Fully Homomorphic Encryption

Mitchell Harper

June 2, 2014

Contents

1 Introduction

Since the creation of public key cryptography, researchers have been searching for an encryption scheme that can allow for arbitrary operations to be performed on a ciphertext while preserving its integrity. This would be widely useful for consumers, businesses, and governments as it would allow for processing of sensitive data to be deferred to other clients or the cloud.

In 2009, the first fully homomorphic encryption was discovered, and since then a variety of optimizations and variations have been published. In this paper, we define the properties of a fully homomorphic scheme and examine a specific scheme over the integers.

2 Cryptography Primer

2.1 Definitions

We begin by defining some terms and concepts in cryptography.

Definition 2.1. We say a function $f(n)$ is $\omega(g(n))$ if f is asymptotically bounded below by a constant multiple of g, or more formally, for all $k > 0$ there exists an n_0 such that for all $n > n_0$, $|f(n)| \geq k \cdot |g(n)|$.

Definition 2.2. We say a function $f(n)$ is $\Theta(g(n))$ if f is asymptotically bounded above *and* below by a constant multiple of g, or more formally, there exists a $k_1 > 0$, $k_2 > 0$, and there exists an n_0 such that for all $n > n_0$, $g(n) \cdot k_1 \leq f(n) \leq g(n) \cdot k_2$.

Definition 2.3. We say an algorithm runs in **polynomial time** if the number of operations needed to complete the algorithm is $O(n^k)$ for some nonnegative integer k. We call n the complexity of the input to the algorithm. In this context, it is often the size of the input. [6]

Definition 2.4. A **plaintext** is a message we wish to encrypt. A **ciphertext** is an encrypted version of an associated plaintext.

We will define a public-key encryption scheme as done in Cramer section 3 [2].

Definition 2.5. A **public-key encryption scheme** is comprised of three algorithms:

- A probabilistic, polynomial-time *key generation algorithm* KeyGen that takes a *security paramater* λ (usually the size of the input) and outputs a public key/secret key pair $(\mathsf{PK}, \mathsf{SK})$.
- Another probabilistic, polynomial-time *encryption algorithm* Encrypt that takes λ, a public key PK, and a plaintext π as input and outputs a ciphertext ψ.
- A deterministic, polynomial-time *encryption algorithm* Decrypt that takes λ, a secret key SK, and a ciphertext ψ and outputs either a plaintext π or the special symbol reject.

2.2 Using a Public-key Scheme

To put these concepts in more accessible terms, we will make an analogy. An apt choice is that of a protected dropbox. When creating such a dropbox using an analog of the defined scheme, a pair of keys is created. The public key can be reproduced as many times as desired, and it allows for anyone to drop a message into the message slot of the dropbox. Using the public key to open the message slot, one can drop a message in the box. Inside the box, we say the message is encrypted. Noably, given a properly designed dropbox, one cannot recover the message using the public key alone. Instead, only the holder of the secret key can open up another door at the back of the dropbox and retrieve messages (the decryption step).

The most common use of public-key encryption alows two parties to communicate securely. In our analogy, both parties build dropboxes and the associated keys, and then each exchanges their public key with the other. This allows for two-way communication where anyone can see the contents of messages, but only the intended recipient can decode and read an encrypted message.

3 Homomorphic Encryption Schemes

In this section, we introduce what it means for a scheme to be homomorphic and give the background necessary for the construction of a specific scheme over the integers.

3.1 An Intuitive Definition

Following the discussion in Gentry, section 1.2 [4], we start with an intuitive definition of a homomorphic scheme: given ciphertexts $\psi_1, \psi_2, \ldots, \psi_n$ that encrypt $\pi_1, \pi_2, \ldots, \pi_n$, a fully homomorphic encryption scheme should allow anyone with a public key to compute a function over those ciphertexts and output a new ciphertext that encrypts $f(\pi_1, \ldots, \pi_n)$ for any arbitrary function f that is efficiently computable. Moreover, no information about the plaintexts leaks and all inputs and outputs are encrypted throughout the process of the evaluation of the function.

