Learning Without Knowing: Applying Homomorphic Encryption to Machine Learning Algorithms

Acknowledgements

This thesis would not have been possible without the support of my advisor, Professor Boaz Barak. I've received plenty of advice from him with regards to both the technical material and presentation of said material in this thesis. As I've discovered throughout writing this thesis, homomorphic encryption is a well-studied field and I am very thankful for Boaz's assistance in navigating the literature in this area.

I am also grateful to Professor Noam Elkies for taking the time to provide some feedback on my drafts in spite of his busy schedule and having to attend to his actual thesis advisees.

Special thanks also go to my friends and classmates for the many intellectual (and not-so-intellectual) conversations that have contributed to my academic and personal growth.

Last but not least, I'd also like to thank my parents for their continued support in all my endeavours. Writing this thesis has been a very rewarding experience for me, and I am very grateful that they encouraged me to do so.

Contents

Chapter 1

Introduction & Background

1.1 Introduction

This thesis will examine applications of homomorphic encryption to machine learning. On a high level, homomorphic encryption gives us the ability to compute functions over ciphertexts instead of having to decrypt and then compute the functions over the decrypted plaintexts. The fact that we are even able to achieve this is quite astounding, and has many practical applications. One of these practical applications is machine learning, in which a goal is often to learn some model based on large datasets. This necessitates computations that make use of the data, which can be challenging if the data is encrypted. There are many settings in which aggregated data is (or should be) stored in some central server in encrypted fashion. For instance, user data on social networks, student data for schools, and patient data in hospitals are all instances where it seems reasonable to need to perform computations over aggregated data while needing to keep the data private.

Without any homomorphic encryption schemes, we'd have to download and decrypt the data before performing computations. This solution, while correct, fails to leverage the computational power that these central servers often provide us. Alternatively, we could store the data unencrypted and rely on other measures like permissions for data security, or equivalently store the secret key on the server for it to use when computations are necessary. This solution is unsatisfactory because permissions can be changed (as we note in an example later) and the data is not secure if an adversary manages to acquire it. Thus, homomorphic encryption provides a way to store data that leverages both the computational power of these servers while maintaining privacy of the data.

In this chapter (Chapter 1), we go over some cryptography preliminaries before presenting the theory of homomorphic encryption, as well as describe some of the machine learning algorithms that we will later see implemented using a homomorphic encryption scheme. Afterwards, in Chapter 2, we will go over some constructions of homomorphic encryption schemes. We will also see an important reduction from a commonly used lattice-based problem (the shortest vector problem) to learning with errors. In Chapter 3, we will examine some implementations of machine learning algorithms that allow them to be carried out by these homomorphic encryption schemes. Finally in Chapter 4, we examine the idea of homomorphic encryption more critically and present some alternatives to fully homomorphic encryption.

1.1.1 Guide for Those With Cryptography Background

For those with an extensive background in cryptography, perhaps even homomorphic encryption, most of Chapter 1 may be skipped. That said, if one has not seen homomorphic encryption before, Chapter 1, Section 3 contains an introduction to the subject matter and a good analogy by Gentry [Gen10]. Furthermore, an introduction to the machine learning algorithms we will explore in this thesis can be found in Chapter 1, Section 4, and may be helpful to those without machine learning background.

1.1.2 Guide for Those With Machine Learning Background

For those with a machine learning background interested in learning cryptography, spending time in Chapters 1 and 2 will be useful to get a better understanding of cryptography. On the other hand, for those only curious about how existing machine learning algorithms can be tailored to work on homomorphic encryption schemes, then Chapter 3 should be sufficient.

1.1.3 Guide for Those Neither Cryptography nor Machine Learning Background

For those who are completely new to the subject matter of this thesis, I recommend spending time understanding the concepts covered in Chapter 1: Introduction, and going through references like [KL07] for Cryptography, and [Bis07] or [Mur12] for the Machine Learning.

1.2 Cryptography Preliminaries

Before we proceed to defining homomorphic encryption, we give a brief introduction to cryptography. For a more thorough treatment of these concepts, we refer the reader to sources like [KL07].

At its most basic, an encryption scheme needs an encryption algorithm (to encrypt), and a decryption algorithm (to decrypt). However, an important principle in cryptography is that the security of a particular encryption scheme should not be based on the secrecy of the encryption/decryption algorithms. Indeed, we assume that adversaries know all the details of these algorithms. Instead, security comes from the fact that the encryption and decryption algorithms are keyed (i.e. take as input a key k which influences their action on the message/ciphertext respectively). This key is generated randomly according to some key generation algorithm `KeyGen`, and typically kept secret. In encryption schemes, this is where the security comes from. Although the adversary knows all the algorithms that are being used, they do not know the particular secret key being used in a given instantiation of the encryption scheme.

As a motivating example for why keys are important, consider the one-time pad, where a message $m \in \{0,1\}^n$ is encrypted by computing the ciphertext $c = m \oplus k$ for some key $k \in \{0,1\}^n$. If k is fixed, this will not be secure because any adversary knows k (it is hardwired into the encryption algorithm) and can compute $m = c \oplus k$. However, if k is chosen uniformly at random from $\{0,1\}^n$, denoted $k \xleftarrow{R} \{0,1\}^n$ and kept secret from the adversary, then the resulting ciphertext will also be uniformly distributed in $\{0,1\}^n$, achieving perfect security.

Although we will keep the length of most keys and messages abstract throughout the thesis, note that key sizes currently tend to be in the 128-256 bit range. For instance, the Advanced Encryption Standard (AES) uses keys that are 128, 192, or 256 bits. Now, we more formally define the notion of a private (symmetric) key encryption scheme.

Definiton 1.2.1: A private-key encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ with key space $\mathcal{K}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$ is a 3-tuple of efficient algorithms where $\texttt{KeyGen} : \lambda \to \mathcal{K}$, $\texttt{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$, and $\texttt{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$. For correctness, we require that $\texttt{Dec}_k(\texttt{Enc}_k(m)) = m$

The key space $\mathcal{K}$ and message space $\mathcal{M}$ often set to be around 128-bit strings. In this thesis, we will often take our message space to be $\mathbb{Z}_q$, the integers modulo q, which is valid because we can identify these bit strings with integers provided that q is large enough. The parameter λ taken in by $\texttt{KeyGen}$ is called the security parameter. We need it because we want our encryption and decryption algorithms to be efficient (i.e. run in polynomial time), which in this case, means polynomial in λ.

Note that we do not require $\texttt{Enc}$ to be a deterministic algorithm. Indeed, a deterministic encryption function $\texttt{Enc}$ has inherent limitations on the security it can provide as we will see later.

1.2.1 Security Definitions for Private-Key Cryptography

Before discussing security, we must formally define what it means to be secure. The security of a cryptographic scheme is judged by whether or not an efficient adversary can successfully distinguish the encryptions of two plaintexts in a given security game. For instance, the security game for EAV-security (referred to as security in the presence of an eavesdropper in [KL07]) is:

Definiton 1.2.2: The EAV-security game, denoted $\texttt{PrivK}^{\texttt{eav}}_{\mathcal{A},\mathcal{E}}(n)$ for a private-key encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ is as follows:

1.) The adversary $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$

2.) A key k is generated using $\texttt{KeyGen}$, unknown to the adversary. Also, a random bit $b \in \{0, 1\}$ is chosen to select m_0 or m_1. We compute $c = \texttt{Enc}_k(m_b)$ and give c to $\mathcal{A}$ as the challenge ciphertext

3.) $\mathcal{A}$ outputs a bit b' with the aim of having $b' = b$

4.) The output of the experiment is 1 if $b' = b$ and 0 otherwise. We refer to the former case as $\mathcal{A}$ succeeding and denote it as $\texttt{PrivK}^{\texttt{eav}}_{\mathcal{A},\mathcal{E}}(n) = 1$.

With this game formalized, we then define EAV-security as follows:

Definiton 1.2.3: A private-key encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ is EAV-secure (or secure in the presence of eavesdroppers) if for all probabilistic polynomial time algorithms $\mathcal{A}$, there is a negligible function $\texttt{negl}$ such that for all n,

$$\Pr[\texttt{PrivK}^{\texttt{eav}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \texttt{negl}(n)$$

where the probability is taken over the randomness in $\mathcal{A}$ and the randomness in the experiment (i.e. choosing the key, bit b, and randomness used by $\texttt{Enc}$).

The chosen-plaintext attack (CPA)-security game is the same as $\texttt{PrivK}^{\texttt{eav}}_{\mathcal{A},\mathcal{E}}$, except that the adversary $\mathcal{A}$ has access to an encryption oracle that it can query at any time during the game. Intuitively, this encryption oracle gives the adversary more power, meaning that CPA-security is a stronger standard than EAV-security. It is easy to see that CPA-security actually implies EAV-security. We present more precise definitions below.

Definiton 1.2.4: The CPA-security game, denoted $\texttt{PrivK}^{\texttt{cpa}}_{\mathcal{A},\mathcal{E}}(n)$ for a private-key encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ is as follows:

1.) A key k is generated using $\texttt{KeyGen}$. The adversary $\mathcal{A}$, who does not know the key, is allowed to make polynomially many queries to an encryption oracle $\texttt{Enc}_k$.

2.) $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$

3.) A random bit $b \in \{0, 1\}$ is chosen. We compute $c = \texttt{Enc}_k(m_b)$ and give c to $\mathcal{A}$ as the challenge ciphertext

4.) $\mathcal{A}$ continues to have access to the encryption oracle $\texttt{Enc}_k$.

5.) $\mathcal{A}$ outputs a bit $b^{'}$ with the aim of having $b^{'} = b$

6.) The output of the experiment is 1 if $b^{'} = b$ and 0 otherwise. We refer to the former case as $\mathcal{A}$ succeeding and denote it as $\texttt{PrivK}^{\texttt{cpa}}_{\mathcal{A},\mathcal{E}}(n) = 1$.

Definiton 1.2.5: A private-key encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ is chosen-plaintext attack (CPA)-secure if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there is a negligible function $\texttt{negl}$ such that

$$\Pr[\texttt{PrivK}^{\texttt{cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \texttt{negl}(n)$$

where the probability is taken over the randomness used by $\mathcal{A}$ as well as the randomness used in the experiment.

Proposition 1.2.6: An encryption scheme $\mathcal{E}$ whose encryption function $\texttt{Enc}$ is deterministic cannot be CPA secure

Proof. Consider the following adversary $\mathcal{A}$ that plays the CPA security game as follows:

1.) Internally select two distinct random messages $m_0, m_1 \in \mathcal{M}$

2.) Use the encryption oracle to obtain encryptions $c_0 = \texttt{Enc}_k(m_0)$ and $c_1 = \texttt{Enc}_k(m_1)$.

3.) Output the two messages $m_0, m_1 \in \mathcal{M}$

4.) On receiving the ciphertext c, if $c = c_0$, output bit 0, otherwise output bit 1

Because $\texttt{Enc}$ is deterministic, the challenge ciphertext c, being an encryption of either m_0 or m_1, will equal either c_0 or c_1. Thus, $\mathcal{A}$ will be able to succeed with non-negligible probability over 1/2 (in particular, $\mathcal{A}$ will succeed always). Thus, $\mathcal{E}$ cannot be CPA secure. □

Finally, chosen-ciphertext attack (CCA)-security allows the adversary both an encryption and decryption oracle that he can query at any time during the game. Other texts distinguish between non-adaptive chosen-ciphertext attacks and adaptive chosen-ciphertext attacks. The former, also called lunchtime attacks, only allow the adversary to use the decryption oracle before receiving the challenge ciphertext. The latter, a more powerful attack model, allows the adversary to continue using the decryption oracle after receiving the challenge ciphertext, but obviously, he is not allowed to query the decryption oracle with the challenge ciphertext after receiving it. When we refer to CCA attacks in this thesis, we mean the latter. As the reader might guess, CCA-security is a stronger standard than CPA- or EAV-security and implies them both. We present formal definitions below.

Definiton 1.2.7: The CCA-security game, denoted $\texttt{PrivK}^{\texttt{cca}}_{\mathcal{A},\mathcal{E}}(n)$ for a private-key encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ is as follows:

1.) A key k is generated using $\texttt{KeyGen}$. The adversary $\mathcal{A}$, who does not know the key, is allowed to make polynomially many queries to an encryption oracle $\texttt{Enc}_k$ and a decryption oracle $\texttt{Dec}_k$.

2.) $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$

3.) A random bit $b \in \{0, 1\}$ is chosen. We compute $c = \texttt{Enc}_k(m_0)$ and give c to $\mathcal{A}$ as the challenge ciphertext

4.) $\mathcal{A}$ continues to have access to the encryption oracle $\texttt{Enc}_k$ and decryption oracle $\texttt{Dec}_k$, now with the restriction that $\mathcal{A}$ cannot run the decryption oracle on the challenge ciphertext c

5.) $\mathcal{A}$ outputs a bit b' with the aim of having $b' = b$

6.) The output of the experiment is 1 if $b' = b$ and 0 otherwise. We refer to this as $\mathcal{A}$ succeeding and denote it as $\texttt{PrivK}^{\texttt{cca}}_{\mathcal{A},\mathcal{E}}(n) = 1$.

Definiton 1.2.8: A private-key encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ is chosen-ciphertext attack (CCA)-secure if for all probabilistic polynomial time adversaries $\mathcal{A}$, there is a negligible function $\texttt{negl}$ such that

$$\Pr[\texttt{PrivK}^{\texttt{cca}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \texttt{negl}(n)$$

where the probability is taken over the randomness used in the experiment.

1.2.2 Public-Key Cryptography

Of course, one drawback of private-key encryption schemes is the fact that a key needs to be shared between the two parties before any encrypted messages can be sent. Indeed, we have this chicken-and-egg like situation wherein the perfect method to securely transmit a secret is itself, a private-key encryption scheme, which in turn needs a secret key to be shared. Thus, the invention of the public-key encryption

scheme was a significant milestone in cryptography. The basic idea is that we have a public key, and secret key. The public key is used for encryption, allowing anyone to send messages to the holder of the secret key, who can use the secret key to decrypt the message. More formally,

Definiton 1.2.9: A public-key encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ with key space $\mathcal{K} = \mathcal{K}_{pk} \times \mathcal{K}_{sk}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$ is a 3-tuple of efficient algorithms where $\texttt{KeyGen} : \lambda \to \mathcal{K}$, $\texttt{Enc}_{pk} : \mathcal{M} \to \mathcal{C}$, and $\texttt{Dec}_{sk} : \mathcal{C} \to \mathcal{M}$. pk is the public key and sk is the secret key. For correctness, we require that $\texttt{Dec}_{sk}(\texttt{Enc}_{pk}(m)) = m$

There are analogous definitions for EAV, CPA and CCA-security for public-key encryption schemes. We refer the reader to [KL07] for the details.

