Computing on Encrypted Data

6.857 Final Project

May 15, 2014

Jiarui Huang
Min Zhang
Weixin Chen
Yi-Shiuan Tung

Abstract: This paper gives an overview of the research that has been done on computing on encrypted data, namely fully homomorphic encryption (FHE) and functional encryption (FE). We first describe public-key encryption schemes and then provide a background and motivation for FHE and FE. We then outline the ideas of the FHE and FE encryption schemes and security. We also tested the performance of a current homomorphic encryption implementation. The goal of the paper is to equip the reader with enough knowledge to pursue deeper understanding of the topics presented.

1 Introduction

Public-key cryptography has provided a way to securely transmit data from one party to another, as well as storing sensitive data and many other applications. As cloud services have become more popular, demands for computing data in the cloud have increased. However, the traditional public-key cryptography doesn't support the computation of encrypted data in the cloud. Given the security guarantees of the public key encryption schemes, the server cannot learn anything about the message from the ciphertext. Unless the server has access to the secret key and simply performs the computation on a decrypted message, the server cannot do any sensible computation. FHE is a type of encryption that would allow the server to compute on the encrypted data. The user with the secret key would then decrypt the result and obtain an answer that is equal to the computation being done directly on the plaintext message. What if the server needs to learn the result of the computation but still not learn anything about the underlying messages? FE encryption schemes solve this type of problem. Before diving into FHE and FE, we first step back a little to look at public key cryptography.

2 Public-key Cryptography

Public-key cryptography, also known as asymmetric cryptography, is a class of cryptographic algorithms which has been widely used in practice. Unlike symmetric cryptography, where the same cryptographic keys are used for both encryption of plaintext and decryption of ciphertext, public-key cryptography uses two keys - a public key known to everyone and a private key (or secret key) known only to the recipient of the message. Sender of the message uses public key to encrypt the message while the recipient is the only one that could decrypt the ciphertext using his/her private key.

Public-key algorithms are based on computationally hard problems such as integer factorization, discrete logarithm, and elliptic curve relationships. However, public-key algorithms should ensure that it is computationally easy for users to create their own pair of cryptographic keys and to use this pair of keys for encryption and decryption but it is infeasible for any adversary to reveal the secret key from knowing the public key. Different from symmetric encryption schemes, public-key algorithms could broadcast its public key without compromising the confidentiality and integrity.

Mathematically, we can describe the public-key scheme as follows:

- Set λ be the security parameter, then let the pair of public key and secret key be generated: $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$.
- Encrypt $m \in M$(message space) to $c \in C$(ciphertext space): $\mathsf{Enc}(pk, m) \rightarrow c$.
- Decrypts $c \in C$(ciphertext space) to $m' \in M$(message space): $\mathsf{Dec}(sk, c) \rightarrow m'$.
- Note that the ecnryption can be randomized but decryption has to be deterministic such that $\forall (pk, sk), \forall m$, $\mathsf{Dec}(sk, Enc(pk, m)) \rightarrow m$.

3 Homomorphic Encryption

3.1 Overview

Is it possible to delegate processing of your data without giving away access to it? Consider the scenario that Bob wants to do some complicated computation on his personal data. He wants to take advantage of the computing power of the cloud, however, to put everything online, unencrypted, is to risk Bob's own privacy. For some certain types of information, such as academic transcript or medical record, storing them unencrypted is against the law. So what should Bob do?

Homomorphic encryption (HE) will be the solution in this case. HE allows the omnipotent cloud to manipulate Bob's encrypted data while Bob doesn't have to sacrifice his own privacy. This notion might seem paradoxical or even logically impossible. In order to give some intuition about the solution, let us consider an analogous problem in real life.

Assume that Alice has some precious raw materials, like diamond, silver and gold, that she wants to be processed into diamond rings. She distrusts her workers and is afraid that her work will steal or replace the valuable materials. In order words, she wants her workers to assemble the raw pieces for her while she doesn't want to give the workers the direct access to the materials. Here's what she does: Alice uses a transparent impenetrable glovebox. The workers can assemble raw materials using the gloves while they can't remove the materials from the box since the glovebox is secured by a lock for which Alice is the only one that has the key. After workers finish processing the materials as intended, Alice unlocks the box and retracts the finished piece.

3.2 Partially Homomorphic Encryption

In 1978, shortly after RSA was invented, Rivest, Adleman, and Dertouzos [RAD78] suggested that fully homomorphic encryption would be possible. For decades, people had been struggling to find such a secure scheme. However, several efficient, partially homomorphic cryptosystems have been developed. Unpadded RSA and ElGamal are both partially homomorphic, which means that the scheme either supports addition or multiplication and not both. If a homomorphic scheme supports both addition and multiplication operations, then any functions can be computed. This type of scheme is termed fully homomorphic. In the following discussion, we will introduce two wildely adopted public-key encryption schemes that are partially homomorphic.

3.2.1 Unpadded RSA

RSA, first proposed in 1977, stands for the initials of the designers: Ron Rivest, Adi Shamir and Leonard Adleman. RSA is one of most widely adopted public-key alogrithms. Its security is based on the compuational difficulty of factoring the product of two large prime integers.

KeyGen: Chooese two distinct prime numbers p and q. p, q should be chosen randomly and of similar length. Compute $n = pq$, and $\phi(n) = \phi(p)\phi(q) = n - (p + q - 1)$, where ϕ is the Euler totient function. Choose an integer e such that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$. Set public key $PK \leftarrow (n, e)$. Compute $d \equiv e^{-1} \pmod{\phi(n)}$, and set secret key $SK \leftarrow d$.