This simple definition does not give a satisfactory definition of a *fully* homomorphic scheme, however we immediately see how such a scheme could be useful. For example, let's assume I store a number of encrypted documents in the cloud. I would like to search for all occurrences of the word "quarterly" in my document store, however it would be cumbersome for me to download all documents, decrypt them, and perform the search. At the same time, I do not fully trust my hosting company and I wish to perform the search on their servers while keeping all documents encrypted. Using a homomorphic scheme, I could define a function f to perform the search, encrypt the query that that function takes using the same key that the documents were encrypted with, and then run the function in the context of the scheme and output a result ciphertext that only I can decrypt that contains the locations of the word in the file.

Beyond this trivial example, there are a number of services that distribute computation across the computers of thousands of volunteers willing to give up spare CPU cycles, including SETI@Home. This program allows one to analyze data from radio telescopes in the search for extraterrestrial life. A homomorphic encryption scheme would allow for this computation to be performed with a higher degree of confidence that the results haven't been tampered with.

3.2 Refining the Definition

The previous definition offers a trivial solution, that is to simply output $\psi = (f, \psi_1, \ldots, \psi_n)$ which merely attaches a description of the function to the original inputs and doesn't actually *process* anything. This is valid in that only the secret key-holder can decrypt the output, however it is exactly the solution we didn't want to deal with in the file searching example.

To exclude this solution, we can require *cicruit privacy*, or that the output reveals nothing about the given function. This is achievable easily by using a construction known as *garbled circuits* that's introduced in Gentry 1.2 [4]. This is not yet satisfactory in that as we compose functions, the length of the output ciphertexts will grow in the number of operations performed.

Then a more useful way to exclude the trivial solution requires that the output of a function under a given homomorphic encryption scheme is independent of the given function. We will actually take a stronger definition for the purpose of this paper: that the Decrypt algorithm can be expressed as a circuit of addition and multiplication gates whose size is a *fixed* polynomial in the security paramater λ.

3.3 Another Analogy

We will recap the jewelry store analogy from Gentry 1.2 [4]. This is similar to the scheme we will develop in the next section. The story is as follows: suppose the owner of a jewelry store, let's call her Alice, wants to use a workforce of employees to assemble raw materials into rings. However, the diamonds and gold are coveted, and Alice wants to prevent her untrustworthy employees from stealing items along the way. Therefore, she hires a contractor to analyze the solution, and they give her the following proposal: allow the workers to handle the jewelry using a locked box which offers a set of gloves that can be used to work with raw materials on the inside (much like what you may have seen in a biology lab).

With this plan, Alice comissions the development of a number of prototype boxes and tests them on the production line. An important feature of the box is that the gloves are strong enough to prevent puncturing and actually keep the raw materials secure. However, she hears complaints from her workers that the gloves become too stiff to work with after a given period of time, and that a ring cannot be completed before the gloves stiffen. Every prototype is succeptible. Moving the unfinished product from one box to another would compromise security.

However, Alice comes up with a brilliant solution. After one box has failed (call it a), it can be placed in another similar box b that also contains the key to a. Then, the worker can use b's gloves, open box a using the key inside, remove the unfinished product, and manipulate it until b's gloves stiffen. This scheme is not guaranteed to work though, as when a box is composed, a worker must be able to open box a, remove the materials, and

perform a non-negligible amount of work before b's gloves stiffen. To Alice's delight, though, the process works and she successfully prevents loss of raw materials, and at the end of the circuit she can take her key, open the last box and obtain the finished product.

One flaw in the analogy is that after a number of steps, there will be many leftover boxes, so let's assume a worker can "will-away" any item in any box as to prevent such clutter. With that taken care of, is there an actual analog that we can apply to the cryptographic domain?

4 Formal Definitions in Homomorphic Encryption

We are now ready to formally define concepts in homomorphic encryption. We will be adapting from the definitions in Van Dijk, et al. [3]

4.1 Notation

We denote paramaters to algorithms using Greek letters with λ always denoting the security paramater. Real numbers and integers will be represented by lowercase English letters. All logarithms are in base-2.

