Advances in Fully Homomorphic Encryption

Amar Singh
ars9he@virginia.edu

April 12, 2018

Contents

Chapter 1

Introduction

1.1 Overview

Definition 1.1.1. A **fully homomorphic encryption** (FHE) scheme allows one to evaluate arbitrary circuits over encrypted data without being able to decrypt. More *informally*, FHE allows us to perform arbitrary computation on encrypted data.

In the 1978 paper titled "On Data Banks and Privacy Homomorphisms", Rivest, Adelman and Dertouzos posed the question: "Can we do arbitrary computations on data while it remains encrypted, without every decrypting it?"[**1**]. Since then, this open problem was known as the "holy grail" of cryptography. Even so, the solution remained elusive. It was left unsolved until Craig Gentry's seminal thesis published in 2009 on the first Fully Homomorphic Encryption Scheme[**2**].

Since then, a number of fully homomorphic encryption schemes have built on Gentry's original construction, revealing clever and innovative ways to apply abstract mathematics within the realm of cryptography. To cover both the build up to Gentry's paper as well as recent advancements in FHE, this paper will be broken into three main parts:

1. Early Homomorphic Encryption
2. "A Fully Homomorphic Encryption Scheme" by Craig Gentry
3. "Secure Search via Multi-Ring Fully Homomorphic Encryption" by Akavia, Feldman, and Shaul

My hope is that by providing a high level explanation of this material, I can motivate others to dive deeper into these papers and consider applying themselves to the unsolved problems in this space.

1.2 Implications

The impact of an efficient and practical fully homomorphic encryption scheme cannot be understated. Progress in this field will substantially increase client privacy while simultaneously minimizing the trust that is necessarily delegated to third party providers for outsourced computation and storage. In this manner, it will help us overcome the current limitations of cloud computing and much more.

Example 1.2.1 (Healthcare)**.** Hospitals collect troves of data. What if they could share that data and perform arbitrary computations on that data without compromising security?

Example 1.2.2 (Finance)**.** Credit card companies, insurers, and banks store a significant amount of data on consumers. What if they could assess credit risk without necessarily exposing the private information of these consumers?

Example 1.2.3 (General Computation)**.** What if instead of relying on Amazon AWS or Google Cloud to store data and run computations, you could outsource both tasks to arbitrary servers? The risks inherent in such a procedure fall dramatically if the data sent to those servers is encrypted (and does not need to be decrypted).

These are only a few of the areas in which practical FHE could dramatically change the way we share and interact with data. The rest of the paper will focus on the progress that has been made to eventually realize the vision of a society in which trust is minimized and data privacy is upheld.

Chapter 2

Homomorphic Encryption

In this section, we will cover a few encryption schemes that are either additively or mutiplicatively homomorphic. It is clear that addition and multiplication form a complete set of operations thereby enabling arbitrary computation on encrypted data. Likewise, until Gentry's thesis (covered at a high level in the next section), constructing an encryption scheme that achieved fully homomorphic encryption ($\iff$ both additively and multiplicative homomorphic) remained an open problem.

2.1 Rivest-Shamir-Adelman (RSA)

Because RSA was covered in depth in Lecture 9, I will only make a few remarks and outline the general principles behind this public encryption scheme.

Specifically, the security of RSA is predicated on the fact that it is practical to find $x, y \in \mathbb{Z}$ such that with modular exponentiation $\forall\ \alpha \in \mathbb{Z}$ over a finite field of order $n \in \mathbb{Z}$,

$$(\alpha^x)^y \equiv \alpha (\text{mod } n)$$

without knowing x and n or even α, it is very difficult to compute y.

The public key is represented by n and x while the private key is represented by y. In this case, α can be seen as the message (which can also be considered part of the private key). The proof of the correctness for RSA is based on Fermat's Little Theorem which states that if p is a prime and, for $a \in \mathbb{Z}$, $p \nmid a \implies a^{p-1} \equiv 1(\text{mod } n)$. This proof of correctness is easy to find online[**3**].

2.2 ElGamal

The ElGamal algorithm provides an alternative to RSA for public key encryption with a few key differences. While we established that the security of RSA depends on the difficulty of factoring large integers, ElGamal conversely relies on the assumption that it is difficult to compute discrete logs in a large prime modulus[**4**].

To explain the semantics of ElGamal, we will proceed with an example:

Alice chooses a large prime p, a generator g of the multiplicative group of order p, and a random $z \in \mathbb{Z}$ such that $2 \leq z \leq p-2$. Alice, then computes $\beta = g^z (\text{mod } p)$. In this scheme, Alice's public key is (p, g, β) and her private key is z.

