

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

UMI Number: 3382729

Copyright 2009 by
Gentry, Craig

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 3382729

Copyright 2009 by ProQuest LLC


All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

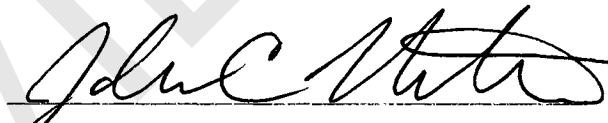
PREVIEW

© Copyright by Craig Gentry 2009
All Rights Reserved

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

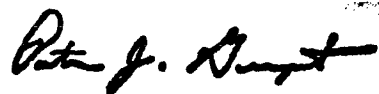
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.


(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 an old open problem. Such a scheme allows one to compute arbitrary functions over encrypted data without the decryption key – i.e., given encryptions $E(m_1), \dots, E(m_t)$ of m_1, \dots, m_t , one can efficiently compute a compact ciphertext that encrypts $f(m_1, \dots, m_t)$ for any efficiently computable function f .

Fully homomorphic encryption has numerous applications. For example, it enables encrypted search engine queries – i.e., a search engine can give you a succinct encrypted answer to your (boolean) query without even knowing what your query was. It also enables searching on encrypted data; you can store your encrypted data on a remote server, and 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, it improves the efficiency of secure multiparty computation.

In our solution, we begin by designing a somewhat homomorphic “bootstrappable” 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.

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

Abstract	iv
Acknowledgments	v
1 Introduction	1
1.1 A Very Brief and Informal Overview of Our Construction	2
1.2 What is Fully Homomorphic Encryption?	5
1.3 Bootstrapping a Scheme that Can Evaluate its Own Decryption Circuit . .	7
1.4 Ideal Lattices: Ideally Suited to Construct Bootstrappable Encryption . . .	10
1.5 Squashing the Decryption Circuit: The Encrypter Starts Decryption!	15
1.6 Security	18
1.7 Performance	20
1.8 Applications	21
2 Definitions related to Homomorphic Encryption	27
2.1 Basic Definitions	27
2.2 Computational Security Definitions	31
3 Previous Homomorphic Encryption Schemes	34
4 Bootstrappable Encryption	43
4.1 Leveled Fully Homomorphic Encryption from Bootstrappable Encryption, Generically	43
4.2 Correctness, Computational Complexity and Security of the Generic Con- struction	48
4.3 Fully Homomorphic Encryption from KDM-Secure Bootstrappable Encryption	51

4.4	Fully Homomorphic Encryption from Bootstrappable Encryption in the Random Oracle Model	53
5	An Abstract Scheme Based on the Ideal Coset Problem	57
5.1	The Ideal Coset Problem	58
5.2	An Abstract Scheme	59
5.3	Security of the Abstract Scheme	62
6	Background on Ideal Lattices I: The Basics	63
6.1	Basic Background on Lattices	63
6.2	Basic Background on Ideal Lattices	65
6.3	Probability Background	68
7	A Somewhat Homomorphic Encryption Scheme	69
7.1	Why Lattices?	69
7.2	Why <i>Ideal</i> Lattices?	70
7.3	A Geometric Approach to Maximizing the Circuit Depth that Can Be Evaluated	70
7.4	Instantiating the Ring: The Geometry of Polynomial Rings	72
7.5	Instantiating Encrypt and Minimizing r_{Enc}	75
7.6	Instantiating Decrypt and Maximizing r_{Dec}	75
7.7	Security of the Concrete Scheme	77
7.8	How Useful is the Somewhat Homomorphic Scheme By Itself?	79
8	Tweaks to the Somewhat Homomorphic Scheme	81
8.1	On the Relationship between the Dual and the Inverse of an Ideal Lattice .	82
8.2	Transference Lemmas for Ideal Lattices	85
8.3	Tweaking the Decryption Equation	86
8.4	A Tweak to Reduce the Circuit Complexity of the Rounding Step in Decryption	88
9	Decryption Complexity of the Tweaked Scheme	90
10	Squashing the Decryption Circuit	98
10.1	A Generic Description of the Transformation	98
10.2	How to Squash, Concretely	100
10.3	Bootstrapping Achieved: The Decryption Circuit for the Transformed System	102