There are a number of rounding operations we will use, namely:

- $\lceil z \rceil$ denotes the rounding up of z, or the unique integer in the half-open interval $[z, z+1)$
- $\lfloor z \rfloor$ represents the down of z, or the unique integer in the interval $(z-1, z]$
- $\lfloor z \rceil$ represents the rounding of z to the nearest integer, or the unique integer in the interval $(z - \frac{1}{2}, z + \frac{1}{2}]$

We will also define the quotient and remainder operations in this context. We say $q_p(z)$ is the quotient of z with respect to z, or $q_p(z) = \lfloor z/p \rceil$. We also say $r_p(z)$ is the remainder of z with respect to p, or $r_p(z) = z - q_p(z) \cdot p$. Note that with our definitions $r_p(z) \in (-p/2, p/2]$. Interchangeably, we denote the remainder as both $[z]_p$ and $(z \mod p)$.

4.2 Classifying Homomorphic Encryption Schemes

As in Van Dijk 2.1 [3], we will examine encryption schemes that are homomorphic with respect to boolean circuits containing addition and multiplication gates only. Note that such a circuit takes a number of bits as input.

Like a regular public-key encryption scheme $\mathcal{E}$, a homomorphic public-key encryption scheme contains the usual algorithms KeyGen, $\mathsf{Encrypt}$, and $\mathsf{Decrpyt}$. We will define such a scheme to also contain the algorithm $\mathsf{Evaluate}$, which takes a public key PK, a boolean circuit C, and a tuple of ciphertexts $\overline{c} = \langle c_1, \cdots, c_t \rangle$ (one for every input bit of c) and produces a new ciphertext c.

Definition 4.1. The scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ is **correct** for a given t-input circuit C if, for any key-pair $(\mathsf{PK}, \mathsf{SK})$ output by $\mathsf{KeyGen}(\lambda)$, any t plaintext bits $\pi_1, \ldots, \pi_t$, and any ciphertexts $\overline{c} = \langle c_1, \ldots, c_t \rangle$ with $c_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{PK}, \pi_i)$, the equation

$$\mathsf{Decrypt}(\mathsf{SK}, \mathsf{Evaluate}(\mathsf{PK}, C, \overline{c})) = C(\pi_1, \ldots, \pi_t)$$

holds.

More informally, a scheme is correct for some functions if the result after decrypting an evaluation of a given function on a ciphertext is result from running that circuit on the original plaintexts.

Definition 4.2. The scheme $\mathcal{E}$ is **homomorphic** for a class $\mathcal{C}$ of circuits if it is correct for all circuits $C \in \mathcal{C}$.

Definition 4.3. $\mathcal{E}$ is **fully homomorphic** if it is correct for all possible boolean circuits.

As stated previously, we need to rule out the trivial solution.

Definition 4.4. The scheme $\mathcal{E}$ is **compact** if there exists a fixed polynomial bound $b(\lambda)$ so that for any pair $(\mathsf{SK}, \mathsf{PK})$ output by $\mathsf{KeyGen}(\lambda)$, any circuit C, and any sequence of ciphertexts $\overline{c} = \langle c_1, \ldots, c_t \rangle$ that was generated with respect to PK, the size of the ciphertext $\mathsf{Evaluate}(\mathsf{PK}, C, \overline{c})$ is no larger than $b(\lambda)$ bits (independently of the size of C).

4.3 Bootstrapping

We will now discuss the concept of bootstrapping, which is necessary in all known fully homomorphic algorithms as of the writing of this paper. The need for such a bootstrapping process was modeled in the jewelry analogy by the stiffening of the gloves. In all known encryption schemes, naively applying the algorithm over polynomials of a high enough degree cause irreversible loss of information about the data being processed. See Gentry, section 3 [4] for further discussion.

Definition 4.5. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ be an encryption scheme, where decryption is implemented by a fixed circuit that depends only on the security paramater λ.

For a given λ, the set of **augmented decryption circuits** consists of exactly two circuits. Both take as input a secret key and two ciphertexts. The first circuit decrypts both ciphertexts and *adds* the resulting plaintext bits mod 2, and the other decrypts both ciphertexts and *multiplies* the resulting plaintext bits mod 2. This set is denoted by $D_{\mathcal{E}}(\lambda)$.

We now come to the critical definition of this paper.

Definition 4.6. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ be a homomorphic encryption scheme (not necessarily fully homomorphic), and for every value of the security paramater λ let $\mathcal{C}_{\mathcal{E}}(\lambda)$ be a set of circuits with respect to λ to which $\mathcal{E}$ is correct. We say that $\mathcal{E}$ is **bootstrappable** if $D_{\mathcal{E}}(\lambda) \subseteq \mathcal{C}_{\mathcal{E}}(\lambda)$ holds for every λ.