Bob encrypts a message $m < p$ and sends it to Alice by first choosing a random $k \in \mathbb{Z}$. Then, Bob uses k to compute $x \equiv g^k (\text{mod } p)$ and $y \equiv \beta^k * m (\text{mod } p)$ before discarding k. Bob sends the encrypted message (x, y) to Alice.

After Alice receives Bob's message, she can easily decrypt by computing xy^{-z}.

Proof. $xy^{-z} \equiv \beta^k * m(g^k)^{-z} (\text{mod } p) \equiv (g^z)^k * m * (g^k)^{-z} (\text{mod } p) \equiv m (\text{mod } p)$ □

Even if Eve intercepts the message sent by Bob (x, y), it is obvious that she can't perform the computation performed by Alice because she does not have z. However, Eve can find z if she computes a discrete log in the large prime modulus p. The hardness of the discrete log problem over a large prime modulus therefore ensures the security of this public key encryption scheme.

While ElGamal provides the advantage that the same plaintext outputs a different ciphertext (in nearly every case) each time it is encrypted, the length of the ciphertext is twice as long as the plaintext. With this in mind, the advantages of this reliably non-deterministic public key encryption scheme may be outweighed by the very long ciphertext output.

2.3 Other Schemes

"Computing Blindfolded: New Developments in Fully Homomorphic Encryption" by Vinod Vaikuntanathan [**5**] details the other developments in homomorphic encryption that occurred before Gentry's thesis on FHE. To name a few,

1. Pallier Encryption Scheme
2. Generalized Pallier Encryption by Damgard and Jurik
3. Lattice-Base Encryption Schemes starting from work by Ajtai and Dwork
4. A proposal for FHE by Fellows and Koblitz

These encryption schemes laid the groundwork for Gentry's breakthrough fully homomorphic encryption scheme (especially the work by Ajtai and Dwork on Lattice-Based Encryption Schemes).

Chapter 3

Fully Homomorphic Encryption

Craig Gentry's groundbreaking thesis [**2**] deserves significantly more attention than it will receive in this paper. The fully homomorphic encryption scheme described in Gentry's paper utilizes abstract algebra in an innovative way to layer the decryption process and allow computation to be performed on encrypted data (without fully decrypting). While the full construction is far to complicated to cover in this paper, I will introduce some of the mathematics necessary to understand the full scheme before describing it at a high level. Likewise, the next section, which is a review of some of the Algebra used in Gentry's paper, is not completely necessary for the reader to understand the rest of this paper.

`This section draws from a few sections of Gentry's paper [2]`

3.1 Algebra

3.1.1 Ring Theory

Definition 3.1.1. Let $f \in \mathbb{Z}[x]$ be a monic polynomial of degree n and consider the quotient ring $\mathbb{Z}[x]/< f >$. Using the standard set of representatives $\{(g \bmod f) \mid g \in \mathbb{Z}[x]\}$, an identification of polynomials with vectors, the quotient ring $\mathbb{Z}[x]/< f >$ is isomorphic (as an additive group) to the integer lattice $\mathbb{Z}^n$ and any ideal $I \subseteq \mathbb{Z}[x]/< f >$ defines a corresponding integer sublattice $\mathcal{L}(I) \subseteq \mathbb{Z}^n$. An **ideal lattice** is an integer lattice $\mathcal{L}(B) \subseteq \mathbb{Z}^n/< f >$ such that $B = \{g \bmod f \mid g \in I\}$ for some monic polynomial f of degree n and ideal $I \subseteq \mathbb{Z}[x]/< f >$.

Gentry's construction uses the polynomial ring $\mathbb{Z}[x]/f(x)$ such that $f(x)$ is a monic polynomial with $deg(f(x)) = n$. An element $\mathbf{v} \in R$ is a coefficient vector $\mathbf{v} \in \mathbb{Z}^n$. The ideal $< \mathbf{v} >$ is generated by the vector $\mathbf{v} \in R$ and corresponds to the lattice generated by the column vectors $\{\mathbf{v}_i \leftarrow \mathbf{v} \times x^i \bmod f(x) \mid i \in [0, n-1]\}$. These column vectors are the **rotation basis** of the *ideal lattice* $< \mathbf{v} >$. For all $\mathbf{w} \in < \mathbf{v} > \exists\ \mathbf{a}$ such that $\mathbf{w} = \mathbf{v} \times \mathbf{a} \implies \mathbf{w} = \sum_i a_i \mathbf{v}_i$.