Enc: Alice sends her public key to Bob. Bob first turns message M into an integer m such that $0 \leq m < n$ and compute the cipher text by $c \equiv m^e \pmod{n}$.

Dec: Alice recovers m from c by using private key d via computing $m \equiv c^d \pmod{n}$.

If the RSA public key is modulus n and exponent e, then the encryption of a message x is given by $Enc(x) = x^e \bmod n$. The homomorphic property is then:
$Enc(x_1) \cdot EncE(x_2) = x_1^e x_2^e \bmod n = (x_1 x_2)^e \bmod n = Enc(x_1 \cdot x_2)$

3.2.2 ElGamal

Proposed by Taher ElGamal in 1985, ElGamal encryption is another widely used public-key encryption scheme. ElGamal encryption scheme is based on Diffie-Hellman key exchage. Its security depends on the computational difficulty of discrete logarithms. The decisional Diffie-Hellman (DDH) assumption is a computational hardness assumption about a certain problem involving discrete logarithms in cyclic groups.

Definition 3.1. *In a multiplicative cyclic group G of order q, and with generator g, decisional Diffie-Hellman (DDH) assumption states that:*

- (g^a, g^b, g^{ab})*, where a and b are randomly and independently chosen from* $\mathbb{Z}_q$
- (g^a, g^b, g^c)*, where a,b,c are randomly and independently chosen from* $\mathbb{Z}_q$

are computationally indistinguishable.

The scheme is provided as follows:

KeyGen: Alice generates an efficient description of a cyclic group G, of order q, with generator g. See below for a discussion on the required properties of this group. Alice chooses a random $x \in \{1 \ldots q - 1\}$. Alice computes $h = g^x$. Alice sets $PK \leftarrow (G, q, g, h)$. Alice retains x, as her private key which must be kept secret.

Enc: To encrypt a message m under Alice's public key (G, q, g, h), Bob chooses a random $y \in \{1 \dots q-1\}$, then calculates $c_1 = g^y$. He calculates the shared secret $s = h^y$, converts his secret message m, into an element m', of G, and calculates $c_2 = m' \cdot s$. Bob then sends the ciphertext, $(c_1, c_2) = (g^y, m' \cdot h^y) = (g^y, m' \cdot (g^x)^y)$, to Alice.

Dec: To decrypt a ciphertext (c_1, c_2), with her private key x, Alice calculates the shared secret $s = c_1^x$, and $m' = c_2 \cdot s^{-1}$, which she then converts back into the plaintext message m, where s^{-1} is the inverse of s in the group G. Note that although encryption might be randomized, the decryption algorithm produces the intended message deterministically, since $c_2 \cdot s^{-1} = m' \cdot h^y \cdot (g^{xy})^{-1} = m' \cdot g^{xy} \cdot g^{-xy} = m'$.

In the ElGamal cryptosystem, in a group G, if the public key is (G, q, g, h), where $h = g^x$, and x is the secret key, then the encryption of a message m is $Enc(m) = (g^r, m \cdot h^r)$, for some random $r \in \{0, \dots, q-1\}$. The homomorphic property is then:
$Enc(x_1) \cdot Enc(x_2) = (g^{r_1}, x_1 \cdot h^{r_1})(g^{r_2}, x_2 \cdot h^{r_2}) = (g^{r_1+r_2}, (x_1 \cdot x_2)h^{r_1+r_2}) = Enc(x_1 \cdot x_2)$.

3.3 Fully Homomorphic Encryption

Most of the early successful attempts start with constructing a private key scheme, and then transform it into a public key version. Rothblum [Rot11] showed that it is possible to extend any compact private key homomorphic scheme into a public key homomorphic scheme that is just slightly less homomorphic. However, since the development will eventually lead to both leveled and fully homomorphic schemes this slight reduction in homomorphic capability is not a very big concern. Here we focus on results of the public key schemes.

A homomorphic public key encryption scheme $\mathcal{E}$ has four algorithms: the usual KeyGen, Enc, and Dec, and an additional algorithm Eval. The algorithm Eval takes as input a public key pk, a circuit C, a tuple of ciphertexts $\vec{c} = \langle c_1, \cdots, c_t \rangle$ (one for every input bit of C), and outputs another ciphertext c.

Definition 3.2. *The scheme $\mathcal{E}$ = (KeyGen, Enc, Dec, Eval) is correct for a given t-input circuit C if, for any key-pair (sk, pk) output by KeyGen(λ), any t plaintext bits $m_1, \cdots, m_t$, and any ciphertexts $\vec{c} = \langle c_1, \cdots, c_t \rangle$ with $c_i \leftarrow$ Enc$_{\mathcal{E}}(pk, m_i)$, it satisfies:*

$$\mathsf{Dec}(sk, \mathsf{Eval}(pk, C, \vec{c})) = C(m_1, ..., m_t).$$

Definition 3.3. *The scheme $\mathcal{E}$ = (KeyGen, Enc, Dec, Eval) is homomorphic for a class $\mathcal{C}$ of circuits if it is correct for all circuits $C \in \mathcal{C}$. $\mathcal{E}$ is fully homomorphic if it is correct for all boolean circuits.*

Note that the above definitions do not exclude trivial constructions such as an Eval that simply outputs $C, c_1, \cdots, c_t$. We rule out the possibility by requiring circular security (ciphertext generated by Eval does not reveal anything about the circuit that it evaluates beyond the output value of that circuit) and compactness (the size of the ciphertext Eval$(pk,\ C, \vec{c})$ is bounded by some fixed polynomial $b(\lambda)$ bits, where λ is the security parameter).