In more general terms, a scheme is bootstrappable if its augmented decruption circuits can be correctly evaluated by the scheme.

With these definitions in hand, we can state the following theorem (see Gentry [4] for the proof).

Theorem 4.1. *There is an (efficient, explicit) transformation that, given a description of a bootstrappable scheme $\mathcal{E}$ and a paramater $d = d(\lambda)$, outputs a description of a another scheme $\mathcal{E}^{(d)}$ such that:*

1. *$\mathcal{E}^{(d)}$ is compact (in particular the* Decrypt *circuit in $\mathcal{E}^{(d)}$ is identical to that in $\mathcal{E}$), and*
2. *$\mathcal{E}^{(d)}$ is homomorphic for all circuits of depth up to d.*

5 A Fully Homomorphic Encryption Scheme over the Integers

5.1 A Well-Posed Problem?

We are about to develop a scheme over the integers that satisfies our definitions of fully homomorphic, but is this the solution to a well-posed problem? We see that there exists a solution to the question: *Is there an encryption scheme over the integers that is fully homomorphic?*. However, this solution is not unique, as variations have already been published (see Coron et al. [1]). Then the above question does not satisfy our definition of a well-posed problem.

However, this does not mean our solution is without merit. As we are developing an algorithm, it is often the case that there are multiple ways to solve a problem. It is often beneficial to have multiple algorithms as they will be applied in varying contexts, with some algorithms being more suited to a given environment than others. We have also constrained our definition of fully homomorphic such that any solution will be interesting.

5.2 A Somewhat Homomorphic Encryption Scheme

We are now ready to develop the fully homomorphic scheme using integer arithmetic as constructed in Van Dijk [3].

5.2.1 Definitions

First, let us define the following paramaters

- γ is the bit-length of the integers in the public key (we shall see in a moment the public key consists of a number of chosen integers),
- η is the bit-length of the secret key,
- ρ is the bit-length of the "noise" (the distance between the public key elements and the nearest multiples of the secret key), and
- τ is the number of integers in the public key.

We must set $\eta \geq \rho \cdot \Theta(\lambda \log^2 \lambda)$ in order to support homomorphism over circuits deep enough to evaluate the "squashed decryption circuit" (to be explained).

We also define another noise paramater: $\rho' = \rho + \omega(\log \lambda)$.

We define, for as specific (η-bit) odd positive integer p, the following distribution over γ-bit integers:

$$\mathcal{D}_{\gamma,\rho}(p) = \{\mathsf{sample}\ q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p), \mathsf{sample}\ r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \mathsf{output}\ x = pq + r\}$$

5.2.2 The Scheme

- $\mathsf{KeyGen}(\lambda)$: The secret key is an odd η bit integer: sample $p \leftarrow (2\mathbb{Z}+1 \cap [2^{\eta-1}, 2^\eta)$. To build the public key, we start by samplining $x_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$ for $i = 0, \ldots, \tau$. Then, relabel so that x_0 is the largest of all values sampled. Restart unless x_0 is odd and $r_p(x_0)$ is even. The public key is then $\mathsf{PK} = \langle x_0, x_1, \ldots, x_\tau \rangle$.

- $\mathsf{Encrypt}(\mathsf{PK}, \pi \in \{0,1\})$: Choose a random subset $S \subseteq \{1, 2, \ldots, \tau\}$ and a random integer r in $(-2^{\rho'}, 2^{\rho'})$, and output $c \leftarrow \left[m + 2r + \sum_{i \in S} x_i\right]_{x_0}$

- $\mathsf{Evaluate}(\mathsf{PK}, C, c_1, \ldots, c_t)$. Given the (binary) circuit $\mathcal{C}_\mathcal{E}$ with t inputs, and t ciphertexts c_i, apply the (integer) addition and multiplication gates of $\mathcal{C}_\mathcal{E}$ to the ciphertexts, performing all the operations over the integers, and return the resulting integer.

- $\mathsf{Decrypt}(\mathsf{SK}, c)$: Output $\pi' \leftarrow (c \mod p) \mod 2$.