Definition 3.1.2. The **rotation basis** for the product of two elements $\mathbf{a}, \mathbf{b} \in \mathbb{Q}[x]/(f(x))$ is the rotation basis of $\mathbf{a} \times \mathbf{b}$. Moreover, the matrix-vector product of a rotation basis $\mathbf{a}$ with the vector $\mathbf{b}$ is the vector $\mathbf{a} \times \mathbf{b}$.

In order to use inverses in the ring $\mathbb{Q}[x]/(f(x))$, Gentry's construction assumes $f(x)$ is irreducible (therefore, all nonzero terms have inverses). It follows from this that $I \lhd R \implies I^{-1}$ is a *fractional ideal* of $R \iff I^{-1} = \{\mathbf{x} \in \mathbb{Q}[x]/(f(x)) \mid \forall\ \mathbf{y} \in I, \mathbf{x} \times \mathbf{y} \in R\}$.

This type of polynomial ring is called a *monogenic* number ring, or *simple algebraic extension*, because these rings are isomorphic to $\mathbb{Z}[\alpha]$ when α is a root of $f(x)$.These rings are easy to work with but are not yet optimized for the FHE scheme. One attractive choice for the fixed ring R is the ring $\mathbb{Z}[x]/(x^n - 1)$.

3.1.2 Ideal Lattices

The problem with previous constructions was that they achieve logarithmic depth by allowing the ciphertext size to scale exponentially with circuit depth. As the ciphertext grows, the decryption circuit must also manage the larger ciphertexts. Under this paradigm, as the scheme permits larger ciphertexts, the evaluation depth becomes very large such that it can eventually never ”catch” up to the depth of the decryption circuit. Conversely, encryption schemes based on *lattices* or *linear codes* have a corresponding decryption scheme whose operations are dominated by an inner product. Likewise, ideal lattices have a multiplicative and additive structure that enables the evaluation of deep arithmetic circuits.

3.2 High Level

Gentry constructs a fully homomorphic encryption scheme using ideal lattices. This scheme can be broken down into three steps:

1. A general ”bootstrapping” result
2. Initial construction using ideal lattices
3. Technique to ”squash the decryption circuit” to permit bootstrapping

Gentry's contributions start during step (2). Specifically, Gentry proposed a public key encryption system that incorporated ideal lattices and is homomorphic for shallow circuits. In the first variation of this scheme $\underline{\mathcal{E}_1}$, a cipher text ψ is encoded as $\mathbf{v} + \mathbf{x}$ such that $\mathbf{v}$ is in the ideal lattice and $\mathbf{x}$ is an error vector used to encode the plaintext π. The ciphertext vectors are encoded as coefficient vectors of elements in the polynomial ring $\mathbb{Z}[x]/f(x)$. In this way, the encrypted ciphertexts can be added and multiplied using ring operations: $\psi \leftarrow \psi_1 + \psi_2$ or $\psi \leftarrow \psi_1 \times \psi_2$. $\mathcal{E}_1$ is homomorphic for circuits with greater multiplicative depth while also allowing unlimited additions.

However, $\mathcal{E}_1$ is only homomorphic for shallow circuits. This follows from the fact that the error vector $\mathbf{x}$ grows with addition and especially multiplication operations. This means that this error vector eventually becomes so long that it will cause a decryption error. The trivial solution would be to decrypt the ciphertext before refreshing the ciphertext, but this protocol would be very limited and require multiple rounds of communication. Instead, bootstrapping allows us to decrypt the ciphertext homomorphically.

Suppose $\mathcal{E}$ is *bootstrappable* with plaintext space $\mathcal{P} = \{0, 1\}$ and with boolean circuits. If we encrypt a plaintext π with public key pk_1 to attain ciphertext ψ_1, we can refresh the ciphertext by first decrypting homomorphically. Suppose we have a secret key sk_1 that corresponds to pk_1 which is also encrypted under a second public key pk_2. The resulting encrypted secret key bits will be denoted $\overline{sk_{1j}}$.

This is best captured in the following algorithm:

```
$Recrypt_{\mathcal{E}}(pk_2, D_{\mathcal{E}}, <\overline{sk_{1j}}>, \psi_1)$
$\text{Set } \overline{\psi_{1j}} \leftarrow Encrypt_{\mathcal{E}}(pk_2, \psi_{1k})$
$\text{Output } \psi_2 \leftarrow Evaluate_{\mathcal{E}}(pk_2, D_{\mathcal{E}}, << \overline{sk_{1j}}>, <\overline{\psi_{1j}}>>)$
```

In this scheme, *Evaluate* takes the bits of sk_1 and ψ_1 which are encrypted under pk_2. Therefore, the output ψ_2 is an encryption under the second public key pk_2. Applying the decryption circuit $D_{\mathcal{E}}$ removes the error vector associated with the first ciphertext but Evaluate$_{\mathcal{E}}$ simultaneously introduces a new error vector. We have made progress as long as the new error vector is shorter than the previous one.