Proposition 1.2.10: If a public-key encryption scheme has indistinguishable encryptions in the presence of an eavesdropper (i.e. is EAV-secure), it is CPA-secure.

Proof. In the public-key version of the EAV-security game, the adversary $\mathcal{A}$ has access to the public key pk. As a result, $\mathcal{A}$ can compute $\texttt{Enc}_{pk}(m)$ himself for any message m, which is equivalent to giving $\mathcal{A}$ access to some encryption oracle. As a result, for public key encryption schemes, EAV-security is equivalent to CPA-security. □

1.3 Homomorphic Encryption

1.3.1 Introduction

Now that we've defined some cryptography basics, we can proceed to outlining the theory of homomorphic encryption. In this section, we introduce the concept of homomorphic encryption both formally and through the jewelry store analogy presented in [Gen10]. Then, we discuss considerations in the security and efficiency of homomorphic encryption schemes.

Definiton 1.3.1: A homomorphic encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}, \texttt{Evaluate}, \mathcal{F})$ with key space $\mathcal{K}$, message space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and security parameter λ is a 5 tuple where $\texttt{KeyGen} : \lambda \to \mathcal{K}$ generates the keys, $\texttt{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ is the encryption function, $\texttt{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ is the decryption function, $\texttt{Evaluate} : \mathcal{F} \times \mathcal{C}^* \to \mathcal{C}$ is the evaluation function that takes a permitted function $f \in \mathcal{F}$ and evaluates them on a set of ciphertexts $(c_1, ..., c_n)$ to product another ciphertext c.

To build intuition, we draw on the analogy presented in Gentry's 2010 survey [Gen10]. Suppose we have a character Alice who owns a jewelry store. In this store, she takes raw materials and turns them into glamorous necklaces and rings. Unfortunately, Alice herself must attend to her customers and so her employees at the store have to take care of assembling the jewelry without her supervision. However, Alice distrusts her employees and is afraid that they will steal the raw materials (analogous to seeing the encrypted data).

Alice's solution to this conundrum is the following. She creates a box that only she has access to via a key. This box comes with gloves that allows people to manipulate the items in the box, but does not allow them

to take anything out of the box. Then, what Alice does is put the raw materials into the box, locks the box so its contents may not be removed, and has her employees assemble the jewelry using the box's provided gloves. At the end of the day, Alice then uses her key to open the box and retrieve the fully assembled piece of jewelry.

As Gentry notes, there are indeed shortcomings to this analogy [Gen10]. For instance, even though the workers cannot take out the materials, they can see whether or not they are present and even more, what they are, thereby not living up to security requirements in cryptography. However, we think that it sufficiently demonstrates the main idea behind homomorphic encryption.

Another, more realistic analogy, is that suppose there is a small college that keeps detailed records about all its students and how well they perform. Because the college lacks resources, it has to store all this data (encrypted) on a third-party cloud service. The college administrators want to see the average GPA of its students. However, they do not want to compromise the privacy of their students by giving the third-party cloud service the secret key. If the college uses a homomorphic encryption scheme, where taking the average is a permitted function, the third-party cloud service can compute some function of the encrypted student data, and return that to the college who can then use their decryption key to decrypt the encrypted average student GPA. Note that we are assuming that the third party cloud service follows an honest-but-curious model, meaning that it will carry out instructions correctly (i.e. compute the desired function of ciphertexts instead of a malicious function of its choosing), but if given the opportunity, will try to gain information about the data underlying the ciphertexts it stores.

However, there exists a trivial homomorphic encryption scheme that satisfies our definition of homomorphic encryption, where the `Evaluate` function just appends the function description to the relevant ciphertexts, and `Decrypt` handles the function evaluation, i.e.

$$\texttt{Evaluate}(f, c_1, ..., c_n) = (f, c_1, ..., c_n) = c$$
$$\texttt{Decrypt}(c) = f(\texttt{Decrypt}(c_1), ..., \texttt{Decrypt}(c_n))$$

Note that this is no better than the client just requesting the decryptions of $c_1, ..., c_n$ and evaluating f himself. This example dodges the point of homomorphic encryption, which is to perform the computation on the server and only have to decrypt the answer. Hence, we impose the *compact ciphertexts requirement*, which asserts that the running time of decrypting c (the output of `Evaluate`) must take the same amount of computation as decrypting c_1 (one of the inputs to `Evaluate`). Furthermore, the size of c cannot be larger than c_1. Finally, the time required to decrypt c must be independent of the complexity of f, the chosen permitted function.

One nice property that compactness gives us is that we can transform a homomorphic private-key encryption scheme into a homomorphic public-key one, only sacrificing one application of the homomorphic evaluation scheme. In particular, we have the theorem below, due to [Rot11].

Theorem 1.3.2 : Any multiple-message secure private-key encryption scheme that is compactly homomorphic with respect to addition modulo 2 can be transformed into a semantically secure public-key

encryption scheme. In particular, if the private-key scheme allows $i + 1$ evaluations of the homomorphic evaluation function, then the corresponding public-key schemes allows i evaluations of the homomorphic evaluation function.

Proof. See [Rot11] for two explicit constructions that transform such private-key encryption schemes into public-key ones □

1.3.2 Security for Homomorphic Encryption Schemes

To reason about the security of the homomorphic encryption schemes we explore, we must first reason about the attack model of the third party server and decide what it is capable of doing. The schemes we discuss assume an honest-but-curious model, which states that the server or 3^{rd} party will follow instructions and conduct the protocol correctly. However, beyond these constraints, it will try and learn as much as possible about the encrypted data. Note that there are other, more adversarial models that we can consider for the server (e.g. they don't always compute the appropriate function), but the obstacles presented by these models can be overcome by using delegation of computation. On high level, delegation of computation tries to establish a protocol for a third party to compute some function and efficiently prove that it has properly computed the function, which is exactly what we would require for an adversarial server. While we don't cover the details of schemes to delegate computation, we refer the interested reader to sources like [CKV10].

One characteristic of homomorphic encryption schemes that has important implications for security is malleability. First, we formally define what it means for an encryption scheme to be malleable.

Definiton 1.3.3: An encryption scheme $\mathcal{E}$ is malleable if given the ciphertext of a message, $c = \texttt{Enc}_k(m)$, there exists a non-constant function f that is not the identity function such that an efficient adversary $\mathcal{A}$ can compute the the ciphertext of some related message $c^{'}$ such that $\texttt{Dec}_k(c^{'}) = f(m)$ with non-negligible probability.

By definition, homomorphic encryption schemes have to be malleable because we allow the computation of various functions over the ciphertexts. Unfortunately, malleable encryption schemes are inherently limited in the security they provide. Indeed, researchers have come up with non-malleable cryptosystems to provide greater security [DDN00]. However, because homomorphic encryption schemes are malleable, we have the following limitation on their security.

Proposition 1.3.4: Any malleable encryption scheme cannot be secure against adaptive chosen-ciphertext attacks

Proof. Suppose we have a malleable encryption scheme $\mathcal{E}$, where for some message m_0, and all $c \in C_m = \{c \in \mathcal{C} \mid \Pr[\texttt{Enc}_k(m_0) = c] > 0$ over the randomness of the key and $\texttt{Enc}\}$, it is known that the message $f(m)$ has ciphertext $g(c)$ for any $c \in C_m$ where f, g can be computed in polynomial time. Because f is non-constant, there exists a message m_1 such that $f(m_1) \neq f(m_0)$. Then, consider the following adversary $\mathcal{A}$:

1.) Submits m_0 to the encryption oracle, and notes down $c_0 = \texttt{Enc}_k(m_0)$

2.) Outputs messages $f(m_0)$ and $f(m_1)$

3.) Upon receiving the challenge ciphertext c, computes $g(c_0)$. If $c = g(c_0)$, then output $b = 0$. Else, flip a coin and output the result.

We can see that such an adversary would gain non-negligible advantage, and so our malleable encrytion scheme $\mathcal{E}$ is insecure against adaptive chosen-ciphertext attacks □

From this, we realize that although there exist encryption schemes, even public-key ones, that are secure against adaptive chosen ciphertext attacks [CS04], it is impossible to construct a homomorphic encryption scheme that satisfies this security requirement. That said, although homomorphic encryption schemes cannot formally meet the CCA requirement, we can augment them with methods for delegating computation that can help ensure that the server does not maliciously alter the ciphertexts it returns, which is the type of attack that motivates the desire for CCA security [CKV10]. There are also many other standards of security that homomorphic encryption schemes are able to attain, such as EAV-security and CPA-security [KL07]. The security definitions for a homomorphic encryption are the same as that for a regular encryption scheme, imposing conditions on the `KeyGen`, `Enc`, `Dec` operations, but being independent of the `Evaluate` operation.

1.4 Machine Learning Preliminaries

In this section, we outline some of the machine learning algorithms that we will see implemented later on using homomorphic encryption schemes. We examine both supervised and unsupervised learning algorithms. Supervised learning involves inferring a function from a set of labelled training data. The two types of supervised learning are classification and regression, wherein the former refers to the case where the labels form a discrete set of class labels, and the latter when the labels take on continuous values. For classification, we examine Fisher's Linear Discriminant, Naïve Bayes Classifiers, and Decision Trees. For regression, we examine linear regression. On the other hand, unsupervised learning involves inferring the hidden structure of unlabelled data. The unsupervised learning algorithm we explore is principal component analysis, which is also an important tool for dimensionality reduction.

1.4.1 Binary Classification

Binary classification refers the problem of taking a feature vector $\mathbf{x}$ and deciding which of two labels (or classes) it came from. For instance, we can take a feature vector containing a person's height and weight and have the classification problem of deciding the person's gender. There are many algorithms that address this problem, including logistic regression, support vector machines etc. In this section, we will examine Fisher's Linear Discriminant, which is a technique to perform binary classification when the features are continuous quantities.

Fisher's Linear Discriminant Analysis approaches the problem of classification by finding a vector $\mathbf{w}$ that we can project the data onto such that it is easiest to define a point that separates the two classes with large margin. To do this, we choose a $\mathbf{w}$ that maximizes the ratio between the inter-class variances and intra-class variances. Intuitively, this makes sense because we want to maximize the separation between different

classes, while minimizing the separation within each class. Once we have this vector $\mathbf{w}$, we can classify a data point $\mathbf{x}$ by computing $\mathbf{w}^T\mathbf{x}$ and comparing it to the threshold c corresponding to our separation point.

More formally, if we let $\mathbf{u}_1$ be the class-conditional means of class 1, and $\mathbf{u}_2$ be the class-conditional means of class 2, then we want to maximize S, which is defined as:

$$\begin{aligned} S = \frac{\sigma_{inter}^2}{\sigma_{intra}^2} &= \frac{(\mathbf{w}^T(\mathbf{u}_2 - \mathbf{u}_1))^2}{\left(\sum_{i,c_i=1} \mathbf{w}^T(\mathbf{x}_i - \mathbf{u}_1)\right)^2 + \left(\sum_{i,c_i=2} \mathbf{w}^T(\mathbf{x}_i - \mathbf{u}_2)\right)^2} \\ &= \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \end{aligned}$$

where $\mathbf{S}_B$ is the between-class scatter matrix calculated using

$$\mathbf{S}_B = (\mathbf{u}_2 - \mathbf{u}_1)(\mathbf{u}_2 - \mathbf{u}_1)^T$$

and if we denote the feature-label pairs as (x_i, c_i), then $\mathbf{S}_W$ is the within-class scatter matrix given by

$$\mathbf{S}_W = \sum_{i,c_i=1} (x_i - \mathbf{u}_1)(x_i - \mathbf{u}_1)^T + \sum_{i,c_i=2} (x_i - \mathbf{u}_2)(x_i - \mathbf{u}_2)^T$$

Following the presentation in [Bis07], we can differentiate with respect to $\mathbf{w}$ to obtain

$$\begin{aligned} \frac{\partial S}{\partial \mathbf{w}} &= 0 \\ \Rightarrow (\mathbf{w}^T\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} &= (\mathbf{w}^T\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w} \end{aligned}$$

Observe that $\mathbf{S}_B\mathbf{w}$ is in the direction of $\mathbf{u}_2 - \mathbf{u}_1$, and that $\mathbf{w}^T\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^T\mathbf{S}_W\mathbf{w}$ are just constants. Since we only care about the direction and not the scale factor of $\mathbf{w}$, we can omit the constants, and multiply both sides by $\mathbf{S}_W^{-1}$ to obtain the optimal solution for $\mathbf{w}$

$$\mathbf{w}^* \propto \mathbf{S}_W^{-1}(\mathbf{u}_2 - \mathbf{u}_1)$$

If we want to classify, we need to calculate the threshold c such that $c = \mathbf{w}^{*T}(\mathbf{u}_1 + \mathbf{u}_2)/2$ where $(\mathbf{u}_1 + \mathbf{u}_2)/2$ is the midpoint between the two means in the direction given by $\mathbf{w}$. Afterwards, given a point $\mathbf{x}$, we calculate $\mathbf{w}^{*T}\mathbf{x}$ and classify depending on whether this is greater or less than c.

1.4.2 Linear Regression

Linear regression is one of the most fundamental algorithms in machine learning. In linear regression, we model a target/output variable as a linear function of input variables (that presumably have some relationship with the target variable) plus a normally distributed error term, i.e.

$$y = \langle \beta, \mathbf{x} \rangle + \epsilon \qquad (\epsilon \sim \mathcal{N}(0, 1))$$

For example, we might model a person's life expectancy as a linear function of his income and number of years of education, so if we have a vector $\mathbf{x} = (\text{income}, \text{years of education})$ that contains a person's income and the number of years he went to school, then we would compute our estimate for that person's life expectancy using

$$\text{life expectancy} = \langle \beta, \mathbf{x} \rangle$$

Of course, the question of what β to pick has not yet been addressed. In ordinary least squares regression, to solve for the β, we compute the β that minimises the least squares error, i.e.

$$\beta = \arg\min_{\beta} \left(\frac{1}{2} \sum_i (y_i - \langle \beta, \mathbf{x}_i \rangle)^2 \right)$$

where the $\frac{1}{2}$ is included for convenience when we take the derivative. If we have n features, and d data points, let X denote the $d \times n$ design matrix whose i^{th} row, j^{th} column contains the value for the j^{th} feature of the i^{th} data point, then the closed form solution to the above optimization problem is

$$\beta^* = (X^T X)^{-1} X^T y$$

1.4.3 Principal Component Analysis

A common issue in machine learning is the curse of dimensionality. This refers to the phenomenon where the volume of a feature space increases exponentially in the number of dimensions (e.g. $\{0, 1\}^d$ has 2^d elements). As a result, a lot of machine learning algorithms do not scale well to large dimensions. One way to deal with this is to reduce the dimensionality of the data while still preserving its essential structure. Making this more precise, if we want to reduce to dimension to size L, we want a set of L vectors W such that we minimize the reconstruction error, defined as

$$J(W, Z) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$$

where $\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i$ and subject to the constraint that $\mathbf{W}$ is orthonormal. Intuitively, W is a $D \times L$ matrix that maps the low dimensional representation $\mathbf{z}_i$ back to D dimension, and $\mathbf{z}_i$ is the low dimension representation of $\mathbf{x}_i$ computed by $\mathbf{Z} = \mathbf{X}\mathbf{W}$.