3.3.1 Learning with Error (LWE)

The existing fully homomorphic encryption (FHE) schemes are based on LWE (on $\mathbb{Z}_q^n$), R-LWE (on $\mathbb{Z}_q[x]/\langle f\rangle$) or the analogs in integer. To keep things simple, we only give examples for LWE on $\mathbb{Z}_q^n$ and the analog on $\mathbb{Z}$.

Definition 3.4. *A pair $(\vec{a}, c)$ follows a LWE$_{\vec{s}}$ distribution if for some fixed $\vec{s} \in \mathbb{Z}_q^n$, $c = \langle \vec{a}, \vec{s}\rangle + e \mod q$, where $\vec{a} \in_R \mathbb{Z}_q^n$, $e \sim \mathcal{N}(0, \alpha q)$. For high dimension, A pair $(A, \vec{c})$ where $A \in_R \mathbb{Z}_q^{n\times m}$ follows a LWE$_{\vec{s}}$ distribution for some fixed $\vec{s} \in \mathbb{Z}_q^n$ if $\vec{c} = \vec{s}A + \vec{e} \mod q$, where $\vec{e} \sim \mathcal{N}(0, \alpha q)^m$.*

Definition 3.5. *The search LWE problem is to find $\vec{s}$ given pairs of $(A, \vec{c})$ (or (a, c)) sampled from LWE$_{\vec{s}}$. The decision LWE problem is to distinguish LWE$_{\vec{s}}$ distribution for a random $\vec{s}$ from uniform distribution on $\mathbb{Z}_q^{(n+1)\times m}$.*

For FHE scheme based on LWE, the decision LWE problem gives the one-wayness. Examples are Gentry-Halevi-Vaikuntanathan scheme ('11) on $\mathbb{Z}_q^n$ with a Alwen-Peikert Trapdoor T_A satisfying $AT_A = 0 \mod q$ that can recover $\vec{e}$ by $((\vec{s}A + \vec{e}) \times T_A \mod q) \times T_A^{-1} = \vec{e}$; Gentry [Gen09] is a GoldreichGoldwasserHalevi type scheme [GGH97] on ideal lattice.

Certainly, IND-CCA2 is not an adequate security concept for FHEs as the homomorphic property implies malleability. They are IND-CPA1 with the assumption of hardness of approximate GCD or Bounded Distance Decoding depending on what ring the scheme dwells in. Furthertmore, Loftus-May-Smart-Vercauteren [LMSV10] claims to achieve IND-CCA1.

As the circuit (computation) grows at scale, the error term accumulates and ciphertext may explode, therefore without further tools, only gives a somewhat homomorphic encryption (SWHE): it can only evaluate circuit whose corresponding polynomial is of low order. The first success in solving this problem is Gentry's phd thesis [Gen09], which gives the first FHE using bootstrapping. Generally, they are two types of methods to construct a FHE from a SWHE:

- Bootstrapping with squashing (Sparse Subset-Sum Problem) or with techniques developed in Gentry-Halevi [GH11].
- Without bootstrapping: Brakerski-Valkuntanathan [BV11] by dimension switching (reducing ciphertext size) and modulus switching (reducing error), and their variants in Brakerski-Gentry-Valkuntanathan [BGV11].

3.3.2 A FHE with Bootstrapping

We present the Dijk-Gentry-Halevi-Vaikuntanathan scheme [DGHV10], which is a simplified version on integer ring of Gentry ideal-lattice based encryption scheme [Gen09]. Here the analog of LWE is to add some noise on the public key and the encryption (bit-by-bit):

$\mathsf{KeyGen}(\lambda)$: The secret key is an odd η-bit integer: $sk = p \in_R (2\mathbb{Z}+1) \cap [2^{\eta-1}, 2^{\eta})$. The public keys $pk = \langle x_0, \cdots, x_\tau \rangle$, where $x_i \in_R \mathcal{D}_{\eta,\lambda}(p) = \{pq + r | q \in_R \mathbb{Z} \cap [0, 2^{\tau-\eta}/p), r \in_R \mathbb{Z} \cap (-2^\lambda, 2^\lambda)\}$. Relabel so that x_0 is the largest.

$\mathsf{Enc}(pk, m \in \{0,1\})$: Choose a random subset $S \subseteq \{1, 2, ..., \tau\}$ and integer $r \in_R (-2^{2\lambda}, 2^{2\lambda})$, and output $c \leftarrow [m + 2r + 2\sum_{i \in S} x_i]_{x_0}$[1].

$\mathsf{Dec}(sk, c)$: Output $m' \leftarrow [[c]_p]_2$.

$\mathsf{Eval}(pk, C, c_1, \cdots, c_t)$: Given the (binary) circuit $C_\mathcal{E}$ with t inputs, and t ciphertexts c_i, apply the addition and multiplication gates of $C_\mathcal{E}$ to the ciphertexts, performing all the operations over the integers, and return the resulting integer. The security is based on the (η, λ)-approximate GCD problem: given polynomially many samples from $\mathcal{D}_{\eta,\lambda}(p)$ for a randomly chosen η-bit odd integer p, output p. This is quite similar to the search-LWE problem.

To achive full homomorphism, we need to deal with the explosion of ciphertext size. Hence it requires some techniques to compress the ciphertext. Similar but more advanced techniques are given in the non-boostrapping scheme below. Finally, after applying these optimization, we invoke Gentry's program of bootstrapping, which also takes care of error control. Intuitively, boostrapping is to augment the public key pk by $Enc(pk, sk)$.

