0.3. *For $i \in [1,t]$, let $a_i = (\ldots, a_{i,1}, a_{i,0}, a_{i,-1}, \ldots)$ be a real number given in binary representation mod $\mathbf{B}_I$ with the promise that $\sum_i a_i \bmod 1 \in [-1/4, 1/4]$. There is a mod-$\mathbf{B}_I$ circuit C for generating $t+1$ integers $z_1, \ldots, z_{t+1}$ (also represented in binary) whose sum is $\lfloor \sum_i a_i \rceil$, such that if the generalized circuit $g(C)$'s inputs are in $\mathcal{B}(r_{in})$, then its outputs are in $\mathcal{B}(r_{out})$ for:*

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

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

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

Proof. Let a_i^* be the integer part of a_i and let $a_i^\dagger = (a_{i,-1}, a_{i,-2}, \ldots)$ be the fractional part. Let $T = \lceil \log t \rceil + 2$. Let $b_i = (a_{i,-1}^\dagger, \ldots, a_{i,-T}^\dagger)$. First, we claim that $\lfloor \sum a_i^\dagger \rceil = \lfloor \sum b_i \rceil$ – i.e., that truncating the least significant bits of the $a_i^\dagger$'s does not affect the rounded sum. This claim follows from the promise that $\sum_i a_i^\dagger$ is within $1/4$ of an integer, and that

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

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

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

This strategy is straightforward when $I = (2 \cdot \mathbf{e}_1)$ and the plaintext space is $\{0,1\}$ mod I.

The binary representation of the Hamming weight of $(x_1, \ldots, x_t)$ is given by

$$(e_{2^{\lfloor \log t \rfloor}}(x_1, \ldots, x_t) \bmod 2, \ldots, e_{2^0}(x_1, \ldots, x_t) \bmod 2)$$

where $e_i(x_1, \ldots, x_t)$ is the ith elementary symmetric polynomial over $x_1, \ldots, x_t$. (See Lemma 4 of [24].) These elementary symmetric polynomials can obviously be computed efficiently. Specifically, one obtains them as the coefficients of the polynomial $p(z) = \prod_{i=1}^{t}(z - x_i)$. The next step would be to bound $\|e_{2^k}(\mathbf{x}_1, \ldots, \mathbf{x}_t)\|$ for $\mathbf{x}_i \in \mathcal{B}(r_{in})$, for $k \in \{0, \ldots, \lfloor \log t \rfloor\}$.

However, for $I \neq (2 \cdot \mathbf{e}_1)$ the situation is complicated by the fact that reduction modulo 2 does not occur automatically in the mod-$\mathbf{B}_I$ circuit. Here we use a slightly different approach (which also works when $I = (2 \cdot \mathbf{e}_1)$). Let $M \in \mathbb{Z}^{(t+1)\times(t+1)}$ be given by $M_{ij} = \binom{i}{j}$ for $i, j \in [0, t]$. Let M^{-1} be a matrix with elements in $R \bmod I$ such that $M^{-1} \cdot M$ is the identity matrix modulo I; M is invertible modulo I, since $\det(M) = 1$. First, our circuit will compute $\mathbf{v} \leftarrow (e_0(b_1, \ldots, b_t), \ldots, e_t(b_1, \ldots, b_t))^T$. Note that $M^{-1} \cdot \mathbf{v} = \mathbf{e}_h$, which is essentially the Hamming weight h of $(b_1, \ldots, b_t)$ in unary. From the unary, we obtain the binary expression by computing the inner product of $\mathbf{e}_h$ with the multi-vector $(\mathbf{c}_0, \ldots, \mathbf{c}_h, \ldots, \mathbf{c}_t)$, where $\mathbf{c}_i$ is the binary representation of i.

Let C be the mod-$\mathbf{B}_I$ sub-circuit above for computing any bit of the binary representation of the Hamming weight. Using $n \cdot \|\mathbf{B}_I\|$ as an upper bound on the length of elements in $R \bmod \mathbf{B}_I$, we have

$$\begin{aligned}
&\|g(C)(\mathbf{x}_1, \ldots, \mathbf{x}_t)\| \\
&\quad \leq \gamma_{\mathsf{Mult}}(R) \cdot n \cdot \|\mathbf{B}_I\| \cdot \Big(\sum_{i \in [0,t]} \|e_i(\mathbf{x}_1, \ldots, \mathbf{x}_t)\| \Big) \cdot t \\
&\quad \leq \gamma_{\mathsf{Mult}}(R) \cdot n \cdot \|\mathbf{B}_I\| \cdot \Big(\sum_{i \in [0,t]} \binom{t}{i} \gamma_{\mathsf{Mult}}(R)^{i-1} \cdot r_{in}^i \Big) \cdot t \\
&\quad = n \cdot \|\mathbf{B}_I\| \cdot (1 + \gamma_{\mathsf{Mult}}(R) \cdot r_{in})^t \cdot t
\end{aligned}$$

At this point, we have generated T numbers, each with $O(T)$ bits, with the same sum as $\sum b_i$. There is a $O(\log T)$-depth constant fan-in boolean circuit for computing this sum, which can be emulated by a $O(\log T)$-depth mod-$\mathbf{B}_I$ circuit. (We omit the details.) Combining the above with results in the proof Theorem 7.3.2, the result follows.

□

Unfortunately, Step 2 uses $t = n$, implying $r_{\mathsf{Dec}}/r_{\mathsf{Enc}} \geq r_{out}/r_{in} \geq 2^n$, and therefore the above analysis cannot show that the scheme is both bootstrappable and secure. However, Lemma 9.0.3 will be relevant to our final scheme, as will the following lemma regarding Step 3:

Lemma 9.0.4. *Using a constant depth circuit having polynomial fan-in* $\mathsf{Add}_{\mathbf{B}_I}$ *gates and constant fan-in* $\mathsf{Mult}_{\mathbf{B}_I}$ *gates, we can compute* $\psi - \mathbf{B}_J^{\mathrm{sk1}} \cdot (\sum \mathbf{y}_i) \bmod \mathbf{B}_I$ *from a binary representation (using the bits* $\{0,1\} \bmod \mathbf{B}_I$*) of the terms of the expression.*

The proof of Lemma 9.0.4 involves converting the binary representation of the terms to a more "natural" mod-$\mathbf{B}_I$ representation, at which point the computation is trivially constant depth. As a toy example for intuition, suppose we have mod-13 gates, where the numbers $0, \ldots, 12$ are represented by 13 different "frequencies" (not in terms of a binary representation), and Add_{13} and Mult_{13} perform addition and multiplication modulo 13 "automatically." Also suppose that we are given a number $b = ...b_1b_0$ in binary representation, where each of the b_i is separately represented by the frequency for '0' or '1' (not by any of the other 11 frequencies). For example, suppose 9 is represented as 1001 rather than by the natural frequency for '9'. From the initial representation of b, how do we compute the "natural" representation of $b \bmod 13$ as a single frequency (from among the 13 different frequencies)? First, we precompute the frequencies $a_j \leftarrow 2^j \bmod 13$. Next, we output $\mathsf{Add}_{13}(..., \mathsf{Mult}_{13}(a_1, b_1), \mathsf{Mult}_{13}(a_0, b_0))$. Using polynomial-fan-in Add_{13} gates, this takes constant depth even if b has a polynomial number of bits. Essentially the same considerations apply in the proof of Lemma 9.0.4. The simplest case is where $I = (2)$ and the conversion is unnecessary.

Proof. For a slightly simpler case, let us first assume that $\mathbf{B}_J^{\mathrm{sk1}}$ is a rotation basis, so that the remaining decryption computation is to compute $\psi - \mathbf{v}_J^{\mathrm{sk1}} \times (\sum \mathbf{y}_i) \bmod \mathbf{B}_I$ for $\mathbf{v}_J^{\mathrm{sk1}} \in R$. Consider one of the vectors – say, $\mathbf{y} \leftarrow \mathbf{y}_1$. How do we compute the "natural" representation of $\mathbf{y} \bmod \mathbf{B}_I$?

Currently, the ith coefficient y_i of $\mathbf{y}$ is represented by the elements $y_{ix} \times \mathbf{e}_1, \ldots, y_{i0} \times \mathbf{e}_1 \in \{0,1\} \bmod \mathbf{B}_I$ where $y_i = \sum_{j=0}^{x} 2^j \cdot y_{ij}$. So, we have

$$\mathbf{y} = \sum_{i \in [1,n], j \in [0,x]} 2^j \times (y_{ij} \times \mathbf{e}_1) \times \mathbf{e}_i \bmod \mathbf{B}_I$$

After pre-computing values $\mathbf{a}_j \leftarrow 2^j \bmod \mathbf{B}_I$ for $j \in [0, x]$, we can compute this representation of $\mathbf{y} \bmod \mathbf{B}_I$ by using two levels of $\mathsf{Mult}_{\mathbf{B}_I}$ gates (since each term in the sum is the product of three terms) and then $\log_{f(n)}(nx)$ levels of $f(n)$-fan-in $\mathsf{Add}_{\mathbf{B}_I}$ gates. Overall, this is constant depth assuming $\mathbf{y}$ was initially represented by a polynomial number of bits.

We obtain the natural mod-$\mathbf{B}_I$ representations of the other vectors in a similar fashion. Thereafter, we compute the result in constant depth, using one level to compute $\mathbf{v}_J^{\mathrm{sk1}} \times \mathbf{y}_i \bmod \mathbf{B}_I$ for each i and a constant number of polynomial fan-in mod-$\mathbf{B}_I$ gates for addition.

The case of a general matrix $\mathbf{B}_J^{\mathrm{sk1}}$ is only slightly more complicated. Basically, since the matrix inhibits our use of ring multiplication, we first compute the "natural" mod-$\mathbf{B}_I$ representation of each individual coefficient (rather than the full vector), multiply the coefficients together in the proper fashion to obtain the natural representations of coefficients in the vectors $\mathbf{B}_J^{\mathrm{sk1}} \cdot \mathbf{y}_i$, and then multiply the representations by the appropriate $\mathbf{e}_i$'s, and add the results modulo $\mathbf{B}_I$.

□

At this point, it may be tempting to ask: how is a mod-$\mathbf{B}_I$ gate implemented, and doesn't this implementation add to the decryption complexity? But we have shown that ring addition and multiplication applied to ciphertexts *induces* mod-$\mathbf{B}_I$ operations over plaintexts: e.g., adding two ciphertexts that encrypt π_1 and $\pi_2 \bmod \mathbf{B}_I$ gives a third ciphertext that encrypts $\pi_3 = \pi_1 + \pi_2 \bmod \mathbf{B}_I$ – i.e., *already reduced* modulo $\mathbf{B}_I$. The mod-$\mathbf{B}_I$ operations, implicitly applied to plaintexts, come for free with the ring operations applied to ciphertexts (up to a point defined by the permitted circuits $\mathcal{C}_{\mathcal{E}}$).

From Lemmas 9.0.3 and 9.0.4, we conclude that, aside from the coefficient multiplication operations in the computation of $\mathbf{B}_J^{\mathrm{sk2}} \cdot \psi$ that we have ignored, the depth of our decryption circuit is $O(\log n)$, where the hidden constant is greater than 1. By Theorem 7.3.2, the maximum depth that we can evaluate is $d = \log\log r_{\mathsf{Dec}} - \log\log \gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}}$. Can we take d to be greater than $\log n$?

Unfortunately, the answer appears to be 'no.' Specifically, the dominant computation in decryption is $\lfloor (\mathbf{B}_J^{\mathrm{sk}})^{-1} \cdot \psi \rceil$, which occurs within the computation of $\psi \bmod \mathbf{B}_J^{\mathrm{sk}}$. Roughly speaking, to ensure that the rounding is correct, one must use a sufficient number of bits of precision. Then, the high precision of each number-number multiplication that occurs within the matrix-vector multiplication forces us to use a high-depth circuit. Specifically, two k-bit numbers can be multiplied together using a $O(\log k)$-depth circuit (with constant fan-in). The precision we seem to need is roughly $\log\det(J) > n \cdot \log r_{\mathsf{Dec}}$ bits, and therefore

we need about a $O(\log n + \log\log r_{\mathsf{Dec}})$-depth circuit.

Unfortunately, for this initial scheme, it seems that no matter how the parameters are set, the decryption circuit is always slightly too complex for the scheme to evaluate.[1] This problem is difficult to fix *post hoc*, in part due to the self-referential nature of the bootstrapability property: intuitively, if one expands the set of circuits that $\mathcal{E}$ can "handle" in an effort to include $D_{\mathcal{E}}$, one seemingly must increase the complexity of $\mathsf{Decrypt}_{\mathcal{E}}$ to accommodate, thereby making the circuit $D_{\mathcal{E}}$ more complex, possibly such that $D_{\mathcal{E}}$ always elusively falls outside of the expanded set. To obtain a bootstrappable encryption scheme, it seems necessary to change the decryption algorithm fundamentally.

[1] However, we do not prove this. It remains possible that the decryption circuit of this initial scheme can be expressed in a way that makes the scheme bootstrappable.

Chapter 10

Squashing the Decryption Circuit

Let $\mathcal{E}^*$ be the encryption scheme described in Chapter 7, modified by Tweak 2 and preferably also Tweak 1 as described in Chapter 8. In this Chapter, we describe how to transform $\mathcal{E}^*$ so as to lower the complexity of the decryption circuit and achieve a bootstrapable scheme. A crucial point is that this transformation *does not reduce the evaluative capacity at all* – i.e., the set of permitted circuits remains fixed. Of course, there is a price: in our new scheme $\mathcal{E}$, we potentially weaken security by including information about the $\mathcal{E}^*$ secret key inside the $\mathcal{E}$ public key. We first describe our transformation generically. We prove security of $\mathcal{E}$ (generically) based on $\mathcal{E}^*$ and the assumed hardness of a certain abstract distinguishing problem, where the latter arises from the new information included in the $\mathcal{E}$ public key. We then instantiate the transformation, where the distinguishing problem becomes a lattice problem that we discuss in Chapter 11.

10.1 A Generic Description of the Transformation

At a high level, our transformation works by splitting the original decryption algorithm into two phases – an initial computationally intensive preprocessing phase performed without the secret key (by the encrypter), followed by a computationally lightweight phase using the secret key (by the decrypter). In short, the encrypter preprocesses its own initial ciphertext, leaving less work for the decrypter to do.

Interestingly, this two-phase approach to decryption is precisely what one finds in *server aided cryptography*. In that setting, a user wants to minimize its cryptographic computation – e.g., because it is using a constrained device, such as a smartcard or handheld. So, it

outsources expensive computations to a server. To set up this arrangement, the user (in some schemes) must give the server a tag τ that is statistically dependent on its secret key sk, but which is not sufficient to permit the server to decrypt efficiently on its own. The processing that the server performs may expand the size of the ciphertext substantially, but nonetheless the processed ciphertext requires less computation for the user to decrypt. In our setting, the encrypter plays the role of the server. We will also use a secret-key-dependent tag τ and suffer from ciphertext expansion.

Now, we describe the transformation in detail. Let $\mathcal{E}^*$ be the initial encryption scheme. We construct a modified scheme $\mathcal{E}$ that uses two new algorithms, $\mathsf{SplitKey}_{\mathcal{E}}$ and $\mathsf{ExpandCT}_{\mathcal{E}}$, that will remain abstract for now.

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

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

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

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

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

Definition 10.1.1 ($\mathsf{SplitKey}$ Distinguishing Problem)**.** The challenger sets $(\mathrm{sk}^*, \mathrm{pk}^*) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}^*}$ and $b \xleftarrow{\mathrm{R}} \{0, 1\}$. If $b = 0$, it sets $(\mathrm{sk}, \tau) \xleftarrow{\mathrm{R}} \mathsf{SplitKey}(\mathrm{sk}^*, \mathrm{pk}^*)$. If $b = 1$, it sets $(\mathrm{sk}, \tau) \xleftarrow{\mathrm{R}} \mathsf{SplitKey}(\perp, \mathrm{pk}^*)$, where $\perp$ is a special symbol. The problem: guess b given $(\tau, \mathrm{sk}^*, \mathrm{pk}^*)$.

Theorem 10.1.2. *Suppose that there is an algorithm $\mathcal{A}$ that breaks the semantic security of $\mathcal{E}$ above with advantage ϵ. Then, there exist algorithms $\mathcal{B}_0$ and $\mathcal{B}_1$, running in about the same time as $\mathcal{A}$, such that either $\mathcal{B}_0$'s advantage against the* $\mathsf{SplitKey}$ *Distinguishing Problem or $\mathcal{B}_1$'s advantage against the semantic security of $\mathcal{E}^*$ is at least $\epsilon/3$.*

Proof. Let Game 0 be the real-world semantic security game. Game 1 is like Game 0, except the challenger generates pk differently. Specifically, instead of inputting sk^* into $\mathsf{SplitKey}$, it

inputs $\perp$ to obtain τ, and adds τ to the pk it sends to $\mathcal{A}$. By assumption, ϵ is $\mathcal{A}$'s advantage in Game 0. Let ϵ' be $\mathcal{A}$'s advantage in Game 1.

$\mathcal{B}_0$ runs as follows. The challenger sets bit $b \xleftarrow{R} \{0,1\}$ and sends a SplitKey Distinguishing Problem instance $(\tau, \mathrm{sk}^*, \mathrm{pk}^*)$ to $\mathcal{B}_0$. $\mathcal{B}_0$ sends $\mathrm{pk} \leftarrow (\mathrm{pk}^*, \tau)$ to $\mathcal{A}$. When $\mathcal{A}$ asks for a challenge ciphertext on one of (π_0, π_1), $\mathcal{B}_0$ sets $\beta \xleftarrow{R} \{0,1\}$ and sends $\psi \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_\beta)$. Eventually, $\mathcal{A}$ sends a bit β'. $\mathcal{B}_0$ sends $b' \leftarrow \beta \oplus \beta'$ to the challenger. Note that the public key pk (and the other aspects of the simulation) is distributed exactly as in Game b. We compute that $\mathcal{B}_0$'s advantage is at least $|\epsilon - \epsilon'|/2$.

$\mathcal{B}_1$ runs as follows. It obtains an $\mathcal{E}^*$ public key pk^* from the challenger. It runs $(\mathrm{sk}, \tau) \xleftarrow{R} \mathsf{SplitKey}(\perp, \mathrm{pk}^*)$ and sends $\mathrm{pk} \leftarrow (\mathrm{pk}^*, \tau)$ to $\mathcal{A}$. When $\mathcal{A}$ asks for a challenge ciphertext on one of (π_0, π_1), $\mathcal{B}_1$ asks the challenger for a challenge ciphertext on one of (π_0, π_1). The challenger sends back ψ^*. $\mathcal{B}_1$ sets ψ to include ψ^* and the output of $\mathsf{ExpandCT}_{\mathcal{E}}(\mathrm{pk}, \psi^*)$ and sends ψ to $\mathcal{A}$. $\mathcal{A}$ sends a bit b', which $\mathcal{B}_1$ forwards to the challenger. We see that the distribution is the same as in Game 1. Also, $\mathcal{B}_1$'s bit is correct if $\mathcal{A}$'s bit is correct; so $\mathcal{B}_1$ has advantage ϵ'.

□

In the next Section, we specify how to instantiate SplitKey, ExpandCT, and the new Decrypt algorithm. After that, we will analyze the new decryption circuit, and prove that we finally have a bootstrappable encryption scheme. We will consider the hardness of our concrete version of the SplitKey Distinguishing Problem in Chapter 11.

10.2 How to Squash, Concretely

Let $\mathbf{v}_J^{\mathrm{sk}^*}$ be the secret key vector of our somewhat homomorphic encryption scheme $\mathcal{E}^*$ after Tweak 1. (Our concrete transformation below can be adapted to handle the scheme without Tweak 1, but handling a secret matrix rather than a secret vector is less efficient.) Recall that this vector is an element of the fractional ideal J^{-1}. Also, recall our decryption equation:

$$\pi = \psi - \lfloor \mathbf{v}_J^{\mathrm{sk}^*} \times \psi \rceil \bmod \mathbf{B}_I$$

The idea of our abstract transformation was to place a "hint" about the $\mathcal{E}^*$ secret key inside the $\mathcal{E}$ public key; what hint do we give about $\mathbf{v}_J^{\mathrm{sk}^*}$?

Our hint will consist of a set of vectors that has a (secret) sparse subset of vectors whose sum is essentially $\mathbf{v}_J^{\mathrm{sk}^*}$. More specifically, the set of vectors τ is $\mathbf{t}_1, \ldots, \mathbf{t}_{\gamma_{setsize}(n)} \in J^{-1}$, where $\gamma_{setsize}(n)$ is a parameter that is polynomial in n. $S \subset \{1, \ldots, \gamma_{setsize}(n)\}$ will be a subset of indices having cardinality $\gamma_{subsetsize}(n)$. And it will hold that $\sum_{i \in S} \mathbf{t}_i = \mathbf{v}_J^{\mathrm{sk}^*} \bmod I$. The new secret key sk is a 0/1-matrix encoding the subset S. The $\mathsf{SplitKey}$ distinguishing problem becomes essentially: given $\mathbf{v}_J^{\mathrm{sk}^*}$ and τ decide whether there is actually a sparse subset whose sum if $\mathbf{v}_J^{\mathrm{sk}^*} \bmod I$, or whether there is a sparse subset whose sum is $\mathbf{0} \bmod I$.

In the $\mathsf{ExpandCT}$ operation, the "encrypter" processes a ciphertext ψ^* output by the original scheme $\mathcal{E}^*$ by computing all of the products $\mathbf{c}_i \leftarrow \mathbf{t}_i \times \psi^* \bmod \mathbf{B}_I$ and including them in the new ciphertext ψ. To decrypt ψ, the user basically extracts the $\gamma_{subsetsize}(n)$ $\mathbf{c}_i$'s that are "relevant" – the $\mathbf{c}_i$'s for which $i \in S$. It then uses the decryption equation

$$\pi = \psi^* - \lfloor \sum_{i \in S} \mathbf{c}_i \rceil \bmod \mathbf{B}_I$$

which can easily be verified to be correct.

This transformation will actually end up increasing the computational complexity of decryption. However, the important point is that the $\mathsf{ExpandCT}$ operation, which *does not* need to be performed homomorphically, prepares a ciphertext that can be decrypted by a *shallower* circuit. The essential reason is that summing up $\gamma_{subsetsize}(n)$ values (in the new decryption equation) requires much less depth – less than $\log n$, as we will see – when $\gamma_{subsetsize}(n)$ is much less than n. We now describe the transformation more formally.

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

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

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

$\mathsf{ExpandCT}_{\mathcal{E}}(\mathrm{pk}, \psi^*)$. Outputs $\mathbf{c}_i \leftarrow \mathbf{t}_i \times \psi^* \bmod \mathbf{B}_I$ for $i \in [1, \gamma_{setsize}(n)]$.

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

Step 0: Set the vectors $\mathbf{w}_{ij} \leftarrow M_{ij} \cdot \mathbf{c}_j$

Step 1: Set the vectors $\mathbf{x}_i \leftarrow \sum_{j=1}^{\gamma_{setsize}(n)} \mathbf{w}_{ij}$

Step 2: From $\mathbf{x}_1, \ldots, \mathbf{x}_{\gamma_{subsetsize}(n)}$, generate integer vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{\gamma_{subsetsize}(n)+1}$ with sum $\lfloor \sum \mathbf{x}_i \rceil$.

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

Remark 10.2.1. To generate τ, one may, for example, just set $\mathbf{t}_1, \ldots, \mathbf{t}_{\gamma_{setsize}(n)-1}$ to be uniformly random vectors in $J^{-1} \cap \mathcal{P}(\mathbf{B}_I)$. Then, one sets $\mathbf{t}_{\gamma_{setsize}(n)} \leftarrow \mathbf{v}_J^{\mathrm{sk}^*} - \sum_{i=1}^{\gamma_{subsetsize}(n)-1} \mathbf{t}_i \bmod \mathbf{B}_I$. Then one permutes the vectors.

Remark 10.2.2. Without Tweak 2, we could have instead used a $\gamma_{setsize}(n)$-sized set of *matrices* with a hidden $\gamma_{subsetsize}(n)$-sized subset whose sum is related to $(\mathbf{B}_J^{\mathrm{sk}})^{-1}$. This would have resulted in a larger public key.

10.3 Bootstrapping Achieved: The Decryption Circuit for the Transformed System

We analyzed Steps 2 and 3 in Chapter 9. It is obvious that Step 0 requires only constant depth. We claim that Step 1 requires only constant depth, but why? Computing $\sum_{j=1}^{\gamma_{setsize}(n)} \mathbf{w}_{ij}$ is very cheap because, in the set $\{\mathbf{w}_{ij} : j \in [1, \gamma_{setsize}(n)\}$, *there is only one nonzero vector*. Therefore, when we add the vectors, no expensive carry operations are required; we simply "XOR" the vectors together using polynomial-fan-in $\mathsf{Add}_{\mathbf{B}_I}$ operations, using constant depth. At last, we have the following theorem.

Theorem 10.3.1. *The scheme $\mathcal{E}$ is bootstrappable when*

$$\gamma_{subsetsize}(n) \cdot \log^{c_1} \gamma_{subsetsize}(n) \leq \left(\frac{\log(r_{\mathsf{Dec}}/m)}{2^{c_2} \cdot \log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})} \right)$$

where $\log^{c_1} \gamma_{subsetsize}(n)$ is the polylog term arising in Lemma 9.0.3, m arises from the redefinition of $\mathcal{C}_{\mathcal{E}}$ in the Tweaks ($m = 2$ when just Tweak 2 is used), and c_2 is a constant representing the depth needed in a circuit having $\mathsf{Add}_{\mathbf{B}_I}$ gates with $\gamma_{\mathsf{Mult}}(R) = n^{\Omega(1)}$ fan-in and $\mathsf{Mult}_{\mathbf{B}_I}$ gates with constant fan-in to sequentially perform $\mathsf{Decrypt}_{\mathcal{E}}$ Steps 0, 1, and 3, and a NAND gate.

Proof. As in the proof of Theorem 7.3.2, for a c-level circuit, if the inputs to the generalized circuit are in $\mathcal{B}(r)$, the outputs are in $\mathcal{B}((\gamma_{\mathsf{Mult}}(R) \cdot r)^{2^c})$. Combining with Lemma 9.0.3, we have that if the inputs to our generalized NAND-augmented decryption circuit are in $\mathcal{B}(r_{\mathsf{Enc}})$, the output is in

$$(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})^{2^{c_2} \cdot (\gamma_{subsetsize}(n) \cdot \mathrm{polylog}(\gamma_{subsetsize}(n)))}$$

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

□

For example, suppose $\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}}$ is polynomial in n, and $r_{\mathsf{Dec}} = 2^{n^C}$ for $C < 1$. In this case, $\gamma_{subsetsize}(n)$ can be polynomial in n (but sub-linear). The constants c_1 and c_2 are not very large, though in practice one would want to optimize them beyond what we have done.

Chapter 11

Security

From Theorem 10.1.2, we know that the bootstrappable encryption scheme described in Chapter 10.2 is semantically secure as long as the SplitKey distinguishing problem (instantiated as described as in Chapter 10.2) is hard and the somewhat homomorphic encryption scheme of Chapter 7 (possibly with the tweaks of Chapter 8) is semantically secure. In other words, the bootstrappable encryption scheme's security is based on two assumptions.

We already addressed the security of the somewhat homomorphic encrypion scheme in Chapter 7.7, basing it on the decision BDDP. Later, we will revisit this scheme, modifying it to obtain a quantum reduction from the shortest independent vector problem (SIVP) over ideal lattices. In the remainder of this Chapter, we will consider the hardness of our concrete version of the SplitKey distinguishing problem. Concretely, the SplitKey distinguishing problem will become the (decision) sparse subset sum problem (SSSP). (See Definition 11.1.4.) We then show how to reduce search SSSP to decision SSSP using Goldreich-Levin [55, 51].

11.1 Regarding the Hint Given in Our "Squashing" Transformation

For the concrete instantiation of SplitKey given in Chapter 10.2, the SplitKey distinguishing problem becomes the following.

Definition 11.1.1 (SplitKey Distinguishing Problem, Concrete Version)**.** Let $\gamma_{setsize}(n)$ and $\gamma_{subsetsize}(n)$ be functions as above, and $\mathbf{B}_I$ a basis of an ideal I. The challenger sets $(\mathrm{sk}^*, \mathrm{pk}^*) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}^*}$ and $b \xleftarrow{\mathrm{R}} \{0,1\}$, where sk^* includes the secret vector $\mathbf{v}_J^{\mathrm{sk}^*} \in J^{-1}$. If

$b = 1$, it sets $\mathbf{v}_J^{\mathrm{sk}^*} \leftarrow \mathbf{0}$. It sets τ to be a set of $\gamma_{setsize}(n)$ vectors $\mathbf{t}_1, \ldots, \mathbf{t}_{\gamma_{setsize}(n)}$ that are uniformly random in $J^{-1} \bmod \mathbf{B}_I$ subject to the constraint that there exists a subset $S \subseteq \{1, \ldots, \gamma_{setsize}(n)\}$ of cardinality $\gamma_{subsetsize}(n)$ such that $\sum_{i \in S} \mathbf{t}_i \in \mathbf{v}_J^{\mathrm{sk}^*} + I$. The problem is to guess b given $(\tau, \mathrm{sk}^*, \mathrm{pk}^*)$.

Here we discuss the hardness of our concrete version of the SplitKey Distinguishing Problem given in Definition 11.1.1. The problem is somewhat unnatural, in the sense that it depends on our key generation algorithm. Below, we base the hardness of our SplitKey Distinguishing Problem on a sparse subset sum problem modulo an integer that is essentially independent of our encryption scheme. We do this in two steps. First, we relate the SplitKey Distinguishing Problem to a sparse subset vector sum problem modulo the lattice IJ, where the problem is independent of the secret key output by our key generation algorithm (but not the public key). Next, as long as I and J satisfy certain criteria, we remove the dependence on I and J.

Here is the intermediate problem that we use.

Definition 11.1.2 (Sparse Vector Subset Sum Problem (SVSSP))**.** Let $\gamma_{setsize}(n)$ and $\gamma_{subsetsize}(n)$ be functions as above, and $\mathbf{B}_I$ a basis of an ideal I. The challenger sets $(\mathrm{sk}^*, \mathrm{pk}^*) \xleftarrow{\mathrm{R}} \mathsf{KeyGen}_{\mathcal{E}^*}$ and $b \xleftarrow{\mathrm{R}} \{0, 1\}$, where the key pair includes bases of an ideal J. It sets $\mathbf{B}_{IJ}$ to be the Hermite normal form of IJ. If $b = 0$ it generates τ as a set of $\gamma_{setsize}(n)$ vectors $\mathbf{u}_1, \ldots, \mathbf{u}_{\gamma_{setsize}(n)}$ that are uniformly random in $\mathbb{Z}^n \cap \mathcal{P}(\mathbf{B}_{IJ})$, subject to the constraint that there exists a subset $S \subseteq \{1, \ldots, \gamma_{setsize}(n)\}$ of cardinality $\gamma_{subsetsize}(n)$ such that $\sum_{i \in S} \mathbf{u}_i \in IJ$. If $b = 1$, it sets the vectors without the constraint. The problem is to guess b given $(\tau, \mathrm{sk}^*, \mathrm{pk}^*)$.

Theorem 11.1.3. *Let $\mathcal{A}$ be an algorithm that decides the concrete version of the* SplitKey *Distinguishing Problem with advantage ϵ. Then, there is an algorithm $\mathcal{B}$, running in about the same time as $\mathcal{A}$, that solves the SVSSP with advantage $(\gamma_{subsetsize}(n)/2\gamma_{setsize}(n)) \cdot \epsilon$.*

Proof. The challenger generates a bit $b \xleftarrow{\mathrm{R}} \{0, 1\}$ and gives $\mathcal{B}$ an appropriate instance $(\tau, \mathrm{sk}^*, \mathrm{pk}^*)$ of SVSSP, where pk^* includes a basis for ideal J, and sk^* contains $\mathbf{v}_J^{\mathrm{sk}^*} \in J^{-1}$. To generate a tag τ' for the SplitKey Distinguishing Problem, $\mathcal{B}$ does the following. Let $\mathbf{B}_{J^{-1}}$ be a basis of J^{-1} and let U be the $n \times \gamma_{setsize}(n)$ matrix formed by the vectors $\{\mathbf{u}_i\}$. $\mathcal{B}$ sets $T' \leftarrow \mathbf{B}_{J^{-1}} \cdot U$, reducing the columns modulo $\mathbf{B}_I$. It sets a bit $\beta \xleftarrow{\mathrm{R}} \{0, 1\}$; if $\beta = 0$ it sets $\mathbf{v} \leftarrow \mathbf{v}_J^{\mathrm{sk}^*}$, otherwise it sets $\mathbf{v} \leftarrow \mathbf{0}$. It adds $\mathbf{v}$ to a random column (say the kth column)

of T', reducing the column modulo $\mathbf{B}_I$, to obtain matrix T. It outputs τ' as the column vectors of T. $\mathcal{A}$ responds with a bit β'. $\mathcal{B}$ outputs $b' \leftarrow \beta \oplus \beta'$.

We have that

$$\Pr[b' = b] = (1/2) \cdot \Pr[b' = 0 | b = 0] + (1/2) \cdot \Pr[b' = 1 | b = 1] = (1/2) \cdot \Pr[b' = 0 | b = 0] + 1/4$$

The last equality follows from the fact that, when $b = 1$, the column vectors of T' are random and independent in $J^{-1} \cap \mathcal{P}(\mathbf{B}_I)$ and thus T is independent of β, β' is independent of β, and b' is uniformly random. We know that the column vectors of T' are random and independent, since multiplication by $\mathbf{B}_{J^{-1}}$ induces a bijection between $\mathbb{Z}^n \cap \mathcal{P}(\mathbf{B}_{IJ})$ and $J^{-1} \cap \mathcal{P}(\mathbf{B}_I)$ that preserves rank: for $\mathbf{c} \in \mathbb{Z}^{\gamma_{setsize}(n)}$, we have

$$T' \cdot \mathbf{c} = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{B}_{J^{-1}} \cdot U \cdot \mathbf{c} = \mathbf{0} \quad \Leftrightarrow \quad U \cdot \mathbf{c} = \mathbf{0}$$

In short, the uniformity of U when $b = 1$ implies the uniformity of T'.

Now, assume $b = 0$. For $i \in \{0, 1\}$, let ϵ_i be the probability that $\mathcal{A}$ outputs 1 when $b^\dagger = i$ in the SplitKey Distinguishing Problem. (We used '$b^\dagger$' to avoid a notation conflict.) We have

$$\Pr[b' = 0] = (1/2) \cdot \big(\Pr[\beta' = 0 | \beta = 0] + \Pr[\beta' = 1 | \beta = 1]\big)$$

If $\beta = 1$, then indeed T has the same distribution as in the $b^\dagger = 1$ case in the SplitKey Distinguishing Problem (i.e., a sparse subset sums to $\mathbf{0}$ modulo I), so $\Pr[\beta' = 1 | \beta = 1] = \epsilon_1$. However, if $\beta = 0$, then T has the same distribution as in the $b^\dagger = 0$ case in the SplitKey Distinguishing Problem (i.e., a sparse subset sums to $\mathbf{v}_J^{\mathrm{sk}^*}$) when $k \in S$, but when $k \notin S$, the distribution is the same as in the $b^\dagger = 1$ case (since $\mathbf{v}_J^{\mathrm{sk}^*}$ is added to a vector that is not a part of the sparse subset and thus is lost in the randomness of the other vectors, while the sparse subset sum is unaffected and is thus still $\mathbf{0}$). Therefore, assuming $\beta = 0$, we have

$$\begin{aligned} \Pr[\beta' = 0] \quad &= \Pr[\beta' = 0 | k \in S] \cdot \Pr[k \in S] + \Pr[\beta' = 0 | k \notin S] \cdot \Pr[k \notin S] \\ &= (1 - \epsilon_0)(\gamma_{subsetsize}(n)/\gamma_{setsize}(n)) + (1 - \epsilon_1) \cdot (1 - \gamma_{subsetsize}(n)/\gamma_{setsize}(n)) \end{aligned}$$

Overall, we have

$$\Pr[b' = 0 | b = 0] = 1/2 + (\epsilon_1 - \epsilon_0)(\gamma_{subsetsize}(n)/2\gamma_{setsize}(n))$$

and thus $\Pr[b' = b] = 1/2 + (\epsilon_1 - \epsilon_0)(\gamma_{subsetsize}(n)/4\gamma_{setsize}(n))$. In other words, $\mathcal{B}$'s advantage is less than $\mathcal{A}$'s advantage by at most a multiplicative advantage of $2\gamma_{setsize}(n)/\gamma_{subsetsize}(n)$. □

Now, we provide a problem that is independent of the particular ideal J output by KeyGen.