The following theorem then tells us that the set of L vectors we want can be obtained from the eigenvectors of the normalized data's empirical covariance matrix.

Theorem 1.4.1 : If we want to find an orthogonal set of L linear basis vectors $\mathbf{w}_j \in \mathbb{R}^D$ and the corresponding scores $\mathbf{z}_i \in \mathbb{R}^L$ such that we minimize the average reconstruction error

$$J(W, Z) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$$

then the optimal solution is to set $\hat{\mathbf{W}} = \mathbf{V}_L$ where $\mathbf{V}_L$ contains the L eigenvectors with largest eigenvalues of the empirical covariance matrix $\hat{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T$ (where the $\mathbf{x}_i$ are assumed to have zero mean)

Proof. See [Mur12] for a proof. □

Thus, we see that the problem of dimensionality reduction via PCA can be solved by obtaining the eigenvectors of the normalized empirical covariance matrix with greatest absolute eigenvalues. In particular, our low(L)-dimension representation of a data vector $\mathbf{x}$ is given by

$$\mathbf{y} = \mathbf{V}_L \mathbf{x} \qquad \text{(where } y \text{ has dimension } L\text{)}$$

The eigenvector with the i^{th} largest absolute eigenvalue is called the i^{th} principal component.

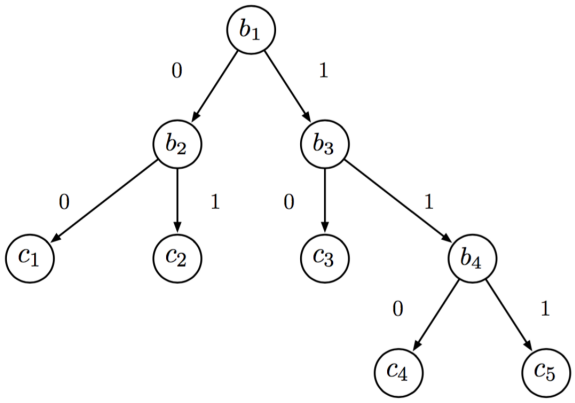


Figure 1.1: Example Classification Decision Tree, from [BPTG14]

1.4.4 Decision Trees

A desirable property of machine learning algorithms is not only predictive power, but also ease of interpretation. Easily interpretable models reveal structure and insights about the data, whereas convoluted models often end up working like a black box. This desire for easily interpretable models makes decision trees a reasonably popular machine learning algorithm, especially when many of them are used together in algorithms like Random Forests.

Decision trees are non-parametric classifiers that partition the feature space into regions corresponding to particular classes. At each node, a boolean function on the features is evaluated and the result of that function determines which sub-trees the data point enters. The leaves of the decision tree correspond to our class labels.

For instance, in Figure 1.1, we have an example decision tree where each b_i is a boolean variable that determines which branch we follow. The leaves are the final classifications. To classify a data point, we first compute the value of b_1 for that data point (e.g. if the feature space consists of the characteristics of fruit, we might have $b_1 = 1$ if the fruit is yellow, and 0 otherwise). If $b_1 = 1$, then we traverse down the right branch, and then have to determine b_3. Otherwise, we go on the left branch and determine b_2. Once we reach a leaf, c_i, that determines the classification of our data point.

1.4.5 Naïve Bayes Classifiers

Definiton 1.4.2: A Naïve Bayes Classifier is a method to classify vectors of discrete valued features. In particular, given a data vector $\mathbf{x} \in \{1, ...K\}^n$, where K is the number of possible values for each feature and D is the number of features. This allows us to write the class conditional density as follows:

$$P(\mathbf{x} \mid C = c, \theta) = \prod_{j=1}^{D} P(x_j \mid C = c, \theta_{js})$$

Naïve Bayes Classifiers are referred to as "naïve" because we assume that the features are conditionally independent given the class label. However, there is no reason we should expect this to be true. That said, even though Naïve Bayes Classifiers make this unwarranted assumption, they often work well in

practice [DP97]. One reason that Naïve Bayes Classifiers work well is because they only have $O(CD)$ parameters, where C is the number of classes, and D is the number of features [Mur12], thereby making the model very simple and resistant to overfitting. That said, Naïve Bayes Classifiers are best used when these independence assumptions are true, or at least close to true.

As the name implies, Naïve Bayes Classifiers are another way to approach the problem of classification. If we have K possible classes, then the model consists of the class priors $\{P(C = c_i)\}_{i=1}^{K}$ (i.e. the probability that each class occurs) and the class-conditional densities $\{P(X = x \mid C = c_i)\}_{i=1}^{K}$, i.e. the probability than an object of class c_i has a particular set of features x.

As a concrete example, suppose we have a list of diseases $\{d_1, ..., d_n\}$ and a vector of symptoms $(s_1, ..., s_m)$ where $s_i = 1$ if the person exhibits symptom i and 0 if he does not. The Naïve Bayes model specifies $P(s_i = 1 \mid d_j)$ for all i, j (i.e. the probability that someone exhibits symptom i given that he has disease j), and $P(d_j)$ for all j (i.e. the probability that someone has disease j). Then, to calculate the probability that someone who has a disease d_k exhibit a particular set of symptoms, $(s_1^{'}, ..., s_m^{'})$, we compute

$$P(s_1^{'}, ..., s_m^{'} \mid d_k) = \prod_{i=1}^{m} P(s_i = s_i^{'} \mid d_k) \qquad \text{(conditional independence given class label)}$$

With this model, classification then works by using a maximum a posteriori decision rule. Given a data point **x**, we compute the posterior probability of it being in each class c_i and classify **x** as the class with the highest posterior probability. More precisely, we compute

$$\begin{aligned} c^* &= \underset{i \in \{1,...,k\}}{\arg\max}\, P(C = c_i \mid X = x) \\ &= \underset{i \in \{1,...,k\}}{\arg\max}\, \frac{P(X = x \mid C = c_i)P(C = c_i)}{P(X = x)} \\ &= \underset{i \in \{1,...,k\}}{\arg\max}\, P(X = x \mid C = c_i)P(C = c_i) \qquad \text{(we can treat } P(X = x) \text{ as a constant)} \end{aligned}$$

Note that in the last step, we omit dividing by $P(X = x)$ because it is applied to the posterior probability for each class, and so omitting it will not affect which one ends up being the largest. As we will see later, not having this division step makes a Naïve Bayes Classifier easier to implement on a somewhat homomorphic encryption scheme.

Chapter 2

Homomorphic Encryption Schemes

In this chapter, we survey the construction of several homomorphic encryption schemes. To demonstrate that homomorphic encryption schemes are not as elusive as they may appear, we start by examining how the RSA algorithm is indeed somewhat homomorphic. Then, in the following section, we examine the Paillier cryptosystem, which is used in some of the machine learning applications we will explore later. Afterwards, we present a fully homomorphic encryption scheme by [GSW13] based on the learning with errors assumption. Finally, although we don't present a homomorphic encryption scheme strictly based on lattice problems, we show a reduction from the shortest vector lattice problem to learning with errors before concluding with a brief section on encoding real numbers.

2.1 Somewhat Homomorphic Properties of RSA

The RSA algorithm devised by Rivest, Shamir, and Adleman is used widely today to securely transmit smaller secret keys between both parties that can then be used to securely communicate larger messages through more efficient private-key encryption schemes. Unfortunately, on its own, plain RSA is not a secure encryption scheme, but in any case, we will explore the homomorphic properties that RSA happens to have, even though it was not specifically designed for this purpose. First, we formally define RSA below:

Algorithm 1 Plain RSA[KL07]

procedure KEY GENERATION(1^λ)
  Choose 2 random λ-bit primes p, q, and compute $N = pq$
  Compute $\phi(N) = (p-1)(q-1)$
  Select $e > 1$ such that $\gcd(\phi(N), e) = 1$, and compute $d = e^{-1} \bmod \phi(N)$
  return (N, e, d)
end procedure

procedure ENCRYPTION(m, N, e)
  return $c = m^e \bmod N$
end procedure

procedure DECRYPTION(c, N, d)
  return $m = c^d \bmod N$
end procedure

We can easily see that the RSA encryption and decryption schemes are correct because by the key generation scheme,

$$\begin{aligned}\texttt{Dec}_{(N,d)}(\texttt{Enc}_{(N,e)}(m)) &= \texttt{Dec}_{(N,d)}(m^e \bmod N) \\ &= m^{de} \bmod N \\ &= m \bmod N \quad \text{(Euler's Totient Theorem (below), and } de = 1 \bmod \phi(N))\end{aligned}$$

Theorem 2.1.1 (Euler's Totient Theorem): If n and a are coprime positive integers, then

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

where ϕ is Euler's totient function.

Furthermore, note that Plain RSA takes for granted that we can generate λ-bit primes for any $\lambda \in \mathbb{Z}^+$. This is definitely not a trivial assumption, but many simple schemes that generate random λ-bit numbers and use primality tests like the Miller-Rabin test work well in practice [KL07]. A more thorough discussion of methods to generate λ-bit primes is beyond the scope of this thesis, so we will assume that we can efficiently generate such primes.

Now, we show that RSA is a valid somewhat homomorphic encryption scheme that supports multiplication modulo N.

Proposition 2.1.2: Plain RSA gives us a valid somewhat homomorphic encryption scheme. In particular multiplication of ciphertexts modulo N corresponds to multiplication of plaintexts modulo N

Proof. Let $\texttt{Enc}_{(N,e)}$ denote the RSA encryption algorithm. Let $m_1, m_2 \in \mathbb{Z}_N$ be our two messages. Observe that

$$\begin{aligned}\texttt{Enc}_{(N,e)}(m_1) \cdot \texttt{Enc}_{(N,e)}(m_2) &= m_1^e \cdot m_2^e \bmod N \\ &= (m_1 m_2)^e \bmod N \\ &= \texttt{Enc}_{(N,e)}(m_1 m_2)\end{aligned}$$

Thus, we can see that multiplying the ciphertexts modulo N corresponds to multiplying the plaintexts modulo N. □

Unfortunately, even if we take the RSA assumption as given, there are a number of attacks on plain RSA that make it insecure [KL07]. In fact, because RSA is deterministic, it cannot be CPA secure. However, at the very least, we can see that the idea of homomorphic encryption is attainable because RSA is able to achieve homomorphism without that being its intended purpose.

2.2 The Paillier Cryptosystem

In this section, we explore an additive homomorphic encryption scheme proposed by Paillier in 1999 [Pai99]. This scheme makes use of the Composite Residuosity Class Problem, which is closely related to RSA. Although not fully homomorphic, the Paillier encryption scheme is used in many of the machine

learning applications we explore in this thesis. A nice feature of the Paillier cryptosystem is that it is homomorphic over a large message space, i.e. $\mathbb{Z}_N$. A common example of a use case for the Paillier cryptosystem is computing encrypted vote counts, since the resulting sums can attain high totals [KL07].

Since its inception, the Paillier cryptosystem has also undergone refinements to make it more efficient [JLMS15]. We outline the algorithm below (Algorithm 2), illustrate the reduction to RSA, and give a proof of security, following the expositions in [Pai99], [KL07]. In particular, we present many of the details from [Pai99] but for clarity of presentation, use notation and terminology from [KL07].

2.2.1 The Encryption Scheme

For the Paillier cryptosystem, the message space $\mathcal{M} = \mathbb{Z}_N$, the key space $\mathcal{K} = \mathcal{K}_{pk} \times \mathcal{K}_{sk}$, where $\mathcal{K}_{pk} = \mathbb{Z}_N, \mathcal{K}_{sk} = \mathbb{Z}_N^*$, and the ciphertext space $\mathcal{C} = \mathbb{Z}_{N^2}$.

Algorithm 2 (from Paillier[Pai99], [KL07])

procedure KEYGEN(1^λ)
    Sample λ-bit primes p, q
    Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$
    return $(N, \phi(N))$
end procedure

procedure ENCRYPTION(N, m)
    $r \xleftarrow{R} \mathbb{Z}_N^*$
    $c = (1+N)^m \cdot r^n \pmod{N^2}$
    return c
end procedure

procedure DECRYPTION($c, N, \phi(N)$)
    Compute $\hat{c} = c^{\phi(N)} \bmod N^2$
    Compute $\hat{m} = \dfrac{\hat{c} - 1}{N}$
    Compute $m = \hat{m} \cdot \phi(N)^{-1} \bmod N^2$
    return m
end procedure

To reason about the correctness of the Paillier cryptosystem, we examine the ciphertext space: the group $\mathbb{Z}_{N^2}^*$ under multiplication modulo N^2 and introduce the notion of N^{th} residues, which are important to the Paillier cryptosystem.

Definiton 2.2.1: An integer $z \in \mathbb{Z}_{N^2}^*$ is an N^{th} residue modulo N^2 if there exists a $y \in \mathbb{Z}_{N^2}^*$ such that

$$z = y^N \pmod{N^2}$$

Returning to correctness, we must prove that the encryption scheme is one-to-one, and that the decryption scheme is correct. The former is important because if two plaintexts are mapped to the same ciphertext, then the decryption algorithm will not be able to decrypt correctly. So, fix an N^{th} residue $g \in \mathbb{Z}_{N^2}^*$, and

denote $\mathcal{E}_g$ as the map:

$$\begin{aligned}\mathcal{E}_g : \mathbb{Z}_N \times \mathbb{Z}_N^* &\to \mathbb{Z}_{N^2}^* \\ (x, y) &\mapsto g^x \cdot y^N \pmod{N^2}\end{aligned}$$

Lemma 2.2.2: If the order of $g \in \mathbb{Z}_{N^2}^*$ is a nonzero multiple of N, then $\mathcal{E}_g$ is bijective.