11 Security	104
11.1 Regarding the Hint Given in Our “Squashing” Transformation	104
11.2 Counterbalancing Assumptions	113
12 Performance and Optimizations	115
12.1 Simple Optimizations	116
12.2 Basic Performance	117
12.3 More Optimizations	117
13 Background on Ideal Lattices II	125
13.1 Overview of Gaussian Distributions over Lattices	125
13.2 The Smoothing Parameter	126
13.3 Sampling a Lattice According to a Gaussian Distribution	128
13.4 Ideal Factorization in Polynomial Rings	129
14 The Somewhat Homomorphic Scheme Revisited	132
14.1 Using Gaussian Sampling in Encrypt	132
14.2 Generating an Ideal with Very Small Norm	133
14.3 Proof of Security Based on the Inner Ideal Membership Problem (IIMP) . .	135
14.4 Success Amplification: Proof of Security Based on the Modified IIMP (MIIMP)	136
14.5 Basing Security on a Search Problem: Bounded Distance Decoding Via Hensel Lifting	138
14.6 Toward Reducing the SIVP to the BDDP: Regev’s Quantum Reduction . .	141
14.7 Summary of Security Results for this Construction So Far	143
14.8 Looking Forward	143
15 Background on Ideal Lattices III	145
15.1 Lemmata Regarding Vectors Nearly Parallel to \mathbf{e}_1	145
15.2 Distribution of Prime Ideals	148
16 Random Self-Reduction of Ideal Lattice Problems	151
16.1 A New Type of Worst-Case / Average-Case Connection for Lattices	151
16.2 Our Average-Case Distribution	153
16.3 How to “Randomize” a Worst-Case Ideal	154
16.4 Why Does the Reduction Require a Factoring Oracle?	157

16.5 Application to our Fully Homomorphic Encryption Scheme	159
17 How to Randomize a Worst-Case Ideal	161
17.1 The RandomizeIdeal Algorithm	161
17.2 Is the Ideal Random? The Proof of Theorem 16.3.4	162
17.3 Reduction of WBDDP to HBDDP and Worst-case IVIP to Average-Case IVIP	164
17.4 An Alternative Way to Randomize an Ideal	166
18 KeyGen per the Average Case Distribution	175
18.1 The Secret Key	175
18.2 Adapting Kalai's Algorithm to Generate a Random Factored Ideal	177
19 Basing Security on Worst-case SIVP in Ideal Lattices	181
19.1 Relationship Among Instances of IVIP	182
19.2 Reduction of SIVP to IVIP	183
20 Circuit Privacy	188
Bibliography	190

List of Tables

PREVIEW

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 Dertouzos [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 $\text{Evaluate}_{\mathcal{E}}$ that, for any valid public key pk , *any* circuit C (not just a circuit consisting of multiplication gates as in RSA), and any ciphertexts $\psi_i \leftarrow \text{Encrypt}_{\mathcal{E}}(\text{pk}, \pi_i)$, outputs

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

a valid encryption of $C(\pi_1, \dots, \pi_t)$ 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 $\text{Evaluate}_{\mathcal{E}}$ 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 $b \leftarrow (c \bmod p) \bmod 2$, 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 π_1, \dots, π_t , fully homomorphic encryption should allow anyone (not just the key-holder) to output a ciphertext that encrypts $f(\pi_1, \dots, \pi_t)$ for any desired function f , as long as that function can be efficiently computed. No information about π_1, \dots, π_t or $f(\pi_1, \dots, \pi_t)$, 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” $f(\pi_1, \dots, \pi_t)$. The minimal requirement is correctness. A fully homomorphic encryption scheme \mathcal{E} should have an efficient algorithm $\text{Evaluate}_{\mathcal{E}}$ that, for any valid \mathcal{E} key pair (sk, pk) , any circuit C , and any ciphertexts $\psi_i \leftarrow \text{Encrypt}_{\mathcal{E}}(\text{pk}, \pi_i)$, outputs

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

This minimal requirement does not seem to be sufficient, however, since it permits the trivial solution, where ψ simply consists of $(C, \psi_1, \dots, \psi_t)$ – i.e., where the $\text{Evaluate}_{\mathcal{E}}$ 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 $\text{Evaluate}_{\mathcal{E}}$ 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, $\text{Evaluate}(\text{pk}, C, \psi_1, \dots, \psi_t)$ constructs a Yao garbling C^\dagger of C , uses the OT queries ψ_1, \dots, ψ_t to construct OT responses $\psi_1^*, \dots, \psi_t^*$ designed to obliviously transfer Yao keys associated to the t input wires in C^\dagger , and outputs $(C^\dagger, \psi_1^*, \dots, \psi_t^*)$. To decrypt this ciphertext, the key holder “decrypts” the OT responses $\psi_1^*, \dots, \psi_t^*$ 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 $C(\pi_1, \dots, \pi_t)$ should “look like” an “ordinary” ciphertext, as long as $C(\pi_1, \dots, \pi_t)$ is a single bit (or element of the same plaintext space that contains $\{\pi_i\}$). For example, the size of the ciphertext output by $\text{Evaluate}(\text{pk}, C, \psi_1, \dots, \psi_t)$ should not depend on C . We focus on this definition. Actually, we use a stronger requirement: that $\text{Decrypt}_{\mathcal{E}}$ 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.,

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

is guaranteed to hold only if $C \in \mathcal{C}_{\mathcal{E}}$ (and, as before, $\psi_i \leftarrow \text{Encrypt}_{\mathcal{E}}(\text{pk}, \pi_i)$). 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} is *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 $\text{Evaluate}_{\mathcal{E}^{(d)}}$ 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 $\text{Evaluate}_{\mathcal{E}^{(d)}}$.*

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

¹We 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 $\text{Evaluate}_{\mathcal{E}}(\text{pk}, C, \psi_1, \dots, \psi_t)$ and $\text{Encrypt}_{\mathcal{E}}(\text{pk}, C(\pi_1, \dots, \pi_t))$ are different. In particular, suppose that there is an “error” associated with each ciphertext, that ciphertexts output by $\text{Encrypt}_{\mathcal{E}}$ have small error, that ciphertexts output by $\text{Evaluate}_{\mathcal{E}}$ 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 $\text{Decrypt}_{\mathcal{E}}$ 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 $\overline{\text{sk}_{1j}}$ be the encryption of the j th bit of sk_1 . Consider the following algorithm.