Note that $c \mod p = c - p \cdot \lfloor c/p \rceil$, and as p is odd we can instead decrypt using $\pi' \leftarrow [c - \lfloor c/p \rceil]_2 = (c \mod 2) \oplus (\lfloor c/p \rceil \mod 2)$.

5.3 Correctness

Definition 5.1. ([3]) We define a **permitted circuit** as one where for any $\alpha \geq 1$ and any set of integer inputs all less than $2^{\alpha(\rho'+2)}$ in abolute value, it holds that the generalized circuit's output has absolute value at most $2^{\alpha(\eta-4)}$

Lemma 5.1. *Let $\mathcal{C}_\mathcal{E}$ denote the be the set of permitted circuits. The scheme from above is correct for $\mathcal{C}_\mathcal{E}$.*

Proof. Consider a "fresh" ciphertext output by $\mathsf{Encrypt}$. We shall start by proving the following lemma saying that each ciphertext is near a multiple of p, and the difference from the closest multiple has the same parity as π.

Lemma 5.0.1. *Let* $(\mathsf{SK}, \mathsf{PK})$ *be output by* $\mathsf{KeyGen}(\lambda)$*. Let* $c \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \pi)$ *for* $\pi \in \{0,1\}$*. Then,* $c = a \cdot p + (2b + \pi)$ *for some integers* a *and* b *with* $|2b + m| \leq \tau 2^{\rho+3}$*.*

Proof. By definition, $c \leftarrow \left[\pi + 2r + \sum_{i \in S} x_i\right]_{x_0}$. Since $|x_0| > |x_i|$ for $i \in \{1, \ldots, \tau\}$, we have that

$$c = \left(\pi + 2r + \sum_{i \in S} x_i\right) + k \cdot x_0 \text{ for some } |k| \leq \tau.$$

For every i, there exist integers q_i and r_i with $|r_i| < 2^\rho$ such that $x_i = q_i \cdot p + 2r_i$, and also $|r_i| \leq 2^\rho$.

We now have

$$c = p \cdot \left(k \cdot q_0 + \sum_{i \in S} q_i\right) + \left(\pi + 2r + k \cdot 2r_0 + \sum_{i \in S} 2r_i\right).$$

Regarding the righmost quantity in parentheses, we see it has the same parity as π, and the its absolute value is at most $4\tau + 3_2^\rho < \tau 2^{rho+3}$ as claimed. □

We now prove the next lemma that says the same is true for ciphertexts output by $\mathsf{Evaluate}$.

Lemma 5.0.2. *Let* $(\mathsf{SK}, \mathsf{PK})$ *be output by* $\mathsf{KeyGen}(\lambda)$*. Let* $\mathcal{C} \in \mathcal{C}_\mathcal{E}$ *be a cirucit with* t *inputs and one output. For* $i \in \{1, \ldots, t\}$ *and* $\pi_i \in \{0,1\}$*, let* $c_i \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \pi_i)$*. Let* $\pi \leftarrow C(\pi_1, \ldots, \pi_t)$ *and* $c \leftarrow \mathsf{Evaluate}(\mathsf{PK}, C, c1, ldots, c_t)$*. Then,* $c = a \cdot P + (2b + m)$ *for some integers* a *and* b *with* $|2b + m| < p/8$*.*

Proof. Let C' be the generalized circuit corresponding to C, which operates over the integers rather than modulo 2. Generally, we have that $C'(c_1, \ldots, c_t) \in C'(2b_1 + \pi_1, \ldots, 2b_t + \pi_t) + p\mathbb{Z}$. Then $[C'(2b_1 + \pi_1, \ldots, 2b_t + \pi_t)]_p$ has the same parity as $\pi = C(\pi_1, \ldots, \pi_t)$. We also have that $|C'(2b_1 + \pi_1, \ldots, 2b_t + \pi_t)| \leq 2^\eta/16 \leq p/8$ by the definition of $\mathcal{C}_\mathcal{E}$, since $|2b + m| \leq \tau 2^{\rho+3}$ by the previous lemma. □