Proof. First, we show that $\mathbb{Z}_N \times \mathbb{Z}_N^*$ and $\mathbb{Z}_{N^2}^*$ have the same number of elements. $\mathbb{Z}_n \times \mathbb{Z}_N^*$ has $N\phi(N)$ elements, where ϕ is the Euler totient function. Let $N = \prod_i p_i^{e_i}$ be the prime factorization of N. Note that,

$$\begin{aligned}\phi(N) &= \prod_i \phi(p_i^{e_i}) && \text{(multiplicativity of } \phi\text{)} \\ &= \prod_i p_i^{e_i - 1}(p_i - 1) && (\phi \text{ for prime powers})\end{aligned}$$

As a result,

$$\begin{aligned}\phi(N^2) &= \prod_i \phi(p_i^{2e_i}) \\ &= \prod_i p_i^{2e_i - 1}(p_i - 1) \\ &= \left(\prod_i p_i^{e_i}\right)\left(\prod_i p_i^{e_i - 1}(p_i - 1)\right) \\ &= N\phi(N)\end{aligned}$$

Thus, $\mathbb{Z}_N \times \mathbb{Z}_N^*$ and $\mathbb{Z}_{N^2}^*$ have the same number of elements, and it suffices to show injectivity to prove $\mathcal{E}_g$ is bijective. Suppose that $g^{x_1} y_1^N = g^{x_2} y_2^N \pmod{n^2} \Rightarrow g^{x_2 - x_1}(y_2/y_1)^N = 1 \pmod{N^2}$. Let λ denote the largest integer for which there exists an element of $\mathbb{Z}_{N^2}^*$ with order λN. Thus, $(y_2/y_1)^{\lambda N} = 1 \pmod{n^2} \Rightarrow g^{\lambda(x_2 - x_1)} = 1 \pmod{N^2}$. This implies that $\lambda(x_2 - x_1)$ is a multiple of g's order and so a multiple of N. Thus, $\gcd(\lambda, N) = 1 \Rightarrow x_2 - x_1 = 0 \pmod{N}, (y_2/y_1)^N = 1 \pmod{N^2}$, which has a unique solution $y_2/y_1 = 1$ over $\mathbb{Z}_{N^2}^*$. Thus, $x_1 = x_2, y_1 = y_2$, and $\mathcal{E}_g$ is bijective. □

It then follows from bijectivity that the encryption algorithm does not map two distinct messages onto the same ciphertext. As for the correctness of the Paillier cryptosystem's decryption algorithm, we have the following proposition.

Proposition 2.2.3: The Paillier Cryptosystem's decryption algorithm is correct. More precisely, for some $m \in \mathbb{Z}_N$, then $\texttt{Dec}_{\phi(N)}(\texttt{Enc}_N(m)) = m$.

Proof. Observe that

$$\begin{aligned}\hat{c} &= c^{\phi(N)} \bmod N^2 \\ &= (1+N)^{m\phi(N)} r^{n\phi(n)} \bmod N^2 \\ &\leftrightarrow (m\phi(N) \bmod (N), r^{\phi(N)} \bmod N) && \text{(using the bijection)} \\ &= (m\phi(N) \bmod (N), 1) && (a^{\phi(N)} = 1 \bmod N \text{ for } a, N \text{ coprime}) \\ &\leftrightarrow (1+N)^{m\phi(N)} \bmod N^2 && \text{(using the bijection)} \\ &= 1 + m\phi(N)N \bmod N^2\end{aligned}$$

It then follows from this that

$$\left(\frac{\hat{c}-1}{N}\right)\cdot\phi(N)^{-1}=m$$

and so the decryption scheme is correct. □

Now that we've verified correctness for the encryption and decryption algorithms, we can proceed to examine the security of the Paillier cryptosystem. On a high level, the "hard problem" that the Paillier cryptosystem is based on is the decisional composite residuosity problem, which relates to distinguishing between uniform elements of $\mathbb{Z}_{N^2}^*$ and n^{th} residues in $\mathbb{Z}_{N^2}^*$. More formally, we define what it means for the decisional composite residuosity problem to be hard, using the definition from [KL07].

Definiton 2.2.4: The decisional composite residuosity problem is hard if for all probabilistic polynomial-time algorithms D, there is a negligible function $\texttt{negl}$ such that for r randomly chosen from $\mathbb{Z}_{N^2}^*$

$$|\Pr[D(N,r^N \bmod N^2)=1]-\Pr[D(N,r)=1]|\leq \texttt{negl}(n)$$

In Paillier's original presentation of this scheme, he explores the n^{th} Residuosity Class Problem and demonstrates that solving this gives us a distinguisher for the decisional composite residuosity problem above. The N^{th} residuosity class is defined as follows:

Definiton 2.2.5: Let $g\in\mathbb{Z}_{N^2}^*$ have order a non-zero multiple of N. We call the N^{th} residuosity class of w with respect to g, the unique integer $[[w]]_g\in\mathbb{Z}_n$ for which there exists y such that

$$\mathcal{E}_g([[w]]_g,y)=([[w]]_g)^g y^N=w$$

Note that this problem is well defined because $[[w]]_g$ is unique, which comes from the fact that $\mathcal{E}_g$ is bijective (by Lemma 2.2.2).

2.2.2 Reduction to RSA

Definiton 2.2.6: The N^{th} Residuousity Class Problem of base g, Class$[N,g]$ is the following problem. Given $w\in\mathbb{Z}_{N^2}^*$ and a base g, compute $[[w]]_g$.

How the N^{th} Residuosity Class Problem gives us a distinguisher for the decisional composite residuosity problem can be seen from the following result:

Lemma 2.2.7: $[[w]]_g=0$ if and only if w is an N^{th} residue modulo N^2

Proof. See [Pai99] for a proof. □

Thus, if we could solve the N^{th} Residuousity Class Problem, a simple distinguisher D could take r and compute $[[r]]_g$ for some suitable g. If this is 0, then it outputs 1 (guessing that it is an N^{th} residue). Otherwise output 0. It is easy to see that this would have non-negligible advantage. This is because by Lemma 2.2.2 and the previous lemma, a non-trivial proportion of elements of $\mathbb{Z}_{N^2}^*$ are not N^{th} residues,

i.e. exactly an $(N-1)/N$ proportion of them.

Currently, a particular instantiation of the N^{th} Residuosity Class Problem depends on several parameters, i.e. N, g, and w. From the following two lemmas, it turns out that not all these parameters influence the difficulty of the problem. First, note that in the construction of this distinguisher, we didn't specify what value of g to choose. Lemma 2.2.8 shows that this does not matter.

Lemma 2.2.8: Class$[N, g]$ is random-self-reducible over all $g^{'}$ whose order is a non-zero multiple of N.

Proof. Observe that $[[w]]_{g_1} = [[w]]_{g_2}[[g_2]]_{g_1} \bmod N$. This gives that $[[g_1]]_{g_2} = [[g_2]]_{g_1}^{-1} \bmod N$, and so $[[g_2]]_{g_1}$ is invertible modulo N. Thus if we have an oracle for Class$[n, g_1]$, then given a g_2 and w, we can solve Class$[n, g_2]$ using

$$[[w]]_{g_2} = [[w]]_{g_1}[[g_2]]_{g_1}^{-1} \bmod N$$

□

With this result, we can fix $g = 1 + N$, as they do in [KL07]. This is a valid choice of g because $(1+N)^N = 1 \bmod N^2$ and so the order of $1 + N$ is a non-zero multiple of N (i.e. N).

Another interesting property of Class$[N, g]$ is the fact that it is random-self-reducible. More precisely, we can reduce an instance of the problem for a given $w \in \mathbb{Z}_{N^2}^*$ into a random instance with $w^{'} \in \mathbb{Z}_{N^2}^*$ drawn from a uniform distribution.

Lemma 2.2.9: Class$[N, g]$ is random-self-reducible over $w \in \mathbb{Z}_{N^2}^*$

Proof. Suppose we are given an instance of the problem with $w \in \mathbb{Z}_{n^2}^*$. We use w to calculate $w^{'} = wg^{\alpha}\beta^N \bmod N^2$ where $\alpha, \beta \overset{R}{\leftarrow} \mathbb{Z}_n$ (note that $\beta \notin \mathbb{Z}_n^*$ occurs with a negligibly small probability for large product of primes $N = pq$). Once we have $[[w^{'}]]_g$, then we can output $[[w]]_g = [[w^{'}]]_g - \alpha \bmod n$. □

Thus, we see that our choices of $w \in \mathbb{Z}_{N^2}^*$ and g an N^{th} residue modulo N don't affect the difficulty of the N^{th} Residuosity Class Problem. Hence, we only need to choose the parameter N, and so we can denote an instantiation of this problem as Class$[N]$.

Theorem 2.2.10 (Reduction to RSA): Class$[N]$ is reducible to RSA

Proof. We know that instances of Class$[N, g]$ are computationally equivalent for g with order a non-zero multiple of N. Using $g = 1 + N$, we only need to show that

$$\text{RSA} \Rightarrow \text{Class}[N, 1+N]$$

Now suppose we have an oracle S for RSA. We have that $c = (1+N)^m y^N \bmod N^2$ for some $m \in \mathbb{Z}_N$ and $y \in \mathbb{Z}_N^*$. Expanding $(1+N)^m$ gives us that $c = y^N \bmod N$ and using our oracle, we get

$y = S(c \bmod N)$. Then, we have that

$$\begin{aligned} c &= (1+N)^m y^N \bmod N^2 \\ \frac{c}{y^N} &= (1+N)^m \bmod N^2 \\ &= 1 + mN \bmod N^2 \end{aligned}$$

This allows us to calculate $m = [[c]]_{1+N}$, as desired, thereby solving Class$[N]$ □

Thus, we've shown that the decisional problem that the Paillier cryptosystem and RSA are closely related encryption schemes in that the former is at most as difficult as factoring or RSA. However, that does not automatically make it secure. Below, we show that the Paillier cryptosystem is CPA-secure assuming the decisional composite residuosity problem is hard, using a neat proof from [KL07].

2.2.3 Proof of Security

Before we present the proof of security, we prove a standard result from group theory.

Proposition 2.2.11: If G is a finite group and $g \in G$ be chosen arbitrarily. Then, if we choose a uniform $k \in G$, then the distribution of $g \circ k \in G$ is also uniform

Proof. Note that for every $g^{'} \in G$, there exists exactly one $k \in G$ (i.e. $k = g^{-1} \circ g^{'}$) such that $g \circ k = g \circ g^{-1} \circ g^{'} = g^{'}$. □

Now, we can proceed to prove the conditional CPA-security of the Paillier cryptosystem.

Theorem 2.2.12 : If the decisional composite residuosity problem is hard, then the Paillier cryptosystem is CPA-secure.

Proof. Let $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ denote the Paillier cryptosystem. Because the Paillier cryptosystem is a public-key encryption scheme, it suffices to show that $\mathcal{E}$ is EAV-secure. Then, by Proposition 1.2.10, this implies that it is CPA-secure. We will do this by constructing a reduction from the decisional composite residuosity problem to breaking EAV-security for the Paillier cryptosystem. Then, because we assume that the decisional composite residuosity problem is hard, we must not be able to break EAV-security for the Paillier cryptosystem.

Let $\mathcal{A}$ be a probabilistic polynomial time adversary. Consider the following algorithm D that attempts to solve the decisional composite residuosity problem by behaving as follows on input N, and y

1.) Set $pk = N$ and run $\mathcal{A}(pk)$ to obtain two messages $m_0, m_1 \in \mathcal{M}$

2.) Choose a uniform bit $b \in \{0, 1\}$ and compute the challenge ciphertext $c = [(1+N)^{m_b} \cdot y] \bmod N^2$

3.) Give $\mathcal{A}$ the challenge ciphertext c, and $\mathcal{A}$ will output a bit $b^{'}$. If $b^{'} = b$ output 1, else 0

First, if the input to D had a y that was drawn from by taking a random $r \in \mathbb{Z}_{N^2}^*$ and raising it to the N^{th} power, we would have $c = (1+N)^{m_b} \cdot y \bmod N^2 = (1+N)^{m_b} \cdot r^N \bmod N^2$. In this case, we

note that if $\mathcal{A}$ was able to correctly distinguish between encryptions of m_0 and m_1 (i.e. if $b = b'$), that is equivalent to D successfully determining whether y was drawn uniformly from $\mathbb{Z}_{N^2}^*$ or from $Res(N^2)$. Hence, we have that

$$\Pr[D(N, r^N \bmod N^2) = 1] = \Pr[\mathtt{PubK}^{\mathtt{eav}}_{\mathcal{A},\mathcal{E}}(n) = 1]$$

Now, we observe the following, if our input to D had a y that was randomly drawn from $\mathbb{Z}_{N^2}^*$, then by Proposition 2.2.11, the ciphertext c is distributed uniformly in $\mathbb{Z}_{N^2}^*$, independent of m. Thus, the probability that $b' = b$ is exactly 1/2 and in particular,

$$\Pr[D(N, r) = 1] = \frac{1}{2}$$

Now, using the assumption that the decisional composite residuosity problem is hard, we have that there exists a negligible function $\mathtt{negl}$ such that

$$\begin{aligned}
&\left|\Pr[D(N, r^N \bmod N^2) = 1] - \Pr[D(N, r) = 1]\right| \leq \mathtt{negl}(n) \\
&\qquad \Rightarrow \left|\Pr[\mathtt{PubK}^{\mathtt{eav}}_{\mathcal{A},\mathcal{E}}(n) = 1] - \frac{1}{2}\right| \leq \mathtt{negl}(n) \qquad \text{(using above results)} \\
&\qquad\qquad \Rightarrow \Pr[\mathtt{PubK}^{\mathtt{eav}}_{\mathcal{A},\mathcal{E}}(n) = 1] \leq \frac{1}{2} + \mathtt{negl}(n)
\end{aligned}$$

This then shows that the Paillier cryptosystem is EAV-secure, and thus CPA-secure, as desired. □

2.3 Fully Homomorphic Encryption from Learning With Errors

So far, the schemes we have examined (i.e. RSA and the Paillier Cryptosystem) have been somewhat homomorphic. Somewhat homomorphic encryption schemes are limited in that they do not support the evaluation of all possible functions f over the ciphertexts. This restriction limits the extent that they can be employed, such as in cases where we may not know what specific operations will be required. Indeed, we'd like to be able to run state-of-the-art machine learning algorithms over our encrypted data without having to change the encryption scheme every time we need to support a new operation, or restrict our algorithms to a certain set of operations.

On the other hand, a fully homomorphic encryption scheme would allow us to evaluate arbitrary functions over the data. The first fully homomorphic encryption scheme was proposed by Gentry [Gen09], using a stronger variant of the Learning with Errors assumption for ideal lattices and an additional assumption about sparse subset sums. We don't cover Gentry's construction in this thesis, but note that it has served as a foundation for many fully homomorphic encryption schemes constructed later on.