Definition 11.1.4 (Sparse Subset Sum Problem (SSSP))**.** Let $\gamma_{setsize}(n)$ and $\gamma_{subsetsize}(n)$ be functions as above, and let q be a prime positive integer. The challenger sets $b \xleftarrow{R} \{0, 1\}$. If $b = 0$ it generates τ as a set of $\gamma_{setsize}(n)$ integers $\{a_1, \ldots, a_{\gamma_{setsize}(n)}\}$ in $[-q/2, q/2]$ that are uniformly random, subject to the constraint that there exists a subset $S \subseteq \{1, \ldots, \gamma_{setsize}(n)\}$ of cardinality $\gamma_{subsetsize}(n)$ such that $\sum_{i \in S} a_i = 0 \bmod q$. If $b = 1$, it sets the elements without the constraint. The problem is to guess b given τ.

The SSSP is a type of knapsack problem; it asks whether there is a sparse knapsack that sums to 0 modulo q. However, the SSSP should not be confused with the low-density knapsack problem. In the latter, $\gamma_{setsize}(n)/\log q$ is small (less than 1). Consequently (though we omit details), one can construct a lattice corresponding to the set of possible knapsack solutions in which the target solution vector corresponding to the subset sum is exponentially shorter than the rest of the solution vectors; this solution vector can then be recovered by a polynomial-time lattice reduction algorithm. In our case, $\gamma_{setsize}(n)/\log q$ will be greater than 1. The consequence of this is that there will be (exponentially) many subsets whose sum is zero modulo q, and known polynomial-time lattice reduction algorithms will fail to extract the sparse solution from the many non-sparse ones.

Theorem 11.1.5. *Assume* $\mathbf{B}_I$ *and* IdealGen *are such that* $\det(I)$ *and* $\det(J)$ *are distinct primes and* $q/\det(IJ)$ *is super-polynomial. Suppose* $\mathcal{A}$ *decides SVSSP with advantage* ϵ *in this setting. Then, there is an algorithm* $\mathcal{B}$ *that decides the SSSP with advantage* $\epsilon/2\gamma_{subsetsize}(n)$*, up to negligible factors.*

The intuition of the proof is that, if there is a sparse subset S such that $\sum_{i \in S} a_i = 0 \bmod q$, then this set sums to zero *over the integers* with non-negligible probability, since

the only possible sums are $k \cdot q$ for $k \in (-\gamma_{subsetsize}(n)/2, \gamma_{subsetsize}(n)/2)$. If this holds, then q is irrelevant; $\sum_{i \in S} a_i = 0 \bmod p$ holds for any p. In particular, $\sum_{i \in S} a_i = 0 \bmod \det(IJ)$.

Accordingly, $\mathcal{B}$'s initial strategy is to set $\mathbf{a}_i \leftarrow a_i \cdot \mathbf{e}_i \bmod \mathbf{B}_{IJ}$ for all i, and ask $\mathcal{A}$ whether these $\mathbf{a}_i$ are statistically uniform or there is a sparse subset of them that sum to $\mathbf{0}$ modulo IJ. There surely is such a sparse subset (namely, S) when $\sum_{i \in S} a_i = 0$. If the a_i's are completely random and independent, then the $\mathbf{a}_i$'s will be statistically random modulo $\mathbf{B}_{IJ}$, since $q/\det(IJ)$ is super-polynomial random and thus the a_i's are statistically random modulo $\det(IJ)$, and because (for technical reasons) multiples of $\mathbf{e}_i$ run over all of the cosets $\mathbb{Z}^n/IJ$.

The difficult case is when $\sum_{i \in S} a_i$ is a nonzero multiple of q. For this case, we would like to map the a_i's to $\mathbf{a}_i$'s so that the $\mathbf{a}_i$'s are statistically uniform, but the initial strategy above does not quite work, since the resulting $\mathbf{a}_i$'s would have a sparse subset that adds up to $k \cdot q \cdot \mathbf{e}_1 \bmod \mathbf{B}_{IJ}$ where $k \in (-\gamma_{subsetsize}(n)/2, \gamma_{subsetsize}(n)/2) \setminus \{0\}$, whereas the $\mathbf{a}_i$'s would be unlikely to have such a sparse subset if they were uniform. So, we revise $\mathcal{B}$'s initial strategy slightly: it chooses a random integer m that is invertible modulo $\det(IJ)$ and sets $\mathbf{a}_i \leftarrow m \cdot a_i \cdot \mathbf{e}_1$. This new strategy still works for the cases when the a_i's are random or have a sparse subset that sums to 0 over the integers; for the case that $\sum_{i \in S} a_i$ is a nonzero multiple of q, the new strategy randomizes the sum of the sparse subset so that it equals $x \cdot \mathbf{e}_1$ for some random x that is invertible modulo $\det(IJ)$. If $\det(I)$ and $\det(J)$ are both super-polynomial, then an overwhelming fraction of numbers are invertible modulo $\det(IJ)$, and the distribution of the $\mathbf{a}_i$'s is thus statistically uniform. If $\det(I)$ is not super-polynomial ($\det(J)$ of course must be), then we can use the Leftover Hash Lemma to prove that the distribution is still statistically uniform.

Overall, if $\mathcal{A}$ has a non-negligible advantage in the SVSSP, then $\mathcal{B}$ can use $\mathcal{A}$ to distinguish when an SSSP instance has a sparse subset that sums to 0 over the integers, which is enough to give $\mathcal{B}$ a non-negligible advantage in the SSSP.

Proof. The challenger generates a bit $b \xleftarrow{R} \{0,1\}$ and gives $\mathcal{B}$ an appropriate instance τ of SSSP. To generate a tag τ' for the SVSSP, $\mathcal{B}$ does the following. $\mathcal{B}$ sets $(\mathrm{sk}^*, \mathrm{pk}^*) \xleftarrow{R} \mathsf{KeyGen}_{\mathcal{E}^*}$ and sets $\mathbf{B}_{IJ}$ to be the Hermite normal form of IJ. It sets m to be random integer that is invertible modulo $\det(IJ)$ and sets τ' to be $\mathbf{u}_i \leftarrow m \cdot a_i \cdot \mathbf{e}_i \bmod \mathbf{B}_{IJ}$.

There are three cases to consider. If $b = 1$, the a_i's are random and independent in $[-q/2, q/2]$. Since $q/\det(IJ)$ is super-polynomial, the a_i's are also (statistically) random and independent modulo $\det(IJ)$. Since $\mathbf{e}_i$ generates all of the cosets $\mathbb{Z}^n/IJ$ (we will show

this momentarily), and m is invertible modulo $\det(IJ)$, the $\mathbf{a}_i$'s are random and independent among the cosets $\mathbb{Z}^n/IJ$.

As to why $\mathbf{e}_i$ generates all of the $\det(IJ)$ cosets of $\mathbb{Z}^n/IJ$, let d be the smallest positive integer such that $d \cdot \mathbf{e}_1 \in IJ$. If $d = \det(IJ)$, then clearly $\mathbf{e}_1$ must traverse all of the $\det(IJ)$ cosets. Otherwise, d is a proper divisor of $\det(IJ)$, either $\det(I)$ or $\det(J)$. But $\det(I) \cdot \mathbf{e}_1$ cannot be in J, since $\det(J) \cdot \mathbf{e}_1 \in J$, which would imply $\mathbf{e}_1 \in J$, since $\det(I)$ and $\det(J)$ are relatively prime. This is impossible, since $\mathbf{e}_1$ generates the entire ring R.

Suppose that $b = 0$ and that $\sum_{i \in S} a_i = 0$ (over the integers). Let S be the set of indices corresponding to the subset whose sum is 0. In this case, $\sum_{i \in S} a_i = 0 \bmod \det(IJ)$, and so $\sum_{i \in S} \mathbf{a}_i = \mathbf{0} \bmod \mathbf{B}_{IJ}$. If we consider any subset of $\gamma_{setsize}(n) - 1$ indices that excludes an index in S, the vectors associated to those indices are random and independent modulo $\mathbf{B}_{IJ}$ for the same reasons as in the first case. Thus, in this case, τ' leads to a (statistically) properly distributed instance of the SVSSP for the $b^\dagger = 0$ case.

Suppose that $b = 0$ and that $\sum_{i \in S} a_i$ is a nonzero multiple of q. Consider the distribution of $\sum_{i \in S} m \cdot a_i \bmod \det(IJ)$; we claim that it is statistically uniform. If this sum is statistically uniform, then the distribution of $\{\mathbf{a}_i\}$ is uniform modulo IJ, since we already know that the distribution is uniform apart from the possibility that there is a sparse subset S with an improbable sum.

First, consider $\sum_{i \in S} m_J \cdot a_i \bmod \det(J)$, where m_J is the residues of m modulo $\det(J)$. We claim that it is statistical uniform and independent of $\sum_{i \in S} m_I \cdot a_i \bmod \det(I)$, where m_I is the residues of m modulo $\det(I)$. Toward this claim, first we note that $\sum_{i \in S} a_i$ is nonzero modulo $\det(J)$, since it equals $k \cdot q$ for some small k, since $\det(J)$ and q are distinct primes, and since k is too small to be divisible by $\det(J)$. We also note that, via CRT, m_J is sampled from $(\mathbb{Z}/\det(J))^*$ randomly and independently of m_I, and, since J is necessarily super-polynomial (for basic security reasons), sampling uniformly from $(\mathbb{Z}/\det(J))^*$ is statistically indistinguishable from sampling uniformly from $(\mathbb{Z}/\det(J))$. The claim follows.

Now, it suffices to show that $\sum_{i \in S} m_I \cdot a_i \bmod \det(I)$ is statistically uniform. If $\det(I)$ is also super-polynomial, then uniformity follows for the same reason it was true wrt $\det(J)$. Otherwise, we apply the Leftover Hash Lemma. Specifically, let $\mathcal{H}$ be a family of hash functions, each hash function h in the family associated a distinct $(h_1, \ldots, h_{\gamma_{setsize}(n)-1}) \in (\mathbb{Z}/\det(I))^{\gamma_{setsize}(n)-1}$. The function maps from the set X of $(\gamma_{subsetsize}(n) - 1)$-sized subsets of $\{1, \ldots, \gamma_{setsize}(n) - 1\}$ to the set $Y = \mathbb{Z}/\det(I)$ via $h(x) = -\sum_{i \in x} h_i \bmod \det(I)$.

This family is clearly 2-universal. By the Leftover Hash Lemma (Lemma 6.3.1), if h and x are selected uniformly and independently, then $(h, h(X))$ is $\frac{1}{2}\sqrt{|\mathcal{Y}|/|\mathcal{X}|}$-uniform. The statistical difference from uniform is negligible when $\det(I) = \binom{\gamma_{setsize}(n)-1}{\gamma_{subsetsize}(n)-1}/n^{\omega(1)}$, which will certainly be true when $\det(I)$ is not super-polynomial. The distribution of $\sum_{i\in S} m_I \cdot a_i \bmod \det(I)$ is even closer to uniform than the distribution induced by the above family of hash functions, since this distribution is equivalent to picking a random hash function from the family above, computing $(h, h(x))$, replacing $h(x)$ with $h(x) + z$ for a uniformly random $z \in (\mathbb{Z}/\det(I))^*$, and then permuting the resulting $\gamma_{setsize}(n)$ elements of $(\mathbb{Z}/\det(I))^*$.

Overall, given that $\sum_{i\in S} a_i = 0 \bmod q$, the most likely multiple of q, out of less than $\gamma_{subsetsize}(n)$ possibilities, is 0 (since the expected mean is 0 when q is odd). Thus, the middle case occurs with probability at least $1/\gamma_{subsetsize}(n)$, and $\mathcal{B}$'s advantage is therefore at least $\epsilon/2\gamma_{subsetsize}(n)$, up to negligible factors. □

Finally, we reduce search SSSP to decision SSSP.

Definition 11.1.6 (Search SSSP)**.** Let $\gamma_{setsize}(n)$ and $\gamma_{subsetsize}(n)$ be functions as above, and let q be a prime positive integer. The challenger generates τ as a set of $\gamma_{setsize}(n)$ integers $\{a_1, \ldots, a_{\gamma_{setsize}(n)}\}$ in $[-q/2, q/2]$ that are uniformly random, subject to the constraint that there exists a subset $S \subseteq \{1, \ldots, \gamma_{setsize}(n)\}$ of cardinality $\gamma_{subsetsize}(n)$ such that $\sum_{i\in S} a_i = 0 \bmod q$. The problem is to output the set S given τ.

Theorem 11.1.7. *Suppose $\mathcal{A}$ decides SSSP with non-negligible advantage in polynomial time. Then, there is an algorithm $\mathcal{B}$ that solves search SSSP with probability* 1/2 *in polynomial time.*

Here is the intuition of the proof. Suppose that we have a flawless oracle $\mathcal{O}$ that decides whether τ' is uniformly random or has a sparse subset that sums to 0. Suppose that we are also given a set $\tau = (a_1, \ldots, a_{\gamma_{setsize}(n)})$ that sums to 0 over a sparse subset S. To decide whether an index $i \in [1, \gamma_{setsize}(n)]$ is in S, we set $r \xleftarrow{R} \mathbb{Z}/(q)$, set $a'_i \leftarrow a_i + r \bmod q$, and give $\tau' = (a_1, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, a_{\gamma_{setsize}(n)})$ to $\mathcal{O}$. If $i \notin S$, then τ' still sums to 0 over S. $\mathcal{O}$ will tell us that there is a sparse subset, and we conclude that $i \notin S$. But if $i \in S$, then τ' is distributed like a random member of $[-q/2, q/2]^{\gamma_{setsize}(n)}$ and we conclude from $\mathcal{O}$'s response that $i \in S$.

Instead of a flawless oracle, we are given algorithm $\mathcal{A}$, which by assumption solves decision SSSP with non-negligible advantage. However, the Goldreich-Levin Theorem [55]

shows us how to use a decision oracle to invert certain functions, even when that oracle is faulty.

Theorem 11.1.8. *(Goldreich-Levin Theorem, restated as in [51]) Suppose we have oracle access to a random process $b_x : \{0,1\}^n \to \{0,1\}$ such that, for some unknown $x \in \{0,1\}^n$, we have*

$$\Pr_{r \in \{0,1\}^n}[b_x(r) = b(x,r)] \geq \frac{1}{2} + \epsilon$$

where the probability is taken uniformly over the internal coin tosses of b_x and all possible choices of $r \in \{0,1\}^n$, and $b(x,r)$ denotes the inner product mod 2 of the binary vectors x and r. Then, we can in time polynomial in n/ϵ output a list of strings that with probability at least 1/2 contains x.

Proof. (Theorem 11.1.7) $\mathcal{B}$ receives a search SSSP instance $\tau = (a_1, \ldots, a_{\gamma_{setsize}(n)})$ from the challenger. Let $x \in \{0,1\}^{\gamma_{setsize}(n)}$ be the (unknown) incidence vector associated to the sparse subset $S \subset \{1, \ldots, \gamma_{setsize}(n)\}$ over which τ sums to 0 modulo q. We will use Goldreich-Levin to recover x.

For $r \in \{0,1\}^n$, define the random process $b_x(r)$ as follows. Sample $c \xleftarrow{R} [-q/2, q/2]$ and $s \xleftarrow{R} \{-1,1\}^{\gamma_{setsize}(n)}$, set

$$\tau' \leftarrow (a_1 + c \cdot s_1 \cdot r_1 \bmod q, \ldots, a_{\gamma_{setsize}(n)} + c \cdot s_{\gamma_{setsize}(n)} \cdot r_{\gamma_{setsize}(n)} \bmod q) \ ,$$

give τ' to $\mathcal{A}$ as its SSSP instance, and output $\mathcal{A}$'s response bit b'. We claim that $b' = \langle x, r \rangle \bmod 2$ with probability non-negligibly bounded away from $1/2$, from which the the result follows by Goldreich-Levin.

For $i \in \{0,1\}$, let ϵ_0 be the probability that $\mathcal{A}$ outputs 1 when a decision SSSP instance has a sparse subset that sums to 0, and let ϵ_1 be the probability that $\mathcal{A}$ outputs 1 otherwise. By assumption, $\epsilon_1 - \epsilon_0$ is non-negligible. Let E_1 and E_2 be the events that $\langle x, r \rangle = 0 \bmod 2$ and $\sum_{i \in S} s_i \cdot r_i = 0$, respectively. Note that E_2 implies E_1. We have

$$\begin{aligned}
&\Pr[b' = \langle x, r \rangle] \\
&= \Pr[b' = 0 | E_2] \cdot \Pr[E_2] + \Pr[b' = 0 | E_1 \wedge \neg E_2] \cdot \Pr[E_1 \wedge \neg E_2] + \Pr[b' = 1 | \neg E_1] \cdot \Pr[\neg E_1] \\
&= (1 - \epsilon_0) \cdot \Pr[E_2] + (1 - \epsilon_1) \cdot ((1/2) - \Pr[E_2]) + \epsilon_1 \cdot (1/2) \\
&= 1/2 + (\epsilon_1 - \epsilon_0) \cdot \Pr[E_2]
\end{aligned}$$

The first equality follows from the fact that, if E_2 occurs, then τ' sums to 0 over S (and is otherwise randomly distributed). However, if E_2 does not occur, then τ' is just a uniformly random member of $[-q/2, q/2]^{\gamma_{setsize}(n)}$, since the sum over S is uniformly random.

We have that $\Pr[E_2]$ is non-negligible – in particular, it is at least $1/(2\gamma_{subsetsize}(n)+1)$ – since there are only $2\gamma_{subsetsize}(n)+1$ possibilities for $\sum_{i \in S} s_i \cdot r_i$ and 0 is the most likely. □

As mentioned in Chapter 10, our approach to reducing the complexity of the decryption circuit is abstractly similar to approaches used in server-aided cryptography to outsource some of the decryption work to an untrusted server. In fact, the similarity is more than just abstract; there are concrete server-aided schemes whose security relies on the SSSP. In particular, in the Matsumoto, Kato and Imai [91] server-aided RSA scheme – called, RSA-S1 – the private exponent d is decomposed into $\gamma_{setsize}(n)$ integers $\{a_i\}$ such that there is a set $\{x_i \in [0, 2^\ell - 1]\}$, only $\gamma_{subsetsize}(n)$ of which are nonzero, such that $\sum_i x_i \cdot a_i = d \bmod \phi(N)$, where N is the RSA modulus. (In our version of SSSP, we restrict the x_i's to $\{0, 1\}$, but this is just for simplicity; like RSA-S1, we could permit the x_i's to have a slightly larger range.) The SSSP is also similar the "full rank avoidance" problem initially proposed by Mulmuley and Sohoni [101] as part of their program to separate P from NP [117]. The full rank avoidance problem asks: given a matrix X with n rows and kn columns grouped as n blocks of k, is it possible to choose one column from each block so that the resulting $n \times n$ matrix M has $\det(M) = 0$? In our setting, we need k to be at least $\gamma_{setsize}(n)/n$. Gurvitz [64] showed the problem to be NP-hard for $k = 2$.

If the SSSP is hard, what are we to make of all of the published attacks against RSA-S1 and its variants [114, 106, 96, 105]? These attacks are feasible only for limited choices of RSA-S1 parameters; they are not polynomial-time in general. Some of these [114] (and previous related results [128, 33]) are essentially time-space tradeoffs, meet-in-the-middle type attacks, whose complexity is exponential in $\gamma_{subsetsize}(n)$; these attacks are not polynomial-time when $\gamma_{subsetsize}(n) = \omega(1)$.

Nguyen and Shparlinski [105] present a lattice-based cryptanalysis of RSA-S1 that succeeds with advantage at least

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

where α is a term that is greater than 1, and where $q = \phi(N)$ and N is the RSA modulus. We can easily avoid the attack by choosing $\gamma_{setsize}(n)$ to be larger than $\log q$. Note that

Theorem 11.1.5 only requires $\log q$ to be larger than $\log(\det(IJ))$ by an additive factor that super-logarithmic in n. So, for example, we could take $\gamma_{setsize}(n)$ to be about $2{\cdot}\log(\det(IJ))$. The intuition is that, once $\gamma_{setsize}(n)$ is sufficiently large, there will be exponentially many subsets in τ (not necessarily sparse) whose vector sum is congruent to $\mathbf{v}_J^{\mathrm{sk}^*}$; lattice reduction techniques have trouble extracting the sparse subset from among the many subset solutions.

For Theorem 11.1.5 to apply to our scheme, we need to use $\mathbf{B}_I$ and instantiate $\mathsf{IdealGen}$ such that I and J satisfy the given requirements on $\det(IJ)$. In Chapter 14, we begin describing a modified version of the somewhat homomorphic scheme that uses an I for which $\det(I)$ is prime. Also, we provide an instantiation of $\mathsf{IdealGen}$ in Chapter 18, which outputs J as a uniformly random invertible prime ideal with norm in a given interval; its norm will most likely be a prime integer.

11.2 Counterbalancing Assumptions

As discussed above, the best known attack on the SSSP is exponential in $\gamma_{subsetsize}(n)$, as long as $\gamma_{setsize}(n)$ is chosen to be large enough to avoid a lattice attack by Nguyen and Shparlinski [105]. I.e., the best attack takes time (roughly) $2^{\gamma_{subsetsize}(n)}$.

On the other hand, by Theorem 10.3.1, our scheme becomes bootstrappable when

$$\gamma_{subsetsize}(n) \cdot \log^{c_1} \gamma_{subsetsize}(n) \leq \left(\frac{\log(r_{\mathsf{Dec}}/m)}{2^{c_2} \cdot \log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})} \right)$$

To allow $\gamma_{subsetsize}(n)$ to be as large as possible for a fixed value of $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$, we let $\gamma_{\mathsf{Mult}}(R)$, r_{Enc} and m be as small as possible (polynomial in n), and r_{Dec} is then approximately $2^{\gamma_{subsetsize}(n)}$. We saw in Chapter 7.7 that the approximation factor of the decision BDDP on which we base security is at least as large as $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$ – i.e., about $2^{\gamma_{subsetsize}(n)}$. We use the rule of thumb that solving $2^{\gamma_{subsetsize}(n)}$-decision BDDP takes time approximately $2^{n/\gamma_{subsetsize}(n)}$ using known attacks [123].

We can set $\gamma_{subsetsize}(n) \approx \sqrt{n}$ to make known attacks on the two problems "equally" expensive (up to the crudeness of our approximations). Or, to put it another way, we can set $\gamma_{subsetsize}(n) \approx \lambda$, where λ is the security parameter of our scheme, and obtain exponential 2^{λ} security against known attacks. Note that this requires our lattice dimension to be quite large: $n \approx \lambda^2$.

Note that in the somewhat homomorphic scheme without bootstrapping, we do not put

any "hint" about the original secret key in the public key, and do not need the second computational assumption. In this case, if we (say) only want to evaluate constant depth, then (as far as we know) it suffices to take n quasi-linear in λ to achieve 2^λ security against known attacks. On the other hand, if we want to evaluate $\theta(\log n)$ depth, this forces us to use a sub-exponential approximation factor in decision BDDP, permitting sub-exponential attacks, and forcing us to take n to be a higher-degree polynomial in λ.

Chapter 12

Performance and Optimizations

In this Chapter, we analyze the performance of our (leveled) fully homomorphic encryption scheme, and describe a few ways to improve the scheme's computational efficiency.

Even after some simple optimizations described in Section 12.1, we find in Section 12.2 that if we want to obtain 2^λ security against known attacks (on each of the two problems on which the security of our scheme is based), the computation per gate in our unoptimized scheme is quasi-linear in λ^9. The computational expense has several sources:

- Homomorphic Decryption: The fact that we perform decryption *homomorphically*, rather than just conventionally, essentially "squares" the computational complexity
- Squashing the Decryption Circuit: This procedure minimized the *depth* of our decryption circuit, but at the expense of substantially increasing the circuit's *size*, and increasing the size of the secret key and ciphertexts
- Counterbalancing assumptions: Make both of our problems hard requires a large lattice dimension

These problems all go away if we only use the somewhat homomorphic encryption scheme (without bootstrapping); this basic scheme is quite efficient.

In Section 12.3, we provide two optimizations, neither substantially decreasing security, that reduce the computation per gate to quasi-linear in λ^6. While still high, this does not seem so unreasonable when one considers that, to get 2^λ security against the number field sieve, one should use an RSA modulus whose bit-length is quasi-linear in λ^3, in which case a full exponentiation takes time quasi-linear in λ^6, even when one uses fast FFT

multiplication. We also provide a third optimization, but where security only holds under the assumption that a different SplitKey Distinguishing Problem is hard. For this different version of the problem, we have no reduction from the SSSP.

12.1 Simple Optimizations

First, we note some very simple optimizations, before getting to the more technical ones described in Section 12.3.

As a preliminary matter, before we begin discussing the scheme's computation per gate, we note that there is some flexibility in how one defines a gate. (See Chapter 4, where we defined what it means for an encryption scheme to be bootstrappable with respect to a set of gates Γ.) In particular, a "gate" could be a "normal" boolean circuit of depth greater than 1. If we use "bigger" gates, then we perform expensive Recrypt operations less often, which may improve efficiency. However, for simplicity of exposition, we will assume in the discussion below that we use NAND gates.

For each NAND gate in C, we evaluate two decryption circuits $D_{\mathcal{E}}$ homomorphically, and then compute NAND homomorphically. In our transformation above from $\mathcal{E}^*$ (the somewhat homomorphic scheme) to $\mathcal{E}$ (with the squashed decryption circuit), we said that a homomorphic Add consisted of extracting (ψ_1^*, ψ_2^*) (the ciphertexts from the somewhat homomorphic scheme) from (ψ_1, ψ_2) (the ciphertexts from the transformed scheme), adding ψ_1^* and ψ_2^* within the somewhat homomorphic scheme (using simple ring addition) to obtain ψ^*, and then setting the output ψ to include ψ^* and the output of $\mathsf{ExpandCT}_{\mathcal{E}}(\mathrm{pk}, \psi^*)$. However, it is actually overkill to use ExpandCT for the interior gates of the NAND-augmented decryption circuit that we are computing; really we only need apply ExpandCT at the end, and can use simple ring addition and multiplication for the interior gate homomorphisms.

Another optimization is that, when applying Recrypt to ψ (the encryption of π under pk_1), we do not really need to first encrypt the bits of ψ under pk_2. Instead, we can view the bits themselves as ciphertexts under pk_2, since there is no requirement that these "ciphertexts" be hiding. In other words, we do not actually need to evaluate the general decryption circuit, but rather merely a non-uniform decryption circuit that takes only the secret key as input and has the ciphertext hard-wired. So, overall, the complexity (per gate in C) is approximately twice the complexity of this simplified version of Recrypt, plus the complexity of ExpandCT.

12.2 Basic Performance

As discussed in Chapter 11.2, the SplitKey Distinguishing Problem becomes harder as $\gamma_{subsetsize}(n)$ increases, while the decision BDDP becomes easier, since increasing $\gamma_{subsetsize}(n)$ increases the approximation factor of the problem. To make both of the problems hard, such that the breaking time of both problems is 2^{λ}, requires us to use a large lattice dimension: $n \approx \gamma_{subsetsize}(n)^2 \approx \lambda^2$.

Now, let us consider the size of an encrypted secret key in our scheme. The secret key (unencrypted) is a $\gamma_{subsetsize}(n) \times \gamma_{setsize}(n)$ matrix of bits. We need $\gamma_{setsize}(n)$ to be rather large – larger than $\log \det(IJ)$ – for our reduction from the SSSP to work. Since J contains a ball of radius r_{Dec}, where the latter is exponential in $\gamma_{subsetsize}(n)$, we have that $\log \det(IJ) > n \log r_{\mathsf{Dec}} > n \cdot \gamma_{subsetsize}(n) \approx \lambda^3$. (And the upper bound $\log \det(IJ) = O(\lambda^3)$ works as well.) So, the secret key (unencrypted) key is approximately λ^4 bits. Encryption multiplies the bit-length by another factor of λ^3, since each ciphertext is a coset of J, where $\log \det(IJ) = O(\lambda^3)$ as described above. Overall, the encrypted secret key is approximately λ^7 bits.

Circumstances become even worse when this encrypted secret key is applied to the ciphertext components output by ExpandCT. Consider a single ciphertext component $\mathbf{c}_i$. It has n coefficients, though each coefficient only needs to have a very small (poly-logarithmic) number of bits, the minimal number needed to ensure that the rounded sum is computed correctly. Each encrypted secret key bit is multiplied with one ciphertext component. Overall, this computation is quasi-linear in $\lambda^7 \times n \approx \lambda^9$, and in fact the result of this intermediate computation also has length quasi-linear in λ^9. The remaining computation is quasi-linear in λ^9, assuming FFT multiplication is used.

12.3 More Optimizations

As our first optimization, we observe that a simplified version of the secret key still works, due to properties of symmetric polynomials.

Optimization 1: Encode the secret key sk as a vector in $\{0,1\}^{\gamma_{setsize}(n)}$, rather than a 0/1 incidence matrix of dimension $\gamma_{subsetsize}(n) \times \gamma_{setsize}(n)$.

Gain: Computational complexity is reduced by a factor of approximately $\gamma_{subsetsize}(n) \approx \lambda$.

In Optimization 1, τ and $\mathsf{ExpandCT}_{\mathcal{E}}$ are as before; the changes are in the format of sk

and in the decryption algorithm, which is as follows.

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

Step 1: Set the vectors $\mathbf{x}_i \leftarrow \mathrm{sk}_i \cdot \mathbf{c}_i$

Step 2: From $\mathbf{x}_1, \ldots, \mathbf{x}_{\gamma_{setsize}(n)}$, generate integer vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{\gamma_{setsize}(n)+1}$ with sum $\lfloor \sum \mathbf{x}_i \rceil$.

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

The key observation here is that all but $\gamma_{subsetsize}(n)$ of the $\mathbf{x}_i$'s are 0, and that, *if we have the promise* that most of the numbers that we are summing up are 0, then we can compute the output above using a shallower circuit. Why? Recall that, in Lemma 9.0.3, we basically reduced computing the sum of t numbers to computing the Hamming weight of a vector $b \in \{0,1\}^t$. Then, we used the fact that the binary representation of this Hamming weight is precisely

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

where e_i is the ith symmetric polynomial. In Optimization 1, we use symmetric polynomials in the same way, but now with the observation that if we have the promise that b has Hamming weight at most k, then there is no need to compute evaluate the polynomials $e_{2^i}(b_1, \ldots, b_t)$ for $i > \lfloor \log k \rfloor$, since they will all be 0 anyway. So, in optimization 1, despite the more concise encoding of sk, we get by with computing the same low-degree elementary symmetric polynomials that we did originally, albeit now with $\gamma_{setsize}(n)$ inputs rather than $\gamma_{subsetsize}(n)$ inputs.

In particular, we have the following lemma, which is analogous to Lemma 9.0.3.

Lemma 12.3.1. *For $i \in [1,t]$, let $a_i = (\ldots, a_{i,1}, a_{i,0}, a_{i,-1}, \ldots)$ be a real number given in binary representation mod $\mathbf{B}_I$ with the promises that $\sum_i a_i \bmod 1 \in [-1/4, 1/4]$ and at most k of the a_i's are nonzero. There is a mod-$\mathbf{B}_I$ circuit C for generating $t+1$ integers $z_1, \ldots, z_{t+1}$ (also represented in binary) whose sum is $\lfloor \sum_i a_i \rceil$, such that if the generalized circuit $g(C)$'s inputs are in $\mathcal{B}(r_{in})$, then its outputs are in $\mathcal{B}(r_{out})$ for:*

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

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

$$r_{out} \leq (\gamma_{\mathsf{Mult}}(R) \cdot r_{in})^{k \cdot \text{polylog}(k)}$$

Proof. The proof is essentially identical to the proof of Lemma 9.0.3 – i.e., we compute the elementary symmetric polynomials up to degree k and use the matrix M^{-1}, now of rank $k+1$. The only real difference is in the value of r_{out}, which is affected by the fact that the polynomials now take more input variables.

Let C be the mod-$\mathbf{B}_I$ sub-circuit for computing any bit of the binary representation of the Hamming weight. Using $n \cdot \|\mathbf{B}_I\|$ as an upper bound on the length of elements in $R \bmod \mathbf{B}_I$, we have

$$\begin{aligned}
&\|g(C)(\mathbf{x}_1, \ldots, \mathbf{x}_t)\| \\
&\quad \leq \gamma_{\mathsf{Mult}}(R) \cdot n \cdot \|\mathbf{B}_I\| \cdot (\sum_{i \in [0,k]} \|e_i(\mathbf{x}_1, \ldots, \mathbf{x}_t)\|) \cdot t \\
&\quad \leq \gamma_{\mathsf{Mult}}(R) \cdot n \cdot \|\mathbf{B}_I\| \cdot (\sum_{i \in [0,k]} \binom{t}{i} \gamma_{\mathsf{Mult}}(R)^{i-1} \cdot r_{in}^i) \cdot t \\
&\quad = t \cdot n \cdot \|\mathbf{B}_I\| \cdot (\sum_{i \in [0,k]} \binom{t}{i} (\gamma_{\mathsf{Mult}}(R) \cdot r_{in})^i) \\
&\quad \leq t \cdot n \cdot \|\mathbf{B}_I\| \cdot (\sum_{i \in [0,k]} (t \cdot \gamma_{\mathsf{Mult}}(R) \cdot r_{in})^i) \\
&\quad \leq k \cdot t \cdot n \cdot \|\mathbf{B}_I\| \cdot (t \cdot \gamma_{\mathsf{Mult}}(R) \cdot r_{in})^k
\end{aligned}$$

At this point, we have generated about $\log k$ numbers, each with $O(\log k)$ bits, with the same sum as $\sum b_i$. There is a $O(\log \log k)$-depth constant fan-in boolean circuit for computing this sum, which can be emulated by a $O(\log \log k)$-depth mod-$\mathbf{B}_I$ circuit. Combining the above with results in the proof Theorem 7.3.2, the result follows.

□

Since r_{out} is similar to before – i.e., exponential in $\gamma_{subsetsize}(n)$ (up to polylogarithmic factors) – one obtains a bootstrappable scheme with Optimization 1 with parameters similar to those required by Theorem 10.3.1.

Now, let us analyze the computation needed after Optimization 1. The more concise

representation of the secret key has size quasi-linear in λ^6 – i.e., $\gamma_{setsize}(n) \approx \lambda^3$ bits, each encrypted in a ciphertext of size approximately $n \cdot \gamma_{subsetsize}(n) \approx \lambda^3$. Multiplying the encrypted secret key balloons the result up to size quasilinear in λ^8. The dominant remaining computation is computing the elementary symmetric polynomials up to degree $\gamma_{subsetsize}(n)$. We need to do one such computation for the least significant bits of the least significant coefficients of the $\mathbf{c}_i$'s, etc.; the total number of such computations is the number of bits in $\mathbf{c}_i$, which is quasi-linear in $n \approx \lambda^2$.

The symmetric polynomials are the coefficients of z^i, $i \in [\gamma_{setsize}(n) - \gamma_{subsetsize}(n), \gamma_{setsize}(n)]$, in the polynomial $p(z) = \prod_{i=1}^{\gamma_{setsize}(n)}(z - b_i)$. Let $f(t)$ be the computation needed to compute the product of t of the $(z - b_i)$'s. Using the recursion that $f(t) = 2 \cdot f(t/2) + \text{polylog}(t/2)$, the total computation needed to compute the symmetric polynomials (non-homomorphically) is $\gamma_{setsize}(n) \cdot \text{polylog}(\gamma_{setsize}(n))$. Since the operations are performed homomorphically – i.e., with ciphertexts of size quasi-linear in λ^3 instead of with bits – the computation needed is quasilinear in λ^6. Since the number of Hamming weight computations is quasi-linear in $n \approx \lambda^2$, the total computation in quasi-linear in λ^8.

Remark 12.3.2. Though it does not affect the asympotics very much, we can optimize Optimization 1 as follows. When a polynomial associated to an interior node has degree $d > \gamma_{subsetsize}(n)$, we can discard its coefficients for z^i for $i < d - \gamma_{subsetsize}(n)$, since they will not affect the end result; thus, at any node, we never maintain more than $\gamma_{subsetsize}(n) + 1$ coefficients.

Optimization 2: Preprocess the initial ciphertext ψ^* even more, collapsing each n-coefficient ciphertext component $\mathbf{c}_i$ into a single coefficient.

Gain: Computational complexity is reduced by a factor of approximately $n \approx \lambda^2$. Combining with Optimization 1, the computational complexity per gate is reduced to λ^6.