These two lemmas immediately imply Lemma 5.1: for any circuit in $\mathcal{C}_\mathcal{E}$ and any encryptions of inputs to that circuit, the integer output by Evaluate is of the form $c = a \cdot p + (2b + \pi)$ with $|2b + \pi| < p/8$ (where pi is the plaintext that c is supposed to encrypt). Then we have $[c]_p = 2b + \pi$, and thus $\pi = [[c]_p]_2$. □

Now, our definition of the set $\mathcal{C}_\mathcal{E}$ isn't entirely straightforward and it's hard to visualize. We have, by the triangle inequality, that a k-input Add gate increases the magnitude of the ciphertext integers by at most a factor of k. However, a 2-input Mult game may *square* the magnitude of the integers (doubling their bit lengths). Then the bottleneck in our scheme is the *multiplicitave depth* of the circuit (the degree of the polynomial computed by the circuit). We then have the following lemma from [3]:

Lemma 5.2. *Let C be a boolean circuit with t inputs and let $C^\dagger$ be the associated integer circuit. Let $f(x_1, \ldots, x_t)$ be the multivariate polynomial computed by $C^\dagger$, and let d be it's degree. If $|\overline{f}| \cdot (2^{\rho'+2})^d \leq 2^{\eta-4}$, (where $|\overline{f}|$ is the l_1 norm of the coefficient vector of f), then $C \in \mathcal{C}_\mathcal{E}$.* □

In particular, $\mathcal{E}$ can handle f as long as

$$d \leq \frac{\eta - 4 - \log |\overline{f}|}{\rho' + 2} \tag{1}$$

We refer to the set of polynomials that satsify (1) as *permitted polynomials* and will denote them by $\mathcal{P}_\mathcal{E}$ and also denote $C(\mathcal{P}_\mathcal{E})$ as the set of circuits that compute them. Then we can summarize our results as $C(\mathcal{P}_\mathcal{E}) \subseteq \mathcal{C}_\mathcal{E}$. For our purposes, we consider settings where $\log |\overline{f}|$ is small in relaiton to η, $\rho' = \omega(\log \lambda)$, and $t, \tau < \lambda^\beta$, and we need to support polynomials of degree up to $\alpha\lambda \log^2 \lambda$ for constants α, β.

5.4 Making the Scheme Fully Homomorphic

We note that our current decryption circuit seems to require boolean circuits that are deeper than what our somewhat homomorphic scheme can handle. We will now use Gentry's transformation to "squash the decryption circuit". This involves adding information about the secret key to the public key, and then using this information to re-process the ciphertext so that it may be decrypted more efficiently. This leads to our neccessary bootstrappability property.

5.4.1 Squashing the Decryption Circuit

We follow the construction in Van Dijk 6.1 [3] and modify our scheme as follows:

We start with new paramaters (functions of λ): $\kappa = \psi\eta/\rho'$, $\theta = \lambda$, and Θ is $\omega(\kappa \cdot \log \lambda)$.

- KeyGen: Generate $\mathsf{SK}^* = p$ and PK^* as before. Set $x_p \leftarrow \lfloor 2^\kappa/p \rceil$, and choose at random a Θ-bit vector with Hamming weight (the sum of its elements in this case) θ, $\overline{s} = \langle s_1, \ldots, s_\Theta \rangle$, and let $S = \{i : s_i = 1\}$.

  Choose at random integers $u_i \in \mathbb{Z} \cap [0, 2^{\kappa+1}), i = 1, \ldots, \Theta$, subject to the condition $\sum_{i \in S} u_i = x_p (\mod 2^{\kappa+1})$. Set $y_i = u_i/2^\kappa$ and $\overline{y} = \{y_1, \ldots, y_\Theta\}$. Then each y_i is a positive (rational) number smaller than two, with κ bits of precision after the binary point. also $[\sum_{i \in S} y_i]_2 = (1/p) - \Delta_p$ for some $|\Delta_p| < 2^{-\kappa}$.

  Output the secret key $\overline{s}$ and the public key $(\mathsf{PK}^*, \overline{y})$.

- Encrypt and Evaluate: Generate a ciphertext π^* as before (an integer). Then for $i \in \{1, \ldots, \Theta\}$, set $z_i \leftarrow [\pi^* \cdot y_i]_2$, keeping only $n = \lceil \log \theta \rceil + 3$ bits of precision for each z_i. Output both π^* and $\overline{z} = \langle z_1, \ldots, z_\Theta \rangle$.