Definition 3.6. *Let $D_\mathcal{E}$ denote the decryption of the scheme $\mathcal{E}$. Let Γ be a set of gates with inputs and output in plaintext space, which includes the trivial gate. We call a circuit composed of multiple copies of $D_\mathcal{E}$ connected by a single g gate. Denote the set of g-augmented decryption circuits ($g \in \Gamma$) by $D_\mathcal{E}(\Gamma)$.*

Definition 3.7. *Let $\mathcal{E}$ be a homomorphic encryption scheme, and for every value of the security parameter λ let $C_\mathcal{E}(\lambda)$ be a set of circuits with respect to which $\mathcal{E}$ is correct. We say that $\mathcal{E}$ is bootstrappable if $D_\mathcal{E}(\lambda) \in C_\mathcal{E}(\lambda)$ holds for every λ.*

Theorem 3.8. *(Leveled FHE). There is an efficient and explicit transformation that given a description of a bootstrappable scheme $\mathcal{E}$ and a parameter $d = d(\lambda)$, outputs a description of another encryption 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 $\mathcal{E}$),
2. $\mathcal{E}^{(d)}$ is homomorphic for all circuits of depth up to d. Moreover, $E^{(d)}$ is semantically secure if $\mathcal{E}$ is.

$\mathcal{E}^{(d)}$ is a key-generation-modified version of $\mathcal{E}$: $\mathsf{KeyGen}(\lambda, d)$: First use $\mathsf{KeyGen}(\lambda)$ to generate d pairs of keys ($(sk_i, pk_i) \xleftarrow{R} \mathsf{KeyGen}(\lambda)$ for $i = 1, \cdots, d$). Let $\bar{sk}_{ij} \xleftarrow{R} \mathsf{Enc}(pk_{i-1}, sk_{ij})$ for $j = 1, \cdots, l$ where l is a polynomial in λ, and sk_{ij} are representations of sk_i in the plaintext space. Output $sk^{(d)} \leftarrow sk_0$ and $pk^{(d)} \leftarrow (\langle pk_i \rangle, \langle sk_{ij} \rangle)$.

[1]We use $[]_q$ to denote the modulo q operation

3.3.3 A FHE without Bootstrapping

One of the first FHE scheme that does not use bootstrapping is provided by Brakerski-Valkuntanathan scheme [BV11] based on lattice like $\mathbb{Z}_q[x]/\langle f\rangle$. Here we apply the tools in their papers to a mimicking scheme based on quadratic Regev's scheme on $\mathbb{Z}_q^n$, notes from Halevi [Hal], and works from [Gen09, BV11, BGV12, Bra12].

$\mathsf{KeyGen}(\lambda = n)$: Choose $A \in_R \mathbb{Z}_q^{n\times m}$, $\vec{s'} \in_R \mathbb{Z}_q^n$, and $\vec{e} \sim \mathcal{N}(0, \alpha q)^m$. Set $\vec{a'} = -\vec{s'}^t A + 2\vec{e}$ mod q, $P^t = (A^t|\vec{a'}^t)$ (hence $P \in \mathbb{Z}_q^{(n+1)\times m}$), and $\vec{s} = (1|\vec{s'}) \in \mathbb{Z}_q^{n+1}$. Finally we set public key $pk = P$, and secret key $sk = \vec{s}$[2].

$\mathsf{Enc}(pk, m \in \{0,1\})$[3]: Denote $\vec{b} = (m, 0\cdots 0)^t \in \mathbb{Z}_q^{n+1}$. Choose $\vec{r} \in \{0,1\}^m$, and output $\vec{c} \leftarrow P\vec{r} + \vec{b}$, which is a vector in $\mathbb{Z}_q^{n+1}$.

$\mathsf{Dec}(sk, \vec{c})$: Output $m' \leftarrow [\langle \vec{s}, \vec{c}\rangle]_q = [2\langle \vec{e}, \vec{r}\rangle + \langle \vec{s}, \vec{b}\rangle]_q$.

We explicitly write the $\mathsf{Eval}(pk, C, \vec{c_1}, \cdots, \vec{c_t})$ for bit product and bit addition on ciphertexts $\vec{c_1}, \vec{c_2}$: $C_1 \equiv \mathsf{Eval}(pk, +, \vec{c_1}, \vec{c_2}) = \vec{c_1} + \vec{c_2}$ since decryption is linear; $C_2 \equiv \mathsf{Eval}(pk, \times, \vec{c_1}, \vec{c_2}) = \vec{c_1} \otimes \vec{c_2}$ as a cross product. To decrypt C_1, C_2, simply output $[[\vec{s}C_i\vec{s}^t]_q]_2$. The case for addition is trivial, here we show the decryption of C_2:

$$[[\vec{s}(\vec{c_1} \otimes \vec{c_2})\vec{s}^t]_q]_2 = [[\langle \vec{s}, \vec{c_1}\rangle \cdot \langle \vec{c_2}, \vec{s}\rangle]_q]_2 = [[(2\langle \vec{e}, \vec{r_1}\rangle + m_1)(2\langle \vec{e}, \vec{r_2}\rangle + m_2)]_q]_2 = [m_1 m_2]_2.$$

If we define an extended secret key and an extended ciphertext as $\vec{s}^* = \mathsf{vec}(\vec{s} \otimes \vec{s})$, $\vec{c}^* = \mathsf{vec}(\vec{c_1} \otimes \vec{c_2})$, then the above decryption can be rewritten as $[[\langle \vec{s}^*, \vec{c}^*\rangle]_q]_2$.