While the fact that we can construct fully homomorphic encryption schemes is a huge success, this flexibility is not without its tradeoffs. Indeed, as noted in [LNV11], even though solutions for fully homomorphic encryption have been constantly proposed and refined, fully homomorphic encryption schemes seem to be a long way away from being efficient enough to be used in practice. However, somewhat homomorphic encryption schemes tend to be much faster, obtaining speed in exchange for only supporting a limited number of operations, which justifies their usage in many machine learning

applications. In any case, we continue in this thesis with an exploration of the fully homomorphic encryption scheme in [GSW13], based on LWE assumptions.

2.3.1 The Learning with Errors Problem/Assumption

The learning with errors problem is the "difficult problem" that the fully homomorphic encryption scheme in [GSW13] is based on. Informally, this assumption states that given $\mathbf{u}$ and $\langle \mathbf{u}, \mathbf{v} \rangle$, it is difficult to solve for $\mathbf{v}$ if a small noise term is added to $\langle \mathbf{u}, \mathbf{v} \rangle$. We give a formal definition below.

Definiton 2.3.1: Given a security parameter λ, let $n = n(\lambda$ be an integer-dimension, $q = q(\lambda) \geq 2$ be an integer and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The Learning With Errors problem (LWE) is to distinguish between the following two distributions:

1.) Sample $(\mathbf{a}_i, b_i) \xleftarrow{R} \mathbb{Z}_q^{n+1}$

2.) Sample $(\mathbf{a}_i, b_i)$ by sampling $\mathbf{a}_i \xleftarrow{R} \mathbb{Z}_q^n$, and sampling b_i by sampling $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$, e_i from the error distribution χ and setting $b_i = \langle \mathbf{a}_i, \mathbf{s}_i \rangle + e_i$

The Learning With Errors assumption is then that the above problem is infeasible.

2.3.2 An Overview of the Scheme: The Approximate Eigenvector Method

The general idea behind this encryption scheme is to achieve LWE-based homomorphic encryption where the addition and multiplication operations correspond directly to matrix addition and multiplication. The general idea is that given a message m and secret key $\mathbf{v}$ with at least one "big" coefficient v_i, the ciphertext is a matrix C such that

$$C \cdot \mathbf{v} = m \cdot \mathbf{v} + \mathbf{e} \qquad \text{(for } \mathbf{e} \text{ "small" error vector)}$$

Therefore, the secret key $\mathbf{v}$ is an approximate eigenvector of the ciphertext C with eigenvalue equal to the message m. To decrypt, we extract the i^{th} row of C (corresponding to the large entry in $\mathbf{v}$, v_i), and compute

$$\begin{aligned} x = \langle C_i, \mathbf{v} \rangle &= m \cdot v_i + e_i \\ \Rightarrow m = \lfloor x / v_i \rceil & \qquad (\lfloor \cdot \rceil \text{ denotes rounding to nearest integer}) \end{aligned}$$

Because the error e_i is small when compared to our "large" coefficient v_i, it will not affect the rounding of x/v_i and distort our decryption of m. Hence, the encryption scheme will be correct.

2.3.3 Tools for Homomorphic Encryption

From our overview, we can see that it is important to keep the error term bounded. Hence, we define what it means to keep a ciphertext strongly bounded and introduce some methods in Algorithm 3 that will allow us to maintain this property. For notation, suppose we have a vector $\mathbf{a}$, then let $a_{i,j}$ to be the j^{th} bit (where $j = 0$ is the least significant bit) of the i element of the vector.

Definiton 2.3.2: A ciphertext C is B-strongly-bounded if its associated message m, and the coefficients of C have magnitude at most 1, while the coefficients of its error term **e** has magnitude at most B

The methods below will be used in the encryption, decryption, and homomorphic evaluation algorithms to keep our ciphertexts strongly bounded. `BitDecomp` simply returns the binary coefficients for each entry in a vector **a**. $\texttt{BitDecomp}^{-1}$ maps these coefficients back to the vector **a**, but note that the inputs to $\texttt{BitDecomp}^{-1}$ need not be in $\{0,1\}$. `Flatten` uses both these operations to take in "ill-formed" binary coefficients (i.e. not in $\{0,1\}$) and represent them with valid binary coefficients without changing the underlying vector's entries. Finally, `PowersOf2` simply creates copies of each entry multiplied by powers of 2.

Algorithm 3 BitDecomp, Flatten, Powersof2 Algorithms

procedure BITDECOMP(**a**)
  return $(a_{1,0}, ..., a_{1,l-1}, ..., a_{k,0}, a_{k,l-1})$
end procedure

procedure $\text{BITDECOMP}^{-1}(a_{1,0}, ..., a_{1,l-1}, ..., a_{k,0}, a_{k,l-1})$
  return $(\sum_j a_{1,j}, ..., \sum_j a_{k,j})$
end procedure

procedure $\text{FLATTEN}(a_{1,0}, ..., a_{1,l-1}, ..., a_{k,0}, a_{k,l-1})$
  return $\texttt{BitDecomp}(\texttt{BitDecomp}^{-1}(a_{1,0}, ..., a_{1,l-1}, ..., a_{k,0}, a_{k,l-1}))$
end procedure

procedure POWERSOF2(**b**)
  return $(b_1, 2_1, ..., 2^{l-1}b_1, ..., b_k, 2b_k, ..., 2^{l-1}b_k)$
end procedure

A few simple facts to note about these methods are

1.) $\langle \texttt{BitDecomp}(\mathbf{a}), \texttt{Powersof2}(\mathbf{b}) \rangle = \langle \mathbf{a}, \mathbf{b} \rangle$

2.) For N-dimensional **a**, if we let $\mathbf{a}' = \texttt{BitDecomp}(\mathbf{a})$, $\langle \mathbf{a}', \texttt{Powersof2}(\mathbf{b}) \rangle = \langle \texttt{BitDecomp}^{-1}(\mathbf{a}'), \mathbf{b} \rangle = \langle \texttt{Flatten}(\mathbf{a}'), \texttt{Powersof2}(\mathbf{b}) \rangle$

Because `Flatten` converts bit decompositions to valid binary representations, i.e. with digits in $\{0,1\}$, it is useful for ensuring that coefficients of our vectors/matrices are small. Recall that this is important for ensuring that our error terms remains bounded throughout the homomorphic evaluation.

2.3.4 The Encryption Scheme

Now, we explore the encryption, decryption, and homomorphic evaluation operations of the fully homomorphic encryption scheme by [GSW13]. Note that although we don't cover it in this thesis, the scheme by [GSW13] also gives us Identity and Attribute-Based Encryption Schemes.

Recall that the ciphertext space consists of $N \times N$ matrices over $\mathbb{Z}_q$ for some dimension parameter N and modulus q. In more detail, suppose C_1 and C_2 are encryptions of m_1 and m_2 respectively in that

$$C_1 \cdot \mathbf{v} = m_1 \cdot \mathbf{v} + \mathbf{e}_1$$
$$C_2 \cdot \mathbf{v} = m_2 \cdot \mathbf{v} + \mathbf{e}_2$$

for small $\mathbf{e}_i$. To see vaguely why this scheme could be homomorphic, we can then see that the sum and products of the ciphertexts are given by

$$\begin{aligned}(C_1 + C_2) \cdot \mathbf{v} &= (m_1 + m_2) \cdot \mathbf{v} + (\mathbf{e}_1 + \mathbf{e}_2) \\ (C_1 \circ C_2) \cdot \mathbf{v} &= C_1 \cdot (m_2 \cdot \mathbf{v} + \mathbf{e}_2) \\ &= m_2 \cdot (m_2 \cdot \mathbf{v} + \mathbf{e}_2) + C_1 \cdot \mathbf{e}_2 \\ &= m_1 \cdot m_2 \cdot \mathbf{v} + (m_2 \cdot \mathbf{e}_1 + C_1 \cdot \mathbf{e}_2)\end{aligned}$$

where the terms $(\mathbf{e}_1 + \mathbf{e}_2)$ and $(m_2 \cdot \mathbf{e}_1 + C_1 \cdot \mathbf{e}_2)$ end up being small. Formally, the scheme is outlined below. Following the presentation in [GSW13], we break up the `KeyGen` algorithm into three parts: `Setup`, `SecretKeyGen`, and `PublicKeyGen`, as below in Algorithm 4:

Algorithm 4 Setup, SecretKeyGen, and PublicKeyGen Algorithms

procedure SETUP($1^\lambda, 1^L$)
    Chose a modulus of $\kappa = \kappa(\lambda, L)$ bits
    Choose lattice dimension $n = n(\lambda, L)$
    Choose error distribution $\chi = \chi(\lambda, L)$ appropriately for LWE that achieves at least 2^λ security against known attacks
    Choose parameter $m = m(\lambda, L) = O(n \log(q))$
end procedure

procedure SECRETKEYGEN(κ, n, χ, m)
    Sample $\mathbf{t} \xleftarrow{R} \mathbb{Z}_q^n$.
    Set $sk = (1, -t_1, ..., -t_n) \in \mathbb{Z}_q^{n+1}$
    return sk
end procedure

procedure PUBLICKEYGEN(κ, n, χ, m, sk)
    Sample a matrix $B \xleftarrow{R} \mathbb{Z}_q^{m \times n}$
    Sample a vector $\mathbf{e} \leftarrow \chi^m$
    Set $\mathbf{b} = B \cdot \mathbf{t} + \mathbf{e}$ where $\mathbf{t}$ can be recovered from sk
    Compute $pk = A = (n+1)$ column matrix where 1^{st} column is $\mathbf{b}$ and next n columns is B
    return pk. (Observe that $pk \cdot sk = \mathbf{e}$ where pk is a matrix and sk and $\mathbf{e}$ are vectors).
end procedure

Below in Algorithm 5 are the main operations of the encryption scheme. Note that there are two decryption algorithms: one if the message comes from a small sample space (e.g. $\{0, 1\}$), and one that works for any message in $\mathbb{Z}_q$, though we only present the case where q is a power of 2. The latter decryption algorithm `MPDecryption` is from [MP12] and a treatment of the general q case can be found there. We also specify how homomorphic operations will be conducted in Algorithm 6 below:

Algorithm 5 The Full Encryption Scheme

procedure ENCRYPTION($(\kappa, n, \chi, m, pk, \mu \in \mathbb{Z}_q)$)
  Sample a uniform matrix $R \in \{0,1\}^{N \times m}$
  Compute $C = \texttt{Flatten}(\mu \cdot I_N + \texttt{BitDecomp}(R \cdot A)) \in \mathbb{Z}_q^{N \times N}$
  return C
end procedure

procedure DECRYPTION($(\kappa, n, \chi, m, sk, c)$)
  Compute $\mathbf{v} = \texttt{Powersof2}(sk)$
  Note that the first l coefficients of $\mathbf{v}$ are $1, 2, ..., 2^{l-1}$
  Choose i such that $v_i = 2^i \in (q/4, q/2]$
  Compute $x_i = \langle C_i, \mathbf{v} \rangle$ where C_i is the i^{th} row of C
  Compute $\mu^{'} = \lfloor x_i / v_i \rceil$
  return $\mu^{'}$
end procedure

procedure MPDECRYPTION($(\kappa, n, \chi, m, sk, c)$)
  Compute $\mathbf{v} = \texttt{Powersof2}(sk)$
  Note that $q = 2^{l-1}$ and the first $l-1$ coefficients of $\mathbf{v}$ are $1, 2, ..., 2^{l-2}$.
  Recover $LSB(\mu)$ from $\mu \cdot 2^{l-2} + small$
  for $i = 3$ to l **do**
    Recover next least significant bits from $(\mu -$ sum of LSBs recovered so far$) \cdot 2^{l-i} + small$
  end for
  Construct $\mu^{'}$ using the recovered bits
  return $\mu^{'}$
end procedure

Algorithm 6 Homomorphic Operations

procedure ADD(C_1, C_2)
  Compute $C_1 + C_2$
  return $\texttt{Flatten}(C_1 + C_2)$
end procedure

procedure MULTCONST(C, α)
  Compute $M_\alpha = \texttt{Flatten}(\alpha \cdot I_N)$
  return $\texttt{Flatten}(M_\alpha \cdot C)$
end procedure

procedure MULT(C_1, C_2)
  return $\texttt{Flatten}(C_1 \times C_2)$
end procedure

procedure NAND(C_1, C_2)
  return $\texttt{Flatten}(I_N - C_1 \cdot C_2)$
end procedure

The statement of security is as follows.

Proposition 2.3.3: Let the set of parameters (n, q, χ, m) be such that the Learning With Errors assumption holds. Then, for $m = O(n \log(q))$ and A, R as generated above, the joint distribution $(A, R \cdot A)$ is computationally indistinguishable from uniform over $\mathbb{Z}_q^{m \times (n+1)} \times \mathbb{Z}_q^{N \times (n+1)}$, and thus, is CPA-secure.

Proof. See [Reg05] for a proof. □

From this, we have a construction of a CPA-secure fully homomorphic encryption scheme based on the learning with errors assumption. This particular construction also uses innovations from [BGV12b] to make the bootstrapping step common to many learning with errors based schemes optional, thereby making it more efficient.

2.4 Reduction of Lattice-Based Schemes to Learning with Errors

Aside from learning with errors, another important cryptographic primitive are lattice-based problems. These are important because the worst-case hardness assumptions for these lattice based problems remain secure even against quantum computers. Moreover, several fully homomorphic encryption schemes have been constructed using these lattice-based problems [BV14]. Although we don't explore any encryption schemes strictly based on lattice-based problems, we will briefly introduce the theory of lattice based cryptography, and then present a reduction from the often used `GapSVP` lattice problem to learning with errors.

Definiton 2.4.1: An n-dimensional (full-rank) lattice is a discrete, additive subgroup of $\mathbb{R}^n$. Equivalently, if we have a basis of n linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \subset \mathbb{R}^n$, then the lattice Λ generated by the basis $\mathbf{B}$ is the set

$$\Lambda = \mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} c_i \mathbf{b}_i \mid c_i \in \mathbb{Z} \right\}$$

Definiton 2.4.2: The dual lattice of a lattice Λ, denoted Λ^* is $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n \mid \text{ for all } \mathbf{v} \in \Lambda, \langle \mathbf{x}, \mathbf{v} \rangle \in \mathbb{Z}\}$.