Suppose the plaintext space is $\{0, 1\}$ and that $I = (2)$.[1] A ciphertext ψ^* from the somewhat homomorphic scheme has the form $\mathbf{m} + \mathbf{j}$, where $\mathbf{m} \in \pi \cdot \mathbf{e}_1 + 2 \cdot \mathbb{Z}^n$ is "short" and $\mathbf{j} \in J$. Addition and multiplication of ciphertexts does not change the essential form of the ciphertext. In particular, the plaintext π always hides in the least significant coefficient of ψ^*; for all of the other coefficients, the offset from the closest J-vector is even. This

[1] Shai Halevi observed the optimization for this case [67].

suggests that, our decryption equation

$$\pi = \psi^* - \lfloor \mathbf{v}_J^{\mathrm{sk}^*} \times \psi^* \rceil \bmod 2$$

we only really care about the least significant coefficient – i.e., π can be recovered from the least significant coefficient of ψ^* and the least significant coefficient of

$$\mathbf{v}_J^{\mathrm{sk}^*} \times \psi^* = \sum_{i \in S} \mathbf{t}_i \times \psi^*$$

In Optimization 2, we modify ExpandCT to output only the least significant coefficients of the ciphertext components $\mathbf{c}_i = \mathbf{t}_i \times \psi^*$, and simplify decryption so that it only sums up these coefficients, reducing decryption computation by a factor of $n \approx \lambda^2$.

In certain cases, we can perform this optimization even when $I \neq (2)$. For example, the optimization works when $\det(I)$ is a small prime p, though the optimization is more complicated in this setting. First, compute a basis $\mathbf{B}'_I$ of I, where the first column vector $\mathbf{b}_0 = (p, 0, \ldots, 0)$, and $\mathbf{b}_i = (a_i, 0, \ldots, 0, 1, 0, \ldots, 0)$ for $i \in [1, n-1]$, where the '1' is in the ith row and $a_i \in (-p/2, p/2)$. (This can easily be done using elementary column operations.) Consider a vector $\mathbf{m} \in \mathbb{Z}^n$. Let

$$\mathbf{m}' \leftarrow \mathbf{m} - \sum_{i=1}^{n-1} m_i \cdot \mathbf{b}_i = \mathbf{m} \bmod \mathbf{B}_I \ ,$$

Then all of the coefficients of $\mathbf{m}'$ are 0, except possibly the least significant coefficient. The idea is that if we could apply this transformation to the value of $\mathbf{m}$ hiding inside the ciphertext (i.e., where $\psi^* = \mathbf{m} + \mathbf{j}$ for $\mathbf{m} \in \pi \cdot \mathbf{e}_1 + I$ and $\mathbf{j} \in J$), then it seems that we could ignore all but the least significant coefficient, as when $I = (2)$. But how do we apply this transformation to ciphertexts, when the value $\mathbf{m}$ is not accessible?

Before we get to how ExpandCT and Decrypt are modified, let us define a convenient notation. For $\mathbf{B}'_I$ and p as above and $\mathbf{x} \in \mathbb{Q}^n$, let

$$\mathbf{x} \text{ red } \mathbf{B}'_I = \mathbf{x} - \sum_{i=1}^{n-1} x_i \cdot \mathbf{b}_i$$

Notice that all of the coefficients of $\mathbf{x}$ red $\mathbf{B}'_I$ are 0, except possibly the least significant one. Also, notice that $\mathbf{x} + \mathbf{y}$ red $\mathbf{B}'_I = (\mathbf{x}$ red $\mathbf{B}'_I) + (\mathbf{y}$ red $\mathbf{B}'_I)$. Finally, notice that $\mathbf{x}$ red $\mathbf{B}'_I$

seems to have a close relationship with $\mathbf{x} \bmod \mathbf{B}'_I$, which equals

$$\mathbf{x} - \sum_{i=1}^{n-1} \lfloor x_i \rceil \cdot \mathbf{b}_i \bmod p$$

The following lemma characterizes this relationship.

Lemma 12.3.3. *Let $\mathbf{B}'_I$ and p be as described above. Let δ and η be positive reals such that $(np/2) \cdot \delta < \eta < 1/2$. Suppose the coefficients of $\mathbf{x}$ are within δ of integers. Then,*

$$\lfloor \mathbf{x} \bmod \mathbf{B}'_I \rceil = \lfloor \mathbf{x} \text{ red } \mathbf{B}'_I \rceil \bmod p$$

Also, the least significant coefficient of $\mathbf{x}$ red $\mathbf{B}'_I$ has size at most $p \cdot \sum_i |x_i|$ and is within η of an integer.

Proof. The upper bound on the magnitude of the least significant coefficient of $\mathbf{x}$ red $\mathbf{B}'_I$ is obvious.

Since the vectors in $\mathbf{B}'_I$ are integer vectors, the coefficients of $(\mathbf{x} \bmod \mathbf{B}'_I)$ are within δ of integers. Also, for some integer k, we have

$$\begin{aligned}(\mathbf{x} \bmod \mathbf{B}'_I) - (\mathbf{x} \text{ red } \mathbf{B}'_I) &= (\mathbf{x} - k \cdot p \cdot \mathbf{e}_1 - \sum_{i=1}^{n-1} \lfloor x_i \rceil \cdot \mathbf{b}_i) - (\mathbf{x} - \sum_{i=1}^{n-1} x_i \cdot \mathbf{b}_i) \\ &= -k \cdot p \cdot \mathbf{e}_1 + \sum_{i=1}^{n-1} (x_i - \lfloor x_i \rceil) \cdot \mathbf{b}_i\end{aligned}$$

Aside from the $-k \cdot p \cdot \mathbf{e}_1$ term, all the coefficients of this difference have magnitude at most $(n-1) \cdot (p/2) \cdot \delta$. Since $\delta + (n-1) \cdot (p/2) \cdot \delta \leq (np/2) \cdot \delta < \eta$, the coefficients of $(\mathbf{x} \text{ red } \mathbf{B}'_I)$ are close (within η) to the same integers that the coefficients of $(\mathbf{x} \bmod \mathbf{B}'_I)$ are close to (up to a multiple of p for the least significant coefficient).

□

With that technical lemma in hand, we modify $\mathsf{ExpandCT}_{\mathcal{E}}$ and $\mathsf{Decrypt}_{\mathcal{E}}$ as follows.

$\mathsf{ExpandCT}_{\mathcal{E}}(\mathrm{pk}, \psi^*)$. Computes $\mathbf{c}'_i \leftarrow \mathbf{t}_i \times \psi^* \bmod \mathbf{B}_I$ for $i \in [1, \gamma_{setsize}(n)]$, and outputs $\mathbf{c}_i \leftarrow \mathbf{c}'_i$ red $\mathbf{B}'_I$. (All but the first coefficient of these vectors is 0, so these coefficients do not actually need to be output.)

$\mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi)$. Takes as input the secret key sk and a ciphertext ψ. It performs the following steps, which are the same as after Optimization 1, but only the least significant coefficients need to be operated on:

Step 1: Set the vectors $\mathbf{x}_i \leftarrow \mathrm{sk}_i \cdot \mathbf{c}_i$

Step 2: From $\mathbf{x}_1, \ldots, \mathbf{x}_{\gamma_{setsize}(n)}$, generate integer vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{\gamma_{setsize}(n)+1}$ with sum $\lfloor \sum \mathbf{x}_i \rceil$.

Step 3: Compute $\pi \leftarrow \psi - (\sum \mathbf{y}_i) \bmod \mathbf{B}'_I$

To show that decryption is correct, it suffices to show that

$$\psi - \lfloor \sum_{i \in S} \mathbf{c}_i \rceil = \psi - \lfloor \sum_{i \in S} \mathbf{c}'_i \rceil \bmod I$$

where the first expression is what is computed in the new decryption algorithm, and the second expression is what was computed prior to Optimization 2. But this follows from Lemma 12.3.3 as long as $\sum_{i \in S} \mathbf{c}'_i$ has coefficients that are sufficiently close to integers.

Modulo I, $\sum_{i \in S} \mathbf{c}'_i$ equals $\mathbf{v}_J^{\mathrm{sk}^*} \cdot \psi^*$. To ensure that this quantity is sufficiently close to an integer vector, we tweak the set of permitted circuits once again, in much the same way as we did in Tweak 2 (see Chapter 8.4). (Recall that in Tweak 2, we changed the set of permitted circuits to require a ciphertext ψ^* to be within $r_{\mathsf{Dec}}/2$ of the J-lattice, so that the coefficients of $\mathbf{v}_J^{\mathrm{sk}^*} \cdot \psi^*$ would be within $1/4$ of integers, thereby simplifying the rounding step.)

Optimization 3: Use the ring $R = \mathbb{Z}[x]/(f(x))$ for $f(x) = x^n + 1$, where n is a power of 2. (Alternatively, one could use some other irreducible $f(x)$ that equals $x^n + h(x)$ for some constant-degree polynomial $h(x)$ with $\|h\| = \mathrm{poly}(n)$.) To set τ, generate $2 \cdot \gamma_{subsetsize}(n)$ random vectors $\mathbf{x}_j \xleftarrow{\mathrm{R}} J^{-1}$ subject to the constraint that there exists a vector $s \in \{0,1\}^{2 \cdot \gamma_{subsetsize}(n)}$ of Hamming weight $\gamma_{subsetsize}(n)$, and a vector $r \in \{0, \ldots, n-1\}^{2 \cdot \gamma_{subsetsize}(n)}$ such that

$$\mathbf{v}_J^{\mathrm{sk}^*} = \sum_{i=1}^{2 \cdot \gamma_{subsetsize}(n)} (s_i \cdot \mathbf{x}_i^{r_i} \bmod f(x)) \bmod I$$

Gain: Computational complexity is reduced by a factor of approximately $n \approx \lambda^2$. With the previous optimizations, the computation per gate is quasilinear in λ^4.

To describe the optimization another way, τ does not consist of $\gamma_{setsize}(n)$ vectors that

are random and independent (aside from the subset sum constraint). Instead, it consists of only $2 \cdot \gamma_{subsetsize}(n)$ vectors that we "unpack" into $2 \cdot n \cdot \gamma_{subsetsize}(n)$ vectors by using all the "rotations" of the original $2 \cdot \gamma_{subsetsize}(n)$ vectors; the vectors are random and independent aside from a subset sum constraint on the $2 \cdot n \cdot \gamma_{subsetsize}(n)$ vectors. The secret key sk consists of $2 \cdot \gamma_{subsetsize}(n)$ ciphertexts encrypting the bits of s, as well as $2 \cdot \gamma_{subsetsize}(n)$ ciphertexts that encrypt the rotations; the value $r_i \in [0, n-1]$ is encrypted in a ciphertext having the form $x^{r_i} + i + j$ for $i \in I$ and $j \in J$. Notice that this secret key is much more concise, by a factor of approximately $\gamma_{setsize}(n)/\gamma_{subsetsize}(n)$.

In $\mathsf{ExpandCT}$, we output $\{\mathbf{x}_i \times \psi^*\}$, much fewer values than before. Combining the secret key with these ciphertext components (in the obvious way) also takes much less computation than before, by a multiplicative factor of approximately λ^2.

The drawback of this optimization is that its security is questionable. In particular, the less random choice of τ prevents the reduction from the SSSP.

The optimizations above are directed toward minimizing the total computational complexity of our scheme. But we note that the *parallel* computational complexity of scheme is already inherently low, precisely because we require the circuit depth of our decryption to be very low. Even with bootstrapping, our scheme could be extremely efficient in a massively parallel implementation.

Chapter 13

Background on Ideal Lattices II

Over the next few Chapters, we revisit the somewhat homomorphic scheme in an effort to base its security on a weaker computational assumption. Recall that, to make the somewhat homomorphic scheme bootstrappable, we needed to squash the decryption circuit and make an *additional* computational assumption; our efforts below will only weaken the *first* assumption.

As our first refinement to the somewhat homomorphic scheme, we modify the Samp algorithm (used in Encrypt) so that it samples from the coset $\pi + I$ according to a discrete Gaussian distribution centered at the origin and with a deviation parameter not much larger than $\|\mathbf{B}_I\|$. When Encrypt uses this distribution and $\det(I) = \text{poly}(n)$, we show in the next Chapter that security can be based (classically) on the (search) BDDP over ideal lattices generated according to the distribution induced by the IdealGen algorithm used in KeyGen. To generate I, we need some results on ideal factorization in polynomial rings, given here. Beginning in Chapter 16, we show how to obtain a worst-case / average-case connection for ideal lattices and show how to instantiate IdealGen so that it generates ideals according to the average-case distribution.

13.1 Overview of Gaussian Distributions over Lattices

For any real $s > 0$, define the Gaussian function on $\mathbb{R}^n$ centered at $\mathbf{c}$ with parameter s as $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x}-\mathbf{c}\|^2/s^2)$ for all $\mathbf{x} \in \mathbb{R}^n$. The associated *discrete* Gaussian distribution

over L is

$$\forall\ \mathbf{x} \in L, D_{L,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(L)}\ ,$$

where $\rho_{s,\mathbf{c}}(A)$ for set A denotes $\sum_{\mathbf{x}\in A} \rho_{s,\mathbf{c}}(\mathbf{x})$. In other words, the probability $D_{L,s,\mathbf{c}}(\mathbf{x})$ is simply proportional to $\rho_{s,\mathbf{c}}(\mathbf{x})$, the denominator being a normalization factor.

We use the *smoothing parameter* [100] associated to a lattice. The gist for our purposes is that if $I_1 \subset I_2$ are two ideals and s exceeds the smoothing parameter of I_1 – in particular, $s = \lambda_n(I_1) \cdot \omega(\sqrt{\log n})$ – then the distribution $D_{I_2,s,\mathbf{c}}$ samples uniformly from the cosets of I_1; intuitively, the Gaussian is so "fat" that it smears uniformly across the cosets. Relatedly, $\rho_{s,\mathbf{c}}(I_2)/\rho_{s,\mathbf{c}}(I_1) = \det(I_1)/\det(I_2)$, up to negligible factors.

We will repeatedly invoke a technical lemma by Gentry, Peikert and Vaikuntanathan (GPV) [49], which basically states that, given a basis $\mathbf{B}$ of L, one can sample vectors in L according to an arbitrarily precise discrete Gaussian distribution, as long as the deviation of the Gaussian is slightly bigger than $\|\mathbf{B}\|$.

Lemma 13.1.1 (Theorem 3.1 of [49])**.** *For any lattice basis $\mathbf{B} \in \mathbb{Z}^{n\times n}$, any real $s \geq \|\mathbf{B}\| \cdot \omega(\sqrt{\log n})$, and any $\mathbf{c} \in \mathbb{R}^n$, GPV's efficient sampling algorithm* Samp*, on input $(\mathbf{B}, s, \mathbf{c})$, has an output distribution that is within negligible statistical distance of $D_{\mathcal{L}(\mathbf{B}),s,\mathbf{c}}$.*

13.2 The Smoothing Parameter

As in [100], for lattice L and real $\epsilon > 0$, we define the *smoothing parameter* $\eta_\epsilon(L)$ to be the smallest s such that $\rho_{1/s}(L^* \setminus \{\mathbf{0}\}) \leq \epsilon$. We also use a couple of lemmata from [100].

Lemma 13.2.1 (Lemma 3.3 from [100])**.** *For any n-dimensional lattice L and positive real $\epsilon > 0$,*

$$\eta_\epsilon(L) \leq \sqrt{\frac{\ln(2n(1+1/\epsilon))}{\pi}} \cdot \lambda_n(L)$$

In particular, for any superlogarithmic function $\omega(\log n)$, there exists a negligible function $\epsilon(n)$ such that $\eta_\epsilon(L) \leq \lambda_n(L) \cdot \sqrt{\omega(\log n)}$.

Another fact we need says that a sample from a discrete Gaussian with parameter s is at most $s \cdot \sqrt{n}$ away from its center with overwhelming probability.

Lemma 13.2.2 (Lemma A.6 of [49], derived from [100])**.** *For any full-rank n-dimensional lattice L, $\mathbf{c} \in \mathbb{R}^n$, real $\epsilon \in (0,1)$, and $s \geq \eta_\epsilon(L)$, we have*

$$\Pr_{x \leftarrow D_{L,s,\mathbf{c}}}\left[\|\mathbf{x}-\mathbf{c}\| > s \cdot \sqrt{n}\right] \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}$$

The above is a stronger form of Banaszyzck's Lemma [11].

Lemma 13.2.3 (Lemma 1.5 of [11])**.** *For any $c > 1/\sqrt{2\pi}$, n-dimensional lattice L, and vector $\mathbf{v} \in \mathbb{R}^n$,*

$$\begin{aligned} \rho(L \setminus c\sqrt{n}\mathcal{B}) &< C^n \cdot \rho(L) \\ \rho((\mathbf{v}+L) \setminus c\sqrt{n}\mathcal{B}) &< 2C^n \cdot \rho(L) \end{aligned}$$

where $C = c\sqrt{2\pi e} \cdot e^{-\pi c^2} < 1$.

When the smoothing parameter exceeds the smoothing parameter (by which we mean $s \geq \eta_\epsilon(M)$ for negligible ϵ) of lattices $M \subseteq L$, we obtain the following nice properties.

Lemma 13.2.4 (Lemma A.4 of [49], implicit in [100])**.** *Let L be any full-rank n-dimensional lattice. Then for any $s \geq \eta_\epsilon(L)$, real $\epsilon \in (0,1)$ and $\mathbf{c} \in \mathbb{R}^n$, we have $\rho_{s,\mathbf{c}}(L) \in [\frac{1-\epsilon}{1+\epsilon}, 1] \cdot \rho_{s,\mathbf{0}}(L)$.*

Lemma 13.2.5 (Lemma A.5 of [49])**.** *Let L, M be full-rank n-dimensional lattices with $M \subseteq L$. Then for any $\epsilon \in (0, 1/2)$, and $s \geq \eta_\epsilon(M)$, and any $\mathbf{c} \in \mathbb{R}^n$, the distribution of $(D_{L,s,\mathbf{c}} \bmod M)$ is within statistical distance at most 2ϵ of uniform over $(M \bmod L)$.*

Lemma 13.2.6. *Let L, M be full-rank n-dimensional lattices with $M \subseteq L$. Then for any $\epsilon \in (0, 1/2)$, and $s \geq \eta_\epsilon(M)$, and any $\mathbf{c} \in \mathbb{R}^n$, $\rho_{s,\mathbf{c}}(L)/\rho_{s,\mathbf{c}}(M)$ equals $\det(M)/\det(L)$, up to a multiplicative factor at most 2ϵ away from 1.*

Approximately, the sum of two discrete Gaussian distributions is another discrete Gaussian distribution, as is the case when the distributions are continuous. The problem is that this is not entirely true; the discreteness of the original distributions introduces some error. Here, we bound the error.

Lemma 13.2.7. *Let L be a lattice, $\mathbf{c} \in \mathbb{R}^n$ be a vector, $s, t > 0$ be two reals, and $r = \sqrt{s^2+t^2}$. Assume that $\epsilon \leftarrow \rho_{r/st}(L^*/\{\mathbf{0}\})$ satisfies $\epsilon < 1/2$. Then, the statistical difference between $D_{L,s,\mathbf{c}_1} + D_{L,t,\mathbf{c}_2}$ and $D_{L,r,\mathbf{c}_1+\mathbf{c}_2}$ is at most 9ϵ.*

Proof. Consider the continuous distribution Y on $\mathbb{R}^n$ obtained by sampling from $D_{L,s,\mathbf{c}_1}$ and then adding a noise vector from $\rho_{t,\mathbf{c}_2}$. We use Lemma 13.2.8 below to conclude that

$$|Y(\mathbf{x}) - \rho_{r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})/r^n| \leq \rho_{r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})/r^n \cdot 4\epsilon$$

But we know that, by definition, $D_{L,r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})$ simply equals $\rho_{r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})/r^n$, up to the normalization factor $n_1 \leftarrow \sum_{\mathbf{x}\in L} \rho_{r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})$. Similarly, $(D_{L,s,\mathbf{c}_1} + D_{L,t,\mathbf{c}_2})(\mathbf{x})$ equals $Y(\mathbf{x})$, up to the normalization factor $n_2 \leftarrow \sum_{\mathbf{x}\in L} Y(\mathbf{x})$. Since $Y(\mathbf{x}) \in [(1-4\epsilon)\rho_{r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x}), (1+4\epsilon)\rho_{r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})]$ for all $\mathbf{x}$, we have that $n_2 \in [(1-4\epsilon)n_1, (1+4\epsilon)n_1]$. Therefore,

$$(D_{L,s,\mathbf{c}_1} + D_{L,t,\mathbf{c}_2})(\mathbf{x}) \in [(1-4\epsilon)^2 \cdot D_{L,r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x}), (1+4\epsilon)^2 \cdot D_{L,r,\mathbf{c}_1+\mathbf{c}_2}(\mathbf{x})]$$

for all $\mathbf{x}$. After replacing the implicit ϵ^2 by ϵ in the equation above, the result follows by integration.

□

Lemma 13.2.8 (Strengthening of Lemma 3.9 of [119])**.** *Let L be a lattice, $\mathbf{c} \in \mathbb{R}^n$ be a vector, $s, t > 0$ be two reals, and $r = \sqrt{s^2+t^2}$. Assume that $\epsilon \leftarrow \rho_{r/st}(L^*/\{\mathbf{0}\})$ satisfies $\epsilon < 1/2$. Consider the continuous distribution Y on $\mathbb{R}^n$ obtained by sampling from $D_{L+\mathbf{c},s,\mathbf{0}}$ and then adding a noise vector from ρ_t. Then, for all $\mathbf{x} \in \mathbb{R}^n$,*

$$|Y(\mathbf{x}) - \rho_r(\mathbf{x})/r^n| \leq \rho_r(\mathbf{x})/r^n \cdot 4\epsilon$$

The lemma above follows from the Regev's proof of Lemma 3.9 [119].

13.3 Sampling a Lattice According to a Gaussian Distribution

Gentry, Peikert and Vaikuntanathan [49] provide a fast algorithm that is able to sample a lattice L according to (a distribution that is statistically indistinguishable from) a discrete Gaussian distribution for parameter s. The algorithm only requires as input a basis $\mathbf{B}$ of L such that $\|\mathbf{B}\|$ is a little bit less than s.

Theorem 13.3.1 (Theorem 3.1 from [49])**.** *For any lattice basis $\mathbf{B} \in \mathbb{Z}^{n\times k}$, any real $s \geq \|\mathbf{B}\| \cdot \omega(\sqrt{\log n})$, there is an algorithm (called* SampleD *in [49]) whose output distribution*

is within negligible distance of $D_{\mathcal{L}(\mathbf{B}),s,\mathbf{c}}$. *The running time of* SampleD *is polynomial in* n *and the size of its input* $(\mathbf{B}, s, \mathbf{c})$.

The algorithm works by iteratively reducing the problem of sampling from L according to a Gaussian distribution centered at $\mathbf{c}$ to the problem of sampling from subspaces of L of smaller and smaller dimension. Specifically, the sampling algorithm SampleD is as follows. On input $(\mathbf{B}, s, \mathbf{c})$, if $k = 0$ (i.e., $\mathbf{B}$ is empty), return 0. Otherwise:

1. Compute $\widetilde{\mathbf{b}_k}$, the (nonzero) component of $\mathbf{b}_k$ orthogonal to $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$. Compute $\mathbf{t}$, the projection of $\mathbf{c}$ onto $\text{span}(\mathbf{B})$, and the scalar value $t = \frac{\langle \mathbf{t}, \widetilde{\mathbf{b}_k} \rangle}{\langle \widetilde{\mathbf{b}_k}, \widetilde{\mathbf{b}_k} \rangle} \in \mathbb{R}$.
2. Choose an integer $z \leftarrow D_{\mathbb{Z}, s/\|\widetilde{\mathbf{b}_k}\|, t}$.
3. Output $z \cdot \mathbf{b}_k + \mathsf{SampleD}(\mathbf{B}', s, \mathbf{t} - z \cdot \mathbf{b}_k)$ via recursion, where $\mathbf{B}' = \{\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}\}$.

Informally, SampleD samples from a k-dimensional lattice by: (1) sampling a $(k-1)$-dimensional hyperplane containing a $(k-1)$-dimensional sublattice of L according to a 1-dimensional Gaussian distribution, and then (2) sampling a lattice point from this hyperplane. In terms of proving that this algorithm outputs according to the desired distribution, one relevant fact is that an n-dimensional Gaussian distribution is equivalent to the direct product of n 1-dimensional Gaussian distributions, each along orthogonal axes parallel to the respective vectors $\widetilde{\mathbf{b}_k}$. Another relevant fact is that the weight allotted to each hyperplane in the *discrete* Gaussian distribution is indeed proportional (up to a negligible factor) to the weight that the hyperplane would receive in a *continuous* Gaussian distribution. Gentry et al. prove this fact using the smoothing parameter – i.e., s is large enough to make the "discreteness" of the points on the hyperplane irrelevant. See [49] for more details.

13.4 Ideal Factorization in Polynomial Rings

Unique factorization of ideals does not necessarily hold for $R = \mathbb{Z}[x]/(f(x))$. Rather, unique factorization of ideals holds for the ring of integers $\mathcal{O}_K = \{x \in K : f^x_{\mathbb{Q}} \in \mathbb{Z}[x]\}$, where K is the field $\mathbb{Q}(x)/(f(x))$, and $f^x_{\mathbb{Q}}$ is the monic irreducible minimal polynomial of x over $\mathbb{Q}$. The ring R is sometimes called an *order* or a *number ring*. The ring of integers is the *maximal order*; it is a ring that contains R. However, the nice algebraic properties of $\mathcal{O}_K$, such as unique factorization, carry over to R modulo some badness that arises from primes that divide the index $[\mathcal{O}_K : R]$.

For general number rings R, a *prime* ideal $\mathfrak{p}$ is an ideal that is *proper* (i.e., does not contain all of R) and such that for any ideals I and J with $I \cdot J \subseteq \mathfrak{p}$, it holds that $I \subset \mathfrak{p}$ or $J \subset \mathfrak{p}$. Every prime ideal is maximal – i.e., there is no ideal I for which $\mathfrak{p} \subset I \subset R$, where the inclusions are proper. The norm of an ideal I, $\mathrm{Nm}(I)$, is the index $[R : I]$. When $R = \mathbb{Z}[x]/(f(x))$ and the lattice associated to R is simply $\mathbb{Z}^n$, we have that $\mathrm{Nm}(I) = \det(I)$. For any prime ideal, the norm is a prime integer power.

For $R = \mathbb{Z}[x]/(f(x))$ and any prime integer p, there is an efficient algorithm to find all of the prime ideals in R with norms that are a power of p. It uses the following theorem.

Theorem 13.4.1 (Kummer-Dedekind, as given in [127])**.** *Consider the factorization $f(x) = \prod_i g_i(x)^{e_i} \bmod p$ for prime integer p. The prime ideals $\mathfrak{p}_i \in \mathbb{Z}[x]/(f(x))$ whose norms are powers of p are precisely*

$$\mathfrak{p}_i = (p, g_i(x))$$

The inclusion $\prod_i \mathfrak{p}_i^{e_i} \subset (p)$ is an equality iff all $\mathfrak{p}_i$ are invertible.

There are polynomial time algorithms for factoring polynomials in $\mathbb{Z}_p[x]$ – e.g., by Kaltofen and Shoup [75].

For ideal $I \subset R$, the fractional ideal I^{-1} is $\{\mathbf{x} \in K = \mathbb{Q}(x)/(f(x)) : \forall \mathbf{y} \in I,\ \mathbf{x} \times \mathbf{y} \in R\}$. I is said to be *invertible* if $I \cdot I^{-1} = R$. Not all ideals are invertible; a non-invertible ideal is said to be *singular*. As an example (from [127]), consider the ring $\mathbb{Z}[x]/(x^2 + 3)$ and the ideal $I = (1 + x, 1 - x)$. I^2 is generated by $(1 + x)^2 = 2x - 2 \bmod x^2 + 3$, $(1 + x)(1 - x) = 4 \bmod (x^2 + 3)$, and $(1 - x)(1 - x) = -2x - 2 \bmod x^2 + 3$, and therefore $I^2 = 2 \cdot I$. Since $I \neq (2)$, it is clear that I is not invertible. The *norm* of an ideal, defined as its index in the ring, is a *multiplicative* map over invertible ideals. However, for a singular ideal one can have a situation where $[R : I]^2 \neq [R : I^2]$ – e.g., for $I = (1 + x, 1 - x)$ in $\mathbb{Z}[x]/(x^2 + 3)$, we have $[R : I] = 2$ and $[R : I^2] = 8$. In general, the norm of a product of ideals is *at least* the product of the norms. All principal ideals are invertible.

However, only a small number of prime ideals are singular. All such prime ideals have norm p^e for some prime integer p that divides $[\mathcal{O}_K : R]$, and it is straightforward to place an upper bound on this index. Specifically, let $\Delta(f)$ be the discriminant of $f(x)$, and Δ_K be the discriminant of the number field $K = \mathbb{Q}(x)/(f(x))$, and $Res(f, f')$ be the resultant

of $f(x)$ and its derivative. All of these terms are integers, and we have

$$\begin{aligned} \Delta(f) &= [\mathcal{O}_K : R]^2 \cdot \Delta_K \\ &= (-1)^{n(n-1)/2} \cdot Res(f, f') \end{aligned}$$

Viewing $Res(f, f')$ as the determinant of the Sylvester matrix associated to f and f', it is easy to see that $|Res(f, f')|$ is bounded by $n^n \|f\|^{2n}$, where $\|f\|$ is the Euclidean length of the coefficient vector associated to f. (Note that our suggested method of selecting $f(x)$ in Chapter 7.4 already places a polynomial bound on $\|f\|$.) We can bound the number of primes that divide $|Res(f, f')|$, and hence $[\mathcal{O}_K : R]$, by $O(n(\log n + \log \|f\|))$. Singular primes are "rare," and we can easily detect and discard them using Dedekind-Kummer. (See [127] for a survey on general number rings and more details on these issues.)

While ideals I that are divisible by singular primes may not have unique factorization, at least there is no "bad interaction" between ideals that are relatively prime (i.e., have no common prime ideal divisors). In particular, one has the isomorphism $R/I \cong \prod_{\mathfrak{p} \supset I} R/I_{(\mathfrak{p})}$, where $I_{(\mathfrak{p})}$ is the $\mathfrak{p}$-part associated to I. For example, if I has norm 15, it can always be uniquely decomposed as the product of two ideals I_3 and I_5 such that $\mathrm{Nm}(I_{(3)}) = 3$ and $\mathrm{Nm}(I_{(5)}) = 5$. In particular, $I_3 = I + (3)$ – the sum of the ideals I and (3); similarly for I_5. In general, if I's norm is divisible by $p_i^{e_i}$, then I's "p_i-part" is simply $I + (p_i^{e_i})$ and has norm $p_i^{e_i}$. The following equation of ideals holds: $I = \prod_i (I + (p_i^{e_i}))$. Also, if $\mathrm{Nm}(I)$ and $\mathrm{Nm}(J)$ are relatively prime, then $\mathrm{Nm}(I \cdot J) = \mathrm{Nm}(I) \cdot \mathrm{Nm}(J)$.

Chapter 14

The Somewhat Homomorphic Scheme Revisited

In this Chapter, we revisit the somewhat homomorphic ideal lattice based scheme described in Chapters 5 and 7. Our objective is to obtain a security proof based on a weaker assumption. Toward that goal, we use this Chapter to describe some different methods to generate the ideal I and to instantiate the algorithm Samp, which is used in $\mathsf{Encrypt}$ to sample from $\pi + I$. We also provide some security reductions for this revised scheme.

14.1 Using Gaussian Sampling in Encrypt

Briefly, we recall the suggestions for instantiating I and Samp in the initial construction. Recall that $\mathsf{IdealGen}$ ensures that I and J are relatively prime. The suggestion was to pick a short generator $\mathbf{s} \in I$. Then, Samp outputs $\pi + \mathbf{s} \times \mathbf{t}$, where $\mathbf{t} \in R$ is a short vector output by algorithm Samp_1.

The image of our original version of Samp is not necessarily very "nice" – e.g., it may not be "spherical," but may rather be distorted in a way that depends on the ring R. Our new instantiation of Samp uses the GPV $\mathsf{SampleD}$ algorithm, described in Chapter 13.3, to obtain a distribution that is "nicer" and simpler to analyze.

$\mathsf{Samp}(\mathbf{B}_I, \pi\}$ outputs $\pi + \mathsf{SampleD}(\mathbf{B}_I, s, -\pi)$ (where s is a global parameter of the system).

As we learned in Chapter 7, it is straightforward to generate $\mathbf{B}_I$ such that $\|\mathbf{B}_I\| = \text{poly}(n)$. Since the GPV algorithm only requires that $s = \|\mathbf{B}_I\| \cdot \omega(\sqrt{\log n})$ (see Lemma

13.1.1), we can also take $s = \text{poly}(n)$, and by Lemma 13.2.2, the sampled vector (and hence r_{Enc}) is longer than $s \cdot \sqrt{n}$ with only negligible probability.

14.2 Generating an Ideal with Very Small Norm

For one of our reductions – namely, the Hensel lifting one described in Chapter 14.5 – we need a stronger requirement on I – namely, that $\det(I)$ be polynomial in n; otherwise, the reduction is inefficient. Restricting I's determinant to be small will allow us to based security on *search BDDP*, essentially by using a "brute-force" search over the $\det(I)$ cosets of I (in R); for this reduction to be efficient, $\det(I)$ must be polynomial in n. One downside of this approach is that it restricts the plaintext space rather severely. Also, it makes the generation of I conceptually more complex; we cannot simply set I to be (m) for small integer m, because (given that $R = \mathbb{Z}[x]/(f(x))$ for $f(x)$ of degree n) the ideal (m) has m^n cosets. The bottom line here is that, to get our Hensel lifting reduction to work, we need to demonstrate two things: 1) if R has an ideal I of norm $\text{poly}(n)$, then there is an efficient algorithm that outputs a basis $\mathbf{B}_I$ of I such that $\|\mathbf{B}_I\| = \text{poly}(n)$; and 2) $R = \mathbb{Z}[x]/(f(x))$ does indeed have an ideal of norm $\text{poly}(n)$ as long as $f(x)$ satisfies certain conditions (which it will already satisfy if $f(x)$ was selected as suggested in Chapter 7.4). We address the first issue first.

Suppose R has an ideal I with $m \leftarrow \det(I) = \text{poly}(n)$; how do we generate a basis of an ideal that has norm m (either I or some other ideal with norm m)? First, we factor m: suppose $m = \prod_i p_i^{e_i}$. Next, for each i, we find bases $\mathbf{B}_{I_i}$ of ideals I_i that have norm $p_i^{e_i}$. We set $I = \prod_i I_i$, and we can compute a preliminary basis $\mathbf{B}'_I$ of I from the bases $\mathbf{B}_{I_i}$. Finally, we set $\mathbf{B}_I$ to the the Hermite normal form of $\mathcal{L}(\mathbf{B}'_I)$. It remains to explain why this gives the desired output.

Recall the basic algebraic fact (see Chapter 13.4) that $I = \prod_i (I + (p_i^{e_i}))$, and the fact that, if $\text{Nm}(I)$ and $\text{Nm}(J)$ are relatively prime, then $\text{Nm}(I \cdot J) = \text{Nm}(I) \cdot \text{Nm}(J)$. From this, we know that if there is an ideal I in R with norm $\prod_i p_i^{e_i}$, then the ideal $I + (p_i^{e_i})$ has norm $p_i^{e_i}$. To find such an ideal, we use Kummer-Dedekind in connection with a polynomial time algorithm for factoring polynomials in $\mathbb{F}_{p_i}[x]$ to obtain all prime ideals whose norms are powers of p_i, and find a product of them whose norm is $p_i^{e_i}$; let this be I_i. Given bases $\mathbf{B}_I$ and $\mathbf{B}_J$ for ideals I and J, one can efficiently compute a basis for the product $I \cdot J$ simply by computing the set of vectors $\{\mathbf{u} \times \mathbf{v} : \mathbf{u} \in \mathbf{B}_I, \mathbf{v} \in \mathbf{B}_J\}$, and then reducing this set to n

vectors (eliminating the linear dependencies) by applying a lattice reduction algorithm such as LLL [81]. Given the initial basis $\mathbf{B}'_I$ of I, computing I's Hermite normal form (HNF) $\mathbf{B}_I$ is efficient. In $\mathbf{B}_I$, which has positive diagonal entries $m_1, \ldots, m_n$ with $\prod_i m_i = m$, the ith basis vector has length at most $m_i + \sum_{j>i} \lfloor m_j/2 \rfloor$, which is at most $m = \text{poly}(n)$.