$\mathsf{Dimension\text{-}Reduction}$: To be able to keep multiplying without the "dimension explosion", we would like to publish some information to allow anyone to convert "extended ciphertexts" (that can be decrypted by "extended secret keys") into "normal ciphertexts" that require only "normal secret key" to decrypt. This can be thought of as a form of proxy re-encryption: we want to publish some information $P(\vec{s}^* \to \vec{t})$ that allows anyone to convert a ciphertext under $\vec{s}^*$ into a ciphertext under $\vec{t}$, without breaking semantic security. In our case it is important that the dimension of $\vec{t}$ is much smaller than the dimension of $\vec{s}^*$. Let us denote the dimensions of $\vec{s}^*$, $\vec{t}$ to be n_1, n_2. The above idea can be formalized as follows. We publish a pseudorandom matrix $P(\vec{s}^* \to \vec{t}) = (\vec{d}^t|D^t) \in \mathbb{Z}_q^{n_1\times n_2}$, where $D \in_R \mathbb{Z}_q^{(n_2-1)\times n_1}$, and $\vec{d} = [-\vec{t}^t D + 2\vec{e} + \vec{s}^*]_q \in \mathbb{Z}_q^{n_1}$, where $\vec{e} \sim \mathcal{N}(0, \alpha q)^{n_1}$. Then for any vector $\vec{c_1} \in \mathbb{Z}_q^{n_1}$, can set $\vec{c_2} = P(\vec{s}^* \to \vec{t}) \cdot \vec{c_1} \in \mathbb{Z}_q^{n_2}$. Moreover,

$$[\langle \vec{t}, \vec{c_2}\rangle]_q = [\vec{t} \cdot P(\vec{s}^* \to \vec{t}) \cdot \vec{c_1}]_q = [\langle 2\vec{e} + \vec{s}^*, \vec{c_1}\rangle]_q,$$

and hence $[[\langle \vec{t}, \vec{c_2}\rangle]_q]_2 = [[\langle \vec{s}^*, \vec{c_1}\rangle]_q]_2$, if $\vec{c_1}$ is a short vector. We can fix the case where $\vec{c_1}$

[2]Note that both $\vec{s}$ and $\vec{s}P$ mod $q = 2\vec{e}$ are short vectors.
[3]Again, encryption is bit-by-bit.

is a long vector by bit decomposing of $\vec{c_1} = \sum_{i=0}^{l-1} 2^i \vec{c^i}$, and take powers of $\vec{s}^*$:

$$\begin{aligned}\mathsf{BitDecomp}(\vec{c_1}) &= (\vec{c^0}|\vec{c^1}|\cdots|\vec{c^{l-1}}) \in \{0,1\}^{n_1 \cdot l},\\ \mathsf{Powers2}_q(\vec{s}^*) &= (\vec{s}^*|[2\vec{s}^*]_q|\cdots|[2^{l-1}\vec{s}^*]_q) \in \mathbb{Z}_q^{n_1 \cdot l}.\end{aligned}$$

where $l = \lceil \log q \rceil$. Next modify the original transform matrix $P^t = (\vec{d^t}|D^t)$ by sampling $D \in_R \mathbb{Z}_q^{(n_2-1)\times(n_1 \cdot l)}$ and setting $\vec{d} = [-\vec{t^t} D + 2\vec{e} + \mathsf{Powers2}_q(\vec{s^*})]_q$. The last step is to reset $\vec{c_2} = [P(\vec{s}^* \to \vec{t}) \cdot \mathsf{BitDecomp}(\vec{c_1})]_q$.

Modulus-Switching: Another requirement for doing more multiplications is to keep the noise in the message small. The noise control can be a [Gen11], for two odd moduli p, q, $\vec{c} \in \mathbb{Z}_q^n$, we can switch modulus by setting i-th entry of $\vec{c'} \in \mathbb{Z}^n$ to be either rounded up or down of $\frac{p}{q} \cdot c_i$, so that $\vec{c'} = \vec{c} \bmod 2$.[4]

With the above two techniques, our Regev-like quadratic LWE scheme can compute circuits that correspond to arbitrary high-order polynomials, achieving the fully homomorphism.

3.3.4 Implementation and Performance

Efficiency is a major issue in Gentry's blueprint. The per-gate computation has poor performance, with overhead of $p(\lambda)$, as a polynomial in the security parameter. Bruce Schneier pointed out that "Gentry estimates that performing a Google search with encrypted keywords - a perfectly reasonable simple application of this algorithm - would increase the amount of computing time by about a trillion".

As for real implementation, HElib (Homomorphic Encryption Library) is a software library on github that implemented HE under the Brakerski-Gentry-Vaikuntanathan scheme [BGV12], which is more efficient than Gentry's [Gen09] as the latter requires an ideal-lattice based PKE[5], along with optimization tools such as bootstrapping, batching, Smart-Vercauteren ciphertext packing, Gentry-Halevi-Smart optimizations.

In order to have some idea of the performance of FHE, we experimented with HElib to compute the component-wise addition and multiplication on two vectors both encrypted and unecrypted. The throughput (operation per second) is shown in Table 1 and 2.

Modulus	Addition per Second	Addition per Second(FHE)	Ratio
2	786885	2437	322.89
5743	609756	153	3985
65537	790909	792	998

Table 1: The performance of addition under encrypted data and initial data.

[4]For more details, see [Gen11].