Definiton 2.4.3: The minimum distance $\lambda_1(\Lambda)$ of a lattice Λ (in the l_2 norm by default) is the minimum distance between any two distinct lattice points in Λ. This is equivalent to the length of the shortest non-zero lattice vector, i.e.

$$\lambda_1(\Lambda) = \min_{\mathbf{x}_1 \neq \mathbf{x}_2; \mathbf{x}_1, \mathbf{x}_2 \in \Lambda} \|\mathbf{x}_1 - \mathbf{x}_2\| = \min_{\mathbf{x} \neq 0; \mathbf{x} \in \Lambda} \|\mathbf{x}\|$$

Definiton 2.4.4: Given a function $\gamma(n) \geq 1$, the shortest vector problem $\texttt{GapSVP}_\gamma(\mathbf{B}, d)$, where $\mathbf{B}$ is a basis of an n-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$ and $d \in \mathbb{R}^+$ is the problem of determining whether $\lambda_1(\Lambda) \leq d$. In particular, it requires an output of 1 if $\lambda_1(\Lambda) \leq d$ and an output of 0 if $\lambda_1(\Lambda) > \gamma(n) \cdot d$.

In 2009, Peikert proved a reduction from a particular variant of the shortest vector problem $\texttt{GapSVP}_{\zeta,\gamma}$ defined below to corresponding versions of the learning with errors problem [Pei09]. For completeness, we present the statement and proof of the theorem in [Pei09]. First we define the $\texttt{GapSVP}_{\zeta,\gamma}$ variant of

the shortest vector problem.

Definiton 2.4.5: Given functions $\zeta(n) \geq \gamma(n) \geq 1$, the ζ-to-γ shortest vector problem $\texttt{GapSVP}_{\zeta,\gamma}(\mathbf{B}, d)$, where

1.) $\mathbf{B}$ is a basis of an n-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$ where $\lambda_1(\Lambda) \leq \zeta(n)$

2.) $\min_i \|\mathbf{b}_i\| \geq 1$

3.) $1 \leq d \leq \zeta(n)/\gamma(n)$

is the problem of determining whether $\lambda_1(\Lambda) \leq d$. In particular, it requires an output of 1 if $\lambda_1(\Lambda) \leq d$ and an output of 0 if $\lambda_1(\Lambda) > \gamma(n) \cdot d$.

Note that for any $\zeta(n) \geq 2^{n/2}$, $\texttt{GapSVP}_{\zeta,\gamma}$ is equivalent to the standard shortest vector problem $\texttt{GapSVP}_{\gamma}$. This is because, by the Lensta-Lenstra-Lovász algorithm [LLL82], an arbitrary basis $\mathbf{B}$ for a lattice Λ can be reduced in polynomial time to another basis $\mathbf{B}^{'}$ that spans the same lattice, and such that

$$\lambda_1(\Lambda) \leq \|\mathbf{b}_1^{'}\| \leq 2^{n/2} \cdot \min_i \|\mathbf{b}_i\|$$

Since, we can always scale the original lattice basis $\mathbf{B}$ such that $\min_i \|\mathbf{b}_i\| = 1$, we can reduce any instance of the shortest vector problem to one where the minimum distance is guaranteed to be less than $2^{n/2}$. This makes the bound imposed by condition (1) trivially true. We don't have to worry about conditions (2) and (3) because we can always scale bases so that (2) is true, and the solution is trivial if (3) does not hold (and both (1) and (2) hold).

Before moving on to the reduction, we state a few lemmas that will be used. For a more comprehensive treatment of the subject matter of these lemmas, see [Pei09]. For notation, let $B_n \subset \mathbb{R}^n$ denote the open unit ball (in the l_2 norm), i.e. the set $B_n = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| \leq 1\}$.

Lemma 2.4.6: Let $\Delta(X, Y)$ denote the statistical difference between distributions X and Y, and $U(S)$ denote the uniform distribution over a set S. For any constants $c, d > 0$ and any $\mathbf{z} \in \mathbb{R}^n$ with $\|\mathbf{z}\| \leq d$, and $d^{'} = d \cdot \sqrt{n/(c\log(n))}$, we get that

$$\Delta(U(d^{'} \cdot B_n), U(\mathbf{z} + d^{'} \cdot B_n)) \leq 1 - \frac{1}{\text{poly}(n)}$$

Proof. See [GG98] for a proof □

We also use the concept of the smoothing parameter from [MR07]. Intuitively, the smoothing parameter is the smallest r that "smooths out" the discrete structure of Λ, up to some statistical error ϵ.

Definiton 2.4.7: For an n-dimensional lattice Λ and positive real number $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest r such that the total Gaussian mass $\rho_{1/r}(\Lambda^* \setminus \{\mathbf{0}\}) = \sum_{\mathbf{w} \in \Lambda^*, \mathbf{w} \neq 0} \rho_{1/r}(\mathbf{w}) \leq \epsilon$, where $\rho_s(\mathbf{x}) = \exp(-\pi\|\mathbf{x}\|^2/s^2)$.

Lemma 2.4.8: For any n-dimensional lattice Λ, we have $\eta_{2^{-n}}(\Lambda) \leq \sqrt{n}/\lambda(\Lambda^*)$

Proof. See Lemma 3.2 in [MR07] □

Lemma 2.4.9: Let $\epsilon = \epsilon(n)$ be a negligible function, $q = q(n) \geq 2$ be an integer, $\alpha = \alpha(n) \in (0,1)$ and $\phi = \Psi_\alpha$. There exists a classical probabilistic polynomial-time reduction $R^{W,D}(\mathbf{B}, r, \mathbf{x})$ that, given as input any basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$, and real $r \geq \sqrt{q} \cdot \eta_\epsilon(\Lambda^*)$, and any target point $\mathbf{x}$ within distance $\alpha q/(\sqrt{2}r) < \lambda_1(\Lambda)/2$ of Λ and given access to

1.) An oracle W that solves $\mathtt{LWE}_{q,\phi}$ using $\mathrm{poly}(n)$ samples

2.) An oracle D that generates samples from $D_{\Lambda^*,r}$

is able to finds the unique $\mathbf{v} \in \Lambda$ closest to $\mathbf{x}$ with overwhelming probability

Proof. See Lemma 3.4 in [Reg05] □

Theorem 2.4.10 : Let $\alpha = \alpha(n) \in (0,1)$ and $\gamma = \gamma(n) \geq n/(\alpha\sqrt{\log(n)})$. Let $\zeta = \zeta(n) \geq \gamma(n)$ and $q = q(n) \geq \zeta(n) \cdot \omega(\sqrt{\log(n)/n})$. Then, there is a probabilistic polynomial-time reduction from solving $\mathtt{GapSVP}_{\zeta,\gamma}$ in the worst case (with overwhelming probability) to solving $\mathtt{LWE}_{q,\Psi_a}$ using $\mathrm{poly}(n)$ samples.

Proof. Suppose we are given an instance of $\mathtt{GapSVP}_{\zeta,\gamma}(\mathbf{B}, d)$. Let $\Lambda = \mathcal{L}(\mathbf{B})$. We run the following reduction:

1.) Chose a point $\mathbf{w}$ uniformly at random from the ball $d' \cdot B_n$, where $d' = d \cdot \sqrt{n/(4\log(n))}$. Let $\mathbf{x} = \mathbf{w} \bmod \mathbf{B}$, i.e. $\mathbf{x}$ is the unique $\mathbf{x} \in \{\sum_{i=1}^n c_i \mathbf{b}_i \mid c_i \in [-1/2, 1/2)\}$ such that $\mathbf{w} - \mathbf{x} \in \mathcal{L}(B)$.

2.) Use the reduction R from the previous lemma on $\mathbf{B}, \mathbf{x}$, and $r = q \cdot \sqrt{2n}/(\gamma \cdot d)$, where the fact that we have an oracle D is given in Proposition 2.13 in [Pei09], and the oracle W is the given oracle for LWE. Let $\mathbf{v}$ be the output of R.

3.) Repeat steps (1) and (2) for $N = \mathrm{poly}(n)$ iterations. If $\mathbf{v} \neq \mathbf{x} - \mathbf{w}$ in any of the N iterations, then return 1 (i.e. accept). Otherwise, return 0 (i.e. reject).

To analyze the reduction, we consider two cases. If $(\mathbf{B}, d)$ is such that $\lambda_1(\Lambda) > \gamma(n) \cdot d$ and requires an output of 0, then by Lemma 2.4.8, we have

$$\eta_\epsilon(\Lambda^*) \leq \frac{\sqrt{n}}{\gamma \cdot d} \qquad \text{(Since } (\Lambda^*)^* = \Lambda\text{)}$$

for $\epsilon(n) = 2^{-n} = \mathtt{negl}(n)$. Hence, our chosen r satisfies $r \geq \sqrt{2}q \cdot \eta_\epsilon(\Lambda^*)$, as required by Lemma 2.4.9. Because $\mathbf{x} - \mathbf{w} \in \Lambda$, the distance from $\mathbf{x}$ to Λ is at most

$$d' = d \cdot \sqrt{\frac{n}{4\log(n)}} \leq \frac{\alpha \cdot d \cdot \gamma}{\sqrt{4n}} = \frac{\alpha q}{\sqrt{2}r}$$

by the hypothesis on γ and the definition of r. Moreover, $\lambda_1(\Lambda) > \gamma \cdot d > 2d'$, so R returns $\mathbf{v} = \mathbf{x} - \mathbf{w}$ in each of the iterations with overwhelming probability, and we return 0 (i.e. reject) as desired.

On the other hand, if $(\mathbf{B}, d)$ is such that $\lambda_1(\Lambda) \leq d$ and requires an output of 1, then let $z \in \Lambda$ be such that $\|z\| = \lambda_1(\Lambda)$. Consider an alternate experiment where the randomly chosen point $\mathbf{w}$ is replaced by

$\mathbf{w}' = \mathbf{z} + \mathbf{w}$, where $\mathbf{w}$ is still chosen uniformly from $d' \cdot B_n$. Let $\mathbf{x}' = \mathbf{w}' \bmod \mathbf{B}$. When R is invoked on $\mathbf{x}'$, by Lemma 2.4.6, as well as the fact that statistical distance cannot increase under any randomized function, we get that

$$\begin{aligned} \Pr[R(\mathbf{x}) = \mathbf{x} - \mathbf{w}] &\leq 1 - \frac{1}{\text{poly}(n)} + \Pr[R(\mathbf{x}') = \mathbf{x}' - \mathbf{w}'] \\ &\leq 2 - \frac{1}{\text{poly}(n)} - \Pr[R(\mathbf{x}' = \mathbf{x}' - \mathbf{w}] \\ \Rightarrow \Pr[R(\mathbf{x}) = \mathbf{x} - \mathbf{w}] &\leq 1 - \frac{1}{\text{poly}(n)} \end{aligned}$$

Thus, if we run the experiment sufficiently many, i.e. $N = \text{poly}(n)$ times, we will have $\mathbf{v} \neq \mathbf{x} - \mathbf{w}$ in at least one iteration with high probability, and will thus return 1 (i.e. accept) as desired. □

However, [Pei09] only works for LWE problems with large moduli $q \geq 2^{n/2}$. Later, [BLP+13] proved a similar reduction that works for polynomial moduli. What these results show is that `GapSVP` can be reduced to LWE.

2.5 Encoding Real Numbers

An important technical detail that should be discussed when considering applications of somewhat/fully homomorphic encryption schemes to machine learning algorithms is the encoding of real numbers. Most homomorphic encryption schemes use $\mathbb{Z}_q$ for some q as their message space. Therefore, they are naïvely unable to encrypt real values. In this section, we briefly explore a method to encode real numbers employed in the machine learning applications used in [AHPW15], [GLN12].

In [AHPW15], real numbers are encoded are as signed bit vectors of size $L + l + 1$ bits, and a real number $a \in \mathbb{R}$ is represented as

$$a = \sum_{k=-L}^{l} a_k 2^k$$

where $a_k \in \{0, 1\}$ if $a \geq 0$ and $a_k \in \{0, -1\}$ if $a < 0$. For operations on real numbers with this encoding we have

$$\begin{aligned} a + b &= \sum_{k=-L}^{l} (a_k + b_k) 2^k = \langle [2^{-L}, ..., 2^0, ..., 2^l], (\texttt{Bits}(a) + \texttt{Bits}(b)) \rangle \\ ab &= \langle [2^{-L}, ..., 2^0, ..., 2^l], \texttt{Bits}(a) \rangle \cdot \langle \texttt{Bits}(b), [2^{-L}, ..., 2^0, ..., 2^l] \rangle \\ &= [2^{-L}, ..., 2^0, ..., 2^l] \cdot \langle \texttt{Bits}(a), \texttt{Bits}(b) \rangle \cdot [2^{-L}, ..., 2^0, ..., 2^l] \end{aligned}$$

Note that this encoding scheme is flexible, as we can choose L beforehand to achieve the desired decimal precision and l to set our maximum integer value. One might worry that this loss of precision may negatively impact the predictive performance of our machine learning algorithms. However, Tschiatschek et al. explored the impact of precision loss on algorithms like Bayesian Network Classifiers and found no significant losses in performance provided that probabilities are not too close to 0 or 1, and the posterior probabilities over the classes are sufficiently different [TRMP12] . This suggests that there are machine learning algorithms that are able to tolerate some level of loss in precision.

Chapter 3

Applications to Machine Learning

In this chapter, we will explore the implementations of popular machine learning algorithms using homomorphic encryption schemes. The ability to implement machine learning algorithms on top of homomorphic encryption schemes is useful because it allows us to perform powerful analytics techniques over encrypted data. For instance, in our earlier example with the small college, we may not just want to compute the average GPA, but also find predictors of good grades for students, or classify students into various categories, all while maintaining the privacy of the data. Note that there is a related problem of not being able to infer information about particular entries in the training set from the results of a particular machine learning algorithm. This problem is that of differential privacy, and will not be considered in this thesis. Instead, the goal of homomorphic encryption is to allow encrypted over encrypted data without revealing any details about the data.

Although there exist fully homomorphic encryption schemes, somewhat homomorphic encryption schemes are sometimes sufficient to implement all the operations needed for certain machine learning algorithms. In other cases, we may use fully homomorphic encryption schemes or make slight modifications to the algorithm to work with somewhat homomorphic schemes [GLN12]. For some sections, we also present the corresponding proofs of security for the respective machine learning algorithms.

3.1 Binary Classification

Recall that the goal of Fisher's Linear Discriminant is to find a vector $\mathbf{w}$ that we can project the data onto that maximises the ratio between the inter-class variances and intra-class variances. Following the presentation in [Bis07], we derived the optimal choice of $\mathbf{w}$ as:

$$\mathbf{w}^* \propto \mathbf{S}_W^{-1}(\mathbf{u}_2 - \mathbf{u}_1)$$

and in particular, if we want to classify, we need to first calculate $c = \mathbf{w}^{*T}(\mathbf{u}_1 + \mathbf{u}_2)/2$ where $(\mathbf{u}_1 + \mathbf{u}_2)/2$ is the midpoint between the two means in the direction given by $\mathbf{w}$. Then, we can classify point $\mathbf{x}_k$ by comparing $\mathbf{w}^{*T}\mathbf{x}_k$ to c.