- Decrypt Output $\pi' \leftarrow [\pi^* - \lfloor \sum_i s_i z_i \rceil]_2$.

The proof of the correctness of this scheme can be found in Lemma 6.1 of [3].

5.4.2 The Scheme is Bootstrappable

Theorem 5.1. *(Van Dijk et. al 6.2) Let $\mathcal{E}$ be the scheme above, and let $D_\mathcal{E}$ be the set of augmented (squashed) decryption circuits. Then, $D_\mathcal{E} \subset C(\mathcal{P}_\mathcal{E})$.*

We will show that $\mathcal{E}$ is bootstrappable. Combined with Theorem 4.1, we obtain homomorphic encryption schemes for circuits of any depth.

Proof. Our objective is to express our modified decryption circuit

$$\pi' \leftarrow c^* - \lfloor \sum_i s_i z_i \rceil \mod 2$$

as a permitted polynomial (satisfying (1)), and show there exists a polynomial size circuit that computes this polynomial.

We split the computation into three steps

$$\begin{array}{cccc} a_{1,0} & a_{1,-1} & a_{1,-2} & a_{1,-n} \\ a_{2,0} & a_{2,-1} & a_{2,-2} & a_{2,-n} \\ a_{3,0} & a_{3,-1} & a_{3,-2} & a_{3,-n} \\ & & \cdots & \\ a_{\Theta,0} & a_{\Theta,-1} & a_{\Theta,-2} & a_{\Theta,-n} \\ \hline W_0 & W_{-1} & W_{-2} & W_{-n} \end{array}$$

Figure 1: The procdure for summing up the $a_i's$: The integer W_{-j} is the Hamming weight of the column of bits $(a_{1,-j}, a_{2,-j}, \ldots, a_{\Theta,-j})$.

1. For $i \in \{1, \ldots, \Theta\}$, set $a_i \leftarrow s_i \cdot z_i$ (remember that the s_i's are indicator bits). Then the a_i's are still rational numbers in $[0, 2)$, given in binary representation with n bits of precision after the binary point.

2. From the Θ rational numbers $\{a_i\}_{i=1}^{\theta}$, generate other $n+1$ rational numbers $\{w_j\}_{j=0}^{n}$, each with less than n bits of precision, such that $\sum_j w_j = \sum_i a_i (\mod 2)$.

3. Output $c^* - \left(\sum_j w_j\right) \mod 2$.

The first step cann be trivially performed with a 1-level sub-circuit of multiplication gates. We then examine the second and third gates.

As noted in [3], computing the sum of k rational numbers in binary representation is well-studied. One scheme involves using the "three-for-two trick" where a constant-depth circuit is used to transform three numbers of arbitrary length into two numbers tha are at most one bit longer, such that the sum of the two result numbers is the same as the sum of the original three. However, this still does not give us a shallow enough circuit. We then use Gentry's technique takining advantage of the fact that all but θ of the $a_i's$ are zero.

Denote the bit representaiton of each number a_i by $a_{i,0}a_{i,-1}\ldots a_{i,-n}$. That is, $\sum_{j=0}^{n} 2^{-j} a_{i,-j}$. The biggest portion of this procedure is a subroutine for computing integers $W_{-j}, j = 0, 1, \ldots n$, where W_{-j} is the Hamming weight of the "column" of bits $(a_{1,-j}, a_{2,-j}, \ldots, a_{\Theta,-j})$. Since at most θ of the $a_i's$ are non-zero, then the W_{-j}'s are no larger than θ, and then can be represented by $\lceil \log(\theta + 1) \rceil < n$ bits. By Lemma 5.3 (to be proven), every bit in the binary rerpesentation of W_{-j} can be expressed as a polynomial of degree at most θ in the Θ variables $a_{i,-j}, i = 1, 2, \ldots \Theta$. Moreover all of these polynomials can be conputed simultaneously by an arithmetic circuit of size $O(\theta \cdot \Theta)$.

With the W_{-j}'s, we can get the sum of the a_i's by taking $\sum_i a_i = \sum_j 2^{-j} W_{-j}$. For $j = 0, 1, \ldots, n$ we set $w_j = (2^{-j} W_{-j}) \mod 2$, so the w_j's are rational numbers with $\lceil \log(\theta + 1) \rceil < n$ bits of precision. We can then use the three-for-two trick, obtaining the sum of the a_i's mod 2.