Now, we need to explain why $R = \mathbb{Z}[x]/(f(x))$ does indeed have an ideal of norm $\text{poly}(n)$ as long as $f(x)$ satisfies certain conditions. When $f(x)$ is such that $\mathbb{Q}(x)/(f(x))$ is an abelian extension of $\mathbb{Q}$ – e.g., when $f(x)$ is a cyclotomic polynomial – then we can invoke the following theorem.

Theorem 14.2.1 (Theorem 8.7.7 of [10])**.** *Assume GRH. Let K/E be a proper abelian extension of number fields with a conductor $\mathfrak{f}$. Let Δ denote the discriminant of E. Then there is a degree 1 prime ideal $\mathfrak{p}$ of E that does not split in K, relatively prime to Δ and $\mathfrak{f}$, satisfying $\text{Nm}(\mathfrak{p}) = O((\log|\Delta| + \log|\text{Nm}(\mathfrak{f})|^2)$. The constant implied by "O" is absolute.*

Δ and the norm of the conductor $\mathfrak{f}$ each divide $\Delta(f)$, the discriminant of $f(x)$, so we have $\text{Nm}(\mathfrak{p}) = O((\log|\Delta(f)|)^2)$. As discussed in Chapter 13.4, $\Delta(f)$ is bounded by $n^n\|f\|^{2n}$, where $\|f\|$ is the Euclidean length of the coefficient vector of $f(x)$, and therefore $\text{Nm}(\mathfrak{p})$ is polynomial in n as long as $\|f\|$ is polynomial in n. In Chapter 7.4, we suggested a method for choosing $f(x)$ so as to minimize $\gamma_{\mathsf{Mult}}(R)$; polynomials $f(x)$ chosen according to this method already satisfy $\|f\| = \text{poly}(n)$. Note that the relative primeness property in Theorem 14.2.1 implies that I will be an invertible ideal in R (see Chapter 13.4).

Rather than fixing $f(x)$ and afterwards finding an ideal in $R = \mathbb{Z}[x]/(f(x))$ with small norm, one can instead choose $f(x)$ and I together. For example, we can select $f(x)$ as follows. Select a prime p, a linear polynomial $(x - b)$, and a monic polynomial $g(x)$ of degree $n - 1$ such that $(x - b)g(x)$ has only lower-order terms – e.g., the coefficient of x^i in $(x - b)g(x)$ is 0 for $i \in (n/2, n - 1]$, like the values of $f(x)$ that we suggested in Chapter 7.4. Set $f(x) \leftarrow (x - b) \cdot g(x) \bmod p$ – i.e., reduce the nonzero coefficients of $(x - b) \cdot g(x)$ so that they have magnitude at most $p/2$. Then, $f(x)$ factors modulo p into factors that include the linear polynomial $x - b$, and therefore $R = \mathbb{Z}[x]/(f(x))$ (by Theorem 13.4.1) has an ideal of norm p. If p happens to divide $[\mathcal{O}_K : R]$, one samples again. The probability that p divides $[\mathcal{O}_K : R]$ is not overwhelming. If p divides $[\mathcal{O}_K : R]$, then it divides $\Delta(f)$ and also $Res(f, f')$, where f' is the derivative of $f(x)$. This only happens when $f(x)$ and $f'(x)$ have a common root modulo p.

14.3 Proof of Security Based on the Inner Ideal Membership Problem (IIMP)

We base the security of our encryption scheme, when using GPV sampling in $\mathsf{Encrypt}$ as described above, directly on the following problem.

Definition 14.3.1 (Inner Ideal Membership Problem (IIMP))**.** Fix ring R, basis $\mathbf{B}_I$ of an ideal $I \subset R$, and algorithm $\mathsf{IdealGen}$ as in the scheme. Fix a positive real s_{IIMP}. The challenger sets $(\mathbf{B}_J^{\mathrm{sk}}, \mathbf{B}_J^{\mathrm{pk}}) \xleftarrow{\mathrm{R}} \mathsf{IdealGen}(R, \mathbf{B}_I)$. The challenger flips a bit $b \xleftarrow{\mathrm{R}} \{0,1\}$. If $b = 0$, it sets $\mathbf{x} \xleftarrow{\mathrm{R}} D_{I, s_{\mathsf{IIMP}}, \mathbf{0}}$. If $b = 1$, it sets $\mathbf{x} \xleftarrow{\mathrm{R}} D_{R, s_{\mathsf{IIMP}}, \mathbf{0}}$. It sets $\mathbf{t} \leftarrow \mathbf{x} \bmod \mathbf{B}_J^{\mathrm{pk}}$. The problem is: given $(\mathbf{B}_J^{\mathrm{pk}}, \mathbf{t})$ (and the fixed values), output b.

Basically, the IIMP is like the IMP (discussed in Chapter 3) – i.e., does $\mathbf{x}$ belong to the "inner" ideal I? – but the instance is perturbed by the "outer" ideal J.

Theorem 14.3.2. *Let $s_{\mathsf{IIMP}} = s/\sqrt{2}$, where s is the Gaussian deviation parameter in the encryption scheme $\mathcal{E}$ described in Chapter 14.1, and suppose that $s_{\mathsf{IIMP}}/\sqrt{2}$ exceeds the smoothing parameter of I – i.e. $\rho_{\sqrt{2}/s_{\mathsf{IIMP}}}(I^{-1}/\{\mathbf{0}\})$ is negligible. Suppose that there is an algorithm $\mathcal{A}$ that breaks the semantic security of $\mathcal{E}$ with advantage ϵ. Then, there is an algorithm $\mathcal{B}$, running in about the same time as $\mathcal{A}$, that solves the IIMP with advantage $\epsilon/2$ (up to negligible factors).*

Proof. The proof is almost a tautology, the only wrinkles being technical issues relating to Gaussian distributions. The basis $\mathbf{B}_J^{\mathrm{pk}}$ was generated by the challenger using the actual KeyGen algorithm, so $\mathcal{B}$ can use it as a properly distributed public key pk to give to $\mathcal{A}$. When $\mathcal{A}$ requests a challenge ciphertext on one of π_0, π_1 in the plaintext space $R \bmod \mathbf{B}_I$, $\mathcal{B}$ sets $\beta \xleftarrow{\mathrm{R}} \{0,1\}$, and $\mathbf{v} \xleftarrow{\mathrm{R}} D_{I, s_{\mathsf{IIMP}}, -\pi_\beta}$. It sends $\psi \leftarrow \pi_\beta + \mathbf{t} + \mathbf{v} \bmod \mathbf{B}_J^{\mathrm{pk}}$ to $\mathcal{A}$. $\mathcal{A}$ sends back a guess β'. $\mathcal{B}$ sends $b' \leftarrow \beta \oplus \beta'$ to the challenger.

When $b = 0$, $\mathcal{B}$'s simulation is almost perfect. In the real world, ψ is chosen according to the distribution $\pi_\beta + D_{I, s, -\pi_\beta}$ (and then reducing modulo $\mathbf{B}_J^{\mathrm{pk}}$). In the simulation, ψ equals $\pi_\beta + \mathbf{t} + \mathbf{v} = \pi_\beta + \mathbf{x} + \mathbf{v} \bmod \mathbf{B}_J^{\mathrm{pk}}$. In other words, the simulated distribution is $\pi_\beta + D_{I, s_{\mathsf{IIMP}}, -\pi_\beta} + D_{I, s_{\mathsf{IIMP}}, \mathbf{0}}$ (and then reducing modulo $\mathbf{B}_J^{\mathrm{pk}}$). By Lemma 13.2.7, the statistical difference between $D_{I, s, -\pi_\beta}$ and $D_{I, s_{\mathsf{IIMP}}, -\pi_\beta} + D_{I, s_{\mathsf{IIMP}}, \mathbf{0}}$ is negligible. (Essentially, this is because the sum of two Gaussians is another Gaussian, even in the discrete setting, up to small error.) Therefore, when $b = 0$, $\mathcal{A}$'s advantage is ϵ by assumption, up to a negligible additive factor.

When $b = 1$, the challenger chose $\mathbf{x}$ according to $D_{\mathbb{Z}^n,s,\mathbf{0}}$. From Lemma 13.2.4, it follows that this distribution is statistically equivalent to setting $\mathbf{y} \xleftarrow{\mathrm{R}} R \bmod \mathbf{B}_I$ and then generating $\mathbf{x}$ according to $\mathbf{y} + D_{I,s,-\mathbf{y}}$. Thus the challenge ciphertext comes from the distribution $\pi_\beta + D_{I,t,-\pi_\beta} + \mathbf{y} + D_{I,s,-\mathbf{y}}$, where $\mathbf{y}$ is uniformly random modulo I. But, by Lemma 13.2.7, this distribution is statistically equivalent to $\mathbf{z} + D_{I,r,-\mathbf{z}}$ for $\mathbf{z}$ uniformly random modulo I. This implies that $\mathcal{B}$'s challenge ciphertext is (essentially) independent of the bit β, and that $\mathcal{A}$ therefore has negligible advantage in this case.

Overall, $\mathcal{B}$'s advantage is $\epsilon/2$, up to negligible factors.

□

14.4 Success Amplification: Proof of Security Based on the Modified IIMP (MIIMP)

The IIMP is "very average-case" in that it depends on Gaussian distributions and the distribution induced by $\mathsf{IdealGen}$. Since the problem looks almost like the semantic security game, the proof of Theorem 14.3.2 is almost a tautology.

In the following problem, we permit more freedom in how $\mathbf{x}$, and hence the target vector $\mathbf{t}$, is chosen. We also amplify the success probability (over values of $\mathbf{x}$) to be overwhelming for a certain fraction of keys output by $\mathsf{IdealGen}$.

Definition 14.4.1 (Modified Inner Ideal Membership Problem (MIIMP))**.** Fix ring R, basis $\mathbf{B}_I$ of an ideal $I \subset R$, and algorithm $\mathsf{IdealGen}$ as in the scheme. Fix a positive real s_{MIIMP}. The challenger sets $(\mathbf{B}_J^{\mathrm{sk}}, \mathbf{B}_J^{\mathrm{pk}}) \xleftarrow{\mathrm{R}} \mathsf{IdealGen}(R, \mathbf{B}_I)$. The challenger flips a bit $b \xleftarrow{\mathrm{R}} \{0,1\}$. If $b = 0$, it sets $\mathbf{x}$ however it likes, except $\mathbf{x} \in I$ and $\|\mathbf{x}\| < s_{\mathsf{MIIMP}}$. If $b = 1$, it does the same thing, except $\mathbf{x} \notin I$. It sets $\mathbf{t} \leftarrow \mathbf{x} \bmod \mathbf{B}_J^{\mathrm{pk}}$. The problem is: given $(\mathbf{B}_J^{\mathrm{pk}}, \mathbf{t})$ (and the fixed values), output b.

Theorem 14.4.2. *Suppose that $s_{\mathsf{MIIMP}} < s_{\mathsf{IIMP}} \cdot \epsilon/(n \cdot \max\{\|\mathbf{B}_I\|\})$. Suppose also that $[R : I]$ is prime. Suppose that there is an algorithm $\mathcal{A}$ that solves the IIMP with advantage ϵ. Then, there is an algorithm $\mathcal{B}$ that, for a $\epsilon/2$ (weighted) fraction of bases $\mathbf{B}_J^{\mathrm{pk}}$ output by* $\mathsf{IdealGen}$*, solves the MIIMP with overwhelming probability. $\mathcal{B}$'s running time is proportional to $1/\epsilon$ times the running time of $\mathcal{A}$.*

Notice that the success amplification in MIIMP came at a cost: s_{MIIMP} is less than $\epsilon \cdot s_{\mathsf{IIMP}}$ – i.e., the MIIMP-solver requires $\mathbf{t}$ to be *much* closer to the ideal lattice J. Similarly, the

running time is much higher. However, the reduction is polynomial when the IIMP solver has non-negligible advantage.

Proof. Let $\epsilon_{\mathbf{B}_J}$ be $\mathcal{A}$'s advantage given that $\mathbf{B}_J^{\text{pk}}$ is chosen. Let $\mathcal{S}$ be the set of bases for which $\epsilon_{\mathbf{B}_J}$ exceeds $\epsilon/2$. By a standard averaging argument, the probability that an ideal in $\mathcal{S}$ is chosen is at least $\epsilon/2$. From now on, suppose $\mathbf{B}_J^{\text{pk}} \in S$.

$\mathcal{B}$ receives a MIIMP instance $\mathbf{t}$ (along with the fixed values). It sets $\mathbf{u} \xleftarrow{\text{R}} R \bmod \mathbf{B}_I$. It sets $\mathbf{x}' \xleftarrow{\text{R}} \mathbf{u} \times \mathbf{t} + D_{I,s_{\mathsf{IIMP}},\mathbf{0}}$ and $\mathbf{t}' \leftarrow \mathbf{x}' \bmod \mathbf{B}_J^{\text{pk}}$. It sends $(\mathbf{B}_J^{\text{pk}}, \mathbf{t}')$ as the instance of IIMP to $\mathcal{A}$. $\mathcal{A}$ sends back a bit b', which $\mathcal{A}$ forwards to the challenger.

Recall that in the IIMP problem, when $b = 0$, the challenger sets $\mathbf{x}' \xleftarrow{\text{R}} D_{I,s_{\mathsf{IIMP}},\mathbf{0}}$, sets $\mathbf{t}' \leftarrow \mathbf{x}' \bmod \mathbf{B}_J^{\text{pk}}$, and sends $(\mathbf{B}_J^{\text{pk}}, \mathbf{t}')$. We claim that, up to the reduction modulo $\mathbf{B}_J^{\text{pk}}$, $\mathcal{B}$ generates $\mathbf{x}'$ according to a nearly identical distribution. In particular,

$$\mathbf{x}' \xleftarrow{\text{R}} \mathbf{u} \times \mathbf{t} + D_{I,s_{\mathsf{IIMP}},\mathbf{0}} \cong \mathbf{u} \times \mathbf{x} + D_{I,s_{\mathsf{IIMP}},\mathbf{0}} \cong D_{I,s_{\mathsf{IIMP}},\mathbf{u}\times\mathbf{x}} \bmod \mathbf{B}_J^{\text{pk}}$$

where $\mathbf{u}$ and $\mathbf{x}$ are very short, and the product is in I. Since the reduction modulo $\mathbf{B}_J^{\text{pk}}$ cannot increase the statistical difference between the distributions (real and simulated) according to which $\mathbf{x}'$ is generated, it suffices to upper bound the statistical difference introduced by translating the distribution $D_{I,s_{\mathsf{IIMP}},\mathbf{0}}$ by $\mathbf{u} \times \mathbf{x}$. Note that s_{IIMP} exceeds the smoothing parameter of I. Also, since an n-dimensional Gaussian is a product of n 1-dimensional Gaussians, and the translation occurs only in the direction of $\mathbf{u} \times \mathbf{x}$, it suffices to consider the statistical difference by translating a 1-dimensional Gaussian. One can show that the statistical difference is at most $\epsilon/4$ when $\|\mathbf{u} \times \mathbf{t}\|$ is less than $s_{\mathsf{IIMP}} \cdot \epsilon/4$. .

When the challenger's bit is 1, the situation is analogous, except that $\mathcal{B}$ simulates sampling from $D_{R,s_{\mathsf{IIMP}},\mathbf{0}}$ by choosing $\mathbf{u}$ randomly modulo I and sampling from $\mathbf{u} \times \mathbf{t} + D_{I,s_{\mathsf{IIMP}},\mathbf{0}} \cong \mathbf{u} \times \mathbf{x} + D_{I,s_{\mathsf{IIMP}},\mathbf{0}} \bmod \mathbf{b}_J^{\text{pk}}$, where $\mathbf{x} \notin I$. Since $[R : I]$ is prime, implying the ideal $(\mathbf{x})$ is relatively prime to I, and since $\mathbf{u}$ is uniform modulo I, the simulated distribution is statistically equivalent.

So, $\mathcal{A}$'s advantage for $\mathcal{B}$'s IIMP instance is at least $\epsilon/2 - \epsilon/4 = \epsilon/4$. With $\theta(1/\epsilon)$ calls to $\mathcal{A}$, $\mathcal{B}$ can solve its MIIMP instance with constant probability, and can make its success probability arbitrarily close to 1 with more calls.

□

14.5 Basing Security on a Search Problem: Bounded Distance Decoding Via Hensel Lifting

Here, we reduce a search problem (BDDP) to a decision problem (MIIMP). To be efficient, this is reduction requires $[R : I] = \text{poly}(n)$.

Definition 14.5.1 (I-Hybrid Bounded Distance Decoding Problem (HBDDP))**.** Fix ring R, basis $\mathbf{B}_I$ of an ideal $I \subset R$, and algorithm $\mathsf{IdealGen}$ as in the scheme. Fix a positive real s_{HBDDP}. The challenger sets $(\mathbf{B}_J^{\text{sk}}, \mathbf{B}_J^{\text{pk}}) \xleftarrow{\text{R}} \mathsf{IdealGen}(R, \mathbf{B}_I)$. The challenger sets $\mathbf{x}$ subject to the constraint that $\|\mathbf{x}\| < s_{\mathsf{HBDDP}}$ and sets $\mathbf{t} \leftarrow \mathbf{x} \bmod \mathbf{B}_J^{\text{pk}}$. The problem is: given $(\mathbf{B}_J^{\text{pk}}, \mathbf{t})$ (and the fixed values), output $\mathbf{x}$.

Theorem 14.5.2. *Suppose* $s_{\mathsf{HBDDP}} \leq (s_{\mathsf{MIIMP}} - 2n \cdot \|\mathbf{B}_I\|)/(\gamma_{\mathsf{Mult}}(R) \cdot (\sqrt{n}/2))$*. Suppose that there is an algorithm* $\mathcal{A}$ *that, for a* ϵ *(weighted) fraction of bases* $\mathbf{B}_J^{\text{pk}}$ *output by* $\mathsf{IdealGen}$*, solves the MIIMP with overwhelming probability. Then, there is an algorithm* $\mathcal{B}$ *that, for a* ϵ *(weighted) fraction of bases* $\mathbf{B}_J^{\text{pk}}$ *output by* $\mathsf{IdealGen}$*, solves HBDDP with overwhelming probability.* $\mathcal{B}$*'s running time is only polynomially larger than that of* $\mathcal{A}$*, as long as* $[R : I] = \text{poly}(n)$*.*

Essentially, the reduction $\mathcal{B}$ works by invoking $\mathcal{A}$ repeatedly to recover $\mathbf{x} \bmod I^k$ for increasingly large k – in effect, a Hensel lift, but with geometric aspects, since $\mathcal{B}$ must construct valid MIIMP instances. Specifically, in the kth iteration, the MIIMP target vector $\mathbf{t}_k$ must actually be close to the lattice J. Once the method recovers the value of $\mathbf{x}$ modulo I^k for sufficiently large k, we can use LLL (more properly, Babai's algorithm for CVP) to recover $\mathbf{x}$ from $\mathbf{x} \bmod I^k$.

More specifically, in the first iteration of the Hensel lift, algorithm $\mathcal{B}$ has $\mathbf{t} = \mathbf{x} \bmod \mathbf{B}_J^{\text{pk}}$ and wants to find $\mathbf{x} \bmod \mathbf{B}_I$. For this, $\mathcal{B}$ we can use a MIIMP-solver $\mathcal{A}$ as follows. It gives $\mathbf{t}$ to $\mathcal{A}$; if $\mathcal{A}$ responds '0' – i.e., that the underlying value of $\mathbf{x}$ is in I – then $\mathcal{B}$ has its answer. Otherwise, it picks some nonzero $\mathbf{u} \in R \bmod \mathbf{B}_I$, and give $\mathbf{t}' \leftarrow \mathbf{t} - \mathbf{u}$ to $\mathcal{A}$. Assuming that this is a valid MIIMP instance, $\mathcal{A}$ returns '0' precisely when $\mathbf{x} = \mathbf{u} \bmod \mathbf{B}_I$. In this fashion, $\mathcal{B}$ searches through the cosets of I until $\mathcal{A}$ indicates that it has found the coset containing $\mathbf{x}$. For this part to be efficient, we require $\det(I)$ to be polynomial in n. Let $\mathbf{r}_0 \in R \bmod \mathbf{B}_I$ be such that $\mathbf{x} \in \mathbf{r}_0 + I$. The value $\mathbf{r}_0$ is certainly useful information in our search for $\mathbf{x}$, but obviously it is not enough information to allow us to recover $\mathbf{x}$ completely. We would like to recover the value of $\mathbf{x}$ modulo a much higher power of I.

How do we perform the next iteration in the Hensel lift? Conceptually, we would like to give our MIIMP-solver $\mathcal{A}$ an instance that is somehow related to $(\mathbf{x} - \mathbf{r}_0)/I$, but what does this mean in practice? Of course, $\mathcal{B}$ has $\mathbf{t}$, not $\mathbf{x}$, and $\mathbf{t} - \mathbf{r}_0$ is not necessarily in I. However, J is relatively prime to I; thus, we can find $\mathbf{v}_0 \in J$ such that $\mathbf{t} - \mathbf{r}_0 - \mathbf{v}_0 \in I$. This vector will have the form $\mathbf{v} + (\mathbf{x} - \mathbf{r}_0)$, where $\mathbf{v} \in IJ$ and $\mathbf{x} - \mathbf{r}_0 \in I$. We then "divide by I" by multiplying by some vector $\mathbf{a} \in I^{-1} \setminus R$; our next MIIMP instance is essentially $\mathbf{t}_1 \leftarrow \mathbf{a} \times (\mathbf{t} - \mathbf{r}_0 - \mathbf{v}_0)$. This equals $\mathbf{a} \times (\mathbf{v} + (\mathbf{x} - \mathbf{r}_0)) = (\mathbf{a} \times \mathbf{v}) + \mathbf{a} \cdot (\mathbf{x} - \mathbf{r}_0)$, where $\mathbf{a} \times \mathbf{v} \in J$ (since $\mathbf{v} \in IJ$), and the new "error" vector $\mathbf{a} \cdot (\mathbf{x} - \mathbf{r}_0)$ is in R, and is also still very short if $\mathbf{a}$ is a suitably short vector. In other words, $\mathbf{t}_1$ becomes a new, potentially valid MIIMP instance. To perform the Hensel lift, we will recursively generate MIIMP instances $\mathbf{t}_k$ that will allow us to recover $\mathbf{x}$ modulo higher powers I^k. However, how do we ensure that $\mathbf{a}$ is a very short vector in $I^{-1} \setminus R$? We know that R is a sub-lattice of I^{-1}; therefore, given any single vector in $I^{-1} \setminus R$, we can reduce it modulo the ideal (1) to obtain a vector whose length is no more than $\sqrt{n}/2$. Extending this approach to later iterations in the most obvious way would involve using powers $\mathbf{a}$, which would be quite long even though $\mathbf{a}$ is short; addressing this problem involves additional subtleties. The detailed proof is below.

Proof. $\mathcal{B}$ is given a HBDDP instance $\mathbf{t}$. $\mathcal{B}$'s goal is to return $\mathbf{x}$, where $\mathbf{t} - \mathbf{x}$ is the closest J-vector to $\mathbf{t}$.

For $i = 0$ to m (for m to be determined), $\mathcal{B}$ does the following. At the beginning of iteration i, $\mathcal{B}$ has values $\mathbf{t}_i$ and $\mathbf{r}_i$. It initializes $\mathbf{t}_0 \leftarrow \mathbf{t}$ and $\mathbf{r}_0 \leftarrow \mathbf{0}$. We will let $\mathbf{x}_i$ denote the implicit unknown vector such that $\mathbf{t}_i - \mathbf{x}_i$ is the closest J-vector to $\mathbf{t}_i$.

$\mathcal{B}$ also uses some values that can be set independently of the $\mathbf{t}_i$'s and $\mathbf{r}_i$'s:

- $\mathbf{a}_i$ is a short vector in $I^{-i} \setminus I^{-i+1}$
- $\mathbf{b}_i \in R \setminus I$ is a vector such that $\mathbf{b}_i \times \mathbf{a}_i^{-1} \in I^i \setminus I^{i+1}$
- $\mathbf{c}_i = \mathbf{b}_i^{-1} \bmod I^{i+1} \in R$

It uses $\mathbf{a}_0 = 1$. Note that, since R is a sub-lattice of I^{-i}, it is easy to compute a vector $\mathbf{a}_i \in I^{-i} \setminus I^{-i+1}$ of length at most $\sqrt{n}/2$ from any vector in $I^{-i} \setminus I^{-i+1}$ by reducing it modulo the cube generated by the vector 1.

For all i, we claim that the following invariants will hold:

1. $\mathbf{r}_i = \mathbf{x} \bmod I^i$

2. $\mathbf{t}_i \in R$ with $\mathbf{x}_i = \mathbf{a}_i \times \mathbf{x} - (\mathbf{a}_i \times \mathbf{r}_i \bmod \mathbf{B}_I)$

The parenthesized term $(\mathbf{a}_i \times \mathbf{r}_i \bmod \mathbf{B}_I)$ means that it is reduced modulo $\mathbf{B}_I$ before interacting with other terms. The invariants clearly hold for $i = 0$.

For $\mathbf{u} \in R \bmod \mathbf{B}_I$, $\mathcal{B}$ sets $\mathbf{t}'_i \leftarrow \mathbf{t}_i - \mathbf{u} \bmod \mathbf{B}_J^{\mathrm{pk}}$ and gives $\mathbf{t}'_i$ to $\mathcal{A}$ as its MIIMP instance. Assuming $\mathbf{t}'_i$ is a valid instance of MIIMP, $\mathcal{A}$ responds by telling $\mathcal{B}$ whether or not $\mathbf{x}_i - \mathbf{u} \in I$ with probability arbitrarily close to 1. $\mathcal{B}$ thereby discovers the value of $\mathbf{u}_i \leftarrow \mathbf{x}_i \bmod \mathbf{B}_I$ in time linear in $[R : I]$.

$\mathcal{B}$ then sets

$$\begin{aligned} \mathbf{r}_{i+1} &\leftarrow \mathbf{r}_i + \mathbf{c}_i \times \mathbf{b}_i \times \mathbf{a}_i^{-1} \times \mathbf{u}_i \\ \mathbf{t}_{i+1} &\leftarrow \mathbf{a}_{i+1} \times (\mathbf{t}_0 - \mathbf{v}_{i+1}) - (\mathbf{a}_{i+1} \times \mathbf{r}_{i+1} \bmod \mathbf{B}_I) \end{aligned}$$

where $\mathbf{v}_{i+1} \in J \cap (\mathbf{t}_0 - \mathbf{r}_{i+1} + I^{i+1})$. (Such a $\mathbf{v}_{i+1}$ exists because I and J are relatively prime.)

Assume that the invariants hold for i; we show that they hold for $i + 1$.

From the second invariant for i, we conclude that the values $\mathbf{t}'_i$ used by $\mathcal{B}$ are indeed valid MIIMP instances, since $\max\{\|\mathbf{x}_i\| + \|\mathbf{u}\|\} \leq \gamma_{\mathsf{Mult}}(R) \cdot (\sqrt{n}/2) \cdot \|\mathbf{x}\| + 2n \cdot \|\mathbf{B}_I\|$, as required. Then, assuming $\mathcal{A}$'s response $\mathbf{u}_i$ is correct (as it should be with probability arbitrarily close to 1), and using our assumption above that $\mathbf{x}_i = \mathbf{a}_i \times \mathbf{x} - (\mathbf{a}_i \times \mathbf{r}_i \bmod \mathbf{B}_I)$, we have that

$$\mathbf{u}_i = \mathbf{a}_i \times (\mathbf{x} - \mathbf{r}_i) \bmod I$$

Multiplying by $\mathbf{a}_i^{-1}$, then $\mathbf{b}_i$, then, $\mathbf{c}_i$, we obtain

$$\begin{aligned} \mathbf{x} &= \mathbf{r}_i + \mathbf{a}_i^{-1} \times \mathbf{u}_i \bmod I \cdot (\mathbf{a}_i^{-1}) \\ \mathbf{b}_i \times \mathbf{x} &= \mathbf{b}_i \times \mathbf{r}_i + \mathbf{b}_i \times \mathbf{a}_i^{-1} \times \mathbf{u}_i \bmod I^{i+1} \\ \mathbf{x} &= \mathbf{r}_i + \mathbf{c}_i \times \mathbf{b}_i \times \mathbf{a}_i^{-1} \times \mathbf{u}_i \bmod I^{i+1} \end{aligned}$$

implying that $\mathbf{r}_{i+1}$ is correct.

As for $\mathbf{t}_{i+1}$, consider the vector

$$\mathbf{t}_{i+1}^{\dagger} \leftarrow \mathbf{a}_{i+1} \times (\mathbf{t}_0 - \mathbf{v}_{i+1} - \mathbf{r}_{i+1}) = \mathbf{a}_{i+1} \times ((\mathbf{t}_0 - \mathbf{v}_{i+1} - \mathbf{x}) + (\mathbf{x} - \mathbf{r}_{i+1}))$$

This vector is in R, since both $\mathbf{t}_0 - \mathbf{v}_{i+1} - \mathbf{x}$ and $\mathbf{x} - \mathbf{r}_{i+1}$ are in I^{i+1}; canceling the

"denominator" of $\mathbf{a}_{i+1}$. We have that $\mathbf{t}_{i+1} - \mathbf{t}^{\dagger}_{i+1} = \mathbf{a}_{i+1} \times \mathbf{r}_{i+1} - (\mathbf{a}_{i+1} \times \mathbf{r}_{i+1} \bmod \mathbf{B}_I) \in I \subset R$. Since $\mathbf{t}^{\dagger}_{i+1} - \mathbf{a}_{i+1} \times (\mathbf{x} - \mathbf{r}_{i+1}) \in J$, the vector $\mathbf{t}_{i+1} - \mathbf{a}_{i+1} \times \mathbf{x} + (\mathbf{a}_{i+1} \times \mathbf{r}_{i+1} \bmod \mathbf{B}_I) \in J$, as required.

Finally, we describe how to recover $\mathbf{x}$ from $\mathbf{x} \bmod I^m$ for large-enough m. Babai's algorithm can recover $\mathbf{x}$ from $\mathbf{x} + I^m$ when $\mathbf{x}$ is exponentially (in n) shorter than any other vector in $\mathbf{x} + I^m$; so, it suffices to show that the $\lambda_1(I^m)$ grows exponentially with m. Let $\mathbf{v}_m \in I^m$ be such that $\|\mathbf{v}_m\| = \lambda_1(I^m)$. Also, assume that the ring R has no zero-divisors, implying that for any $\mathbf{v} \in R$, the vectors $\{\mathbf{v} \times x^i : i \in [0, n-1]\}$ are linearly independent. We have

$$\det(I^m) \leq \prod_{i=0}^{n-1} \|\mathbf{v}_m \times x^i\| \leq \gamma_{\mathsf{Mult}}(R)^{n-1} \|\mathbf{v}_m\|^n$$

which implies that

$$\lambda_1(I^m) \geq (\det(I))^{m/n} / \gamma_{\mathsf{Mult}}(R)$$

□

14.6 Toward Reducing the SIVP to the BDDP: Regev's Quantum Reduction

Here we base the security of our scheme on the following problem.

Definition 14.6.1 (Ideal Independent Vector Improvement Problem (IVIP))**.** Fix ring R and a positive real s_{IVIP}. Let $\mathbf{B}_J$ be a basis for an ideal lattice J of R. The problem is: given $\mathbf{B}_J$ (and the fixed values), output an independent set $\mathbf{B}_{J^{-1}}$ of the fractional ideal J^{-1} such that $\|\mathbf{B}_{J^{-1}}\| \leq 1/s_{\mathsf{IVIP}}$.

Since J is an integer (non-fractional) ideal of R, we know that R (i.e., $\mathbb{Z}^n$) is a sub-lattice of J^{-1}. Therefore, we "trivially" know the independent set $\{\mathbf{e}_i\}$ of J^{-1}. When $s_{\mathsf{IVIP}} > 1$, the IVIP asks one to "improve" upon this trivial independent set. We have the following theorem.

Theorem 14.6.2. *Let J be an ideal lattice in R. Suppose $s_{\mathsf{IVIP}} \leq s_{\mathsf{HBDDP}}/(n^{1.5} \cdot \|f\|)$ and $s_{\mathsf{HBDDP}} \leq \lambda_1(J)/2$. Suppose that there is a classical algorithm $\mathcal{A}$ that solves s_{HBDDP}-HBDDP*

for J (with probability 1). Then, there is a quantum algorithm $\mathcal{B}$ that solves s_{IVIP}-IVIP for J.

Actually, "hybridness" of the hybrid BDDP does not manifest itself in Theorem 14.6.2, since the ideal J is fixed. For every algorithm $\mathsf{IdealGen}$ that generates J according to some distribution, we can say that IVIP for that distribution reduces to HBDDP for that distribution. So, this reduction will also be useful for reducing worst-case IVIP to worst-case BDDP.

The proof of Theorem 14.6.2 invokes the following lemma by Regev [119].

Lemma 14.6.3 (Lemma 3.13 in [119])**.** *Let L be an n-dimensional lattice and assume that there exists an oracle that answers $CVP_{L,d}$ (the target vector's distance from L is bounded by d) queries for some $d < \lambda_1(L)/2$. Then, there exists a* quantum *algorithm for sampling from the distribution $D_{L^*,\sqrt{n}/d}$.*

Notice that Regev's reduction is not exactly from the SIVP, since the vectors output by the sampling algorithm, which by Lemma 13.2.2 will be of length less than n/d with overwhelming probability, are not necessarily short relative to the shortest vectors in L^*. We prefer to think of Regev's reduction as being more analogous to the IVIP, since when L is an integer lattice, it outputs an independent set of L^* that is better than the trivial set $\{\mathbf{e}_i\}$ when $n/d < 1$.

Proof. (Theorem 14.6.2) Algorithm $\mathcal{A}$ solves $\mathrm{CVP}_{J,s_{\mathsf{HBDDP}}}$ by assumption. So, by Lemma 14.6.3, there is a quantum algorithm that samples from the distribution $D_{J^*,\sqrt{n}/s_{\mathsf{HBDDP}}}$.

We have $\sqrt{n}/s_{\mathsf{HBDDP}} \geq 2\sqrt{n}/\lambda_1(J) \geq 2\lambda_n(J^*)$ (the latter inequality by transference theorems for general lattices). Let $\mathbf{v}$ be the shortest nonzero vector in J^*. There is a non-negligible probability that a vector drawn according to the distribution above is a nonzero vector. By Lemma 13.2.3, the probability that a vector longer than n/s_{HBDDP} is drawn is negligible.

Let $\mathbf{w} \in J^*$ be a vector of length at most n/s_{HBDDP} drawn by Regev's algorithm. By Lemma 8.1.2, we can use $\mathbf{w}$ to generate an independent set $\mathbf{B}_{J^{-1}}$ of J^{-1} with $\|\mathbf{B}_{J^{-1}}\| \leq \sqrt{n} \cdot \|f\| \cdot \|\mathbf{w}\| \leq n^{1.5} \cdot \|f\|/s_{\mathsf{HBDDP}}$.

□

Interestingly, we do not need to use Regev's algorithm to generate an independent set of J^*. We only need the algorithm to generate a *single* vector of J^*, which we can use to

generate an independent set of J^{-1}.

14.7 Summary of Security Results for this Construction So Far

Collecting Theorems 14.3.2, 14.4.2, 14.5.2 and 14.6.2, we have the following corollary.

Corollary 14.7.1. Suppose that

$$\begin{aligned} s_{\mathsf{IVIP}} &\leq s_{\mathsf{HBDDP}}/(n^{1.5} \cdot \|f\|) \\ s_{\mathsf{HBDDP}} &\leq (s_{\mathsf{MIIMP}} - 2n \cdot \|\mathbf{B}_I\|)/(\gamma_{\mathsf{Mult}}(R) \cdot (\sqrt{n}/2)) \\ s_{\mathsf{MIIMP}} &< s_{\mathsf{IIMP}} \cdot \epsilon/(n \cdot \max\{\|\mathbf{B}_I\|\}) \\ s_{\mathsf{IIMP}} &= s/\sqrt{2} \end{aligned}$$

where s is the Gaussian deviation parameter in the encryption scheme $\mathcal{E}$ described in Chapter 14.1. Also suppose that $s/2$ exceeds the smoothing parameter of I, that $\mathsf{IdealGen}$ always outputs an ideal J with $s \cdot \sqrt{n} < \lambda_1(J)$, and that $[R : I]$ is prime. Finally, suppose that there is an algorithm $\mathcal{A}$ that breaks the semantic security of $\mathcal{E}$ with advantage ϵ. Then, there is a classical algorithm $\mathcal{B}_1$ that solves s_{HBDDP}-HBDDP for an $\epsilon/4$ (up to negligible factors) weight fraction of bases output by $\mathsf{IdealGen}$. The running time of $\mathcal{B}_1$ is $\mathcal{O}([R : I] \cdot (1/\epsilon) \cdot \mathrm{time}(\mathcal{A}))$. Also, there is a quantum algorithm that solves s_{IVIP}-IVIP for an $\epsilon/4$ (up to negligible factors) weight fraction of bases output by $\mathsf{IdealGen}$.