[5]Construction of an efficient ideal-lattice based PKE is still an open problem

Modulus	Multiplication per Second	Multiplication per Second(FHE)	Ratio
2	958083	552	1735
5743	518134	102	5079
65537	650062	538	1208

Table 2: The performance of multiplication under encrypted data and initial data.

4 Functional Encryption

Another encryption concept that is closely related to homomorphic encryption is functional encryption. The idea of functional encryption came from a generalization of Identity Based Encryption and Attribute Based Encryption [SW05][GPSW06]. The formal definition of functional encryption was first given by Boneh, Sahai and Waters in [BSW11]. O'Neil also contributed to the formalization of functional encryption and its security definition in [ON10].

Imagine two scenarios.

- An email user wants his email service to filter out spam emails according to some policies the user specifies. However, he does not want to release the content of his emails.
- Inside a large corporation, one wants to share data with a certain set of people.

The current public key encryption scheme would fail on both. In the first scenario, the email server needs to get the user's secret key to run spam filtering algorithm on the data, but the secret key will release the original message to the server completely. In the second scenario, the person who wants to share data need to exchange public/private key pairs with everyone he wants to share data with, which would be impossble for data sharing in large communities. The problem is that current public key encryption schemes only allow for "all or nothing" decryption through a single secret key - one can either decrypt the whole message, or know nothing.

Such problems motivate the emergenge of function encryption. The difference between functional encryption and traditional public key encryption is the addition of a *keygen* algorithm which generates secret keys allowing computaiton of certain functions on the plaintexts. The owner of the data holds a master secret key, which will be used to generate secret decryption key associated with functions. If the owner wants to grant another party the access to $f(m)$ without releasing m, it uses its master secret key to generate a function associated decryption key sk_f and passes the key to the . The party can then calculate $f(m)$ using the secret decryption key, but it would learns nothing else about m.

Functinal encryption and homomorhpic encryptions both support some form of computation on plaintexts, they supports very different functionalities. To illustrate their difference, consider the example of spam filter. With functional encryption, the user can grant the email server the ability to run algorithm f on his messages by giving the server the secret key sk_f associated with f. The email server can compute $f(m)$ for a message

m to filter spam. However, using homomorphic encryption, the email server can only compute $Enc(f(m))$. The server needs the secret key to decrypt $f(m)$, but that will release m to the server as well.

4.1 Definition

The formal definition is given below. The definition comes from [BSW11].

Definition 4.1. *Functional encryption for a functionality F defined over (K,X) is a tuple of the four probabilistic polynomial time (PPT) algorithms $(setup, keygen, enc, dec)$ satisfying the following correctness condition for all $k \in K$ and $x \in X$. K is referred to as the key space of the functional encryption.*

- $(pp, mk) \leftarrow setup(1^\lambda)$
  A public key pp and master secret key mk pair is generated by a master key generation algorithm that takes as input a security parameter 1^λ.

- $sk \leftarrow keygen(mk, k)$
  A function-specific secret key sk is generated by a key generation algorithm that takes as input the master secret key mk and a function parameter k.

- $c \leftarrow enc(pp, x)$
  The ciphertext c is generated by the encryption algorithm that takes as input the public key pp and the plaintext message k.

- $y \leftarrow dec(sk, c)$
  A function of the message y specified by parameter k is generated by the secret key sk and the ciphertext.

Taking an example from [site paper], let $K = \{1, \epsilon\}$ and F be defined over (K, X) for some plaintext space X such that:

$$F(k,x) = \begin{cases} x, & \text{if } k = 1. \\ len(x), & \text{if } k = \epsilon. \end{cases} \tag{1}$$

4.2 Security

Given the definition of functional encryption, a natural question that arises is how to define the security of a functional encryption scheme. Informally, a FE scheme is secure if given a set of secret keys, any third party only learns the function of the plaintext but nothing else about the plaintext. Despite the simplicity of the informal statement, formalizing the notion of security is surprisingly difficult. Definitional issues of functional encryption and its security were first demonstrated in [BSW11] and [ON10], where the authors showed that a natural game-based definition would fail and suggested a simulation-based definition. These definitions will be summarized in the following section.

4.2.1 Game-based Definition

We begin with a natural game-based definition given in [BSW11], which is similar to the Chosen Plaintext Attack (CPA) security. A challenger who wants to prove the security of an encryption scheme will play a game with the adversary. The game has the following stages:

1. Setup: $(pp, mk) \leftarrow setup(1^\lambda)$
2. Query: The adversary submits key queries k_i. The challenger replies with corresponding decryption keys $\{sk_{k_i}\}$.
3. Challenge: The adversary suggests a message challenge pair $m_0 = m_1$ which satisfies
$$F(k_i, m_0) = F(k_i, m_1) \forall i. \tag{2}$$
4. Reply: The challenger randomly select a bit $b \in \{0, 1\}$ and replies to the adversary with $Enc(pp, m_b)$.
5. Query Again: The adversary continues to issue key queries subject to (2).
6. Decide: The adversary outputs a bit $b' \in \{0, 1\}$.

The adversary wins if $b = b'$. And define:

Definition 4.2. *A FE scheme is secure if for all probablistic polynomial time (PPT) adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins is $\frac{1}{2} + \epsilon$ where ϵ is negligible.*

The main difference of this definition from CPA security is the non-triviality contraint (2) for the message challenge pair. (2) is necessary under a functional encryption scheme. Otherwise if $F(k_i, m_0) \neq F(k_i, m_1)$, the adversary can calculate $F(k_i, m_b) = Eval(s_{k_i}, Enc(pp, m_b))$ and ouputs b.