As explicitly examined in [3], the degree of the polynomials in the first step is two, the degree of the polynomials in the second step is at most θ, and the degree of the polynomial in the last step is at most

$$32(n+1)^{1/\log(3/2)} < 32\lceil \log \theta + 4 \rceil^{1.71} < 32 \log^2 \theta.$$

Then the total degree of the decryption circuit is bounded by $2 \cdot \theta \cdot 3 \log^2 \theta = 64\theta \log^2 \theta$. By our choice of $\theta = \lambda$, we have degree at most $64\lambda \log^2 \lambda$.

Then we see that the augmented decryption circuits $D_{\mathcal{E}}$ can be expressed as polynomials of degree at most $128\lambda \log^2 \lambda$ in the Θ variables s_i. Since the logaithm of the l_1 norm of this polynomial is small in relation to η, and since $\Theta = n\gamma/\rho \cdot \omega(\log \lambda) < \lambda^7$, and also $\tau < \lambda^7$, the argument at the end of section 5.3 gives us that we can get $D_{\mathcal{E}} \subset C(\mathcal{P}_{\mathcal{E}})$ with $\alpha = 128$ and $\beta = 7$. Then the scheme is bootstrappable. □

Finally, we give an explicit method to compute the W_j's using polynomials of degree no larger than θ.

Lemma 5.3. *(Lemma 6.3 of [3]) Let $\overline{\sigma} = \langle \sigma_1, \sigma_2, \ldots, \sigma_t \rangle$ be a binary vector, let $W = W(\overline{\sigma})$ be the Hamming weight of $\overline{\sigma}$, and denote the binary representation of W by $W_n \ldots W_1 W_0$.*

Then for every $i \leq n$, the bit W_i can be expressed as a binary polynomial of degree exactly 2^i in the variables $\sigma_1, \ldots, \sigma_t$. Moreover, there is an arithmetic circuit of size $2^i \cdot t$ that simultaneously computes all the polynomials for $W_0, \ldots, W_i$.

Proof. A well known method for getting the ith bit in the binary representation of the Hamming weight of a bit vector $\overline{\sigma}$ is to take $e_{2^i}(\overline{\sigma}) \mod 2$, where e_k is the kth elementary symmetric polynomial (see [3] Lemma 6.3). This is

$$W_i(\overline{\sigma}) = e_{2^i}(\overline{\sigma}) \mod 2 = \left(\sum_{|S| = 2^i} \prod_{j \in S} \sigma_j \right) \mod 2.$$

We see immediately the degree of e_{2^i} is exactly 2^i.

To complete the circuit, we compute the elementary symmetric polynomials in the σ_i's as the coefficiients of the polynomial $P_{\overline{\sigma}} = \prod_i = 1^t(z - \sigma_i)$, with $e_k(\overline{\sigma})$ being the coefficient of z^{t-k} in the result. This is exactly the polynomial we need to complete the proof. We can do better: to compute only the first few bits $W_0, \ldots, W_i$, we can throw away the lower order terms of $P_{\overline{\sigma}}$, so we do not need the coefficients of z_j where $j < t - 2^i$. The authors of [3] go on to explain further optimizations. □

6 Conclusion

We have demonstrated a fully homomorphic encryption scheme over the integers. Notably, we have not discussed the security of such a scheme, but Van Dijk et al. [3] discuss the security of our specific scheme at length.

The reader may wonder how this scheme is being used today. Unfortunately, this scheme is too impractical to implement on a large dataset (see the bound given in Theorem 5.1), and only recently have there been published implementations that can run on medium-sized datasets. [5] This is the main open problem, to optimize efficiency while preserving security.

References

[1] Coron, Naccache, and Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2012*, pages 446–464. Springer, 2012.

[2] Cramer, Ronald, and Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[3] Van Dijk, Gentry, Halevi, and Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[4] Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[5] Halevi. Helib, an implementation of homomorphic encryption. `https://github.com/shaih/HElib`.

[6] Terr. Polynomial time. `http://mathworld.wolfram.com/PolynomialTime.html`.