14.8 Looking Forward

So far, we have managed to base the security of our revised initial construction on average-case IVIP – average-case in the sense that the IVIP instance depends on the average-case distribution induced by the algorithm $\mathsf{IdealGen}$. This is quite nice; IVIP is closely related to SIVP (a natural hard problem over lattices), and it is certainly not uncommon for a cryptosystem to be based on the average-case hardness of the underlying hard problems – i.e., the assumption is that the problem is hard over a sampleable distribution of instances.

All other things being equal, however, it would be preferable to base the security on the hardness of solving *worst-case* instances of the problem. Ajtai [2] established that, for certain lattice problems (e.g., SVP), one can reduce worst-case instances to average-case instances,

though the approximation factor in the worst-case instance is larger by a factor polynomial in n. The possibility of using such a *worst-case / average-case connection*, thereby basing security on worst-case hardness, is part of the appeal of lattice-based cryptography.

Over the next few Chapters, we will describe an IdealGen algorithm that generates a "nice" average-case distribution of "random" ideal lattices. We will also describe how to "randomize" a worst-case BDDP instance over ideal lattices to obtain an average-case instance according to the same average-case distribution generated by IdealGen – i.e., we describe a worst-case / average-case reduction. Our worst-case / average-case reduction is qualitatively different from previous ones, including such reductions involving ideal lattices [98, 111, 112, 88, 99]. Most notably, unlike previous examples, our worst-case and average-case lattice problems are over lattices of the same dimension. This average-case / worst-case connection is rather technical, and requires us to closely consider the distribution of ideals in number rings. We provide some background on these issues in the next Chapter.

Chapter 15

Background on Ideal Lattices III

15.1 Lemmata Regarding Vectors Nearly Parallel to $\mathbf{e}_1$

In some of our reductions, and also in our instantiation of $\mathsf{IdealGen}$, we will generate a vector $\mathbf{v} \in t \cdot \mathbf{e}_1 + \mathcal{B}(s)$ where s is much smaller than t – i.e., $\mathbf{v}$ is "nearly parallel" to $\mathbf{e}_1$.

Since such a $\mathbf{v}$ is very close to being simply the scalar t, we would expect $\mathbf{v}$ to behave almost like a real number, and the lattice $(\mathbf{v})$ to behave almost like a scaling of $\mathbb{Z}^n$. The following lemmata characterize this intuition more formally. These lemmata apply to $\mathbf{v} \in \mathbb{Q}[x]/(f(x))$.

The first lemma basically says that if $\mathbf{v}$ is nearly parallel to $\mathbf{e}_1$, then it is within a small factor of being the shortest nonzero vector in the lattice $(\mathbf{v})$.

Lemma 15.1.1. *Let* $\mathbf{v} = \mathbf{e}_1 + \mathbf{u}$. *Then,* $\lambda_1((\mathbf{v})) \geq 1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|$.

Proof. Let $\mathbf{a} \in (\mathbf{v}) \setminus \mathbf{0}$. Then, there is some vector $\mathbf{b} \in R \setminus \mathbf{0}$ such that $\mathbf{a} = \mathbf{b} \times \mathbf{v}$. We have

$$\begin{aligned} \|\mathbf{a}\| &= \|\mathbf{b} + \mathbf{b} \times \mathbf{u}\| \\ &\geq \|\mathbf{b}\| - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{b}\| \cdot \|\mathbf{u}\| \\ &\geq \|\mathbf{b}\| \cdot (1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|) \\ &\geq 1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| \end{aligned}$$

□

We have a similar statement regarding $\lambda_n((\mathbf{v}))$.

Lemma 15.1.2. *Let* $\mathbf{v} = \mathbf{e}_1 + \mathbf{u}$. *If the ring* R *has no zero divisors, then* $\lambda_n((\mathbf{v})) \leq 1 + \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|$. *In particular, this upper bound holds for the linearly independent vectors* $\{\mathbf{v} \times x^i : i \in [0, n-1]\}$.

Proof. For the rotation basis $\{\mathbf{v} \times x^i : i \in [0, n-1]\}$ of $(\mathbf{v})$, we have $\|\mathbf{v} \times x^i\| = \|\mathbf{e}_i + \mathbf{u} \times x^i\| \leq 1 + \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|$.

□

When we sample a vector $\mathbf{v}$ according to the Gaussian distribution $D_{\mathbb{Z}^n, s, t \cdot \mathbf{e}_1}$, we would like to say that, for well-chosen s and t, $\mathbf{v}$ is the only vector $(\mathbf{v})$ that is contained in the ball $t \cdot \mathbf{e}_1 + s\sqrt{n}\mathcal{B}$, and that actually the weight of the Gaussian distribution is negligible over $(\mathbf{v}) \setminus \{\mathbf{v}\}$.

Lemma 15.1.3. *Let* $\mathbf{v} = \mathbf{e}_1 + \mathbf{u}$ *with* $\|\mathbf{u}\| = 1/(\delta \cdot \gamma_{\mathsf{Mult}}(R))$ *for* $\delta > 3$ *and* $\gamma_{\mathsf{Mult}}(R) > 1$. *Then,* $\mathbf{v}$ *is the only vector in* $(\mathbf{v})$ *that is within a distance of* $((\delta - 2)/\delta)$ *of* $\mathbf{e}_1$.

Proof. By Lemma 15.1.1,

$$\lambda_1((\mathbf{v})) \geq 1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| = 1 - 1/\delta$$

Therefore, any vector in $(\mathbf{v})$ other than $\mathbf{v}$ is at least

$$(1 - 1/\delta) - 1/(\delta \cdot \gamma_{\mathsf{Mult}}(R)) > (\delta - 2)/\delta$$

away from $\mathbf{e}_1$.

□

The next two lemmas say that if $\mathbf{v}$ is close to 1, then so is its inverse, and, in fact, inversion nearly preserves distance from 1 in a way that also preserves certain Gaussian quantities.

Lemma 15.1.4. *If* $\|\mathbf{u}\| < 1/\gamma_{\mathsf{Mult}}(R)$, *then*

$$\mathbf{e}_1/(\mathbf{e}_1 - \mathbf{u}) = \mathbf{e}_1 + \mathbf{u} + \mathbf{x} \quad \text{for} \quad \|\mathbf{x}\| \leq \frac{\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|^2}{1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|}$$

Proof. Since $\|\mathbf{u}\| < 1/\gamma_{\mathsf{Mult}}(R)$, we have that

$$\lim_{k \to \infty} \|\mathbf{u}^k\| \leq \lim_{k \to \infty} \gamma_{\mathsf{Mult}}(R)^{k-1} \|\mathbf{u}\|^k = 0$$

Thus, $\mathbf{e}_1/(\mathbf{e}_1 - \mathbf{u}) = \mathbf{e}_1 + \mathbf{u} + \mathbf{u}^2 + \cdots$. The length of this vector's difference from $\mathbf{e}_1 + \mathbf{u}$ is at most:

$$\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|^2 + \gamma_{\mathsf{Mult}}(R)^2 \cdot \|\mathbf{u}\|^3 + \cdots = \frac{\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|^2}{1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|}$$

□

Lemma 15.1.5. *Let* $\mathbf{v} = \mathbf{e}_1 + \mathbf{u}$ *and* $\mathbf{w} = 1/\mathbf{v}$ *(the inverse in* $\mathbb{Q}(x)/((f(x)))$*. Suppose* $\|\mathbf{u}\| \leq 1/2\gamma_{\mathsf{Mult}}(R)$ *and* $\|\mathbf{u}\|^3 \leq \beta \cdot \sigma^2/\gamma_{\mathsf{Mult}}(R)$*. Then*

$$\rho_{\sigma,\mathbf{e}_1}(\mathbf{w})/\rho_{\sigma,\mathbf{e}_1}(\mathbf{v}) \in [e^{-6\beta\cdot\pi}, e^{6\beta\cdot\pi}]$$

Proof. Let $\mathbf{x}$ be such that $\mathbf{w} = \mathbf{e}_1 - \mathbf{u} + \mathbf{x}$. By Lemma 15.1.4,

$$\|\mathbf{x}\| \leq \frac{\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|^2}{1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|}$$

We have

$$\frac{\rho_{\sigma,\mathbf{e}_1}(\mathbf{w})}{\rho_{\sigma,\mathbf{e}_1}(\mathbf{v})} = \frac{e^{-\pi\|(-\mathbf{u}+\mathbf{x})\|^2/\sigma^2}}{e^{-\pi\|\mathbf{u}\|^2/\sigma^2}}$$

This latter quantity is clearly in the interval

$$\left[e^{-\pi\|\mathbf{x}\|(2\|\mathbf{u}\|+\|\mathbf{x}\|)/\sigma)^2}, e^{-\pi\|\mathbf{x}\|(-2\|\mathbf{u}\|+\|\mathbf{x}\|)/\sigma^2}\right]$$

We claim that the magnitude of the exponents in the left and right terms is small. In particular, we have

$$\begin{aligned}
\pi\|\mathbf{x}\|(2\|\mathbf{u}\| + \|\mathbf{x}\|)/\sigma^2 &\leq \pi \cdot \sigma^{-2} \cdot \left(\frac{2\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|^3}{1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|} + \frac{\gamma_{\mathsf{Mult}}(R)^2 \cdot \|\mathbf{u}\|^4}{(1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|)^2}\right) \\
&\leq \pi \cdot \sigma^{-2} \cdot \left(4\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|^3 + 4\gamma_{\mathsf{Mult}}(R)^2 \cdot \|\mathbf{u}\|^4\right) \\
&\leq 4\beta \cdot \pi + 4\beta \cdot \pi \cdot \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| \\
&\leq 6\beta \cdot \pi
\end{aligned}$$

□

The next lemma states that if $\mathbf{v}$ is very close to $t \cdot \mathbf{e}_1$, then the determinant (or norm)

of $(\mathbf{v})$ is close to t^n.

Lemma 15.1.6. *Let* $\mathbf{v} = \mathbf{e}_1 + \mathbf{u}$. *Then,* $\det(\mathbf{v}) \leq e^{n \cdot \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|}$. *If* $\|\mathbf{u}\| \leq 1/n \cdot \gamma_{\mathsf{Mult}}(R)$, *we obtain* $1/3 \leq \det(\mathbf{v}) \leq e$.

Proof. We know that $\det((\mathbf{v}))$ is the volume of the rotation basis of $\mathbf{v}$. So,

$$\det((\mathbf{v})) \quad \leq \quad \prod_{i=0}^{n-1} \|\mathbf{v} \times x^i\| \leq (1 + \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|)^n \leq e^{n \cdot \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|}.$$

Now suppose $\|\mathbf{u}\| \leq 1/n \cdot \gamma_{\mathsf{Mult}}(R)$; then, the term on the right is simply e. Also, from Lemma 15.1.4, we know that

$$\mathbf{w} \leftarrow \mathbf{e}_1/\mathbf{v} = \mathbf{e}_1 - \mathbf{u} + \mathbf{x} \quad \text{for} \quad \|\mathbf{x}\| \leq \frac{\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|^2}{1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|} \leq \frac{1}{(1 - 1/n) \cdot n^2 \cdot \gamma_{\mathsf{Mult}}(R)}$$

Since $\| - \mathbf{u} + \mathbf{x}\| \leq (1 + 1/(n-1))/n \cdot \gamma_{\mathsf{Mult}}(R)$,

$$\det((1/\mathbf{v})) \leq e^{1+1/(n-1)} \implies \det((\mathbf{v})) \geq e^{-1-1/(n-1)} \geq 1/3 \ ,$$

the latter inequality for reasonable values of n.

□

15.2 Distribution of Prime Ideals

Recall that the *ring of integers* of a number field K is:

$\mathcal{O}_K = \{x \in K : f^x_{\mathbb{Q}} \in \mathbb{Z}[x]\}$, where $f^x_{\mathbb{Q}}$ is the (monic) minimal polynomial of x in $\mathbb{Q}[x]$

$\mathcal{O}_{\mathbb{Q}[\alpha]}$ *contains* $\mathbb{Z}[\alpha]$, and the former's nice properties carry over to $\mathbb{Z}[\alpha]$, modulo some detectable and fixable "badness" due to prime ideals that divide the conductor of $\mathcal{O}_{\mathbb{Q}[\alpha]}$. These prime ideals are "rare" in the sense that they all have norms that are powers of some prime integer that divides the discriminant of $f(x)$, which (for suitable $f(x)$), is only $n^{O(n)}$. (See Chapter 13.4 for more details, and see [127] for a nice survey of algorithmic aspects of general number rings.)

The distribution of prime ideals in number fields is quite analogous to the distribution of primes in the integers. Just as the prime number theorem states that the number of

primes less than x is approximately $x/\ln x$, we have Landau's prime ideal theorem.

Theorem 15.2.1 (Theorem 8.7.2 from [10])**.** *Let K be an algebraic number field of degree n. Let $\pi_K(x)$ denote the number of prime ideals whose norm is $\leq x$. Let $\lambda(x) = (\ln x)^{3/5}(\ln \ln x)^{-1/5}$. There is a $c > 0$ (depending on K) such that*

$$\pi_K(x) = x/\ln x + O(xe^{-c\lambda(x)})$$

With the Generalized Riemann Hypothesis, one can make a stronger statement.

Theorem 15.2.2 (Theorem 8.7.4 from [10])**.** *Assume GRH. Let K be an algebraic number field of degree n and discriminant Δ. For $x \geq 2$, we have*

$$|\pi_K(x) - x/\ln x| = O(\sqrt{x}(n \ln x + \ln |\Delta|))$$

The constant implied by the "O" symbol is absolute.

In later Chapters, we will require some results on the distribution of prime ideals, but nothing as strong as Theorem 15.2.2. Rather, we will require only that, for certain intervals $[a, b]$ (e.g., where $a = b/2$), the number of prime ideals with norms in $[a, b]$ is a non-negligible fraction of b. In particular, we use the following lemma.

Lemma 15.2.3. *Let M be an ideal with norm in $[N, 2N]$, with $\log N$ polynomial in n. Let $\mathcal{I}_{a,b}$ be the set of invertible prime ideals with norms in $[a, b]$. Let $s = n \cdot \|f\| \cdot (b/N)^{1/n} \cdot \omega(\sqrt{\log n})$ and $t \geq \gamma_{\mathsf{Mult}}(R) \cdot s \cdot n^{1.5}$. Suppose $\mathbf{v}$ is chosen according to the distribution $D_{M^{-1},s,t\cdot\mathbf{e}_1}$. If $|\mathcal{I}_{a,b}|/b$ is non-negligible, then the probability that the ideal $M \cdot (\mathbf{v})$ has a divisor in $\mathcal{I}_{a,b}$ is non-negligible.*

Remark 15.2.4. Above, notice that $M \cdot (\mathbf{v})$ is an integer (non-fractional) ideal, since $\mathbf{v} \in M^{-1}$.

Proof. As a preliminary matter, we have that s exceeds the smoothing parameter of all ideals I with norm at most b/N, since by Lemma 8.2.2, $\lambda_n(I) < n \cdot \|f\| \cdot \det(I)^{1/n} = s/\omega(\sqrt{\log n})$.

Now, we want to prove that the quantity

$$\frac{\sum_{\mathbf{v} \in \mathfrak{p}M^{-1} \text{ for some } \mathfrak{p} \in \mathcal{I}_{a,b}} \rho_{s,t\cdot\mathbf{e}_1}(\mathbf{v})}{\sum_{\mathbf{v} \in M^{-1}} \rho_{s,t\cdot\mathbf{e}_1}(\mathbf{v})}$$

is non-negligible.

Consider the numerator of the expression above. Let us restrict the summation to $\mathbf{v}$'s in the ball $t \cdot \mathbf{e}_1 + \mathcal{B}(s \cdot \sqrt{n})$. For convenience, let $p(n) = \log(2N \cdot et^n)$. We have

$$\begin{aligned}
\sum_{\mathbf{v} \in \mathfrak{p}M^{-1} \text{ for some } \mathfrak{p} \in \mathcal{I}_{a,b}} \rho_{s,t\cdot\mathbf{e}_1}(\mathbf{v}) \quad &\geq \sum_{\mathfrak{p} \in \mathcal{I}_{a,b}} (\log \mathrm{Nm}(\mathfrak{p})) \cdot \sum_{\mathbf{v} \in \mathfrak{p}M^{-1}} \rho_{s,t\cdot\mathbf{e}_1}(\mathbf{v}) / (\log \mathrm{Nm}(M \cdot (\mathbf{v}))) \\
&\geq (1/p(n)) \cdot \sum_{\mathfrak{p} \in \mathcal{I}_{a,b}} (\log \mathrm{Nm}(\mathfrak{p})) \cdot \sum_{\mathbf{v} \in \mathfrak{p}M^{-1}} \rho_{s,t\cdot\mathbf{e}_1}(\mathbf{v}) \\
&\geq (1/p(n)) \cdot \sum_{\mathfrak{p} \in \mathcal{I}_{a,b}} \sum_{\mathbf{v} \in \mathfrak{p}M^{-1}} \rho_{s,t\cdot\mathbf{e}_1}(\mathbf{v}) \\
&\approx (1/p(n)) \cdot \sum_{\mathfrak{p} \in \mathcal{I}_{a,b}} \rho_{s,t\cdot\mathbf{e}_1}(\mathfrak{p}M^{-1}) \\
&\gtrapprox (1/p(n)) \cdot \rho_{s,t\cdot\mathbf{e}_1}(M^{-1}) \cdot \sum_{\mathfrak{p} \in \mathcal{I}_{a,b}} 1/\mathrm{Nm}(\mathfrak{p}) \\
&\geq (1/p(n)) \cdot \rho_{s,t\cdot\mathbf{e}_1}(M^{-1}) \cdot \sum_{\mathfrak{p} \in \mathcal{I}_{a,b}} 1/b
\end{aligned}$$

The first inequality follows from the fact that $\log \mathrm{Nm}(M \cdot (\mathbf{v})) \geq \sum_i \log \mathrm{Nm}(\mathfrak{p}_i)$, where $\{\mathfrak{p}_i\}$ consists of distinct invertible prime ideal divisors of $M \cdot (\mathbf{v})$. The second inequality follows from the fact that the invertibility of $(\mathbf{v})$ implies that $\mathrm{Nm}(M \cdot (\mathbf{v})) = \det(M) \cdot \det((\mathbf{v}))$, and from Lemma 15.1.6, which implies that $\det((\mathbf{v})) \leq e \cdot t^n$ when $\mathbf{v}$ is in the ball above, where e is Euler's constant. The "$\approx$" equation holds up to a multiplicative factor that is negligibly close to 1, because on the rhs we go back to including vectors that are outside of the ball, which carry only a negligible fraction of the weight since s exceeds the smoothing parameter of all ideals with norm at most b/N. By Lemma 13.2.6, since (again) s exceeds the smoothing parameter of all ideals with norm at most b/N, the "$\gtrapprox$" inequality is true up to a multiplicative factor that is negligibly close to 1. Overall, since $\sum_{\mathfrak{p} \in \mathcal{I}_{a,b}} 1/b$ is non-negligible by assumption, the result follows. □

Chapter 16

Random Self-Reduction of Ideal Lattice Problems

In this Chapter, we provide an overview of our worst-case / average-case connection for ideal lattices, and its relevance to our scheme. We provide the formal details in later Chapters.

16.1 A New Type of Worst-Case / Average-Case Connection for Lattices

Inherently, public-key cryptography is based on *average-case hardness*. For example, in the Rabin encryption scheme, the KeyGen algorithm induces an (average-case) distribution of public keys. In the random oracle model, one can prove that Rabin encryption is secure if it is infeasible to factor moduli generated *according to the average-case distribution*. As far as we know, any efficient instantiation of Rabin encryption could be insecure even if factoring is hard in the *worst-case* – i.e., for some (possibly negligible fraction of) moduli.

Sometimes, however, one can prove an *worst-case / average-case* connection – i.e., if one can solve an average-case instance of problem A (generated according to some distribution D), then one can solve *any* instance of problem B. If D is efficiently samplable and has enough min-entropy to be used in KeyGen, then the worst-case / average-case connection may allow one to base the scheme's security on the *worst-case hardness* of problem B. This gives more assurance that the scheme is secure.

For example, one can establish something similar to a worst-case / average-case connection for schemes based on Diffie-Hellman. The Diffie-Hellman problem is *random self-reducible* – i.e., given *any* instance I_1 of Diffie-Hellman over a fixed group G, one can use I_1 to generate a *random* instance I_2 of Diffie-Hellman over G, such that an algorithm that solves I_2 can be used as a sub-routine in an algorithm to solve I_1. Thus, a scheme whose security is based on Diffie-Hellman in G is based on the problem's worst-case hardness in G. However, since G is fixed – i.e., we use the same G in the worst-case and average-case instances – the random self-reducibility of Diffie-Hellman in G is typically not considered to be a full-fledged worst-case / average-case connection.

In 1996, Ajtai [2] found a surprising reduction of worst-case lattice problems to an average-case ones, and Ajtai and Dwork [4] proposed an encryption scheme based on this worst-case hardness. The good news is that, unlike in the Diffie-Hellman case, the worst-case problem is a completely general problem (over lattices) that is unconstrained by any parameters in the average-case problem – a full-fledged worst-case / average-case connection. The bad news is that Ajtai's reduction has a price. In the Diffie-Hellman case, the worst-case and average-case problems are of the same type – i.e., Diffie-Hellman over G. In Ajtai's reduction, however, the lattice in the worst-case instance has smaller dimension than the average-case instance – i.e., his reduction allows one to base security on worst-case hardness, but over a seemingly easier set of lattices.

Another problem with Ajtai's approach is that it seems difficult to adapt it to output average-case *ideal lattices*. Obviously, our KeyGen algorithm generates a key according to an average-case distribution, and we need KeyGen to generate an ideal lattice since our scheme depends crucially on the multiplicative structure of ideal lattices. Prior work [98, 111, 112, 88, 99] adapts Ajtai's approach to establish a worst-case / average-case connection where only the *worst-case problem* is directly over ideal lattices; the average-case lattice is generated as a sort of "knapsack" of vectors from the worst-case lattice, a process which ruins the ideal lattice structure. To obtain an average-case instance that is directly over ideal lattices, we apparently need an approach fundamentally different from Ajtai's and other prior work.

Our worst-case / average-case connection is conceptually similar to a random self-reduction, where both the worst-case and average-case instances are directly over ideal lattices in the same polynomial ring R. One difference is that the ideal lattices in the

average-case instance correspond to invertible prime ideals (see Chapter 13.4 for definitions), whereas this is not necessarily the case of the ideal in the worst-case instance. Also, the approximation factor in the worst-case instance is larger by a multiplicative factor that depends on the ring R. For an exponential-sized family of rings R – in particular, when R is selected as suggested in Chapter 7.4 – this factor is only polynomial in n.

16.2 Our Average-Case Distribution

As mentioned above, we want our average-case problem to be "directly over" ideal lattices. We use a natural lattice problem – the bounded distance decoding problem (BDDP) – which (informally) asks: given a lattice L and a vector $\mathbf{t}$ with the promise that $\text{dist}(L, \mathbf{t}) \leq s$, output some $\mathbf{v} \in L$ with $\|\mathbf{t} - \mathbf{v}\| \leq s$. (The notation $\text{dist}(L, \mathbf{t})$ denotes $\min\{\|\mathbf{v} - \mathbf{t}\| : \mathbf{v} \in L\}$; equivalently, the problem could ask for $\mathbf{t} - \mathbf{v}$.) Usually s is chosen to be less than $\lambda_1(L)/2$, where $\lambda_1(L)$ is the shortest nonzero vector in L, so that there is a unique solution.

The average-case problem that we consider in the main reduction is actually a "hybrid" of worst-case and average-case. The ideal lattice is generated according to an average-case distribution induced by an algorithm $\mathsf{IdealGen}$. However, the vector $\mathbf{t}$ is "worst-case." The only requirement on $\mathbf{t}$ is that it be within a certain distance of the lattice; it need not be chosen according to any known (or even samplable) distribution. Here is formal statement of the problem.

Definition 16.2.1 (Hybrid Bounded Distance Decoding Problem (HBDDP))**.** Fix ring R and algorithm $\mathsf{IdealGen}$ that samples bases of ideals in R. Fix a positive real s_{HBDDP}. The challenger sets $\mathbf{B}_J^{\text{pk}} \xleftarrow{\text{R}} \mathsf{IdealGen}(R)$. The challenger sets $\mathbf{x}$ subject to the constraint that $\|\mathbf{x}\| < s_{\mathsf{HBDDP}}$ and sets $\mathbf{t} \leftarrow \mathbf{x} \bmod \mathbf{B}_J^{\text{pk}}$. The problem is: given $(\mathbf{B}_J^{\text{pk}}, \mathbf{t})$ (and the fixed values), output $\mathbf{x}$.

This problem is the same as I-HBDDP (see Chapter 14.5), except that in I-HBDDP, the algorithm $\mathsf{IdealGen}$ took the basis of an ideal I as input and was required to generate bases for an ideal J relatively prime to I.

Our average-case distribution is uniform over invertible prime ideals in R that have norms in some specified interval $[a, b]$. In practice, the fact that J is prime will mean that J is automatically relatively prime to I.

One reason behind our choice of average-case distribution is that, crudely speaking, there is almost a "bijection" between integers and norms of ideals in R; so, sampling a random

prime ideal amounts to sampling a random prime integer, and outputting an ideal whose norm is (a power of) that integer. This crude intuition is inaccurate, but "close enough" to be useful. Another reason that this average-case distribution is attractive is that we are also able to take worst-case ideal lattices, and "randomize into" this average-case distribution, though there are considerable technical difficulties.

We remark that while there are some difficulties in using our average-case distribution, *defining* a random ideal lattice seems simpler than defining a "random lattice" in the general setting [2, 3]. To see how different the settings are, notice that for any integer d, there are at least d^{n-1} distinct lattices of determinant d and dimension n (since there are at d^{n-1} distinct Hermite normal form bases with $(1, \ldots, 1, d)$ along the diagonal), whereas the number of ideal lattices with determinant in $[d, 2d]$ is $\theta(d)$.

16.3 How to "Randomize" a Worst-Case Ideal

Our worst-case problem is also BDDP over ideal lattices in R:

Definition 16.3.1 (Worst-Case Bounded Distance Decoding Problem (WBDDP))**.** Fix ring R and positive real s_{WBDDP}. The problem is: given $(\mathbf{B}_M, \mathbf{t})$, where $\mathbf{B}_M$ is a basis for an ideal lattice M of R and $\mathbf{t}$ satisfies $\mathrm{dist}(M, \mathbf{t}) \leq s_{\mathsf{WBDDP}}$, output $\mathbf{y} \in \mathbf{t} + M$ such that $\|\mathbf{y}\| \leq s_{\mathsf{WBDDP}}$.

How do we "randomize" a worst-case lattice into our distribution on HBDDP lattices, in such a way that an algorithm that solves HBDDP can be used as a subroutine in an algorithm that solves WBDDP? Let us start with some high-level intuition. One intuition is that if we start from an arbitrary ideal lattice M and look at coarse enough sub-lattices of it, we will find sub-lattices of every "geometric shape". That is, if we choose a "random" ideal K, then the "geometric shape" of MK will be essentially independent of the "geometric shape" of M. Roughly speaking, one may view the "geometric shape" as the shape of the parallelepiped formed by a basis of very short vectors in the lattice, which is unchanged by rigid transformations.

But clearly MK is not independent of MK, since this lattice will actually have some "algebraic properties" that depend on M. In particular, the ideal MK is divisible by the ideal M. Of course, this is unacceptable in terms of obtaining a worst-case / average-case connection; perhaps our average-case instance-solver fails precisely when its instance is divisible by M.

To destroy these algebraic properties, consider the following approach. Choose an element $\mathbf{v} \in (MK)^{-1}$ and output $J \leftarrow MK(\mathbf{v})$ as the average-case ideal. (The *fractional ideal* $(MK)^{-1}$ is simply the inverse of the ideal lattice MK.) More specifically, choose $\mathbf{v}$ according to the discrete Gaussian distribution $D_{(MK)^{-1},s,t\cdot\mathbf{e}_1}$, where $t \gg s$ so that $\mathbf{v}$ is "nearly" parallel to the scaled unit vector $t \cdot \mathbf{e}_1$, and the lattice $(\mathbf{v})$ is "close" to the lattice $t \cdot \mathbb{Z}^n$ in that it has nearly the same "shape". The idea is that, when $\mathbf{v}$ is chosen in this way, the "shape" of J is essentially the same as that of MK since multiplication by $(\mathbf{v})$ is close to being a rigid transformation, but the M^{-1} factor in $(\mathbf{v})$ removes the "algebraic properties" that come from M. Intuitively, $J = MK \cdot (\mathbf{v})$ has a "shape" independent of M and does not inherit the "algebraic" properties of M, so it should be "random".

The outline of the approach above, of course, is merely intuition; it only serves to explain why we may hope to find a randomization procedure. However, we are able to formalize the approach. In Chapter 17.4, we define an algorithm $\mathsf{RandomizeIdeal}'$ that takes as input an ideal M and a bound $b > \det(M)$. $\mathsf{RandomizeIdeal}'$ invokes a sub-routine $\mathsf{RandomIdealGen}(R, i, j)$ that outputs an ideal K that is uniformly random among invertible ideals with determinant in $[i, j]$, together with a short vector $\mathbf{w}_K \in K$ satisfying $\|\mathbf{w}_K\| < X$ for some X. The particular value of $[i, j]$ that it uses is $[bt^{2n}/\det(M), 4bt^{2n}/\det(M)]$, where t is a factor depending only on R and that may be polynomial in n, and the value X is preferably only polynomial in n. We prove the following theorems.

Theorem 16.3.2. *For any $\alpha \geq 1$, assuming* $\mathsf{RandomIdealGen}$ *is efficient,* $\mathsf{RandomizeIdeal}'$ *efficiently outputs an invertible ideal J that is statistically uniform subject to the constraint that* $\det(J) \in [2bt^{3n} \cdot \alpha^n, 3bt^{3n} \cdot \alpha^n]$.

Assuming that invertible prime ideals are not too sparse among invertible ideals in the interval, $\mathsf{RandomizeIdeal}'$ can easily be adapted to output ideals according to our average-case distribution.

Theorem 16.3.3. *Assume* $\mathsf{RandomIdealGen}$ *is efficient. Also assume that there is an algorithm $\mathcal{A}$ that, for some $\alpha \geq 1$, solves the HBDDP with overwhelming probability (over the random coins chosen by $\mathcal{A}$) for a ϵ fraction of ideals J output by* $\mathsf{IdealGen}$*, where* $\mathsf{IdealGen}$ *outputs an invertible ideal J that is uniformly random with determinant in $[2bt^{3n} \cdot \alpha^n, 3bt^{3n} \cdot \alpha^n]$. Then, there is an algorithm $\mathcal{B}$, with running time approximately $O(1/\epsilon)$ that of $\mathcal{A}$, that solves the WBDDP over any ideal M with $\det(M) < b$ and $s_{\mathsf{WBDDP}} \leq s_{\mathsf{HBDDP}}/(t^2 \cdot \alpha \cdot X)$.*

RandomizeIdeal′ is fairly similar to what we outlined above, but includes some rejection sampling to fine tune the sampling distribution. The distribution analysis relies heavy on properties of Gaussian distributions over lattices, and on the relation between $\mathbf{v}$ and its inverse in the field $\mathbb{Q}(x)/(f(x))$ overlying R – e.g., that the fractional ideal $(1/\mathbf{v})$ is "close" to being a scaling of $\mathbb{Z}^n$ when $(\mathbf{v})$ is.

Though we defer the formal details of the proof of Theorem 16.3.3 until Chapter 17, the high-level idea is simple. From its WBDDP instance $(M, \mathbf{u})$, $\mathcal{B}$ uses RandomizeIdeal′ to generate an average-case ideal lattice $J = MK(\mathbf{v})$ for $\mathcal{A}$, and generates the target vector $\mathbf{t}$ for the average-case instance as $\mathbf{u} \times \mathbf{v} \times \mathbf{w}_K \bmod \mathbf{B}_J$. Once $\mathcal{A}$ finally succeeds, $\mathcal{B}$ can easily convert $\mathcal{A}$'s solution into a WBDDP solution. The reduction requires $\mathbf{w}_K$ to be short, so that the smallness of $\mathrm{dist}(M, \mathbf{u})$ implies the smallness of $\mathrm{dist}(J, \mathbf{t})$ – i.e., that the average-case instance is well-formed.

However, an unfortunate wrinkle in this approach is that we do not know how to instantiate RandomIdealGen efficiently. In the next Section, we outline how to instantiate RandomIdealGen efficiently given access to a *factoring oracle*, which could be instantiated with quantum computation. We also outline our difficulties in instantiating it with an efficient classical algorithm. We stress that we still use RandomizeIdeal′. Specifically, as sketched in Chapter 16.5 and detailed in Chapter 18, we use a weak version of RandomizeIdeal′ in our (classical, efficient) instantiation of IdealGen in KeyGen. This weak version of RandomizeIdeal′ uses a weak version of RandomIdealGen that does not generate a short vector $\mathbf{w}_K \in K$, which is sufficient for KeyGen. (The short vector $\mathbf{w}_K$ is need only in the worst-case / average-case reduction to translate a solution to the average-case problem into a solution to the worst-case one.)

Since we need a factoring oracle anyway, we use a different reduction (given in Chapter 17.1-17.3) that requires weaker statements about the distribution of prime ideals in number fields than we require to instantiate RandomIdealGen efficiently. Specifically, we provide an algorithm RandomizeIdeal, which invokes a factoring oracle, and prove the following theorems.

Theorem 16.3.4. *Let $\mathcal{I}_{a,b}$ be the set of invertible prime ideals with norm in $[a, b]$. Suppose that $|\mathcal{I}_{a,b}|/b$ is non-negligible, $\log b$ is polynomial in n, and $a^2 > 2N \cdot t_0^n$ where $t_0 = t + s \cdot \sqrt{n}$ for $s = \omega(\sqrt{\log n})$, $s = n^3 \cdot \|f\|^3 \cdot (b/N)^{1/n} \cdot \omega(\sqrt{\log n})$, and $t \geq \gamma_{\mathsf{Mult}}(R) \cdot n^{1.5} \cdot s$. Then, given an ideal M with norm in $[N, 2N]$ as input, and with access to a factoring oracle,* RandomizeIdeal *efficiently outputs the basis of an invertible prime ideal J that is statistically*

uniform, and independent of M, subject to the constraint that $\mathrm{Nm}(J) \in [a, b]$.

Theorem 16.3.5. *Let a, b, $\mathcal{I}_{a,b}$, N, and t be as in Theorem 16.3.4. Suppose that there is an algorithm $\mathcal{A}$ that solves s_{HBDDP}-HBDDP with overwhelming probability (over the random coins chosen by $\mathcal{A}$) for a ϵ (weighted) fraction of invertible prime ideals $J \in \mathcal{I}_{a,b}$. (Assume $s_{\mathsf{HBDDP}} < \lambda_1(J)/2$.) Then, there is an algorithm $\mathcal{B}$, with running time approximately $O(1/\epsilon)$ that of $\mathcal{A}$, that solves the s_{WBDDP}-WBDDP for any (worst-case) ideal M with* $\det(M) \in [N, 2N]$ *when $2t \cdot s_{\mathsf{WBDDP}} + \sqrt{n} \leq s_{\mathsf{HBDDP}}$. When $N \geq b$, for any $g(n)$ that is $\omega(\sqrt{\log n})$, we can set $t = \gamma_{\mathsf{Mult}}(R) \cdot n^{4.5} \cdot \|f\|^3 \cdot g(n)$.*

16.4 Why Does the Reduction Require a Factoring Oracle?

The algorithm $\mathsf{RandomizeIdeal}'$, and hence the worst-case / average-case reduction, is efficient and classical (non-quantum), with the possible exception of the sub-routine $\mathsf{RandomIdealGen}$. An obvious attempt to instantiate $\mathsf{RandomIdealGen}$ is the following: generate a random short vector $\mathbf{w}_K$ and set $K \leftarrow (\mathbf{w}_K)$. But, among other problems, the output K is obviously a *principal* ideal (i.e., has a single generator). Typically, most ideals in R are not principal, so K cannot be said to be random. (We could restrict the worst-case and average-case problems to principal ideals, but this leads to other technical difficulties.) It seems quite challenging to efficiently generate a non-principal ideal together with a short vector in it. If one, say, generates two "random" vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}^n$, one of them short, and sets K to be the ideal $(\mathbf{v}) + (\mathbf{w})$, most likely K will be all of R, or some uninteresting ideal with very small norm.