$C_{\mathcal{E}}$ (permitted set of circuits) contains $D_{\mathcal{E}}$ and the augmentation of $D_{\mathcal{E}}$ by a NAND gate $\implies$ $\mathcal{E}$ is $\underline{\text{bootstrappable}}$.

Theorem 3.2.1. *One can construct a family $\{\mathcal{E}^{(d)}\}$ of leveled fully homomorphic encryption schemes from any bootstrappable encryption scheme $\mathcal{E}$.*

3.2.1 Summary

The construction described above is a simplified version of Gentry's scheme in which homomorphic encryption using ideal lattices takes advantage of *bootstrappability* in order to achieve fully homomorphic encryption. In this context, to be bootstrappable, the encryption scheme $\mathcal{E}$ must be able to evaluate its own decryption circuit (which includes recryptions of the same plaintext) as well as slightly augmented versions of this decryption circuit. This provides the flexibility necessary to perform operations on plaintexts and progress through a circuit.

Chapter 4

Secure Search via Multi-Ring FHE

A recent paper titled "Secure Search via Multi-Ring Fully Homomorphic Encryption" [**6**] by Adi Akavia, Dan Feldman, and Hayim Shaul improves upon previous secure search algorithms by presenting a secure search algorithm that achieves a polynomial of logarithmic degree $log^{O(1)}m$ for m the number of database records of existing solutions. This significantly improves upon the $\Omega(m)$ complexity achieved by previous secure search algorithms.

One key motivating sentence from the paper reads, "With e-mail, medical, financial, and other personal information transferring to the cloud, it is paramount to guarantee privacy on top of data availability while keeping correctness of the computations."[**6**] This sentence underscores the necessity of developing more efficient fully homomorphic encryption algorithms.

`This section draws from a few sections of [6]`

4.1 Secure Search

Definition 4.1.1 (Secure Search)**.** The *server* holds an unsorted array of encrypted values (previously uploaded to the server, and where the server has no access to the secret decryption key):

$$[\![array]\!] = ([\![x_1]\!], ..., [\![x_m]\!])$$

The client sends the server an encrypted lookup value $[\![l]\!]$. The server returns to the client an encrypted index and value

$$[\![y]\!] = (]\!]i]\!], [\![x_i]\!])$$

satisfying the condition that $isMatch(x_i, l) = 1$.

Definition 4.1.2 (Efficiency)**.** The **client** is deemed **efficient** if its running time is polynomial in the output length $|i|= O(logm)$ and $|x_i|$ and in the time to encrypt/decrypt a single ciphertext. The **server** is **efficient** if the polynomial $f([\![array]\!], [\![l]\!])$ the server evaluates to obtain $[\![y]\!]$ is of degree polynomial in $log(m)$ and the degree of $isMatch()$ and of size polynomial in m and the size of $isMatch$. The **protocol** is **efficient** if both the client and server are efficient.

A major challenge for designing algorithms that run on data encrypted with FHE is to present the computation as a low degree polynomial. Moreover, previous secure search protocols suffer from one of the following shortcomings:

1. Restricted search functionality (i.e. PIR, PSI)
2. Inefficient in the sense of having at least linear dependence on the database size m or either the client's running time or the degree of the polynomial computed by the server (i.e. natural folklore secure search solution, secure pattern matching)
3. Security is weakened to leak vital search information (i.e. searchable encryption)

4.2 High Level

The server computes and sends to the client a short list of candidates and the client chooses the smallest candidate to decode the correct value. To simplify the presentation of the protocol, it returns the index i^*. Even so, standard PIR techniques with no further interactions can return (index, value) pairs (in which case the client can efficiently decode to obtain the desired pair: $(i^*, array(i^*))$.

The low-degree polynomial proposed consists of two separate steps. In the first step, some function takes as input $(array, l)$ and outputs a binary vector indicator $indicator \in \{0,1\}^m$ that indicates $\forall$ $i \in [m]$ whether the record $array(i)$ matches the lookup value l. This is done by the server side and can incorporate any generic pattern matching polynomial $isMatch()$ given as part of the given problem specification.