However, the non-triviality constraint is sometimes overly restrictve, rendering the insufficiency of Definition 4.2 to capture security notions for certain FE schemes. A clearly insecure FE scheme can be still proven to be secure under Definition 4.2. The constrait requires that for all key queries submitted by the adversary has to satisfy $F(k_i, m_0) = F(k_i, m_1)$. If the functionality F only provides functions that have different values for different plaintexts, the constraint (2) requires $m_0 = m_1$. Then the adversary cannot distinguish (m_0, m_1) at all for any encryption. Therefore, every encryption, including the clear-text encryption $Enc(pp, m) = m$ is considered secure in this case and Definition 4.2 completely fails.

4.2.2 Simulation-based Definition

To address the insufficiency of the game-based definition, [BSW11] and [ON10] proposed an alternative security definition in the simulation paradigm. There are some subtle differences in their definition notions, but all versions follow the same scheme. We will introduce a definition given in [?] to give a sense how simulation-based security definition work. Intuitively, in the real world, the adversary has access to the public key, the decryption keys and the ciphertext while in the ideal world it should only get $F(k_i, m)$. Under a secure functional scheme, these two cases should not be able to be distinguished. The formal definition is given below.

Real Distribution

1. $(pp, mk) \leftarrow Setup(1^\lambda)$
2. $k_i \leftarrow A_1$, $sk_{k_i} \leftarrow keygen(pp, k_i)$
3. $(m_1, m_2, \ldots, m_n) \leftarrow M$
4. $c_i = enc(pp, m_i)$
5. $k_i \leftarrow A_2$, $sk_{k_i} \leftarrow keygen(pp, k_i)$
6. $\alpha \leftarrow A_3(pp, c_1, \ldots, c_n, sk_{k_1}, \ldots, sk_{k_l})$
7. output $(\alpha, pp, m_1, \ldots, m_n, k_1, \ldots, k_l)$

Ideal Distribution

1. $(pp, mk) \leftarrow Setup(1^\lambda)$
2. $k_i \leftarrow A_1$, $sk_{k_i} \leftarrow keygen(pp, k_i)$
3. $(m_1, m_2, \ldots, m_n) \leftarrow M$
4. $c_i = enc(pp, m_i)$
5. $k_i \leftarrow A_2$, $sk_{k_i} \leftarrow keygen(pp, k_i)$
6. $\alpha \leftarrow S(pp, f(k_1, x_1), \ldots, f(k_i, x_j), \ldots, f(k_l, x_n))$
7. output $(\alpha, pp, m_1, \ldots, m_n, k_1, \ldots, k_l)$

We brieftly explain the process here. First the public/secret master key setup is generated. Then an adversary algorithm A_1 can issue key requests to *keygen* and receives the corresponding secret keys. Note that A_1 do not need to generate all at once and can generate key issues adaptively. A message generator algorithm M then generates message m_i and encryptes them. After seeing the ciphertext, another adversary algorithm A_2 can adaptively generates more key requests. Finally, in the real distribution, an adversary algorithm A_3 generates a random number α given the public key, the ciphter texts and all the secret keys while in the ideal distribution, a simulator generates α only given the function values $f(k_i, x_j)$.

Definition 4.3. *A functional encryption FE is secure if for any PPT adversary algorithm A_1, A_2, A_3, M, there exists a simulator algorithm S such that real distribution and ideal distribution is indistinguishable.*

Definition 4.3 is referred to as the adaptive security because the adversary can generate key requests adatively after it sees the ciphertexts. In the non-adative version, the adversary must generate all key requests before it sees the ciphertexts. Many construction of functional encryption schemes are only proved non-adaptively secure, as we will introduce later.

4.3 Functional Encryption Schemes

The elegant definition of functional encryption captures many other important encryption schemes. In this section, we will introduce two subclasses of functional encryption.

4.3.1 Identity Based Encryption

Identity based encryption (IBE) is a type of public key encryption where any string can serve as public keys, for example, email addresses, social security numbers, locations, IP addresses, or a combination of the above. Public keys in IBE are often referred to as identities. The advantage of IBE over tradictional public key encryption is that it removes the need for generating random public/private key pairs and the trust of a third party for key management.

We illustrate how IBE works in an example of email services using IBE. Alice wants to send a message to Bob, she encrypts the message with Bob's identity (e.g. email address) as the public key. After receiving the message, Bob contacts a trusted key generation source and verifies himself to retrieve his secret key. This step is only required once if Bob publishes a consistent identity as his public key. Otherwise, Alice could concatenate the date or Bob's other identities in addition to his email. Bob would then have to fetch a different secret key each time in order to decrypt the message.

The notion of IBE was first introduce by Adi Shamir in 1985[Sha85]. Shamir proposed a signature scheme and speculated the existence of an encryption scheme. The first practical encryption system was published in 2001 by Dan Boneh and Matthew Franklin. [BF03] The scheme used Weil pairings over elliptic curves and finite fields. Many others have also proposed IBE schemes: [GPV08, CHKP10, ABB10].

Now we give a formal definition of IBE under the notion of functional encryption. For a functional encryption scheme with public key pp, when the encryptor wants to encrypt message m with identity id, it contatenates m with id and encrypts the whole

$$c = Enc(pp, (id, m))$$

The key space for the FE is the set of identities id and the functionality F is defined as follows:

$$F(id, (id', m)) = \begin{cases} m & \text{if } id = id' \\ \perp & \text{if } id \neq id'. \end{cases}$$

where $\perp$ means a randomly generated message.