One approach to fixing the "attempt" above is to use a factoring oracle. For example, we can sample $\mathbf{w}_K \in \mathbb{Z}^n$ from a Gaussian ball of small radius s centered at $t \cdot \mathbf{e}_1$, so that $\mathbf{w}_K$ is nearly parallel to $\mathbf{e}_1$. We then factor the ideal $(\mathbf{w}_K)$ and set K to be a random divisor of $(\mathbf{w}_K)$, subject to the norm requirements on K. However, there is no known efficient classical algorithm to factor $(\mathbf{w}_K)$ in $R = \mathbb{Z}[x]/(f(x))$. But if we have an integer factoring oracle, we can factor $(\mathbf{w}_K)$ easily: factor $\det((\mathbf{w}_K))$ into $\prod_i p_i^{e_i}$, then (for each p_i) use an efficient classical polynomial factorization algorithm (e.g., Kaltofen-Shoup [75]) to factor $f(x) = \prod_j g_{ij}(x) \bmod p_i$, and finally test whether the various (possibly non-principal) ideals $(g_{ij}(x), p_i)$ divide $(\mathbf{w}_K)$. All of the prime ideal factors have this form [127] and will be efficiently discovered. Shor [124] describes a quantum integer factoring algorithm, which can be used to instantiate the oracle efficiently.

Why should the ideal K be random? The probability that an ideal K with $\det(K) \in [a, b]$ *divides* $(\mathbf{w}_K)$ is negligibly close to $1/\det(K)$, for suitably chosen parameters, using results on Gaussian distributions over lattices. If this immediately translated into some way to *choose* K with probability negligibly close to $1/\det(K)$ (or $c/\det(K)$ for some non-negligible c), then we could make the distribution uniform by outputting the sampled K only with probability $\det(K)/b$, where b is an upper bound on $\det(K)$, and otherwise resampling.

However, actually picking a divisor of $(\mathbf{w}_K)$ leads to some complications. One can think of it like a balls and bins problem. For $\mathbf{w}_K$ chosen according to the distribution above, we know that K divides $(\mathbf{w}_K)$ with probability very close to $1/\det(K)$ – i.e., in some sense K is in a $1/\det(K)$ fraction of the bins. However, some of the K's tend to be in crowded bins – i.e., some K's tend to divide only the values of $(\mathbf{w}_K)$ that have many (potentially a super-polynomial number of) divisors. In fact, we know exactly which K's are the problematic ones: they are the K's that themselves have many divisors. To address this problem, we could show that the fraction of "bad" K's is small, and that we can sample uniformly from among the remaining K's. But this requires rather strong effective results on the distribution of prime ideals – e.g., one can try to use an Erdös-Kac-like lemma that characterizes the number of prime ideal divisors that $(\mathbf{w}_K)$ is likely to have. This was our original approach, but ultimately we opted for a reduction that still requires a factoring oracle, but only needs weaker results on the distribution of prime ideals.

In the simpler reduction, given the WBDDP instance $(M, \mathbf{u})$, we start off by generating a vector $\mathbf{v} \xleftarrow{\mathrm{R}} D_{M^{-1},s,t\cdot\mathbf{e}_1}$ for $t \gg s$ and setting $L \leftarrow M \cdot (\mathbf{v})$ and $\mathbf{t}' \leftarrow \mathbf{u} \times \mathbf{v}$. The idea here is that the BDDP instance $(L, \mathbf{t}')$ is just like the instance $(M, \mathbf{u})$, except everything has been multiplied by $\mathbf{v}$, which is very close to a rigid transformation that essentially preserves the "shape" of the BDDP instance. Consequently, a solution to $(L, \mathbf{t}')$ would give a solution to $(M, \mathbf{u})$. Since $\mathbf{v} \in M^{-1}$, L is an integer (non-fractional) ideal. Moreover, since L inherits some randomness from $\mathbf{v}$'s distribution, we can show (for proper settings of parameters) that the probability that L is divisible by an invertible prime ideal J with norm in our desired interval $[a, b]$ is proportional to $1/\det(J)$. So, to sample uniformly from among the candidate J's, we apply our factoring oracle to L, tentatively grab a candidate J, and make the probability uniform over the J's through rejection sampling. The average-case BDDP instance is essentially $(J, \mathbf{t}' \bmod \mathbf{B}_J)$, except that $\mathbf{t}'$ is first rounded to an integer vector. Since J is a super-lattice of L, the fact that $\mathbf{t}'$ is a small distance from L implies that it is a small distance from J, and therefore forms a valid BDDP instance, though of course this

distance is larger in relation to $\lambda_1(J)$ than to $\lambda_1(L)$. We begin describing this reduction formally in Chapter 17.1.

We still describe RandomizeIdeal$'$ (the "other" approach) for two reasons. First, as discussed above, we use a weak version of our RandomizeIdeal$'$ algorithm to obtain a IdealGen algorithm (used in KeyGen) that efficiently generates ideals (classically) according to the average-case distribution. The weak version requires only a weak version of RandomIdealGen, which (as before) generates an ideal K that is uniformly random among ideals with norm in a certain interval, but does not generate a short vector $\mathbf{w}_K \in K$; this weak version can be instantiated classically. (The techniques used in RandomizeIdeal do not appear to be useful for generating a classical IdealGen algorithm that generates ideals according to the average-case distribution.) Second, we speculate that there may be a classical way to instantiate RandomIdealGen efficiently, in which case the worst-case / average-case connection would be entirely classical.

16.5 Application to our Fully Homomorphic Encryption Scheme

The KeyGen algorithm in our fully homomorphic encryption scheme uses an algorithm IdealGen that outputs the basis of a (random) ideal J, together with a short independent set $\mathbf{B}_{J^{-1}}$ of J^{-1}. One technical requirement is that J must be relatively prime to an ideal I, where I is a fixed global ideal used by everybody.

We give a specific instantiation of IdealGen that outputs ideals J according to our average-case distribution, together with a short independent set $\mathbf{B}_{J^{-1}}$ of J^{-1} as needed for decryption. Since, in our average case distribution, J is prime, it will automatically be relatively prime to I.

It may seem counterintuitive that we can efficiently generate (classically) a random ideal J and an associated secret key, even though we do not know how to instantiate RandomIdealGen efficiently (classically). As a rough explanation, the reason is that the secret key for J is a short independent set for the *inverse* ideal, while RandomIdealGen is supposed to generate a short vector in the ideal itself.

A better explanation is simply to sketch why a weak version of RandomizeIdeal$'$, which uses a weak version of RandomIdealGen that does not generate the short vector $\mathbf{w}_K$, suffices to generate a random J together with a short independent set for J^{-1}. In this weak version of RandomizeIdeal$'$, we "randomize" the "worst-case" ideal $M = R$; that is, we generate

K and $\mathbf{v}$ as we did before and set $J = K \cdot (\mathbf{v})$ (instead of $J = MK \cdot (\mathbf{v})$). So, how do we obtain a short independent set for J^{-1}? Given an independent set $\mathbf{B}_{K^{-1}}$ of K^{-1}, we can obtain an independent set $\mathbf{B}_{J^{-1}}$ of J^{-1} simply by dividing each column vector in $\mathbf{B}_{K^{-1}}$ by $\mathbf{v}$. We trivially have the independent set $\{\mathbf{e}_i\}$ for K^{-1}, so the rotation basis of $1/\mathbf{v}$ forms an independent set of J^{-1}. When $\mathbf{v}$ is long enough, this independent set is suitable for decryption. In particular, we prove that when we generate our secret key in this way, we can obtain a value of r_{Dec} that is within a polynomial factor of $\lambda_1(J)$. Then, the analysis given in Chapter 7.7 applies, where we showed that we could permit $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$ to be as large as the approximation factor of our BDDP instance, up to a polynomial factor, thereby maximizing the circuit depth that we can evaluate.

In Chapter 14, we described a sequence of reductions that, subject to certain conditions – e.g., that $\det(I)$ is prime and $\mathrm{poly}(n)$, and s_{HBDDP} is sufficiently small – reduces the I-HBDDP with the distribution from $\mathsf{IdealGen}$ to the semantic security of our scheme. Later, we give another sequence of reductions that reduces the (worst-case) SIVP to the WBDDP. Overall, with the worst-case / average-case connection given here, this bases the security of our scheme on quantum worst-case SIVP over ideal lattices in the ring R.

Chapter 17

How to Randomize a Worst-Case Ideal

17.1 The RandomizeIdeal Algorithm

In this Section, we present our algorithm RandomizeIdeal and prove Theorems 16.3.4 and 16.3.5.

RandomizeIdeal$(R, \mathbf{B}_M, N, s, t, a, b)$. Takes as input the ring R, a basis $\mathbf{B}_M$ of an ideal M of R whose norm is in $[N, 2N]$ where $\log N$ is polynomial in n, and parameters s, t, a, and b such that:

- $s = \omega(\sqrt{\log n})$,
- $s = n^3 \cdot \|f\|^3 \cdot (b/N)^{1/n} \cdot \omega(\sqrt{\log n})$,
- $t \geq \gamma_{\mathsf{Mult}}(R) \cdot n^{1.5} \cdot s$,
- the number of invertible prime ideals with norms in $[a, b]$ is a non-negligible fraction of b,
- a/b is non-negligible,
- $\log b$ is polynomial in n,
- $a^2 > 2N \cdot t_0^n$ where $t_0 = t + s \cdot \sqrt{n}$.

The algorithm does the following.

1. It generates a vector $\mathbf{v}$ per the discrete Gaussian distribution $D_{M^{-1},s,t\cdot\mathbf{e}_1}$; it sets $L \leftarrow M \cdot (\mathbf{v})$.

2. It uses a factoring oracle to compute bases of the invertible prime ideal divisors $\{\mathfrak{p}_i\}$ of L.

3. It sets J to be an ideal in $\{\mathfrak{p}_i\}$ that has norm in $[a, b]$, if there is one; otherwise, it restarts.

4. With probability $\det(J)/b$ it outputs a basis $\mathbf{B}_J$ of J, along with the vector $\mathbf{v}$; otherwise, it restarts.

Remark 17.1.1. In Step 1, one can sample from the distribution $D_{M^{-1},s,t\cdot\mathbf{e}_1}$ by using the GPV algorithm (see Chapter 13.3) with the independent set $\{\mathbf{e}_i\}$ in M^{-1}.

Remark 17.1.2. See Chapter 13.4 for more details on how an integer factoring oracle can be used to compute in Step 2 to compute the bases of prime ideal factors of L.

17.2 Is the Ideal Random? The Proof of Theorem 16.3.4

Before proving that our RandomizeIdeal algorithm is efficient and has the proper distribution, we provide a preliminary fact regarding $\mathbf{v}$.

Lemma 17.2.1. *The vector* $\mathbf{v}$ *drawn in Step 1 is in* $t \cdot \mathbf{e}_1 + \mathcal{B}(s \cdot \sqrt{n})$ *with overwhelming probability, where* $\mathcal{B}(r)$ *is the open ball of radius* r*.*

Proof. First, observe that $\mathbb{Z}^n$ is a sub-lattice of M^{-1}. Therefore, regarding smoothing parameters, we have $\eta_\epsilon(M^{-1}) \leq \eta_\epsilon(\mathbb{Z}^n) \leq \sqrt{\pi^{-1} \cdot \ln(2n(1+1/\epsilon))}$. Since $s = \omega(\sqrt{\log n})$, there is a negligible ϵ for which $s > \sqrt{\pi^{-1} \cdot \ln(2n(1+1/\epsilon))}$ – i.e., s exceeds the smoothing parameter of M^{-1}. By Lemma 13.2.2, we have

$$\Pr_{\mathbf{v}\leftarrow D_{M^{-1},s,t\cdot\mathbf{e}_1}}\left[\|\mathbf{v} - t\cdot\mathbf{e}_1\| > s\cdot\sqrt{n}\right] \leq \frac{1+\epsilon}{1-\epsilon}\cdot 2^{-n}$$

□

Now, we prove Theorem 16.3.4.

Theorem 16.3.4. We claim that the probability of restarting in Steps 3 and 4 is non-overwhelming, from which the efficiency of our algorithm follows. For Step 3, this is true because a/b is non-negligible. Regarding Step 4, we use Lemma 15.2.3, which establishes that, for our choices of s, t, a, and b, there is a non-negligible probability that $M \cdot (\mathbf{v})$ has an invertible prime ideal divisor with norm in $[a, b]$ when $\mathbf{v}$ is sampled according to the above distribution.

Now, consider the probability that an invertible prime ideal J with norm in $[a, b]$ is chosen as the ideal J in Step 3 in a single trial (without restarts). Assuming $\mathbf{v} \in t \cdot \mathbf{e}_1 + \mathcal{B}(s \cdot \sqrt{n})$ (which is indeed the case with overwhelming probability by Lemma 17.2.1), we claim that J is chosen iff $\mathbf{v} \in JM^{-1}$.

For the 'if' direction of our claim, if $\mathbf{v} \in JM^{-1}$, then J divides (is a super-lattice of) $L \leftarrow M \cdot (\mathbf{v})$. Since $(\mathbf{v})$ is an invertible ideal (being principal), we have that $\det(L) = \det(M) \cdot \det((\mathbf{v})) \leq 2N \cdot t_0^n < a^2 \leq \det(J)^2$. Consequently, besides J, L cannot have any other prime ideal divisors with norm in $[a, b]$, and J is chosen. For the 'only if' direction, that J is chosen implies that J divides (is a super-lattice of) $L = M \cdot (\mathbf{v})$. But then JM^{-1} is a super-lattice of $M^{-1}M \cdot (\mathbf{v})$. M is not necessarily invertible; in particular, $M^{-1}M$ may be divisible by singular prime ideals that divide the conductor. (See Chapter 13.4.) However, we also trivially have that JM^{-1} is a super-lattice of $J \cdot (\mathbf{v})$ since $\mathbf{v} \in M^{-1}$. Therefore, all vectors in the sum of ideals $J \cdot (\mathbf{v}) + M^{-1}M \cdot (\mathbf{v}) = (J + M^{-1}M) \cdot (\mathbf{v})$ are contained in JM^{-1}. However, J and $M^{-1}M$ are relatively prime, since J is invertible while $M^{-1}M$ is divisible by only singular (non-invertible) prime ideals. Therefore, $(\mathbf{v})$ is contained in JM^{-1}; in particular, $\mathbf{v} \in JM^{-1}$.

Therefore, for fixed M, we have

$$\Pr[J] = \frac{\sum_{\mathbf{v} \in JM^{-1}} \Pr[v]}{\sum_{\mathbf{v} \in M^{-1}} \Pr[v]} = \frac{\rho_{s,t \cdot \mathbf{e}_1}(JM^{-1})}{\rho_{s,t \cdot \mathbf{e}_1}(M^{-1})}$$

Since J is an invertible ideal, we have $\det(JM^{-1}) = \det(J) \cdot \det(M^{-1})$. Assuming that s exceeds the smoothing parameter of JM^{-1}, Lemma 13.2.6 thus implies that

$$\frac{\rho_{s,t \cdot \mathbf{e}_1}(JM^{-1})}{\rho_{s,t \cdot \mathbf{e}_1}(M^{-1})} \approx 1/\det(J)$$

where "$\approx$" means up the equation is correct up to a multiplicative factor that is negligibly close to 1. Step 4 adjusts this probability so that it is $1/b$ – i.e., uniform (and independent

of M) over all prime ideals with norms in $[a, b]$.

It remains only to show that s exceeds the smoothing parameter of the lattice JM^{-1}. We have that

$$\begin{aligned} s &= n^3 \cdot \|f\|^3 \cdot (b/N)^{1/n} \cdot \omega(\sqrt{\log n}) \\ &\geq n^3 \cdot \|f\|^3 \cdot \det(J)^{1/n} / \det(M)^{1/n} \cdot \omega(\sqrt{\log n}) \\ &\geq n^2 \cdot \|f\|^2 \cdot \det(J)^{1/n} \cdot \det(M^{-1})^{1/n} \cdot \omega(\sqrt{\log n}) \\ &\geq n \cdot \|f\| \cdot \det(JM^{-1})^{1/n} \cdot \omega(\sqrt{\log n}) \\ &\geq \lambda_n(JM^{-1}) \cdot \omega(\sqrt{\log n}) \end{aligned}$$

by the lemmata in Chapter 8.2. □

17.3 Reduction of WBDDP to HBDDP and Worst-case IVIP to Average-Case IVIP

Next, we prove Theorem 16.3.5, showing how to use the procedure RandomizeIdeal to reduce WBDDP to HBDDP.

Proof. (Theorem 16.3.5) $\mathcal{B}$ wants to solve the WBDDP instance $(M, \mathbf{u})$. It does the following:

1. It runs $(\mathbf{B}_J, \mathbf{v}) \xleftarrow{\mathrm{R}} \mathsf{RandomizeIdeal}(R, \mathbf{B}_M, N, s, t, a, b)$.
2. It sets $\mathbf{t}' \leftarrow \mathbf{u} \times \mathbf{v}$ and $\mathbf{t} \leftarrow \lfloor \mathbf{t}' \rceil \bmod \mathbf{B}_J$; let $\mathbf{c} \in [-1/2, 1/2)^n$ be vector such that $\mathbf{t}' - \mathbf{t} - \mathbf{c} \in J$.
3. It runs $\mathcal{A}$ on the instance $(J, \mathbf{t})$, receiving back a vector $\mathbf{y}$ such that $\mathbf{t} - \mathbf{y} \in J$. (If $\mathcal{A}$ does not solve this instance, restart.)
4. It outputs $\mathbf{x} \leftarrow (\mathbf{y} + \mathbf{c})/\mathbf{v}$.

First, we verify that $(J, \mathbf{t})$ is a valid HBDDP instance that should be solvable by $\mathcal{A}$. By Theorem 16.3.4, RandomizeIdeal outputs the basis of an ideal J that is statistically uniform among invertible prime ideals with norm in $[a, b]$.

Now consider $\mathbf{t}$. By assumption, there exist $\mathbf{m} \in M$ and $\mathbf{z}$ with $\|\mathbf{z}\| \leq s_{\mathsf{WBDDP}}$ such that $\mathbf{u} = \mathbf{m} + \mathbf{z}$. So, $\mathbf{t}' = \mathbf{m}' + \mathbf{z}'$, where $\mathbf{m}' \in M \cdot (\mathbf{v})$ and (assuming $\mathbf{v} \in t \cdot \mathbf{e}_1 + \mathcal{B}(s \cdot \sqrt{n})$)

we have

$$\|\mathbf{z}'\| = \|\mathbf{z} \times \mathbf{v}\| \leq t \cdot \|\mathbf{z}\| + \gamma_{\mathsf{Mult}}(R) \cdot s \cdot \sqrt{n} \cdot \|\mathbf{z}\| \leq 2t \cdot s_{\mathsf{WBDDP}}$$

Since $M \cdot (\mathbf{v})$ is a sub-lattice of J, we have that $\mathbf{t}' = \mathbf{j}' + \mathbf{z}'$ for some $\mathbf{j}' \in J$. When we round $\mathbf{t}'$ to obtain $\mathbf{t}$ (and reduce modulo $\mathbf{B}_J$), this shifts the vector by a distance of at most $\sqrt{n}$ (modulo J), and thus $\mathbf{t} = \mathbf{j} + \mathbf{z}''$ for some $\mathbf{j} \in J$ and $\|\mathbf{z}''\| \leq 2t \cdot s_{\mathsf{WBDDP}} + \sqrt{n} \leq s_{\mathsf{HBDDP}}$.

By the analysis above, $\mathcal{A}$ should solve the instance $(J, \mathbf{t})$ with probability at least ϵ. If $\mathcal{A}$ solves this instance – i.e., $\mathcal{B}$ receives from $\mathcal{A}$ a vector $\mathbf{y}$ with $\|\mathbf{y}\| < s_{\mathsf{HBDDP}}$ such that $\mathbf{t} - \mathbf{y} \in J$ – then, since $\mathbf{z}''$ is also a valid solution and since $s_{\mathsf{HBDDP}} < \lambda_1(J)/2$, we have that $\mathbf{y} = \mathbf{z}''$. Then, $\mathbf{z}'' + \mathbf{c} = \mathbf{z}'$ and $\mathbf{z}'/\mathbf{v} = \mathbf{z}$, and $\mathcal{B}$ solves its WBDDP instance.

We aim to set t as small as possible. For some function $g(n)$ that is $\omega(\sqrt{\log n})$, when $N \geq b$, we can set $s = n^3 \cdot \|f\|^3 \cdot g(n)$, and then $t = \gamma_{\mathsf{Mult}}(R) \cdot n^{4.5} \cdot \|f\|^3 \cdot g(n)$. □

The reduction from worst-case to average-case IVIP is very similar.

Theorem 17.3.1. *Let $\mathcal{I}_{a,b}$ be the set of invertible prime ideals with norm in $[a, b]$. Suppose that $|\mathcal{I}_{a,b}|/b$ is non-negligible and $a^2 > 2N \cdot t_0^n$ where $t_0 = t + s \cdot \sqrt{n}$ for $s = \omega(\sqrt{\log n})$, $s = n^3 \cdot \|f\|^3 \cdot (b/N)^{1/n} \cdot \omega(\sqrt{\log n})$, and $t \geq \gamma_{\mathsf{Mult}}(R) \cdot n^{1.5} \cdot s$. Suppose that there is an algorithm $\mathcal{A}$ that solves s_{IVIP}-IVIP with overwhelming probability (over the random coins chosen by $\mathcal{A}$) for a ϵ (weighted) fraction of invertible prime ideals $J \in \mathcal{I}_{a,b}$. Then, there is an algorithm $\mathcal{B}$, with running time approximately $O(1/\epsilon)$ that of $\mathcal{A}$, that solves IVIP for any (worst-case) ideal M with $\det(M) \in [N, 2N]$ for parameter $s_{\mathsf{IVIP}}/2t$. In particular, if $N \geq b$, for any $g(n)$ that is $\omega(\sqrt{\log n})$, it solves IVIP for parameter $s_{\mathsf{IVIP}}/(2\gamma_{\mathsf{Mult}}(R) \cdot n^{4.5} \cdot \|f\|^3 \cdot g(n))$.*

Proof. (Theorem 17.3.1) $\mathcal{B}$ is a given an ideal lattice M as its IVIP instance. It does the following:

1. It runs $(\mathbf{B}_J, \mathbf{v}) \xleftarrow{\mathrm{R}} \mathsf{RandomizeIdeal}(R, \mathbf{B}_M, N, s, t, a, b)$.
2. It runs $\mathcal{A}$ on the instance J. If it does not receive back an independent set $\mathbf{B}_{J^{-1}}$ of J^{-1} that satisfies s_{IVIP}-IVIP, it restarts.
3. It sets the ith column of $\mathbf{B}_{M^{-1}}$ to be the ith column of $\mathbf{B}_{J^{-1}}$ times $\mathbf{v}$; it outputs $\mathbf{B}_{M^{-1}}$.

Since RandomizeIdeal generates J as a uniformly random invertible prime ideal with norm in $[a, b]$, $\mathcal{A}$ outputs a satisfactory independent set in Step 2 with probability ϵ.

By the properties of $\mathsf{RandomizeIdeal}$, $J^{-1}(\mathbf{v})$ is a sub-lattice of M^{-1}. By multiplying the generating columns of $\mathbf{B}_{J^{-1}}$ with $\mathbf{v}$, we obtain an independent set $\mathbf{B}_{M^{-1}}$ of M^{-1} that satisfies

$$\|\mathbf{B}_{M^{-1}}\| \leq \|\mathbf{B}_{J^{-1}}\| \cdot t + \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{B}_{J^{-1}}\| \cdot s \cdot \sqrt{n} \leq 2t \cdot \|\mathbf{B}_{J^{-1}}\| \leq 2t/s_{\mathsf{IVIP}} \ ,$$

as required.

We aim to set t as small as possible. For some function $g(n)$ that is $\omega(\sqrt{\log n})$, when $N \geq b$, we can set $s = n^3 \cdot \|f\|^3 \cdot g(n)$, and then $t = \gamma_{\mathsf{Mult}}(R) \cdot n^{4.5} \cdot \|f\|^3 \cdot g(n)$. □

17.4 An Alternative Way to Randomize an Ideal

In this Section, we present an algorithm $\mathsf{RandomizeIdeal}'$ and prove Theorems 16.3.2 and 16.3.3. This algorithm will be used in our (classical) instantiation of $\mathsf{IdealGen}$ in KeyGen. The algorithm invokes an as-yet-unspecified algorithm $\mathsf{RandomIdealGen}(R, i, j)$ that generates the basis of a uniformly random invertible ideal K with $\det(K) \in [i, j]$, together with a short vector $\mathbf{w}_K \in K$. Let $s = \omega(\sqrt{\log n})$ and $t \geq 20 \cdot \gamma_{\mathsf{Mult}}(R) \cdot s \cdot n^2$. Let $S = s \cdot \alpha$ and $T = t \cdot \alpha$ for $\alpha \geq 1$.

$\mathsf{RandomizeIdeal}'(R, \mathbf{B}_M, b, s, t, \alpha)$. Takes as input the ring R, a basis $\mathbf{B}_M$ of an invertible ideal M of R, a bound $b \geq \det(M)$, and s, t, and α (and hence S and T) as above. It does the following.

1. Runs $(\mathbf{B}_K, \mathbf{w}_K) \xleftarrow{\mathrm{R}} \mathsf{RandomIdealGen}(R, bt^{2n}/\det(M), 4bt^{2n}/\det(M))$
2. Generates a vector $\mathbf{v}$ per the discrete Gaussian distribution $D_{(MK)^{-1},S,T\cdot\mathbf{e}_1}$
3. Sets $J \leftarrow M \cdot K \cdot (\mathbf{v})$
4. Let $c_1 = 4bt^{2n}$, an upper bound on $\det(MK)$. It continues to Step 5 with probability $\det(MK)/c_1$; otherwise, it returns to Step 1.
5. With probability $c_2 \cdot \frac{\rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(1/\mathbf{v})}{\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v})}$, continues to Step 6, where c_2 is a constant to be defined later; otherwise, it returns to Step 1.
6. Returns to Step 1 if $\det(J) \notin [2bt^{2n}T^n, 3bt^{2n}T^n]$.

7. With probability $2bt^{2n}T^n/\det(J)$, outputs $\mathbf{B}_K$, $\mathbf{w}_K$, $\mathbf{v}$, and the Hermite normal form of J; otherwise, it returns to Step 1.

Remark 17.4.1. In Step 2, one can sample from the distribution $D_{(MK)^{-1},S,T\cdot\mathbf{e}_1}$ by using the GPV algorithm (see Chapter 13.3) with the independent set $\{\mathbf{e}_i\}$ in $(MK)^{-1}$.

Remark 17.4.2. We will show that $1/C \leq \rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(1/\mathbf{v})/\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v}) \leq C$, where $C = e^{6\pi\sqrt{1/n}}$. (See the proof of Theorem 16.3.2.) Therefore, it suffices to take $c_2 \leftarrow 1/C$ in Step 5.

Remark 17.4.3. $\mathsf{RandomizeIdeal}'$ outputs a uniformly random invertible ideal with norm in a certain interval. It is straightforward to modify the algorithm so that it outputs a uniformly random invertible *prime* ideal – per our preferred average case distribution – simply by running the algorithm repeatedly until the output ideal is prime. This is efficient as long as prime invertible ideals are a non-negligible fraction of invertible ideals in the interval.

Before proving Theorem 16.3.2, we provide some lemmata regarding the properties of $\mathbf{v}$, the vector selected in Step 2.

Lemma 17.4.4. *The vector* $\mathbf{v}$ *drawn in Step 2 is in* $T\cdot\mathbf{e}_1 + \mathcal{B}(S\cdot\sqrt{n})$ *with overwhelming probability, where* $\mathcal{B}(r)$ *is the open ball of radius* r*.*

Proof. Similar to the proof of Lemma 17.2.1. □

Since $\mathbf{v}$ is in the ball $T\cdot\mathbf{e}_1 + \mathcal{B}(S\cdot\sqrt{n})$ with overwhelming probability – i.e., $\mathbf{v}$ is "almost parallel" to the vector $T\cdot\mathbf{e}_1$ and is therefore "almost a real number" – we would expect $\mathbf{v}$ to "behave" almost like a real number, and the lattice $(\mathbf{v})$ to behave almost like a scaling of $\mathbb{Z}^n$. The following lemmas, which borrow from lemmas in Chapter 15.1 characterize this intuition more formally.

Lemma 17.4.5. *When* $\mathbf{v} \in T\cdot\mathbf{e}_1 + \mathcal{B}(S\cdot\sqrt{n})$ *in Step 2, it is the only vector in* $(\mathbf{v})$ *inside that ball. In fact,* $\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v})/\rho_{S,T\cdot\mathbf{e}_1}((\mathbf{v})) = 1-\epsilon$ *for negligible* ϵ*.*

Proof. Let $\mathbf{u}$ be such that $\mathbf{v} = T(\mathbf{e}_1+\mathbf{u})$. We have that $\|\mathbf{u}\| \leq S\cdot\sqrt{n}/T < 1/(\delta\cdot\gamma_{\mathsf{Mult}}(R))$ for $\delta = n$. By Lemma 15.1.3, $\mathbf{v}$ is the only vector in $(\mathbf{v})$ that is within a distance of

$T \cdot ((n-2)/n) > T/2$ of $T \cdot \mathbf{e}_1$. By Lemma 13.2.3,

$$\begin{aligned}\rho_{S,T\cdot\mathbf{e}_1}((\mathbf{v}) \setminus \{\mathbf{v}\}) &\leq \rho_{S,T\cdot\mathbf{e}_1}((\mathbf{v}) \setminus [T \cdot \mathbf{e}_1 + \mathcal{B}(T/2)]) \\ &\leq \rho_{S,T\cdot\mathbf{e}_1}((\mathbf{v}) \setminus [T \cdot \mathbf{e}_1 + \mathcal{B}(S \cdot \gamma_{\mathsf{Mult}}(R) \cdot n^{1.5}/2)]) \\ &\leq 2C^n \cdot \rho_S((\mathbf{v}))\end{aligned}$$

where $C = c\sqrt{2\pi e} \cdot e^{-\pi c^2}$ and $c = \gamma_{\mathsf{Mult}}(R) \cdot n/2$ and

$$\begin{aligned}\rho_S((\mathbf{v})) = 1 + \rho_S((\mathbf{v}) \setminus \mathcal{B}(T/2)) &\leq 1 + C^n \cdot \rho_S((\mathbf{v})) \\ &\leq 1/(1 - C^n)\end{aligned}$$

Thus, $\rho_{S,T\cdot\mathbf{e}_1}((\mathbf{v}) \setminus \{\mathbf{v}\})$ is *extremely* small – approximately, $\exp(-\gamma_{\mathsf{Mult}}(R)^2 \cdot n^3)$. One the other hand, $\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v})$ is not nearly as small, being at least (approximately) $\exp(-n)$, since $\mathbf{v}$ is at most $S \cdot \sqrt{n}$ distant from $T \cdot \mathbf{e}_1$. □

In the next lemma, we apply Lemma 15.1.5 to the vector $\mathbf{v}$.

Lemma 17.4.6. *Suppose $\mathbf{v} \in T \cdot \mathbf{e}_1 + \mathcal{B}(S \cdot \sqrt{n})$ in Step 2. Let $\mathbf{v}' = \mathbf{v}/T$, and $\sigma = S/T$. Then, $\mathbf{v}'$ satisfies the conditions of Lemma 15.1.5 for $\beta = \sqrt{1/n}$. In particular, if $\mathbf{w}' = 1/\mathbf{v}' \in \mathbb{Q}[x]/(f(x))$, then*

$$\rho_{\sigma,\mathbf{e}_1}(\mathbf{w}')/\rho_{\sigma,\mathbf{e}_1}(\mathbf{v}') \in [e^{-6\pi\sqrt{1/n}}, e^{6\pi\sqrt{1/n}}]$$

and consequently

$$\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{w})/\rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(\mathbf{v}) \in [e^{-6\pi\sqrt{1/n}}, e^{6\pi\sqrt{1/n}}]$$

where $\mathbf{w} = 1/\mathbf{v}$.

Proof. Let $\mathbf{v}' = \mathbf{e}_1 + \mathbf{u}$. We have that $\|\mathbf{u}\| \leq S \cdot \sqrt{n}/T < 1/(n^{1.5} \cdot \gamma_{\mathsf{Mult}}(R))$. Also, $\|\mathbf{u}\|^3 \leq \beta \cdot \sigma^2/\gamma_{\mathsf{Mult}}(R) \leq \beta \cdot n^{-4} \cdot \gamma_{\mathsf{Mult}}(R)^{-3}$ for $\beta = \sqrt{1/n}$. □

Now we prove Theorem 16.3.2.

Proof. (Theorem 16.3.2) First, we prove that $\mathsf{RandomizeIdeal}'$ is correct – i.e., that it outputs J according to a distribution that is uniform "up to negligible error" – i.e., $\Pr[J_1]/\Pr[J_2] \leq 1 + \epsilon$ for all J_1, J_2 with norms in the prescribed interval, for some negligible ϵ. (In general,

we use "up to negligible error" to mean an equality holds up to such a $1+\epsilon$.) Afterwards, we prove efficiency. To prove efficiency, it suffices to prove that neither of the rejection sampling in Steps 5 and 6 rejects with overwhelming probability.

Correctness. For fixed M, consider the probability that IdealGen generates the pair (K, J) at Step 3, for some K with $\det(K)$ in the required interval.

$$\Pr[K \wedge J] = \Pr[K] \cdot \Pr[J|K] = c \cdot \Pr[J|K] \ ,$$

where, by the assumption on RandomIdealGen, c is some factor independent of K and M. The ultimate goal is to show that, for all J whose norms are in the prescribed interval, $\sum_K \Pr[J|K]$ is statistically the same.

We claim that

$$\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v})/\rho_{S,T\cdot\mathbf{e}_1}((MK)^{-1}) \quad \leq \quad \Pr[J|K] \quad \leq \quad \rho_{S,T\cdot\mathbf{e}_1}((\mathbf{v}))/\rho_{S,T\cdot\mathbf{e}_1}((MK)^{-1})$$

The left inequality follows from the fact that we implicitly choose the ideal J when $\mathbf{v}$ is chosen in Step 2 (out of all possible vectors in $(MK)^{-1}$). However, $\mathbf{v}$ is not necessarily the only vector that induces J; in fact, for every unit $\mathbf{u} \in R$ (whose norm is 1), sampling $\mathbf{v} \times \mathbf{u}$ in Step 2 induces J. All such $\mathbf{v} \times \mathbf{u}$ are in the ideal $(\mathbf{v})$; hence, the second inequality. From Lemmas 17.4.4 and 17.4.5, $\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v}) = \rho_{S,T\cdot\mathbf{e}_1}((\mathbf{v}))$, up to negligible error; hence the inequalities above are very tight. Thus $\Pr[J|K]$ equals $\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v})/\rho_{S,T\cdot\mathbf{e}_1}((MK)^{-1})$, up to negligible error.

Now, consider the denominator $\rho_{S,T\cdot\mathbf{e}_1}((MK)^{-1})$; we claim that, for fixed (S, T, M), this summation is proportional to $\det(MK)$, up to negligible error. This follows from Lemma 13.2.6, and the fact that S exceeds the smoothing parameter of $(MK)^{-1}$ for any integer ideal M (since $\mathbb{Z}^n$ is a sub-lattice of $(MK)^{-1}$). So, after Step 3, we have

$$\Pr[J|K] = c_1 \cdot \rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v})/\det(MK)$$

up to negligible error for some constant c_1 that is independent of K, and thus, after the rejection sampling in Step 4, we have

$$\Pr[K \wedge J] = c_2 \cdot \rho_{S,T\cdot\mathbf{e}_1}(\mathbf{v})$$

up to negligible error for some constant c_2 that is independent of K.

Let $\mathbf{w} = 1/\mathbf{v}$. After Step 5, we have

$$\Pr[K \wedge J] = c_3 \cdot \rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(\mathbf{w}) \tag{17.1}$$

up to negligible error for some constant c_3 that is independent of K. The rationale for this step is rather technical; essentially, performing this inversion will allow us to compute a certain sum over the K's (rather than awkwardly over K^{-1}'s). This will become clearer momentarily.

In step 6, we eliminate J's that have norm falling outside of the interval. Intuitively, the reason we discard J's that were on the edge of the interval is that these J's tend to be "associated" to K's that were on the edge of their interval. (Roughly, we say J is "associated" to a K when $\Pr[J \wedge K]$ is not absurdly small – i.e., the pair induces a $\mathbf{v}$ that is in the ball $T\cdot\mathbf{e}_1+\mathcal{B}(S\cdot\sqrt{n})$.) The problem with such J's is that they would also be associated to some K's with norms just outside the interval, if we had permitted RandomIdealGen to generate such K's – i.e., these J's have a "truncated" set of associates. It is easier to discard these bad J's and just consider ones whose associates are not truncated.