In the second step, the client receives the output from the server in the first step and identifies the index i^*, the first match in $array$ for l, namely the first index such that

$$isMatch(array(i^*, l) = 1$$

The natural polynomial for computing this first positive index i^* (by the client) is of high degree $\Omega(m)$, resulting in an inefficient protocol. The paper's main contribution is proposing a novel way to compute this first positive index i^* via low degree polynomial of degree $log^{O(1)}m$, resulting in an efficient protocol.

The naive sketch parameterized by the length m of the input vector and the modulus p for arithmetic operations is a degree $(p-1)^2$ polynomial. The problem with this naive sketch is that to guarantee that the output is the first positive index, as desired, the scheme requires a large modulus $p = \Omega(m)$ which result in a polynomial $SPiRiT_{m,p}$ of high degree $\Omega(m^2)$.

To resolve this problem, the authors replace the evaluation of $SPiRiT_{m,p}$ on a single large ring $p = \Omega(m)$ by few $k = o(log^2 m)$ evaluations of such polynomials but on small rings moduli $p_1, ..., p_k = O(log^2 m) \implies$ evaluating in parallel k polynomials each of low degree $O(log^4 m)$. Instead of the server returning the desired value b^* to the client, the server returns the k values from the aforementioned computations over multiple rings. The additional time on the client side that is required to extract the desired value i^* from the k outputs is very fast with only a small overhead over receiving b^* by a factor of $o(log^2 m)$. In summary, they redefine a hard problem (that requires a single output from a large polynomial ring) to an easier problem that uses multiple outputs (from a few small polynomial rings).

In known single ring arithmetic circuits, all known secure search solutions have multiplicative depth $\Omega(log(m))$ (where the multiplicative depth is essentially equal to the logarithm of the degree of the polynomial evaluated by the circuit and therefore is not considered practical).

In this work, the authors propose a multi ring FHE evaluation that consists of computing an arithmetic circuit with gates for addition/multiplication over several ring moduli $p_1, ..., p_k$. The paper reveals a multi-ring arithmetic circuit for secure search of multiplicative depth $O(loglog(m))$. This presents an exponential improvement over the previous scheme.

4.3 In-Depth

4.3.1 Defining the Client-Server Relationship

Using the data upload protocol picture to describe the client-server relationship. I think generalizing to blockchains could be useful?. In terms of generalizing to blockchains, what if you could split up tasks that are involved in one bigger task in this manner and offer permissioned rights on data achieved by proxy-re encryption. In this way, you could outsource consensus and computation to blockchains in a more efficient yet still very secure way.

4.3.2 Complexity Goal

Their primary compexity goal involves minimizing the client's latency (as in the wait time between sending a search query and obtaining the search result). This latency accounts for both the server's time for evaluating the search polynomial as well as the client's time for encryption, decryption, and decoding of the received evaluation outcome.

4.3.3 Black Box Usage of Semantically Secure FHE

4.4 Results

Akavia, Feldman and Shaul presented the first *efficient protocol for secure search* that is applicable to large datasets with unrestricted search functionality. Specifically, the protocol provides:

1. *Efficient Client*:The client's running time is proportional to the time to compute $log^{O(1)}m$ encryption/decryption operations plus the time to read the retrieved record $(1, x_i)$.

2. *Efficient Server*: The server evaluates a polynomial of degree $log^{O(1)}(m) * deg(isMatch)$ and size $m * size(isMatch)$ (where we denoted $deg(f)$ and $size(f)$ the degree and size of the polynomial f).

3. *Unrestricted search functionality*: the protocol is applicable to any data *array* and lookup value l, with no restrictions on the number of records in *array* that match the lookup value l.

4. *Full security*: The input data *array* and lookup value l are encrypted with fully, or leveled, homomorphic encryption (FHE) achieving the strong property of semantic security both for data at rest and during searching.

Bibliography

1. ”On Data Banks and Privacy Homomorphisms”. Rivest, Adelman, Dertouzos. 1978.
2. ”A Fully Homomorphic Encryption Scheme”. Craig Gentry. 2009.
3. https://www.geeksforgeeks.org/rsa-algorithm-cryptography/
4. http://homepages.math.uic.edu/ leon/mcs425-s08/handouts/el-gamal.pdf
5. http://www.cs.toronto.edu/ vinodv/FHE-focs-survey.pdf
6. ”Secure Search via Multi-Ring Fully Homomorphic Encryption”. Adi Akavia, Dan Feldman, and Hayim Shaul. https://eprint.iacr.org/2018/245.pdf.