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

Set $\overline{\psi_{1j}} \xleftarrow{R} \text{Encrypt}_{\mathcal{E}}(\text{pk}_2, \psi_{1j})$

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

Above, 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 $\text{Decrypt}_{\mathcal{E}}(\text{sk}_1, \psi_1) = \pi$.² In other words, Recrypt decrypts homomorphically using the encrypted secret key, thus obtaining a new ciphertext that encrypts the same thing as the original one.

² 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 $\text{Evaluate}_{\mathcal{E}}$ to remove the *inner* encryption. Applying the decryption circuit $D_{\mathcal{E}}$ removes the “error” associated to the first ciphertext under pk_1 , but $\text{Evaluate}_{\mathcal{E}}$ 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 $\langle \overline{\text{sk}_{1j}} \rangle$ 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 $\text{pk}_1 = \text{pk}_2$. 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 **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 $(\pi_1 \text{ NAND } \pi_2)$ under pk_2 using the encrypted secret key $(\text{sk}_1 \text{ under } \text{pk}_2)$ 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 ‘×’, 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 $i_1, \dots, i_t \in I$ and $r_1, \dots, r_t \in R$. 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 ‘×’:

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

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

$$\text{Evaluate}_{\mathcal{E}}(\text{pk}, C, \psi_1, \dots, \psi_t) \in C(\pi_1, \dots, \pi_t) + I$$

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 $\mathbf{v} \bmod \mathbf{B}_I = \mathbf{v} - \mathbf{B}_I \cdot \lfloor \mathbf{B}_I^{-1} \cdot \mathbf{v} \rfloor$, where \mathbf{B}_I^{-1} is the matrix inverse of \mathbf{B}_I and $\lfloor \cdot \rfloor$ 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 $\mathbf{r} \bmod \mathbf{B}_I = \mathbf{0} \bmod \mathbf{B}_I$. 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^{sk} and \mathbf{B}_J^{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 $\pi^* \xleftarrow{D} \pi + I$, and sends $\psi \leftarrow \pi^* \bmod \mathbf{B}_J^{\text{pk}}$. 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

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

For decryption to work, the secret key \mathbf{B}_J^{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^{sk} . In this case, the $\text{mod-}\mathbf{B}_J^{\text{sk}}$ 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:

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

where $i_1, \dots, i_t \in I$. (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 ‘ \times ’, but we will use this for now.) In this case, for decryption to work, we need $C(\pi_1 + i_1, \dots, \pi_t + i_t)$ to be the distinguished representative of $C(\pi_1 + i_1, \dots, \pi_t + i_t) + J$ w.r.t. \mathbf{B}_J^{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 $C(\pi_1 + i_1, \dots, \pi_t + i_t)$ is always the distinguished representative of $C(\pi_1 + i_1, \dots, \pi_t + i_t) + J$ w.r.t. \mathbf{B}_J^{sk} when \mathbf{B}_J^{sk} is generated according to $\text{KeyGen}_{\mathcal{E}}$ and π_k and i_k are chosen according to $\text{Encrypt}_{\mathcal{E}}$. In this case, the $\text{mod-}\mathbf{B}_J^{\text{sk}}$ operation recovers $C(\pi_1 + i_1, \dots, \pi_t + i_t)$, after which the decrypter easily recovers $C(\pi_1, \dots, \pi_t)$ 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 $C(\pi_1 + i_1, \dots, \pi_t + i_t)$ 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^{sk} are precisely the points in \mathbb{Z}^n that are contained inside the parallelepiped $\mathcal{P}(\mathbf{B}_J^{\text{sk}}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \sum x_i \cdot \mathbf{b}_i, x_i \in [-1/2, 1/2)\}$ associated to the basis $\mathbf{B}_J^{\text{sk}} = \{\mathbf{b}_i\}$. Decryption works as long as the “error vector” is never so long that it falls outside of $\mathcal{P}(\mathbf{B}_J^{\text{sk}})$.³ Once we specify some radius r_{Dec} such that the parallelepiped $\mathcal{P}(\mathbf{B}_J^{\text{sk}})$ 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 $\text{Encrypt}_{\mathcal{E}}$) the vector $\pi^* \stackrel{D}{\leftarrow} \pi + I$ always falls within a ball of radius r_{Enc} , the bootstrappability question becomes: is $C(\mathbf{x}_1, \dots, \mathbf{x}_t) \in \mathcal{B}(r_{\text{Dec}})$ whenever $\mathbf{x}_i \in \mathcal{B}(r_{\text{Enc}})$ for all i and C is an (augmented)

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