We claim that for the J's in this interval, after Step 5 (and Step 6), it holds that

$$\sum_K \Pr[K \wedge J] = c_4 \cdot \sum_{\mathbf{w}\in J^{-1}M} \rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(\mathbf{w})$$

up to negligible error for some constant c_4 that is independent of K.

It suffices to show that

$$\sum_K \rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(\mathbf{w}) = \sum_{\mathbf{w}\in J^{-1}M} \rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(\mathbf{w})$$

– i.e., the only issue difference is that (using Equation 17.1) we sum up over K's on the left-hand side, defining $\mathbf{w}$ to be the inverse of $\mathbf{v}$ where $\mathbf{v}$ is the element in $(MK)^{-1}J$ that is inside the ball $T\cdot\mathbf{e}_1+\mathcal{B}(T/2)$ (if there is one; if not, this value of K contributes nothing to the sum), whereas on the right-hand side we are summing up directly over elements of $J^{-1}M$. For fixed J and M, each distinct K on the lhs that could have been chosen (i.e., that leads to a $\mathbf{v}$ in the ball) maps to a *distinct* $\mathbf{v}$ in $(\mathbf{v}) = (MK)^{-1}J$ and thus a distinct $\mathbf{w} \in J^{-1}M$. (Note: $(\mathbf{w}) = J^{-1}MK$.) Thus, the terms summed on the lhs are a subset of

the terms summed on the rhs. To show that the two sides are equal up to negligible error, it suffices to show that the terms omitted on the lhs contribute a negligible fraction of the weight.

So, consider the values of $\mathbf{w} \in J^{-1}M$ that are inside the ball $(1/T)\mathbf{e}_1 + \mathcal{B}(S \cdot \sqrt{n}/T^2)$; these values contribute the overwhelming fraction of the sum on the right, assuming that S/T^2 exceeds the smoothing parameter of $J^{-1}M$, which we will establish shortly. We claim that, that all such $\mathbf{w}$'s are included in the sum on the lhs – i.e., that for every $\mathbf{w}$ in this ball, there is some K with norm in the interval $[bt^{2n}/\det(M), 4bt^{2n}/\det(M)]$ such that $(\mathbf{w}) = J^{-1}MK$ (for our fixed J). In particular, set $K \leftarrow (\mathbf{w}) \cdot JM^{-1}$; it remains to check that K has norm in the correct interval. Since the norm is a multiplicative map among invertible ideals, we have $\det(K) = \det((\mathbf{w})) \cdot \det(J) \cdot \det(M^{-1})$. Consider $\det((\mathbf{w})) = 1/\det((\mathbf{v}))$. We can lower- and upper-bound $\det((\mathbf{v}))$ by finding hypercubes that circumscribe and are circumscribed by the parallelepiped $\mathcal{P}(\mathbf{B}_\mathbf{v})$, where $\mathbf{v}$ is the rotation basis of $\mathbf{v}$. Let $\mathbf{v}_i = \mathbf{v} \times x^i \bmod f(x)$. For every point $\mathbf{a}$ on the surface of this parallelepiped, there is an i such that

$$\mathbf{a} = (\pm 1/2) \cdot \mathbf{v}_i + \sum_{j \neq i} x_j \cdot \mathbf{v}_j$$

for $x_j \in [-1/2, 1/2]$. So,

$$T/2 - n \cdot \gamma_{\mathsf{Mult}}(R) \cdot S \leq |\langle \mathbf{a}, \mathbf{e}_i \rangle| \leq T/2 + n \cdot \gamma_{\mathsf{Mult}}(R) \cdot S$$

So, the parallelepiped inscribes a hypercube, centered at the origin, with sides of length $T - 2n \cdot \gamma_{\mathsf{Mult}}(R) \cdot S \geq T(1 - 1/10n)$ and volume approximately $T^n/1.1$. Similarly, it is circumscribed by a hypercube with volume approximately $T^n \cdot 1.1$. Given these bounds on $\det((\mathbf{w}))$, and since $\det(J) \in [2t^{2n}T^n b, 3t^{2n}T^n b]$, we have that $\det(MK) \in [2t^{2n}b/1.1, 3t^{2n}b \cdot 1.1]$, which proves the claim.

We claim that S/T^2 exceeds the smoothing parameter of $J^{-1}M$. The norm of J^{-1} is at most $1/(2bt^{2n}T^n)$ and the norm of M is at most b. Therefore, the norm of $J^{-1}M$ is at most $1/(2t^{2n}T^n)$. By Minkowski, $\lambda_1(J^{-1}M)$ is at most $\sqrt{n}/(t^2T)$; let $\mathbf{a}$ be a nonzero vector in $J^{-1}M$ whose length is at most this value. Then we have:

$$\lambda_n(J^{-1}M) \leq \max_i \{\|\mathbf{a} \times x^i\|\} \leq \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{a}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot \sqrt{n}/(t^2T) \leq 1/(n^{1.5}tT)$$

The value $S/T^2 > 1/tT$ is much larger; in particular, it is $\lambda_n(J^{-1}M) \cdot \omega(\sqrt{\log n})$ – i.e., larger than the smoothing parameter of $J^{-1}M$.

Finally, we consider the sum

$$\sum_{\mathbf{w} \in J^{-1}M} \rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(\mathbf{w})$$

Since S/T^2 exceeds the smoothing parameter of $J^{-1}M$, the sum $\sum_{\mathbf{w}\in J^{-1}M} \rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(\mathbf{w})$ is proportional to $\det(J)$, for reasons similar to those discussed above. The final step gets rid of the dependence on $\det(J)$ – i.e., $\Pr[J]$ becomes constant, up to negligible error.

Efficiency. Assuming RandomIdealGen is efficient, the efficiency of RandomizeIdeal′ follows from our claims that the algorithm does not return to Step 1 with overwhelming probability in Steps 5-6.

Regarding Step 5, we invoke Lemma 17.4.6, which says that with overwhelming probability it holds that

$$\rho_{S,T\cdot\mathbf{e}_1}(\mathbf{w})/\rho_{S/T^2,(1/T)\cdot\mathbf{e}_1}(\mathbf{v}) \in [e^{-6\pi\sqrt{1/n}}, e^{6\pi\sqrt{1/n}}]$$

and therefore we can take $c_2 \leftarrow e^{-6\pi\sqrt{1/n}}$ as claimed, so that the probability of rejecting is very small.

As for the probability of rejecting J in Step 6, we see that this only occurs if $\det(J)$ is outside of $[2t^{2n}T^n b, 3t^{2n}T^n b]$. But since the associated value of $(\mathbf{w}) = J^{-1}MK$ has norm in the interval $[T^{-n}/1.1, T^{-n} \cdot 1.1]$ with overwhelming probability, this implies (with overwhelming probability) that MK has norm outside of $[2t^{2n}b \cdot 1.1, 3t^{2n}b/1.1]$ – i.e., K has norm outside of $[2t^{2n}b/\det(M) \cdot 1.1, 3t^{2n}b/1.1]$. By the distribution of ideals (see Chapter 15.2), it seems reasonable to believe that this occurs only with only constant probability, in which case the probability of rejecting in Step 6 is a constant. □

Next, we prove Theorem 16.3.3, showing how to use the procedure RandomizeIdeal′ to reduce WBDDP to HBDDP.

Proof. (Theorem 16.3.3) $\mathcal{B}$ wants to solve the WBDDP instance $(M, \mathbf{u})$. It does the following:

1. It runs $(\mathbf{B}_K, \mathbf{w}_K, \mathbf{v}, J) \xleftarrow{\mathrm{R}} \mathsf{RandomizeIdeal}'(R, \mathbf{B}_M, b, s, t, \alpha)$, using $\alpha = 1$.
2. It sets $\mathbf{t} \leftarrow \mathbf{u} \times \mathbf{v} \times \mathbf{w}_K \bmod \mathbf{B}_J$.

3. It runs $\mathcal{A}$ on the instance $(J, \mathbf{t})$, receiving back a vector $\mathbf{y}$ such that $\mathbf{t} - \mathbf{y} \in J$.

4. It outputs $\mathbf{x} \leftarrow \mathbf{y}/(\mathbf{v} \times \mathbf{w}_K)$.

Regarding Step 1, by assumption we have that the algorithm $\mathsf{RandomIdealGen}$ outputs a basis $\mathbf{B}_K$ for an ideal K that is uniformly random subject to the constraint that its $\det(K)$ is in the specified interval. Therefore, by Theorem 16.3.2, $\mathsf{RandomizeIdeal}'$ outputs the basis $\mathbf{B}_J$ of an ideal J statistically uniformly, subject to the constraint that $\det(J) \in [2bt^{3n}, 3bt^{3n}]$. It also outputs $\mathbf{v}$ satisfying $J = M \cdot K \cdot (\mathbf{v})$ and $\mathbf{v} \in t \cdot \mathbf{e}_1 + \mathcal{B}(s \cdot \sqrt{n})$, the latter with overwhelming probability. Also, we know that $\|\mathbf{w}_K\| < X$ by assumption.

We verify that $\mathcal{A}$ can perform Step 3 – i.e., that $(J, \mathbf{t})$ is a valid HBDDP instance, and that $\mathcal{A}$ therefore should solve it with overwhelming probability for a ϵ fraction of J. Specifically, we have that $\mathbf{u} = \mathbf{m} + \mathbf{z}$ for $\mathbf{m} \in M$ and $\|\mathbf{z}\| \leq s_{\mathsf{WBDDP}}$. We have

$$\begin{aligned} \mathbf{t} &= (\mathbf{m} + \mathbf{z}) \times \mathbf{v} \times \mathbf{w}_K \bmod \mathbf{B}_J \\ &= \mathbf{m} \times \mathbf{v} \times \mathbf{w}_K + \mathbf{z} \times \mathbf{v} \times \mathbf{w}_K \bmod \mathbf{B}_J \\ &= \mathbf{m}' + \mathbf{z}' \bmod \mathbf{B}_J \\ &= \mathbf{z}' \bmod \mathbf{B}_J \end{aligned}$$

where $\mathbf{m}' \in J$ because $M(\mathbf{v})K = J$, and where $\mathbf{z}' = \mathbf{z} \times \mathbf{v} \times \mathbf{w}_K$.

Now, we claim that $\mathbf{z}'$ is short, so that $(J, \mathbf{t})$ is a valid HBDDP instance. Note that $\mathbf{v} = t \cdot \mathbf{e}_1 + s\sqrt{n} \cdot \mathbf{x}$ where $\|\mathbf{x}\| \leq 1$ with overwhelming probability. So,

$$\begin{aligned} \|\mathbf{z}'\| &\leq \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{z} \times \mathbf{v}\| \cdot \|\mathbf{w}_K\| \leq \gamma_{\mathsf{Mult}}(R) \cdot (t + \gamma_{\mathsf{Mult}}(R) \cdot s\sqrt{n}) \cdot \|\mathbf{z}\| \cdot \|\mathbf{w}_K\| \\ &\leq \gamma_{\mathsf{Mult}}(R) \cdot (t + \gamma_{\mathsf{Mult}}(R) \cdot s\sqrt{n}) \cdot s_{\mathsf{WBDDP}} \cdot X \leq t^2 \cdot s_{\mathsf{WBDDP}} \cdot X \leq s_{\mathsf{HBDDP}} \end{aligned}$$

where the penultimate inequality holds for the value of t used in $\mathsf{RandomizeIdeal}'$.

Since the HBDDP instance is valid, $\mathcal{A}$ solves it with probability ϵ. When $\mathcal{A}$ solves an instance and outputs $\mathbf{z}' = \mathbf{z} \times \mathbf{v} \times \mathbf{w}_K$, $\mathcal{B}$ outputs $\mathbf{z}$. □

The reduction from worst-case to average-case IVIP is very similar.

Theorem 17.4.7. *Let s, t, and b be defined as in $\mathsf{RandomizeIdeal}'$. Suppose that there is an algorithm $\mathcal{A}$ that solves s_{IVIP}-IVIP with overwhelming probability (over the random coins chosen by $\mathcal{A}$) for a ϵ (weighted) fraction of invertible ideals J with norm in $[2bt^{3n}, 3bt^{3n}]$.*

Suppose that $\mathsf{RandomIdealGen}$ *(efficiently) outputs* $\mathbf{B}_K$ *that includes a vector* $\mathbf{w} \in K$ *with* $\|\mathbf{w}_K\| < X$*. Then, there is an algorithm* $\mathcal{B}$*, with running time approximately* $O(1/\epsilon)$ *that of* $\mathcal{A}$*, that solves IVIP for any (worst-case) ideal* M *with* $\det(M) < b$ *for parameter* $s_{\mathsf{WIVIP}} \leq s_{\mathsf{IVIP}}/((t + s \cdot \sqrt{n} \cdot \gamma_{\mathsf{Mult}}(R)) \cdot \gamma_{\mathsf{Mult}}(R) \cdot X)$*.*

Proof. (Theorem 17.4.7) $\mathcal{B}$ is a given an ideal lattice M as its IVIP instance. It does the following:

1. It runs $(\mathbf{B}_K, \mathbf{w}_K, \mathbf{v}, J) \xleftarrow{\mathrm{R}} \mathsf{RandomizeIdeal}'(R, \mathbf{B}_M, b, s, t, \alpha)$ using $\alpha = 1$.
2. It runs $\mathcal{A}$ on the instance J. If it does not receive back an independent set $\mathbf{B}_{J^{-1}}$ of J^{-1} that satisfies s_{IVIP}-IVIP, it restarts.
3. It sets the ith column of $\mathbf{B}_{M^{-1}}$ to be the ith column of $\mathbf{B}_{J^{-1}}$ times $\mathbf{w}_K \times \mathbf{v}$; it outputs $\mathbf{B}_{M^{-1}}$.

Since $\mathsf{RandomizeIdeal}'$ generates J as a uniformly random invertible ideal with norm in $[2bt^{3n}, 3bt^{3n}]$, $\mathcal{A}$ outputs a satisfactory independent set in Step 2 with probability ϵ.

By the properties of $\mathsf{RandomizeIdeal}'$, $M^{-1} = J^{-1}K(\mathbf{v})$, and thus $J^{-1}(\mathbf{w}_K \cdot \mathbf{v})$ is a sublattice of M^{-1}. By multiplying the generating columns of $\mathbf{B}_{J^{-1}}$ with $\mathbf{w}_K \cdot \mathbf{v}$, we obtain an independent set $\mathbf{B}_{M^{-1}}$ of M^{-1} that satisfies $\|\mathbf{B}_{M^{-1}}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{B}_{J^{-1}}\| \cdot \|\mathbf{w}_K \cdot \mathbf{v}\|$. Note that $\mathbf{v} = t \cdot \mathbf{e}_1 + s\sqrt{n} \cdot \mathbf{x}$ where $\|\mathbf{x}\| \leq 1$ with overwhelming probability. We have that

$$\|\mathbf{w}_K \cdot \mathbf{v}\| \leq t \cdot X + \gamma_{\mathsf{Mult}}(R) \cdot s \cdot \sqrt{n} \cdot \|\mathbf{x}\| \cdot X < (t + s \cdot \sqrt{n} \cdot \gamma_{\mathsf{Mult}}(R)) \cdot X$$

Thus,

$$\|\mathbf{B}_{M^{-1}}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot (t + s \cdot \sqrt{n} \cdot \gamma_{\mathsf{Mult}}(R)) \cdot X / s_{\mathsf{IVIP}}$$

as required. □

Chapter 18

KeyGen per the Average Case Distribution

The first step in RandomizeIdeal$'$, which is invoked by IdealGen, is to generate the basis of an ideal K that is uniformly random subject to its norm being in a prescribed interval. We describe the algorithm to do this, called WeakRandomIdealGen, in this Chapter. The main difference between RandomIdealGen and WeakRandomIdealGen is that, in the latter, we do not generate a short vector $\mathbf{w}_K \in K$. As described below, this weak version suffices for KeyGen.

18.1 The Secret Key

For the moment, let us assume that we have an algorithm WeakRandomIdealGen that outputs a basis $\mathbf{B}_K$ of an ideal that is uniformly random, subject to the constraint that its norm is in some specified interval $[a, b]$. Define WeakRandomizeIdeal$'$ exactly like RandomizeIdeal$'$, except that WeakRandomizeIdeal$'$ uses WeakRandomIdealGen instead of RandomIdealGen, and the output of WeakRandomizeIdeal$'$ includes only $\mathbf{B}_K$, the vector $\mathbf{v}$, and the Hermite normal form $\mathbf{B}_J$ of J. Also, WeakRandomizeIdeal$'$ uses the modification mentioned in Remark 17.4.3; it outputs a random invertible *prime* ideal rather than a random invertible one. As before, $s = \omega(\sqrt{\log n})$ and $t \geq 20 \cdot \gamma_{\mathsf{Mult}}(R) \cdot s \cdot n^2$, $S = s \cdot \alpha$, and $T = t \cdot \alpha$. However, whereas we used $\alpha = 1$ to obtain as tight a reduction as possible (between the worst-case ideal M and the average-case ideal J), here we set M to be $\mathbb{Z}^n$ and use a large value of α – e.g., on the order of $2^{\sqrt{n}}$ – which implicitly induces a large value of r_{Dec}, which permits greater depth

to be evaluated before bootstrapping is needed. We instantiate IdealGen as follows.

$\mathsf{IdealGen}(R, \mathbf{B}_I)$:

1. Run $(\mathbf{B}_K, \mathbf{v}, \mathbf{B}_J) \leftarrow \mathsf{WeakRandomizeIdeal}(R, R, 1)$.
2. Set $\mathbf{B}_J^{\mathrm{pk}} \leftarrow \mathbf{B}_J$.
3. Set $\mathbf{B}_J^{\mathrm{sk}} \leftarrow \mathbf{B}_{\mathbf{v}}$, where $\mathbf{B}_{\mathbf{v}}$ is the rotation basis of $\mathbf{v}$.

Remark 18.1.1. In practice, we will always choose J to have much larger determinant than I, and thus the fact that J is prime will automatically imply that it is relatively prime to I.

Remark 18.1.2. Recall that decryption works fine as long as the ideal lattice $\mathcal{L}(\mathbf{B}_J^{\mathrm{sk}})$ *contains* J. (In our case, this holds since $J = K \cdot (\mathbf{v})$ – i.e., $(\mathbf{v})$ contains J.) Intuitively, if ψ is a ciphertext very close to a J-vector, then it is also very close to a vector in $\mathcal{L}(\mathbf{B}_J^{\mathrm{sk}})$ – though, of course, to decrypt with $\mathbf{B}_J^{\mathrm{sk}}$, this distance needs to be comfortably less than the first minimum of $\mathcal{L}(\mathbf{B}_J^{\mathrm{sk}})$.

Let us consider the value of r_{Dec} associated to $\mathbf{B}_J^{\mathrm{sk}}$, and show that it is only polynomially smaller than $\lambda_1(J)$. This implies that $\mathbf{B}_J^{\mathrm{sk}}$ is the best secret key for J, up to a polynomial factor.

Lemma 18.1.3. *Suppose* $\mathbf{B}_J^{\mathrm{sk}}$, $\mathbf{B}_J^{\mathrm{pk}}$, $\mathbf{v}$ *and* $\mathbf{B}_K$ *are generated by* $\mathsf{IdealGen}(R, I)$*, as described above. Then,* $\lambda_1(J)/r_{\mathsf{Dec}} = \mathrm{poly}(n)$ *if* $\gamma_{\mathsf{Mult}}(R)$ *is polynomial in* n.

Proof. First, let us consider $\lambda_1(J)$. By Theorem 16.3.2, $\det(J) \in [2bt^{2n}T^n, 3bt^{2n}T^n]$. Therefore, $\lambda_1(J) < 3^{1/n} \cdot \sqrt{n} \cdot t^2 \cdot T$ by Minkowski.

Now, consider r_{Dec}, the radius of the largest circle circumscribed by the secret basis $\mathbf{B}_J^{\mathrm{sk}} = \mathbf{B}_{\mathbf{v}}$. By Lemma 8.1.1, we have that $r_{\mathsf{Dec}} \geq 1/(2n \cdot \|(\mathbf{B}_J^{\mathrm{sk}})^{-1}\|)$, where in our case $(\mathbf{B}_J^{\mathrm{sk}})^{-1}$ is the rotation basis of the vector $1/\mathbf{v} \in \mathbb{Q}[x]/(f(x))$. Since $\mathbf{v}$ is of the form $T \cdot (\mathbf{e}_1 + \mathbf{u})$ for $\|\mathbf{u}\| \leq S \cdot \sqrt{n}/T \leq 1/(\gamma_{\mathsf{Mult}}(R) \cdot n^{1.5})$, we can apply Lemma 15.1.4 to obtain that

$$1/\mathbf{v} = (1/T) \cdot (\mathbf{e}_1 - \mathbf{u} + \mathbf{x}) \quad \text{for} \quad \|\mathbf{x}\| \leq \frac{\gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|^2}{1 - \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\|}$$

The rotation basis consists of the vectors $\mathbf{e}_i/\mathbf{v}$, which has length at most

$$(1/T) \cdot (\|\mathbf{e}_i\| + \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{u}\| + \gamma_{\mathsf{Mult}}(R) \cdot \|\mathbf{x}\|) \leq (1/T) \cdot \left(1 + 1/n^{1.5} + 2/n^3\right) \leq 2/T$$

Consequently, $r_{\mathsf{Dec}} \geq T/4n$.

The ratio of $\lambda_1(J)$ and r_{Dec}, which is bounded by $4(3^{1/n})n^{1.5}t^2$, is therefore polynomial in n, assuming t is polynomial in n. We can set t to be polynomial in n if $\gamma_{\mathsf{Mult}}(R)$ is polynomial in n. □

18.2 Adapting Kalai's Algorithm to Generate a Random Factored Ideal

Now we describe how to instantiate WeakRandomIdealGen, an efficient classical algorithm for generating a basis $\mathbf{B}_K$ of an invertible ideal K that is uniformly random, subject to the constraint that its norm is in a prescribed interval. Several difficulties conspire to make this less trivial than it sounds.

1. The ring $\mathcal{O}_{\mathbb{Q}(\alpha)}$, which contains our ring $\mathbb{Z}[\alpha]$ where α is a root of $f(x)$, enjoys unique factorization of ideals. However, $\mathcal{O}_{\mathbb{Q}(\alpha)}$ certainly does not have an ideal $\mathfrak{p}$ of norm p for every integer prime p. In other words, one cannot simply pick an integer N and expect that $\mathcal{O}_{\mathbb{Q}(\alpha)}$ that has an ideal of norm N.

2. If there is an ideal of norm N, there may be *multiple* ideals of norm N. In particular, for prime p, if there is one ideal of norm p, there may be up to n ideals of norm p, where n is the degree of $f(x)$. If N has many prime factors, the number of ideals with norm N may be very large: up to about $n^{\log N/\log\log N}$.

3. If $\mathcal{O}_{\mathbb{Q}(\alpha)}$ has an ideal I of norm N, it is still not necessarily easy to find a *basis* of I unless N is prime. If N is prime, one can find the ideals of norm N^e (for some $e \geq 1$) essentially by factoring $f(x)$ modulo N. (See Chapter 13.4.) However, this is not efficient if N is composite.

These considerations, particularly the last one, lead us to construct an algorithm for generating a random *factored* ideal whose norm is in the prescribed interval, even though, in principle, we do not need the factorization. Our algorithm for generating a random factored

ideal is a modification of Kalai's algorithm for generating a random factored number [74]. Let us recall Kalai's algorithm.

Kalai's Algorithm for Generating a Random Factored Number.

Input: Integer $b > 0$.

Output: A uniformly random number $1 \leq N \leq b$, with its factorization.

1. Generate a sequence $b \geq s_1 \geq s_2 \geq \cdots \geq s_\ell = 1$ by uniformly choosing $s_1 \in \{1, \ldots, b\}$ and $s_{i+1} \in \{1, \ldots, s_i\}$.
2. Let N be the product of the prime s_i's.
3. If $N > b$, restart.
4. Output N and the prime s_i's with probability N/b; otherwise, restart.

The main idea behind Kalai's algorithm is that the probability that a prime $p \leq b$ is in the sequence is $1/p$, since p is not chosen iff one of $1, \ldots, p-1$ is chosen before p (which occurs with probability $(p-1)/p$). Overall, the probability that exactly p^e divides N is $1/p^e - 1/p^{e+1} = (p-1)/p^{e+1}$, and (subject to the constraint that all of N's factors must be at most b) the probability that $N = \prod_i p_i^{e_i}$ is chosen is $\prod_j (p_j - 1)/p_j^{e_j+1} = (1/N) \cdot \prod_j (p_j - 1)/p_j$, where p_j runs over *all* primes up to b. Thus, the probability that N is chosen is exactly $1/N$, up to some factor that does not depend on N. Two rejection sampling steps ensure the correct distribution. By Mertens' theorem, the algorithm will *not* restart in Step 3 with probability $\theta(1/\log b)$.

We would like our algorithm for generating a random factored ideal to have a property similar to Kalai's algorithm: that, before some rejection sampling steps, the probability that a prime ideal $\mathfrak{p}$ is in the list is $1/\det(\mathfrak{p})$. Kalai's strategy for ensuring this property is very elegant; ideally, we would like to replicate it. However, as mentioned above, there is a considerable difficulty: there may be many (up to n) prime ideals whose norms are to the same prime integer. This makes it difficult to generate a sequence from large to small, as Kalai does, since some ideals are "tied" in terms of largeness. We break these ties by mapping prime ideals of norm q to distinct integers $[n(q-2)+1, n(q-1)]$. We then use Kalai's algorithm to sample from the integers, and then pick prime ideals as pre-images of these integers according to this map. Then, similar to Kalai, we multiply together the prime powers in our list, and perform a couple of rejection steps before outputting the

result. From the bases of the individual prime factors, it easy to generate a basis of the composite ideal (essentially by multiplying the bases together). By an analogue of Mertens' theorem for ideals in number fields, our random factored ideal will have a norm less than our desired bound with inverse-log probability.

We now present the algorithm; afterwards, we review the purpose of the individual steps. Let $h : \mathcal{I} \to \mathbb{N}$ be an efficiently computable injective map from invertible prime ideals in R to positive integers, such that $h(\mathfrak{p}) \in [n \cdot (\mathrm{Nm}(\mathfrak{p}) - 2) + 1, n \cdot (\mathrm{Nm}(\mathfrak{p}) - 1)]$. (Such an h can always be constructed (incrementally), since there are at most n prime ideals of a given norm.)

WeakRandomIdealGen(R, a, b).

Input: The ring R and integers $b > a > 1$ with $(b - a)/b$ non-negligible.

Output: A basis $\mathbf{B}_K$ of a uniformly random invertible ideal K with norm in $[a, b]$.

1. Generate n integer sequences $b \geq s_{i1} \geq s_{i2} \geq \cdots \geq s_{i\ell_i} = 1$ as in Kalai's algorithm.
2. Set S_0 to be the set of integers in the sequences, with no duplications.
3. Set S_1 to be the subset of S_0 consisting of those integers that equal $h(\mathfrak{p})$ for some invertible prime ideal $\mathfrak{p}$.
4. Generate S_2 from S_1 as follows. For each $r \in S_1$, replace r with the prime ideal $\mathfrak{p}$ such that $h(\mathfrak{p}) = r$ with probability $1/(N_{\mathfrak{p}}(1 - (1 - 1/r)^n))$; otherwise, nothing replaces r. (We use $N_{\mathfrak{p}}$ to abbreviate $\mathrm{Nm}(\mathfrak{p})$.)
5. Construct a set S_3 by, for each $\mathfrak{p} \in S_2$, replacing $\mathfrak{p}$ with the power $\mathfrak{p}^e$ with probability $(\mathrm{Nm}(\mathfrak{p}) - 1)/\mathrm{Nm}(\mathfrak{p})^{e+1}$.
6. Set K to be the product of the prime power ideals in S_3, and $\mathbf{B}_K$ to be a basis for K.
7. If $\det(K) \in [a, b]$, output $\mathbf{B}_K$ with probability $\det(K)/b$; otherwise, restart.

Remark 18.2.1. Regarding Step 3, as discussed in Chapter 13.4, there are efficient algorithms for recognizing when a prime power integer is the norm of a prime ideal, and for computing a basis of such a prime ideal.

Regarding WeakRandomIdealGen, we have the following theorem.

Theorem 18.2.2. *The algorithm uniformly samples an invertible ideal $K \subset R$ with norm in $[a, b]$. The algorithm takes time $b/(a-b) \cdot \mathrm{poly}(n, \log b)$.*

Proof. (Theorem 18.2.2) Consider an invertible prime ideal $\mathfrak{p}$. What is the probability that $\mathfrak{p}$ is in S_2? Let $r = h(\mathfrak{p})$, and consider the probability that r is in S_0. For any integer $r' \in [1, b]$, and any *single* sequence, the probability that r' is in that sequence is $1/r'$, just as in Kalai's algorithm. So, the probability that r is in at least one of the sequences is $1-(1-1/r)^n$. Then, the probability that $\mathfrak{p} \in S_2$ is $1/N_{\mathfrak{p}}$, assuming $1/(N_{\mathfrak{p}}(1-(1-1/r)^n))$ is in fact a quantity between 0 and 1 (so that Step 4 is possible).

This assumption is true, since

$$\begin{aligned} r < n \cdot (N_{\mathfrak{p}} - 1) + 1 \quad & \Rightarrow n/(r-1) > 1/(N_{\mathfrak{p}} - 1) \\ & \Rightarrow (r/(r-1))^n > N_{\mathfrak{p}}/(N_{\mathfrak{p}} - 1) \\ & \Rightarrow (1-1/r)^n < 1 - 1/N_{\mathfrak{p}} \\ & \Rightarrow 1-(1-1/r)^n > 1/N_{\mathfrak{p}} \end{aligned}$$

Step 5 adjusts the probabilities so that they are analogous to those in Kalai's algorithm, and the rest of the steps and analysis mirror his algorithm. Specifically, for each invertible prime ideal $\mathfrak{p}_i$ of norm at most b, the probability that $\mathfrak{p}_i^{t_i}$ is chosen, and not $\mathfrak{p}_i^{t_i'}$ for some $t_i' > t_i$, is $(\mathrm{Nm}(\mathfrak{p}_i))^{t_i} \cdot (1 - 1/\mathrm{Nm}(\mathfrak{p}_i))$. The probabilities associated to different prime ideals are independent. So, for an invertible ideal $M = \prod_{\mathrm{Nm}(\mathfrak{p}_i) \le b} \mathfrak{p}_i^{t_i}$, we have

$$\Pr[M] = 1/\mathrm{Nm}(M) \cdot \prod_{\mathrm{Nm}(\mathfrak{p}_i) \le b} (1 - 1/\mathrm{Nm}(\mathfrak{p}_i))$$

through Step 6. After Step 7, $\Pr[M]$ becomes $(1/b) \cdot \prod_{\mathrm{Nm}(\mathfrak{p}_i) \le b}(1 - 1/\mathrm{Nm}(\mathfrak{p}_i))$, which is independent of M. By Merten's theorem for number fields, we have

$$\prod_{\mathrm{Nm}(\mathfrak{p}_i) \le b} (1 - 1/\mathrm{Nm}(\mathfrak{p}_i)) = \frac{e^{-\gamma}}{a_K} \frac{1}{\log b} + O(\frac{1}{\log^2 b})$$

where a_K is the residue of $\zeta_K(s)$, the Dedekind zeta-function, at $s = 1$, and γ denotes Euler's constant $0.577\ldots$. Since there are $\theta(b)$ ideals of norm at most b, there is an inverse-log probability that $\mathrm{Nm}(K) \le b$. Among K's with norm at most b, approximately a $(b-a)/b$ fraction of them have norm at least a. The result follows. □

Chapter 19

Basing Security on Worst-case SIVP in Ideal Lattices

So far, we have based the security of our system on worst-case BDDP (WBDDP) over ideal lattices (along with the additional problem SSSP induced by our technique for squashing the decryption circuit). Theorem 14.6.2 uses Regev's reduction [119] to establish that, for each individual ideal lattice J, IVIP reduces to BDDP. Since this holds whether J is a worst-case or an average-case instance, this bases the security of our system on the worst-case IVIP.

There is a subtlety here; the worst-case / average-case reduction requires a lower bound on the determinant of the ideal used in the average-case instance in terms of the worst-case instance – i.e., worst-case ideal might be required to have a smaller determinant than the average-case ideal. But what if IVIP is hard only when the determinant of the ideal lattice in question is large, and our worst-case instances are easy?

In this Chapter, we continue the series of reductions. First, we establish that IVIP instances tend to be harder when the determinant of the lattice in question is small. (This reduction uses a factoring oracle, and therefore the reduction is polynomial time in the quantum setting.) Next, we show that the shortest independent vector problem (SIVP) over ideal lattices is hard if the IVIP problem is hard for all lattices whose determinants exceed a particular bound. This bases the security of our system on the SIVP over ideal lattices.

19.1 Relationship Among Instances of IVIP

The following theorem clarifies that, if one has access to a factoring oracle (which can be instantiated efficiently with quantum computation), the harder instances of IVIP involve ideal lattices with smaller determinants.

Theorem 19.1.1. *Suppose that there is an algorithm $\mathcal{A}$ that solves s_{IVIP}-IVIP whenever the given ideal has $\det(J) \in [a, b]$ for $[a, b] = [d^n_{\mathsf{IVIP}}, 2 \cdot d^n_{\mathsf{IVIP}}]$. Let M be an ideal with norm in $[N, 2N]$ with $N \geq a$. Assume that invertible prime ideals with norms in $[a, b]$ are not negligibly sparse. Then, there is an algorithm $\mathcal{B}$ that solves IVIP for M for parameter $s_{\mathsf{IVIP}}/(2\gamma_{\mathsf{Mult}}(R) \cdot n^{2.5} \cdot \|f\| \cdot g(n))$ for any $g(n)$ that is $\omega(\sqrt{\log n})$.*

Intuitively, it makes sense that solving IVIP for ideal lattices of smaller determinant is the harder case. For any $m > 1$, if M has large enough determinant, then $\lambda_n(M^{-1}) < 2^{-n}/m$. In this case, LLL will return an independent set of M^{-1} of length at most $1/m$, thus solving m-IVIP. It seems reasonable to guess that, even when $\det(M)$ is not so large, IVIP should become easier as the determinant of M becomes larger. Theorem 19.1.1 establishes that this is indeed true (with access to a factoring oracle).

Proof. (Theorem 19.1.1) If $\det(M) \in [a, b]$, then solve IVIP for M immediately using $\mathcal{A}$. Otherwise, assume $N \geq b$.

Our proof uses techniques similar to those in RandomizeIdeal. Let s and t be such that $s = \omega(\sqrt{\log n})$, $s = n \cdot \|f\| \cdot (b/N)^{1/n} \cdot \omega(\sqrt{\log n})$ and $t \geq \gamma_{\mathsf{Mult}}(R) \cdot n^{1.5} \cdot s$. Let $\mathcal{I}_{a,b}$ be the set of invertible prime ideals with norms in $[a, b]$. Generate a vector $\mathbf{v}$ per the discrete Gaussian distribution $D_{M^{-1},s,t\cdot\mathbf{e}_1}$ and set $L \leftarrow M \cdot (\mathbf{v})$. By Lemma 15.2.3, if $|\mathcal{I}_{a,b}|/b$ is non-negligible, then the probability that the ideal $M \cdot (\mathbf{v})$ has a divisor in $\mathcal{I}_{a,b}$ is non-negligible. Use a factoring oracle to discover whether there is such a factor. If there is not, choose a new $\mathbf{v}$. If there is, set J to be such a factor.

$\mathcal{B}$ obtains from $\mathcal{A}$ an independent set $\mathbf{B}_{J^{-1}}$ of J^{-1} satisfying $\|\mathbf{B}_{J^{-1}}\| \leq 1/s_{\mathsf{IVIP}}$. This independent set is also an independent set of L^{-1}, and we obtain an independent set $\mathbf{B}_{M^{-1}}$ of M^{-1} from it by multiplying the vectors of $\mathbf{B}_{J^{-1}}$ by $\mathbf{v}$. As in the proof of Theorem 17.3.1, we obtain $\|\mathbf{B}_{M^{-1}}\| \leq 2t/s_{\mathsf{IVIP}}$.