Although [GLN12] do not specify a particular encryption scheme, they do mention that they intend for this algorithm to work on a somewhat homomorphic encryption scheme that supports additions and limited

multiplications. As a result, they define the notion of an algorithm being D-polynomial, which means that the algorithm can be expressed as a polynomial of bounded degree.

However, as we can see, determining this optimal $\mathbf{w}$ requires inverting the matrix $\mathbf{S}_W^{-1}$, which in turn requires division and thus, is not D-polynomial [GLN12]. In [GLN12], they present a least-squares approach to computing an approximation of the optimal vector $\mathbf{w}^*$. The idea is that we can define an error function and minimize that using algorithms like gradient descent.

In more detail, let $\mathbf{d} = \mathbf{u}_2 - \mathbf{u}_1$ be the difference in class-conditional means. Then, suppose we try to minimise a sensible cost function (note that the optimal $\mathbf{w} = \mathbf{S}_W^{-1}\mathbf{d} = \mathbf{S}_W^{-1}(\mathbf{u}_2 - \mathbf{u}_1)$).

$$E(\mathbf{w}) = \frac{1}{2}\|\mathbf{S}_W\mathbf{w} - \mathbf{d}\|$$

with the exception that instead of the standard Euclidean norm, as in [GLN12], we use $\|\mathbf{v}\|^2 = \mathbf{v}^T\mathbf{S}_W\mathbf{v}$ for better conditioning. To minimize this cost function, we first take the gradient, giving

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \mathbf{S}_W\mathbf{w} - \mathbf{d}$$

Thus, if we apply gradient descent with a learning rate η, we find that at each step, we do

$$\begin{aligned}\mathbf{w}_{i+1} &= \mathbf{w}_i - \eta(\mathbf{S}_W\mathbf{w}_i + \mathbf{d}) \\ &= (\mathbf{I} - \eta\mathbf{S}_W)\mathbf{w}_i - \eta\mathbf{d}\end{aligned}$$

Then, if we start with $\mathbf{w}_0 = 0$, we can analyse the recurrence to see that the k^{th} approximation for $\mathbf{w}^*$ is

$$\mathbf{w}_k = \eta\sum_{i=0}^{k-1}(\mathbf{I} - \eta\mathbf{S}_W)^i\mathbf{d}$$

This will converge if the spectral radius (or the magnitude of the eigenvalue corresponding to its dominant eigenvector) of $\mathbf{I} - \eta\mathbf{S}_W$ is less than 1. However, we can always achieve this by picking η sufficiently small. It is easy to see that this algorithm is D polynomial with $S = 2(k-1) + 1$.

3.2 Linear Regression

In this section, we explore how we can implement linear regression, a very foundational machine learning algorithm, using a homomorphic encryption scheme. Note that the earlier construction shows how we can a gradient descent algorithm for Fisher's Linear Discriminant and that can also be applied to linear regression, but we will not go further into that. The construction we explore here is due to [WH12] and allows us to perform the regression on the encrypted data. They implement a fully homomorphic encryption scheme similar to that discussed in [Bra12]. Thus, their limitations are not so much on the operations they can perform, but rather on the performance overhead of using a fully homomorphic encryption scheme.

Recall briefly that in linear regression, we model the output variable as a linear function of the input variables plus a normally distributed error term, i.e.

$$y = \langle\beta, \mathbf{x}\rangle + \epsilon \qquad (\epsilon \sim \mathcal{N}(0,1))$$

To solve for the β, we compute the β that minimizes the least squares error. Recall that if X denotes the $d \times n$ design matrix, then the closed form solution to the above optimization problem is

$$\beta^* = (X^T X)^{-1} X^T y$$

Note that matrix multiplications and matrix-vector multiplications can be handled easily by somewhat homomorphic encryption schemes such as the Paillier cryptosystem because they support addition and multiplication. However, matrix inversion is a little more problematic. Fortunately, [WH12] mention that we can use Cramer's Rule to compute these inverses. First we mention Cramer's Rule

Theorem 3.2.1 (Cramer's Rule): Let A be an $n \times n$ non-singular matrix, and $\mathrm{Adj}(A)$ be its adjugate matrix. Then, its inverse is given by

$$A^{-1} = \frac{1}{\det(A)} \mathrm{Adj}(A)$$

Proof. For a proof, see [Cra50] □

Note that computing the adjugate of a matrix and determinant of matrix can be achieved with addition and multiplication. As for the division by the determinant, if we use one of the schemes for encoding real numbers, we should be able to divide by $\det(A)$ if we know beforehand how large $\det(A)$ is, so that we can set our encoding to be able to represent $1/\det(A)$. What this means is that our cryptosystem would have to calculate $\det(X^T X)$ before specifying an encoding scheme. One way this can be done is that the client can compute $\det(X^T X)$ and send it the server in encrypted form. Using fast matrix multiplication, we can compute the determinant of an $n \times n$ matrix in $O(n^{2.376})$ time using Dodgson condensation and Strassen's algorithm for matrix multiplication [Dod67] [Ski98]. However, if we realize that we don't necessarily need the exact determinant, but rather just an upper bound, then we can use Hadamard's inequality. Specifically, if a matrix A has N columns $\{v_1, ..., v_N\}$, then its determinant is bounded by

$$|\det(A)| \leq \prod_{i=1}^{n} \|v_i\|$$

This can be computed in $O(n^2)$ time, but sacrifices the extra space needed to store a potentially unnecessarily large encoding or real numbers, albeit on the server.

Alternatively, [WH12] suggests that the third-party server can return the numerators and denominators separately and have the client perform the division. Although not ideal, this solution should suffice in most practical situations since division by a known constant is not a costly operation.

As mentioned before, since [WH12] use a fully homomorphic encryption scheme, performance is a major concern for them. In particular, multiplications are a costly operation because the fully homomorphic encryption scheme in [Bra12] is based on the Ring Learning With Errors (RLWE) assumption, and the error terms grows faster on multiplications. Encoding of integers is done using bit vectors, i.e. a bitwise representation of $(a_0, ..., a_k)$ corresponds to $\sum_{i=0}^{k} a_i 2^i$. This means that addition and multiplication translates to polynomial addition and multiplication. One constraint that [WH12] have to address is

ensuring that the coefficients of the resulting polynomial do not grow too large. To deal with this, they make clever use of the Chinese Remainder Theorem, as stated below:

Theorem 3.2.2 (Chinese Remainder Theorem): Suppose we have an integer n with prime factorization $n = \prod_{i=1}^{k} p_i^{e_i}$. Then, we have the following isomorphism

$$\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/p_1^{e_1}\mathbb{Z} \oplus ... \oplus \mathbb{Z}/p_k^{e_k}\mathbb{Z}$$

The Chinese Remainder Theorem is useful for [WH12] because it allows them to support computations over $\mathbb{Z}/n\mathbb{Z}$ for large n. Basically, they take $n = \prod_{i=1}^{k} p_i$, a product of many small prime factors. Then, to evaluate the arithmetic circuit over $\mathbb{Z}/n\mathbb{Z}$, they can just evaluate it over $\mathbb{Z}/p_1\mathbb{Z}, ..., \mathbb{Z}/p_k\mathbb{Z}$ and apply the Chinese Remainder Theorem to recover the result in $\mathbb{Z}/n\mathbb{Z}$.

They also make use of the Chinese Remainder Theorem for polynomials to batch multiplications. In particular, they can pack multiple plaintext messages in one ciphertext block. Basically, the plaintext space can be broken up into a series of non-interacting plaintext slots, i.e. a single message m can encode messages $m_0, ...m_k$. Then, if we have another packed message $m^{'}$ encoding $m_0^{'}, ..., m_k^{'}$, we can compute $m + m^{'}$ or $m \cdot m^{'}$ to compute k sums/products with a single homomorphic addition/multiplication. One instance where this becomes very useful is when evaluating matrix products, which requires computing a lot of inner products. With this scheme, we can compute an inner product with a single multiplication operation. To then add the entries in the vector, they use an algorithm in [BGV12a] to permute the entries and add all these permutations such that the first entries is the desired sum.

3.3 Principal Component Analysis

As discussed earlier, principal component analysis (PCA) is a very important tool for machine learning. In particular, it is used heavily in exploratory data analysis and dimensionality reduction. Recall that the idea behind PCA is to find the vectors along which the variance of the given data projected onto those vectors is maximized. We can calculate these vectors by computing the eigenvectors of the covariance matrix of the data. In particular, we know that the eigenvector whose eigenvalue has the i^{th} largest absolute value is the i^{th} vector along which the data's variance is maximized [Jol02]. As a result, the eigenvector with the largest absolute eigenvalue, called the dominant eigenvector is that which maximizes projected variance.

Pereira and Aranha demonstrate an algorithm to perform PCA over encrypted data [PA15]. In particular, they acknowledge the performance overhead of using a fully homomorphic encryption scheme, and adapt their implementation of PCA to work with a somewhat homomorphic encryption scheme that supports an arbitrary number of additions and a few multiplications. Note that they also use division by fixed constants, which is allowed by the method of encoding real numbers discussed earlier. In this section, we present the algorithm and prove its correctness.

Because PCA involves extracting the dominant eigenvectors of the covariance matrix, we will be using some basic results from linear algebra that we state below.

Theorem 3.3.1 (Spectral Theorem): Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix. Then, we can write $A = UDU^T$, where U is an orthogonal matrix with its columns as the normalized eigenvectors of A, and D is a diagonal matrix, where the i^{th} entry on the diagonal is the eigenvalue associated with the i^{th} column of U.

Proof. For a proof, refer to any suitable linear algebra text, such as [Axl95]. □

Corollary 3.3.2: Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix. If we denote the eigenvalues and corresponding eigenvectors of A as $(\lambda_1, v_1), ..., (\lambda_n, v_n)$, then A may be written as

$$A = \sum_{i=1}^{n} \lambda_i v_i v_i^T$$

3.3.1 The Power Method

To find a dominant eigenvector of a matrix, we can use the power method. Given a real matrix $A \in \mathbb{R}^{n \times n}$, we can sample a vector $u \in \mathbb{R}^n$ and then repeated multiple u by A, thereby giving the sequence $Au, A^2u, ...$ We now prove the following claim:

Proposition 3.3.3: The sequence $Au, A^2u, ...$ converges to the dominant eigenvector of A up to a scalar multiple.

Proof. Let $v_1, ..., v_n$ be the eigenvectors of A arranged by descending absolute value, so v_1 is the dominant eigenvector. Then we can write $u = \sum_{i=1}^{n} c_i v_i$. Then we can write $A^k u$ as

$$\begin{aligned} A^k u &= A^k \left(\sum_{i=1}^{n} c_i v_i \right) \\ &= \sum_{i=1}^{n} \lambda_i^k c_i v_i \\ \Rightarrow \frac{A^k u}{\lambda_1^k} &= c_1 v_1 + \sum_{i=2}^{n} \frac{\lambda_i^k}{\lambda_1^k} c_i v_i \end{aligned}$$

Because v_1 is the dominant eigenvalue, $|\lambda_1| > |\lambda_i|$ for $i > 1$. Thus, as $k \to \infty$, we get that

$$\lim_{k \to \infty} \frac{A^k u}{\lambda_1^k} = c_1 v_1$$

We can then normalize the resulting vector if necessary. □

This proposition means that we can use the following algorithm to obtain the dominant eigenvector of a matrix A as follows:
Of course, this method only works if the dominant eigenvector is strictly dominant (i.e. $\lambda_1 > \lambda_2$) and not weakly dominant. We also require that u, the random vector has some component in the direction of v_1. These event are low probability events, so we omit the discussion of these complications.

Note that in each step, we perform an $O(n^2)$ operation (i.e. matrix-vector multiplication) and the number of steps to convergence depends on the ratio $|\lambda_2/\lambda_1|$. If this is close to 1, then the algorithm will converge more slowly, and if close to 0, it will converge more quickly.

Algorithm 7 Power Method

```
1: procedure POWERMETHOD(A)
2:     /* set N to be the number of rows/cols of A */
3:     N = dim(A)
4:     u = randomVector(N)
5:     for i = 1 to NUM_STEPS do
6:         u = (1/θ_i) Au
7:     end for
8:     return u
9: end procedure
```

3.3.2 Applying the Power Method

The power method allows us to find the dominant eigenvector (i.e. first principal component) of a given matrix. However, in general, we may have to find the k principal components of the covariance matrix. To solve this, Prereira and Aranha present a method, eigenShift, to convert the original matrix A into one whose second dominant eigenvalue becomes the dominant eigenvalue.

Algorithm 8 EigenShift Operation to Remove Dominant Eigenvector

```
1: procedure EIGENSHIFT(A, v /* dominant eigenvector */)
2:     α = ⟨v, v⟩
3:     Compute B = αA − Avv^T
4:     return B
5: end procedure
```

Note that above, instead of normalizing v, we scale the whole computation of B by $\alpha = \langle v, v \rangle$. This is to remove the need to divide by the norm of v, since our encryption scheme is not capable of division by values that are not known constants. To show correctness, we must prove the following:

Theorem 3.3.4 : Let A be an $n \times n$ real symmetric matrix. Then, eigenShift shifts the eigenvalues of A. In particular, the eigenvalue associated with the dominant eigenvector of A becomes 0.