4.3.2 Attribute Based Encryption

In [SW05], Amit Sahai and Brent Waters extended the concept of IBE that allows more complex data access control. As they proposed, messages are encrypted with identity w and can be decrypted with identity w' if and only if w and w' are similar based on some "set overlap" distance metric. They extended the notion of identity to be a set of arbitrary attributes such as name, age, address, etc. Unlike IBE which only allows decryption when the encryption identity and decryption identity matches perfectly, the new scheme allows for some error tolerance and hence, it is named "fuzzy" identity based encryption. The error tolerant property of the encryption allows for some interesting applications such as using biometric identities as the keys for encryption.

Goyal, Pandey, Sahai and Waters [GPSW06] formalized the above idea to two types of ABE: Key-Policy ABE and Ciphertext-Policy ABE. Ciphertext policy ABE schemes are decribed in [BSW]. For Key-Policy ABE, every recipient are associated with a set of attributes. The user sending the message can specify a policy for which only receivers with attributes satisfying the policy can decrypt. For example, a MIT student wants to share a document with all college students satisfy the following requirement:

("Class of 2015" OR "Major Computer Science") AND (NOT "harvard")

He can encrypt the message with a boolean formula

$$\phi(x_1, x_2, x_3) = (x_1 || x_2) \& (!x_3)$$

where x_1, x_2, x_3 denotes the three attributes.

Every user will be assigned with an attribute vector describing his attributes. For example, a MIT student majoring in Biology in the class of 2015 will have an attribute vector $(true, false, true)$. When receiving a message, the recipient contacts the key generation source to submit his attribute vector v. The key generation source verifies the user and responds to him with a secret key sk_v corresponding to v. Then, the user can use sk_v to decrypt the message. The decryption algorithm ensures that $Dec(sk_v, c) = m$ if and only if $\phi(sk_v) = true$. Otherwise, it will return a random message.

We formalize the definition of ABE under the concept of functional encryption. If an encryptor wants to encrypt the message m and specify a policy ϕ, it encrypts

$$c = Enc(pp, (\phi, m)).$$

The key space of the FE is defined as all possible attribute vectors. For example, if there are n attributes, then the size of the key space will be 2^n. The functionality F is defined as

$$F(v, (\phi, m)) = \begin{cases} m & \text{if } \phi(v) = 1 \\ \perp & \text{if } \phi(v) = 0. \end{cases}$$

5 Conclusion

Being able to compute on encrypted data brings about new applications that have never been imagined before. Email filtering, secure cloud computing, and software obfuscation are some of the applications that are currently in development. The field of computing on encrypted data still has many challenges. No efficient implementation of FHE has been developed because of the noise built up from performing the operations. Some have proposed encryption schemes that have bounded depth but does not apply to more complicated functions. FE is similar to FHE but gives more access control and can be applied to multiple different scenarios. We have provided an overview of the accomplishments of computing on encrypted data and hope to see future advancements in the field.

6 Acknowledgements and References

6.1 Acknowledgements

We would like to thank Professor Ron Rivest for the initial idea and our TA Justin for his feedback on our project proposal. We would also love to thank all the scholars who devoted themselves in this field to provide us all the information.

6.2 References

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In EUROCRYPT, 2010, pp.553572.
[Bra12] Zvika Brakerski, Fully homomorphic encryption without modulus switching from classical gapsvp, Advances in Cryptology - CRYPTO'12, Lecture Notes in Computer Science, vol.7417, Springer, 2012, pp.868-886.
[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. SIAM J. Comput., 32(3), 2003.
[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, (leveled) fully homomorphic encryption without bootstrapping, Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, ACM, 2012, pp.309-325.
[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Denitions and challenges. In TCC, 2011.
[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. Manuscript, 2011.
[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In EUROCRYPT, pages 2443, 2010.
[Gen09] Craig Gentry, A fully homorphic encryption scheme. (PhD thesis) Stanford University, 2009.
[Gen10] Craig Gentry, Computing arbitrary functions of encrypted data, Communications of the ACM, v.53 n.3, March 2010.
[Gen11] Craig Gentry. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive: 2011/277.
[GGH97] Oded Goldreich, Sha Goldwasser, and Shai Halevi. Eliminating decryption errors in the ajtai-dwork cryptosystem. In Burton S. Kaliski Jr., editor, CRYPTO, volume 1294 of Lecture Notes in Computer Science, Springer, 1997, pp.105111..
[GH11] Craig Gentry and Shai Halevi, Fully Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits. FOCS 2011, pp.107-116.
[GKPVZ12] Sha Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan and Nickolai Zeldovich. "Reusable Garbled Circuits and Succinct Functional Encryption." Cryptology ePrint Archive, Report 2012/733.
[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions, STOC'08, pp.197206.
[LMSV10] On CCA-Secure Fully Homomorphic Encryption, Cryptology ePrint Archive: 2010/560

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. Advances in Cryptology, EUROCRYPT 2012, pages 700-718, Heidelberg, Germany, 2012. Springer.
[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, STOC'05, pp. 84-93.
[Rot11] Ron Rothblum, Homomorphic encryption: From private-key to public-key, Theory of Cryptography (Yuval Ishai, ed.), Lecture Notes in Computer Science, vol. 6597, Springer Berlin Heidelberg, 2011, pp.219-234.
[RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, Academic Press, 1978, pp.169177.
[Sha85] A. Shamir. Identity-based cryptosystems and signature schemes. In CRYPTO 1984, vol. 196 of LNCS, 1985, pp.4753.
[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In EUROCRYPT, 2005, pp.457-473.