We aim to set t as small as possible. For some function $g(n)$ that is $\omega(\sqrt{\log n})$, since $N \geq b$, we can set $s = n \cdot \|f\| \cdot g(n)$, and then $t = \gamma_{\mathsf{Mult}}(R) \cdot n^{2.5} \cdot \|f\| \cdot g(n)$. □

19.2 Reduction of SIVP to IVIP

In SIVP, the length requirement on the output basis is stated in more absolute terms as a multiple of the nth minimum of the lattice, rather than relative to a trivial known basis.

Definition 19.2.1 (Ideal SIVP)**.** Fix ring R and a positive real $d_{\mathsf{SIVP}} \geq 1$. Let $\mathbf{B}_M$ be a basis for an ideal lattice M of R. The problem is: given $\mathbf{B}_M$ (and the fixed values), output an independent set $\mathbf{B}_M$ of M for which $\|\mathbf{B}_M\| \leq d_{\mathsf{SIVP}} \cdot \lambda_n(M)$.

Toward reducing Ideal SIVP to Ideal IVIP, it is convenient to use an intermediate problem.

Definition 19.2.2 (Inverse Ideal SIVP)**.** Fix ring R and a positive real $d_{\mathsf{ISIVP}} \geq 1$. Let $\mathbf{B}_M$ be a basis for an ideal lattice M of R. The problem is: given $\mathbf{B}_M$ (and the fixed values), output an independent set $\mathbf{B}_{M^{-1}}$ of M^{-1} for which $\|\mathbf{B}_{M^{-1}}\| \leq d_{\mathsf{ISIVP}} \cdot \lambda_n(M^{-1})$.

It is easy to reduce Ideal SIVP to Inverse Ideal SIVP. (For convenience, we will use SIVP and ISIVP to refer to these two problems.)

Theorem 19.2.3. *Suppose that there is an algorithm $\mathcal{A}$ that solves d_{ISIVP}-ISIVP. Then there is an algorithm $\mathcal{B}$ that solves d_{ISIVP}-SIVP.*

Proof. $\mathcal{B}$ is given the basis $\mathbf{B}_M$ of an ideal M for which it wants to solve SIVP. It gives to $\mathcal{A}$ a basis $\mathbf{B}_J$ of the ideal $J \leftarrow \det(M) \cdot M^{-1} \subset R$. $\mathcal{A}$ sends back an independent set $\mathbf{B}_{J^{-1}}$ of J^{-1} for which $\|\mathbf{B}_{J^{-1}}\| \leq d_{\mathsf{ISIVP}} \cdot \lambda_n(J^{-1})$. We know that $J^{-1} = (1/\det(M)) \cdot M$; so, by multiplying $\mathbf{B}_{J^{-1}}$ by $\det(M)$, we obtain an independent set $\mathbf{B}_M$ of M that satisfies $\|\mathbf{B}_M\| \leq d_{\mathsf{ISIVP}} \cdot \lambda_n(M)$. □

The following theorem states the reduction from ISIVP to IVIP.

Theorem 19.2.4. *Let $d_{\mathsf{ISIVP}} = (3 \cdot e)^{1/n} \cdot d_{\mathsf{IVIP}}$, where e is Euler's constant. Suppose that there is an algorithm $\mathcal{A}$ that solves IVIP for parameter $s_{\mathsf{IVIP}} > 8 \cdot \gamma_{\mathsf{Mult}}(R) \cdot n^{2.5}\omega(\sqrt{\log n})$ for all ideals with determinant at least d^n_{IVIP}. Then, there is an algorithm $\mathcal{B}$ that solves d_{ISIVP}-ISIVP. The running time is approximately $\log d_{\mathsf{IVIP}}$ times that of $\mathcal{A}$.*

Combining Theorem 19.1.1 with Theorem 19.2.4, we have the following corollary.

Corollary 19.2.5. Suppose that there is an algorithm $\mathcal{A}$ that solves s_{IVIP}-IVIP for $s_{\mathsf{IVIP}} > 16 \cdot \gamma_{\mathsf{Mult}}(R)^2 \cdot n^5 \cdot \|f\| \cdot g(n)$ for some $g(n)$ that is $\omega(\log n)$, whenever the given ideal has

$\det(J) \in [a, b]$ for $[a, b] = [d^n_{\mathsf{IVIP}}, 2 \cdot d^n_{\mathsf{IVIP}}]$. Assume that invertible prime ideals with norms in $[a, b]$ are not negligibly sparse. Then, there is an algorithm $\mathcal{B}$ that solves worst-case d_{ISVIP}-ISIVP for $d_{\mathsf{ISVIP}} = (3 \cdot e)^{1/n} \cdot d_{\mathsf{IVIP}}$, where e is Euler's constant.

Roughly speaking (and somewhat inaccurately), our reduction from ISIVP to IVIP will work as follows. We are given an ideal M for which we want to solve ISIVP – i.e., find a short independent set for M^{-1}. Our reduction will solve the ISIVP by using our IVIP algorithm $\mathcal{A}$ recursively. We feed M to $\mathcal{A}$. $\mathcal{A}$ sends back an "improved" basis $\mathbf{B}_{M^{-1}}$ of M^{-1} for which $\|\mathbf{B}_{M^{-1}}\| \leq 1/s$ for some $s > 1$. We use this improved basis to find a different ideal lattice, J_1, that has the basically same "shape" as M, but a smaller determinant. Our method for doing this is to use the GPV algorithm to sample $\mathbf{v}$ from the intersection of M^{-1} and the translated ball $(1/2)\mathbf{e}_1 + \mathcal{B}(\sqrt{n}(\log n)/s)$, and then to set $J_1 \leftarrow (\mathbf{v}) \cdot M$, which is an integer ideal lattice. GPV is able to use the *improved* basis to sample from this ball, even though the ball's radius is somewhat small (when s is large enough). The vector $\mathbf{v}$ is very close, and nearly parallel, to $(1/2)\mathbf{e}_1$. For this reason, the rotation basis of $\mathbf{v}$ is a nearly orthogonal matrix. In particular, the mapping from J_1^{-1} to M^{-1} given by multiplication by the vector $\mathbf{v}$ roughly preserves "shape." Also, we have $\det(J_1) = \det(\mathbf{v}) \cdot \det(M) \approx 2^{-n} \cdot \det(M)$ – i.e., the determinant decreases. Ultimately, through recursion, we end up with a lattice J_i whose determinant is less than d^n_{IVIP}, and which has basically the same shape as M – i.e., again, a known nearly orthogonal matrix transforms one ideal to the other. Since J_i has determinant less than d^n_{IVIP}, and therefore $\lambda_n(J_i^{-1}) \geq 1/d_{\mathrm{IVIP}}$, the "unimproved" basis $\{\mathbf{e}_i\}$ is a d_{IVIP}-ISIVP solution for J_i. We then use the nearly orthogonal matrix that transforms J_i^{-1} to M^{-1} to obtain a $(3 \cdot e)^{1/n} \cdot d_{\mathrm{IVIP}}$-approximate solution to ISIVP for M.

Proof. (Theorem 19.2.4) Given basis $\mathbf{B}_M$ of ideal M, $\mathcal{B}$ does the following pre-processing operation.

1. It generates a basis of M^{-1} and runs LLL on it to obtain a basis $\mathbf{B}_{M^{-1}}$ of M^{-1} that satisfies $\|\mathbf{B}_{M^{-1}}\| \leq 2^n \cdot \lambda_n(M^{-1})$.

2. It generates a vector $\mathbf{v}_0 \in M^{-1}$ that is in

$$s_{\mathrm{IVIP}} \cdot \|\mathbf{B}_{M^{-1}}\| \cdot \big(\ (1/2)\mathbf{e}_1 + \mathcal{B}(\sqrt{n}(\log n)/s_{\mathrm{IVIP}}) \ \big)$$

3. It sets $J_0 \leftarrow (\mathbf{v}_0) \cdot M$ and computes a basis $\mathbf{B}_0$ of J_0.

It sets $i = 0$ and enters the following loop.

1. If $\det(J_i) < d^n_{\text{IVIP}}$, it outputs the rotation basis $\mathbf{B_w}$ of the vector $\mathbf{w} \leftarrow \prod_{j=0}^{i} \mathbf{v}_j$ and $i_{end} \leftarrow i$; break.

2. It runs $\mathcal{A}$ on $\mathbf{B}_i$ and receives from $\mathcal{A}$ a basis $\mathbf{B}^*_i$ of J_i^{-1} with $\|\mathbf{B}^*_i\| \leq 1/s_{\text{IVIP}}$.

3. It generates a vector $\mathbf{v}_{i+1} \in J_i^{-1} \cap ((1/2)\mathbf{e}_1 + \mathcal{B}(\sqrt{n}(\log n)/s_{\text{IVIP}}))$.

4. It sets $J_{i+1} \leftarrow (\mathbf{v}_{i+1}) \cdot J_i$, computes a basis $\mathbf{B}_{i+1}$ of J_{i+1}, and increments i.

(Notice that all of the J_i are integer ideals, even though the $\mathbf{v}_i$'s are not necessarily elements of R, but rather elements of $\mathbb{Q}[x]/(f(x))$. For example, J_0 is an integer ideal since it is generated by $\mathbf{v}_0 \times \mathbf{m}_i$ for $\mathbf{m}_i \in M$, and all of these generators are elements of R since $\mathbf{v}_0 \in M^{-1}$.)

First, we check that $\mathcal{B}$ can perform all of the steps efficiently; this is obvious aside except for Step 2 of the preprocessing operation and Step 3 of the loop. In both cases, we pick a point from the intersection of a lattice and a ball whose radius is at least $\sqrt{n}\log n$ times the length of some independent set that we have for that lattice. (In the preprocessing step, we trivially have the independent set $\{\mathbf{e}_i\}$ of M^{-1}, and in the loop step we have $\mathbf{B}^*_i$, an independent set of J_i^{-1} received from the IVIP-solver $\mathcal{A}$ that must satisfy $\|\mathbf{B}^*_i\| \leq 1/s_{\text{IVIP}}$ by assumption.) We can use the GPV algorithm (see Chapter 13) to sample a point from the lattice according to a discrete Gaussian distribution (centered anywhere we want) whose deviation parameter is less than $\log n$ times the length of our independent set. By Lemma 13.2.2, with overwhelming probability, the sampled point falls within a ball of radius $\sqrt{n}$ times that deviation parameter. This implies that we can efficiently sample the $\mathbf{v}_i$ as claimed.

Next, we claim that the algorithm terminates in polynomial time with an i_{end} such that $\det(J_{i_{end}}) < d^n_{\text{IVIP}}$. Obviously, the algorithm does not break until $\det(J_i) < d^n_{\text{IVIP}}$. Also, we have that $\det(J_{i+1}) = \det((\mathbf{v}_i)) \cdot \det(J_i) \approx (1/2)^n \cdot \det(J_i)$ – i.e., the determinant decreases substantially with each iteration.

A bit more accurately, Lemma 15.1.6 tells us that $\det((\mathbf{v}_i)) < e/2^n$. Suppose $\det(J_i) > d^n_{\text{IVIP}}$. We have that $\det(J_i) < (e/2^n)^i \cdot \det(J_0)$, implying

$$i < \log_{2^n/e}(\det(J_0)/\det(J_i)) < \log_{2^n/e}(\det(J_0)/d^n_{\text{IVIP}}) \leq \log_{2^n/e}\det(J_0)$$

We also have that

$$\begin{aligned}
\det(J_0) &= \det((\mathbf{v}_0)) \cdot \det(M) \\
&\leq e \cdot (s_{\mathrm{IVIP}}/2)^n \cdot \|\mathbf{B}_{M^{-1}}\|^n \cdot \det(M) \\
&\leq e \cdot (s_{\mathrm{IVIP}}/2)^n \cdot (2^n \cdot \gamma_{\mathsf{Mult}}(R) \cdot \det(M)^{-1/n})^n \cdot \det(M) \\
&\leq e \cdot (2^n \cdot s_{\mathrm{IVIP}} \cdot \gamma_{\mathsf{Mult}}(R)/2)^n
\end{aligned}$$

Since $(2^n/e)^{2n-1} > e \cdot (2^n \cdot s_{\mathrm{IVIP}} \cdot \gamma_{\mathsf{Mult}}(R)/2)^n$ for reasonable values of s_{IVIP} and $\gamma_{\mathsf{Mult}}(R)$, we have that $\det(J_i) > d_{\mathrm{IVIP}}^n$ implies $i < 2n-1$. Therefore, $i_{end} \leq 2n-1$ and the algorithm terminates in polynomial time.

Finally, let us consider the output basis $\mathbf{B}_{\mathbf{w}}$, the rotation basis of $\mathbf{w} \leftarrow \prod_{j=0}^{i_{end}} \mathbf{v}_j$. Since $(\mathbf{w}) = J_{i_{end}} \cdot M^{-1}$, we have that $(\mathbf{w})$ is a sub-lattice of M^{-1} and therefore the rotation basis $\mathbf{B}_{\mathbf{w}}$ is an independent set of M^{-1}. It remains to compute $\|\mathbf{B}_{\mathbf{w}}\|$. From the fact that $\det(J_{i_{end}}) < d_{\mathrm{IVIP}}^n$, we have that

$$\det((\mathbf{w})) < d_{\mathrm{IVIP}}^n / \det(M)$$

We claim that

$$\|\mathbf{B}_{\mathbf{w}}\| \leq (3e)^{1/n} \cdot \det((\mathbf{w}))^{1/n}$$

The theorem follows from this claim, since

$$\det((\mathbf{w}))^{1/n} \leq d_{\mathrm{IVIP}} \cdot \det(M^{-1})^{1/n} \leq d_{\mathrm{IVIP}} \cdot \lambda_n(M^{-1})$$

For $j > 0$, we have $\mathbf{v}_j = (1/2)(\mathbf{e}_1 + \mathbf{x}_j)$, where $\|\mathbf{x}_j\| \leq 2\sqrt{n}(\log n)/s_{\mathrm{IVIP}}$; for $j = 0$ these equations are just multiplied by $s_{\mathrm{IVIP}} \cdot \|\mathbf{B}_{M^{-1}}\|$ – i.e., $\mathbf{v}_0 = (s_{\mathrm{IVIP}} \cdot \|\mathbf{B}_{M^{-1}}\|/2)(\mathbf{e}_1 + \mathbf{x}_0)$, where $\|\mathbf{x}_0\| \leq 2\sqrt{n}(\log n)/s_{\mathrm{IVIP}}$. So,

$$\prod_{j=0}^{i_{end}} \mathbf{v}_j = s_{\mathrm{IVIP}} \cdot \|\mathbf{B}_{M^{-1}}\|/2^{i_{end}+1} \cdot \prod_{j=0}^{i_{end}} (\mathbf{e}_1 + \mathbf{x}_j)$$

We bound the distance of $\prod_{j=0}^{i_{end}}(\mathbf{e}_1 + \mathbf{x}_j)$ from $\mathbf{e}_1$:

$$\begin{aligned}
\left\|\mathbf{e}_1 - \prod_{j=0}^{i_{end}}(\mathbf{e}_1 + \mathbf{x}_j)\right\| &\leq \sum_{\emptyset \neq S \subset [0, i_{end}]} \left\|\prod_{j \in S} \mathbf{x}_j\right\| \\
&\leq \sum_{\emptyset \neq S \subset [0, i_{end}]} \gamma_{\mathsf{Mult}}(R)^{|S|-1} \cdot (2\sqrt{n}(\log n)/s_{\mathrm{IVIP}})^{|S|} \\
&= (1/\gamma_{\mathsf{Mult}}(R)) \sum_{\emptyset \neq S \subset [0, i_{end}]} (\gamma_{\mathsf{Mult}}(R) \cdot 2\sqrt{n}(\log n)/s_{\mathrm{IVIP}})^{|S|} \\
&= (1/\gamma_{\mathsf{Mult}}(R)) \left(-1 + (1 + \gamma_{\mathsf{Mult}}(R) \cdot 2\sqrt{n}(\log n)/s_{\mathrm{IVIP}})^{i_{end}+1}\right) \\
&\leq (1/\gamma_{\mathsf{Mult}}(R))(-1 + e^{(i_{end}+1)\gamma_{\mathsf{Mult}}(R) \cdot 2\sqrt{n}(\log n)/s_{\mathrm{IVIP}}}) \\
&\leq (2/\gamma_{\mathsf{Mult}}(R)) \cdot ((i_{end}+1)\gamma_{\mathsf{Mult}}(R) \cdot 2\sqrt{n}(\log n)/s_{\mathrm{IVIP}}) \\
&\leq 8n\sqrt{n}(\log n)/s_{\mathrm{IVIP}} \\
&\leq 1/n\gamma_{\mathsf{Mult}}(R)
\end{aligned}$$

The third-from-last inequality holds for reasonable values of n. Specifically, $-1 + e^t \approx t$ for small values of t, and the exponent in the fourth-from-last expression is at most $1/2n$ (from the upper bound on i_{end} and lower bound on s_{IVIP}). For $n \geq 1/2$, t is small enough so that indeed $-1 + e^t \leq 2t$, as needed for the inequality.

So, for some $\mathbf{x}$ with $\|\mathbf{x}\| \leq 1/n\gamma_{\mathsf{Mult}}(R)$, we have

$$\mathbf{w} = s_{\mathrm{IVIP}} \cdot \|\mathbf{B}_{M^{-1}}\|/2^{i_{end}+1} \cdot (\mathbf{e}_1 + \mathbf{x})$$

Thus, by Lemma 15.1.1, $\|\mathbf{B}_{\mathbf{w}}\| \leq (s_{\mathrm{IVIP}} \cdot \|\mathbf{B}_{M^{-1}}\|/2^{i_{end}+1}) \cdot (1 + 1/n)$.

By Lemma 15.1.6, we obtain $\det((\mathbf{w})) \geq (s_{\mathrm{IVIP}} \cdot \|\mathbf{B}_{M^{-1}}\|/2^{i_{end}+1})^n/3$, from which the claim follows.

□

Chapter 20

Circuit Privacy

Recall our definition of circuit privacy (Definition 2.1.6). We say that a homomorphic encryption scheme $\mathcal{E}$ is circuit-private for circuits in $\mathcal{C}_{\mathcal{E}}$ if, for any key-pair $(\mathrm{sk}, \mathrm{pk})$ output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, any circuit $C \in \mathcal{C}_{\mathcal{E}}$, and any fixed ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$ that are in the image of $\mathsf{Encrypt}_{\mathcal{E}}$ for plaintexts $\pi_1, \ldots, \pi_t$, the following distributions (over the random coins in $\mathsf{Encrypt}_{\mathcal{E}}$, $\mathsf{Evaluate}_{\mathcal{E}}$) are (statistically) indistinguishable:

$$\mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, C(\pi_1, \ldots, \pi_t)) \quad \approx \quad \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \Psi)$$

where correctness obviously still must hold.

So far, our scheme may not be circuit private. In fact, ciphertexts output by $\mathsf{Evaluate}$ clearly come from a different distribution than those output by $\mathsf{Encrypt}$, since ciphertexts output by $\mathsf{Evaluate}$ will tend to be further away from the lattice J (since they are not as "fresh").

However, obtaining circuit privacy for our scheme is quite straightforward. Our approach is to use a public (i.e., not using the secret key) algorithm $\mathsf{RandomizeCT}_{\mathcal{E}}$ that, applied *post hoc*, *induces* the same distribution (statistically) to ciphertexts output by $\mathsf{Encrypt}_{\mathcal{E}}$ and $\mathsf{Evaluate}_{\mathcal{E}}$, while preserving correctness.

The idea is simple: to construct a *random* encryption ψ' of π from a *particular* encryption ψ of π, we simply add an encryption of 0 that has a *much* larger random "error" vector than ψ – super-polynomially larger, so that the new error vector statistically obliterates all information about ψ's error vector. However, this description is not entirely accurate, since a "proper" encryption of '0,' whether output by $\mathsf{Encrypt}_{\mathcal{E}}$ or $\mathsf{Evaluate}_{\mathcal{E}}$, is a vector lying inside

$J + \mathcal{B}(r_{\mathsf{Dec}}/m)$ – i.e., a vector whose distance from J is at most r_{Dec}/m, where m depends on which tweaks we use. On the other hand, our randomizing encryption of '0' will be much further away from J. In particular, it will be chosen from $J+\mathcal{B}(\alpha \cdot r_{\mathsf{Dec}}/m)$ where α is super-polynomial, so that the "noise" from this randomizing encryption statistically obliterates any information about the initial ciphertext's offset from J. We need $\mathcal{B}(\alpha \cdot r_{\mathsf{Dec}}/m) \subset X_{\mathsf{Dec}}$ to ensure correct decryption; so, this tweak once again entails increasing m.

Bibliography

[1] N. Ahituv, Y. Lapid, and S. Neumann. Processing Encrypted Data, In *Comm. of the ACM*, vol. 20, pages 777–780, 1987.

[2] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proc. of STOC '96*, pages 99–108, 1996.

[3] M. Ajtai. Generating hard instances of the short basis problem. In *Proc. of ICALP '99*, pages 1-9, 1999.

[4] M. Ajtai and C. Dwork. A public key cryptosystem with worst-case / average-case equivalence. In *Proc. of STOC '97*, pages 284–293, 1997.

[5] M. Ajtai, R. Kumar, and D. Sivakumar. A Sieve Algorithm for the Shortest Lattice Vector Problem. In *Proc. of STOC '01*, pages 601–610, 2001.

[6] J.H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Proc. of Eurocrypt '02*, LNCS 2332, pages 83–107. Springer, 2002.

[7] F. Armknecht and A.-R. Sadeghi. A New Approach for Algebraically Homomorphic Encryption. Cryptology ePrint Archive: Report 2008/422.

[8] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. J. of the ACM, vol. 45, no. 3, 1998, pages 501–555.

[9] L. Babai. On Lovaszs lattice reduction and the nearest lattice point problem, Combinatorica, 6 (1986), pp. 113. Preliminary version in STACS 1985.

[10] E. Bach and J. Shallit. Algorithmic Number Theory, Volume 1, 1996.

[11] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. Mathematische Annalen 296(4) (1993) 625–635.

[12] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, K. Yang. On the (Im)possibility of Obfuscating Programs. In *Proc. of Crypto '01*, LNCS 2139, pages 1–18.

[13] D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. In *Proc. of STOC '86*, pages 1–5.

[14] D. Beaver. Minimal-latency secure function evaluation. In *Proc. of Eurocrypt '00*, pages 335350. Springer, 2000.

[15] M. Bellare and A. Sahai. Non-malleable encryption. Equivalence between two notions, and an indistinguishability-based characterization. In *Proc. of Crypto '99*, LNCS 1666, pages 519–536. Springer, 1999.

[16] M. Bellare, A. Boldyreva, and S. Micali. Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In *Proc. of Eurocrypt '00*, pages 259–274. Springer, 2000.

[17] J. Benaloh. Verifiable secret-ballot elections. Ph.D. thesis, Yale Univ., Dept. of Comp. Sci., 1988.

[18] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Proc. of SAC '02*, LNCS 2595, pages 62–75. Springer, 2002.

[19] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. *Eurocrypt '98*, LNCS 1403, pp. 127–144.

[20] D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. J. ACM, vol. 48, no. 4. Pages, 702-722. ACM, 2001. Preliminary version in Crypto 1997.

[21] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. *TCC '05*, LNCS 3378, pp. 325–341.

[22] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-Secure Encryption from Decision Diffie-Hellman. In *Proc. of Crypto '08*, LNCS 5157, pages 108–125.

[23] D. Boneh and R. Lipton. Searching for Elements in Black-Box Fields and Applications. In *Proc of Crypto '96*, LNCS 1109, pages 283–297. Springer, 1996.

[24] J. Boyar, R. Peralta, and D. Pochuev. On the Multiplicative Complexity of Boolean Functions over the Basis $(\wedge, \oplus, 1)$. Theor. Comput. Sci. 235(1), pp. 43–57, 2000.

[25] E. Brickell and Y. Yacobi. On Privacy Homomorphisms. In *Proc. of Eurocrypt '87*, LNCS 304, pages 117–125. Springer, 1988.

[26] J.Y. Cai and A. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *Proc. of FOCS '97*, pages 468–477.

[27] R. Canetti. Personal communication, 2008.

[28] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proc. of STOC '98*, pages 209–218. ACM, 1998.

[29] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proc. of ACM CCS '07*.

[30] R. Canetti, H. Krawczyk, and J.B. Nielsen. Relaxing chosen-ciphertext security. In *Proc. of Crypto '03*, pages 565–582. Springer, 2003.

[31] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. TR CS0917, Dept. of Computer Science, Technion, 1997.

[32] M. Christodorescu. Private Use of Untrusted Web Servers via Opportunistic Encryption. In Web 2.0 Security and Privacy, 2008.

[33] D. Coppersmith and G. Seroussi. On the minimum distance of some quadratic residue codes. In IEEE Trans. Inform. Theory 30 (1984), 407–411.

[34] D. Coppersmith and A. Shamir. Lattice Attacks on NTRU. In *Proc. of Eurocrypt '97*, LNCS 1233, pages 52–61.

[35] R. Cramer, I. Damgaard, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of Crypto '01*, LNCS 2045, pages 280–300.

[36] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. *Crypto '98*, LNCS 1462, pp. 13–25.

[37] W. van Dam, S. Hallgren, and L. Ip. Quantum algorithms for some hidden shift problems. In *Proc. of SODA '03*, pages 489–498. Full version in SIAM J. Comput. 36(3): 763–778 (2006).

[38] I. Damgard and M. Jurik. A Length-Flexible Threshold Cryptosystem with Applications. *ACISP '03*, LNCS 2727, pages 350–356.

[39] I. Damgard and J.B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Proc. of Crypto '03*, LNCS 2729, pages 247–264. Springer, 2003.

[40] M. van Dijk Interval Obfuscation. To be published as an MIT-CSAIL Technical Report in 2009. Also, personal communication, 2009.

[41] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.* 30(2):391–437 (electronic), 2000. Preliminary version in STOC 1991.

[42] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *Crypto '84*, pp. 469–472.

[43] R. Endsuleit, W. Geiselmann, and R. Steinwandt. Attacking a polynomial-based cryptosystem: Polly Cracker. *Int. Jour. Information Security*, (1):143–148, 2002.

[44] M. Fellows and N. Koblitz. Combinatorial cryptosystems galore! In *Contemporary Mathematics*, volume 168 of *Finite Fields: Theory, Applications, and Algorithms*, FQ2, pages 51–61, 1993.

[45] M. Franklin and S. Haber. Joint encryption and message-efficient secure computation. Journal of Cryptology, 9(4):217–232, 1996.

[46] W. Geiselmann and R. Steinwandt. Cryptanalysis of Polly Cracker. *IEEE Trans. Information Theory*, (48):2990–2991, 2002.

[47] C. Gentry. Key Recovery and Message Attacks on NTRU-Composite. *Eurocrypt '01*, LNCS 2045, pp. 182–194.

[48] C. Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proc. of STOC '09*, pages 169–178.

[49] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. *STOC '08*, pp. 197–206.

[50] C. Gentry and M. Szydlo. Cryptanalysis of the Revised NTRU Signature Scheme. *Eurocrypt '02*, LNCS 2332, pp. 299–320.

[51] O. Goldreich. Modern Cryptography, Probabilistic Proofs and Pseudorandomness. 1998.

[52] O. Goldreich. Foundations of Cryptography: Basic Applications, vol. 2, Cambridge University Press, 2004.

[53] O. Goldreich, S. Goldwasser, and S. Halevi. Collision-free hashing from lattice problems. Technical Report TR96-056, Electronic Colloquium on Computational Complexity (ECCC) (1996).

[54] O. Goldreich, S. Goldwasser, and S. Halevi. Public-Key Cryptosystems from Lattice Reduction Problems. In *Proc. of Crypto '97*, LNCS 1294, pages 112-131.

[55] O. Goldreich and L. Levin. Hard-Core Predicates for Any One-Way Function. In *Proc. of STOC '89*. ACM, 1989.

[56] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game – a completeness theorem for protocols with honest majority. J. of the ACM, vol. 38, no. 1, pp. 691-729, 1991. Preliminary version in FOCS '86.

[57] O. Goldreich and R. Ostrovsky. Software protection and simulation by oblivious RAMs. JACM, 1996.

[58] S. Goldwasser. Personal communication, 2009.

[59] S. Goldwasser, Y. T. Kalai, and G. Rothblum. One-Time Programs. In *Proc. of Crypto '08*, LNCS 5157, pages 39–56. Springer, 2008.

[60] S. Goldwasser and D. Kharchenko. Proof of plaintext knowledge for the Ajtai-Dwork cryptosystem. In *Proc. of TCC 2005*, pages 529-555, 2005.

[61] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proc of STOC '82*, pages 365–377, 1982.

[62] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[63] J. Groth, R. Ostrovsky, and A. Sahai. Perfect Non-Interactive Zero Knowledge for NP. *Eurocrypt '06*, LNCS 4004, pp. 339–358.

[64] L. Gurvitz. On the Complexity of Mixed Discriminants and Related Problems.

[65] I. Haitner. Personal communication, 2008.

[66] I. Haitner and T. Holenstein. On the (im)possibility of key dependent encryption. In *Proc. of TCC '09*, LNCS 5444, pages 202–219. Springer, 2008.

[67] S. Halevi. Personal communication, 2009.

[68] S. Halevi and H. Krawczyk. Security under key-dependent inputs. In *Proc. of ACM CCS '07*, 2007.

[69] J. Hoffstein, J. Pipher and J. Silverman. NTRU: A Ring Based Public Key Cryptosystem. In *Proc. of ANTS '98*, LNCS 1423, pages 267–288.

[70] S. Hohenberger. Personal communication, 2009.

[71] S. Hohenberger, G. Rothblum, A. Shelat, V. Vaikuntanathan. Securely Obfuscating Re-encryption. In *Proc. of TCC '07*, LNCS 4392, pages 233-252.

[72] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random Generation from One-Way Functions (Extended Abstracts). In *Proc. of STOC '89*, pages 12–24.

[73] Y. Ishai and A. Paskin. Evaluating Branching Programs on Encrypted Data. In *Proc. of TCC '07*.

[74] A. Kalai. Generating Random Factored Numbers, Easily. J. Cryptology, vol. 16, no. 4, pages 287–289. 2003. Preliminary version in SODA 2002.

[75] E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. In *Proc. of STOC '95*, pages 398–406. ACM, 1995.

[76] R. Karp. A Survey of Parallel Algorithms for Shared Memory Machines.

[77] A. Kawachi, K. Tanaka, K. Xagawa. Multi-bit cryptosystems based on lattice problems. In *Proc. of PKC '07*, LNCS 4450, pages 315–329. Springer, 2007.

[78] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *Proc. of STOC '92*, pages 723–732.

[79] J. Kilian. Improved Efficient Arguments. In *Proc. of Crypto '95*, LNCS 963, pages 311–324.

[80] E. Landau. Neuer Beweis des Primzahlsatzes und Beweis des Primidealsatzes. Mathematische Annalen 56: 645-670.

[81] A.K. Lenstra, H.W. Lenstra, L. Lovsz. Factoring polynomials with rational coefficients. Math. Ann. 261(4) (1982) 515–534.

[82] F. Levy-dit-Vehel, M.G. Marinari, L. Perret, and C. Traverso. A Survey On Polly Cracker Systems.

[83] F. Levy-dit-Vehel and L. Perret. A Polly Cracker system based on satisfiability. In *Coding, Crypt. and Comb., Prog. in Comp. Sci. and App. Logic*, v. 23, pp. 177–192.

[84] H. Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In *Proc. of ICS '05* pages 314-328, 2005.

[85] L. Ly. A public-key cryptosystem based on Polly Cracker, Ph.D. thesis, Ruhr-Universität Bochum, Bochum, Germany 2002.

[86] L. Ly. Polly two – a new algebraic polynomial-based public-key scheme. AAECC, 17(3-4), 2006.

[87] V. Lyubashevky. Personal communication, 2009.

[88] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *Proc. of ICALP '06*. Springer, 2006.

[89] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. Full version.

[90] V. Lyubashevky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In *Proc. of TCC '08*.

[91] T. Matsumoto, K. Kato, and H. Imai. Speeding up secret computations with insecure auxiliary devices. In *Proc. of Crypto '88*, LNCS 403, pages 497-506. Springer, 1988.

[92] U. Maurer and D. Raub. Black-Box Extension Fields and the Inexistence of Field-Homomorphic One-Way Permutations. *Asiacrypt '07*, pp. 427–443.

[93] A. May, Cryptanalysis of NTRU-107, manuscript, 1999. Available from `http://www.informatik.uni- frankfurt.de/~alex/crypto.html`.

[94] C.A. Melchor, G. Castagnos, and P. Gaborit. Lattice-based homomorphic encryption of vector spaces. ISIT '08, pp. 1858–1862.

[95] C.A. Melchor, P. Gaborit, and J. Herranz. Additive Homomorphic Encryption with t-Operand Multiplications. Eprint 2008/378.

[96] J. Merkle. Multi-round passive attacks on server-aided RSA protocols. In *Proc. of ACM CCS '00*, pages 102-107. ACM, 2000.

[97] D. Micciancio. Improving Lattice Based Cryptosystems Using the Hermite Normal Form. In *Proc. of CaLC '01*, LNCS 2146, pages 126–145. Springer, 2001.

[98] D. Micciancio. Improved cryptographic hash functions with worst-case / average-case connection. In *Proc. of STOC '02*, pages 609–618.

[99] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *Proc. of FOCS '02*, pages 356–365.

[100] D. Micciancio and O. Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *FOCS '04*, pp. 372–381.

[101] K. Mulmuley and M. Sohoni. Geometric complexity theory I: An approach to the P vs. NP and related problems. SIAM J. Comput., 31(2):496–526, 2002.

[102] D. Naccache and J. Stern. A New Public-Key Cryptosystem Based on Higher Residues. *ACM CCS '98*.

[103] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. of STOC '01*, pages 590–599, 2001.

[104] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proc. of STOC '90*, pages 427–437. ACM, 1990.

[105] P.Q. Nguyen and I. Shparlinski. On the Insecurity of Some Server-Aided RSA Protocol. *Asiacrypt '01*, LNCS 2248, pp. 21–35.

[106] P.Q. Nguyen and J. Stern. The BeguinQuisquater server-aided RSA protocol from Crypto '95 is not secure. In *Proc. of Asiacrypt '98*, pages 372–379. Springer, 1998.

[107] A.M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Crypt. and Comp. Num. Th.*, Proc. Sympos. Appl. Math., vol. 42, AMS, 1990, pp. 75–88.

[108] T. Okamoto and Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. *Eurocrypt '98*, LNCS 1403, pp. 308–318.

[109] R. Ostrovsky and W. E. Skeith. Private Searching on Streaming Data. In *Proc. of Crypto '05*, LNCS 3621, pp. 223–240.

[110] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *Eurocrypt '99*, pp. 223–238.

[111] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Proc. of TCC '06*, pages 145166.

[112] C. Peikert and A. Rosen. Lattices that Admit Logarithmic Worst-Case to Average-Case Connection Factors. In *Proc. of STOC '07*, pages 478–487.

[113] C. Peikert and B. Waters. Lossy Trapdoor Functions and Their Applications. *STOC '08*, pp. 187–196.

[114] B. Pfitzmann and M. Waidner. Attacks on protocols for server-aided RSA computation. In *Proc. of Eurocrypt '92*, LNCS 658, pages 153-162. Springer, 1993.

[115] M. Prabhakaran and M. Rosulek. Homomorphic Encryption with CCA Security. In *Proc. of ICALP '08*. Springer, 2008.

[116] C. Rackoff and D.R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proc. of Crypto '91*, LNCS 576, pages 433-444. Springer, 1991.

[117] K. W. Regan. Understanding the Mulmuley-Sohoni Approach to P vs. NP.

[118] O. Regev. New lattice-based cryptographic constructions. Journal of the ACM 51(6) (2004) 899942. Extended abstract in STOC '03.

[119] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proc. of STOC '05*, pages 84–93, 2005.

[120] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180, 1978.

[121] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Comm. of the ACM*, 21:2, pages 120–126, 1978.

[122] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC1. In *Proc. of FOCS '99*, pages 554–567, 1999.

[123] C.P. Schnorr. A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms. *Theoretical Computer Science*, 53(2-3):201–224, 1987.

[124] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5): 1484–1509, 1997. Extended abstract in FOCS '94.

[125] D.X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[126] R. Steinwandt. A ciphertext-only attack on Polly Two, 2006.

[127] P. Stevenhagen. The Arithmetic of Number Rings. Algorithmic Number Theory, MSRI Publications, Volume 44, 2008. See also Stevenhagen's course notes "Number Rings."

[128] D.R. Stinson. Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. Mathematics of Computation, vol. 71, no. 237, pages 379–391, 2001.

[129] A.C. Yao. Protocols for secure computations (extended abstract). FOCS '82, pages 80–91.

[130] A. C. Yao. How to generate and exchange secrets. FOCS '86, pages 162-167.