Proof. We can see that $u = v_1$, since we normalise the dominant eigenvector. Thus,

$$\begin{aligned} Bu &= Au - (Auu^T)u \\ &= Au - Au(u^T u) && \text{(matrix operations are associative)} \\ &= Au - Au\|u\| \\ &= Au - Au && (\|u\| = 1) \\ &= 0 \end{aligned}$$

Hence, $u = v_1$ is an eigenvector of B with eigenvalue 0. By Corollary 3.3.2, we know

$$\begin{aligned} A &= \sum_{i=1}^{n} \lambda_i v_i v_i^T \\ \Rightarrow B &= A - Av_1 v_1^T \\ &= A - \lambda_1 v_1 v_1^T \\ &= \lambda_2 v_2 v_2^T + ... + \lambda v_n v_n^T \end{aligned}$$

Now we want to show that the eigenvectors (other than the dominant eigenvector) of B are the same as those for A and have the same eigenvalues. Observe that for $i \in \{2, ..., n\}$,

$$\begin{aligned} Bv_j &= \sum_{i=2}^{n} \lambda_i v_i v_i^T v_j \\ &= \lambda_j v_j \qquad (v_i^T v_j = 0 \text{ if } i \neq j \text{ and } 1 \text{ otherwise}) \end{aligned}$$

Hence, A and B share all the same eigenvectors and eigenvalues with the exception of A's dominant eigenvector, which has eigenvalue 0 for B. As a result, the second dominant eigenvalue of A is the dominant eigenvalue of B □

With the EigenShift and PowerMethod algorithms, we are now in a position to perform PCA, since we can extract the dominant eigenvector of the covariance matrix, and then transform that matrix so that the second dominant eigenvector becomes the dominant eigenvector and so on. More precisely, we can use the following algorithm:

Algorithm 9 PCA using PowerMethod and Eigenshift

```
1:  procedure PCA(A, v /* dominant eigenvector */)
2:      Σ = covariance matrix, K = number of principal eigenvectors desired
3:      principal_components = []
4:      for i = 1 to K do
5:          principal_component_i = powerMethod(Σ)
6:          principal_components.append(principal_component_i)
7:          Σ = eigenShift(Σ)
8:      end for
9:      return principal_components
10: end procedure
```

3.4 Binary Decision Trees

As mentioned earlier, decision trees are commonly used for regression or classification. In this section, we outline a method presented in [BPTG14] to evaluate binary decision trees used for classification. The key idea is that we can express the classification of these decision trees as polynomials. More precisely, this polynomial uses the values of all the booleans in the decision tree's nodes and the class labels on its leaves to return the classification result. Then, because polynomials only involve additions and multiplication, they do not require a terribly sophisticated homomorphic encryption scheme to compute them on the ciphertexts privately.

3.4.1 Expressing Decision Trees as Polynomials

Consider a decision tree where at each node, we make a boolean decision b_i and at the leaves, we have classes labelled c_i. One such example is given is Figure 3.1. We can see that the outcome of following this decision tree is equivalent to evaluating the polynomial

$$P(b_1, ..., b_4, c_1, ...c_5) = b_1 \cdot (b_3 \cdot (b_4 \cdot c_5 + (1 - b_4) \cdot c_4) + (1 - b_3) \cdot c_3) + (1 - b_1) \cdot (b_2 \cdot c_2 + (1 - b_2) \cdot c_1)$$

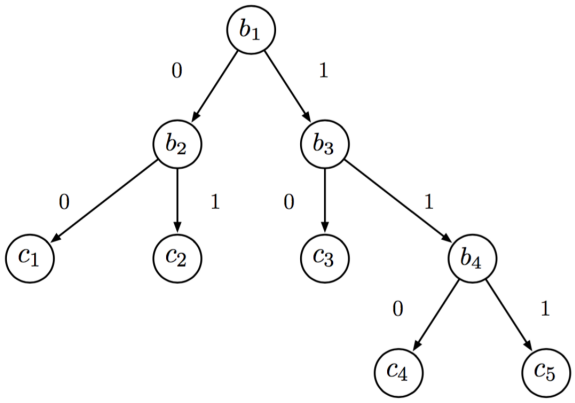


Figure 3.1: Example Classification Decision Tree, from [BPTG14]

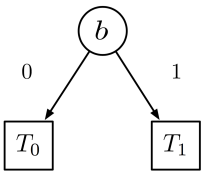
In [BPTG14], Bost, Popa, Tu & Goldwasser then present a recursive procedure for constructing the polynomial P given some binary decision tree T, which we outline below.

Algorithm 10 Transforming a Binary Decision Tree into a Polynomial [BPTG14]

```
1: procedure TRANSFORMTREE(root)
2:     if root is a class then
3:         return root
4:     else    /* root is a boolean */
5:         b = root
6:         T_0 = subtree if b = 0; T_1 = subtree if b = 1;
7:         return (1 − b) · TransformTree(T_0) + b · TransformTree(T_1)
8:     end if
9: end procedure
```

To illustrate the intuition behind Algorithm 10, we use the diagram below.

(a) Base Case: output c



(b) Recursive Case: output $(1 - b) \cdot \texttt{TransformTree}(T_0) + b \cdot \texttt{TransformTree}(T_1)$

Figure 3.2: Illustration of base and recursive cases for Algorithm 10 (`TransformTree`), from [BPTG14]

3.4.2 Efficiency & Security Considerations

In terms of concrete implementation details, [BPTG14] describe the following. Suppose we have our server S and client C. To obtain the boolean values $\{b_i\}_{i=1}^{n}$, the client C uses the garbling scheme given in [BHKR13], the short circuit for comparison in [KSS09], and the well-known oblivious transfer (OT) scheme given in [NP01]. C sends these in encrypted form to the server. The server S evaluates the polynomial on these inputs using the homomorphic properties of the encryption scheme.

There are many ways to further optimize this algorithm. One way that [BPTG14] describe is to lower the multiplicative depth of the given polynomials, since multiplications on a homomorphic encryption scheme are still a performance bottleneck. More concretely, suppose that our decision tree corresponds to evaluating a polynomial of degree d. Then, we will have to evaluate some d-degree term $a_1 \cdot ... \cdot a_d$. This has multiplicative depth d, which is slow on an FHE scheme, since the performance of FHE depends on the depth of multiplications in the circuit [BPTG14]. However, we can use tree-based evaluation and construct a binary tree over these values $a_1, ..., a_d$ and multiply them in pairs according to this tree. By doing so, we lower the multiplicative depth to $\lceil \log(d) \rceil$, making the fully homomorphic encryption scheme's evaluation substantially more efficient.

Other optimizations include employing a leveled fully homomorphic encryption scheme, and using SIMD slots, [BPTG14]. Now, we formally state the security theorem for this scheme.

Proposition 3.4.1: The Decision Tree Classifier is CPA-secure in the honest-but-curious model.

Proof. Interested readers can find the full proof in [BPTG14]. For succinctness, we only provide the intuition. Basically, because we are using a fully homomorphic encryption scheme, the server does not learn anything about the inputs that the client provides. Furthermore, the client also does not learn anything about the structure of the tree because the server performs the evaluation of the polynomial. □

3.5 Naïve Bayes Classifiers

Recall that if we want to perform classification using a Naïve Bayes classifier, then if we are given the priors probabilities for each class and the class conditional densities, we can use Bayes Rule to calculate

$$\begin{aligned} P(C = c^{'} \mid \mathbf{x}, \theta) &= \frac{P(\mathbf{x} \mid C = c^{'}, \theta) P(C = c^{'})}{P(\mathbf{x})} \\ &= \frac{P(\mathbf{x} \mid C = c^{'}, \theta) P(C = c^{'})}{\sum_c P(\mathbf{x} \mid C = c, \theta) P(C = c)} \end{aligned}$$

In the implementation of the Naïve Bayes Classifier in [BPTG14], the choice of homomorphic encryption scheme is the Paillier Encryption Scheme. Although it isn't fully homomorphic, its additive homomorphic properties are sufficient to implement this algorithm. Also because the Paillier Encryption schemes works over integers, we note that we use an encoding scheme for real numbers similar to that described earlier. In particular, we require that the modulus for the Paillier scheme N satisfies $\log(N) > l_{\max} + 1 + \lambda$, where $l_{\max}$ is the number of bits we need for computation and λ is the security parameter.

Now, let each our feature vector be $\mathbf{x} = (x_1, ..., x_n)$, and let feature x_j take on values in the domain D_j. We also assume we have the following tables:

- One table for the class log-priors, i.e. table P where the i^{th} entry $P(i) = \log(p(C = c_i))$
- kd tables: one per feature j per class i, i.e. table $T_{i,j}$ where the v^{th} entry $T_{i,j}(v) = \log(p(X_j = v \mid C = c_i))$ for all $v \in D_j$.

Before outlining the algorithm, we note that the situation used in the presentation of Naïve Bayes in [BPTG14] is slightly different from that we've seen in most of the other applications in this thesis. In particular, most of the other examples assume that all the data comes from the client and the goal is to keep it secret from the server. However, in their exposition of the secure Naïve Bayes classifier, [BPTG14] detail the situation where the server has the table of priors and class conditional densities for each feature (i.e. all $kd + 1$ tables detailed above). These tables are calculated from data that already exists on the server and not from the client. From a security standpoint, this isn't an issue because this data can be encrypted using the Paillier encryption scheme and a series of homomorphic operations can establish these tables. From a performance standpoint, note that these tables can be prepared in advance, so their cost can be amortized over several calls [BPTG14].

Another important difference is that the algorithm presented by [BPTG14] is interactive. As we can see below, the client computes the "log-probabilities" (up to a constant factor) that their given data point belongs to a particular class (encrypted under the Paillier encryption scheme), and then runs an argmax protocol with the client to determine which class is the most likely. The intuition behind the argmax protocol is that the server randomly permutes the data, has the client determine index of the largest value in the permuted array, and then inverts the permutation to compute the index of the largest value in the unpermuted array as i_0. That said, there are issues with the server learning the ordering of these values, but as mentioned, see [BPTG14] for a more thorough description of the argmax protocol.

Then, the algorithm to perform such a computation is:

Algorithm 11 Naïve Bayes Classifier[BPTG14]

procedure NAÏVEBAYES
    Client Input: $x = (x_1, ..., x_d) \in \mathbb{Z}^d$, public key pk_1, secret key sk_2
    Server Input: Secret key sk_1, public key pk_2, probability tables $\{\log(p(C = c_i))\}_{1 \leq i \leq k}$ and $\{\{\log(p(X_j = v \mid C = c_i))\}_{v \in D_j}\}_{1 \leq j \leq d, 1 \leq i \leq k}$
    Client's Output: i^* such that $p(x, c_{i^*})$ is maximized.

    The server prepares the tables P and $\{T_{i,j}\}_{1 \leq i \leq k, 1 \leq j \leq d}$
    The server sends $[[P]]$ and $\{[[T_{i,j}]]\}_{i,j}$ to the client
    For all $1 \leq i \leq k$, the client computes $[[\log(p_i)]] = [[P(i)]] + \sum_{j=1}^{d} [[T_{i,j}(x_j)]]$
    The client runs the argmax protocol with the server and gets $i^* = \arg\max_i p_i = \arg\max_i(\log(p_i))$
    C outputs i^*
end procedure

Hence, we see that computing over logarithms of probabilities in this case does not only provide us with better numerical stability, but in the case of homomorphic encryption also allows us to use somewhat homomorphic encryption schemes that allow addition like the Paillier cryptosystem.

Proposition 3.5.1: Algorithm 11 is secure in the honest-but-curious attack model

Proof. Refer to [BPTG14], but the intuition is that the Paillier cryptosystem is secure and because we only perform homomorphic operations, Algorithm 11 is also secure. □

Chapter 4

Alternatives & Limitations for Homomorphic Encryption

4.1 Limitations of Homomorphic Encryption

4.1.1 Computational Efficiency

One notable limitation of homomorphic encryption schemes, especially fully homomorphic ones is their (lack of) computational efficiency. Most current schemes are still too computationally intensive to implement in practice, although there are schemes that have been construction for the sole purpose of being implementable. For instance, an R package `HomomorphicEncryption` is able to conduct thousands of additions per second, but unfortunately, is only able to conduct about 50 multiplications per second [LJMA15]. As we can see, multiplication is still a significant bottleneck in this particular scheme, and so the computational cost is an important consideration when deciding whether or not to use homomorphic encryption for a particular application: machine learning or not.

However, Graepel et al. note that in most cases, a somewhat homomorphic encryption suffices for most cases [GLN12]. For instance, in their work, they make use of a somewhat homomorphic encryption scheme able to carry out additions and multiplications. Thus, they are restricted to algorithms that are involve evaluating polynomials with additions and multiplications. In their discussions of Fisher's Linear Discriminant, they discuss a Division-Free Integer (DFI) implementation of the algorithm, wherein they multiply all the terms by any possible denominator they could encounter during the algorithm, letting them carry out Fisher's Linear Discriminant with a limited somewhat homomorphic encryption scheme.

4.1.2 Security

As mentioned before, homomorphic encryption schemes are malleable by definition. Because of this, they are not secure against adaptive chosen-ciphertext attacks. Furthermore, throughout this thesis, we've assumed that third party servers follow an honest-but-curious model. What this means is that the servers follow instructions but try to acquire information about the data within these constraints. If however, we consider the situation where we deal with a different adversary model, homomorphic encryption faces some issues.

One such issue is that the server can intentionally compute a false response and send it back to the client. Aside from obvious issues with receiving a false result, the server could do so maliciously to try to prompt a certain response from the user in order to gain information about the encryption scheme. This isn't a massive concern because there exist schemes for delegating computation that allow the server to prove that they've performed the appropriate computations [CKV10]. This resolution also helps protect homomorphic encryption schemes against certain types of CCA attacks, because the server must prove that it has returned the correct result to the client, making it difficult for the adversary to manipulate the ciphertext it returns to elicit some response from the client.

4.1.3 Ethical Considerations

Another, broader critique of the morality of homomorphic encryption schemes is raised by Rogaway in [Rog15]. He notes that even though the notion of fully homomorphic encryption seems empowering, its impact on academic cryptography and possibly society as a whole may not be positive.

Firstly, Rogaway brings up the current lack of practicality of fully homomorphic encryption and the possibility that it may never be computationally efficient enough to be used outside academic contexts. Because fully homomorphic encryption, on the surface, is a very exciting avenue for research, many cryptography researchers have gravitated towards it, even though this may not be the area of cryptography that can generate the greatest amount of social good. A key premise of Rogaway's paper is that cryptographers should not only be academics but should be deeply aware of the political implications of their work. Thus, because fully homomorphic encryption encourages cryptographers to pursue something intellectually interesting but possibly inconsequential in practice, Rogaway argues that it has a negative impact on the cryptographic community.

As for broader societal implications, Rogaway notes that the idea of fully homomorphic encryption, especially when not fully understood, gives people a greater sense of security towards having their data stored in large central databases. However, in [Rog15], he cites a 2014 interview by Defense Advanced Research Projects Agency (DARPA) Program Director Dan Kaufman indicating that law enforcement intend to hold the keys to decrypt the data and use them if need be. In particular, the interview describes a situation where a homomorphic encryption scheme is used, allowing authorities to query encrypted records and find how many people that match certain criteria. Then, after obtaining approval from the relevant government bodies, the individual personal records can then be decrypted.

The interview cited in [Rog15] highlights an issue with homomorphic encryption in practice in that it grants a large amount of power to the party controlling the central database. Although we can obtain theoretical proofs of security for particular homomorphic encryption schemes, an individual would be better off from a privacy point of view by not allowing their data to exist on this server at all. But because the notion of homomorphic encryption suggests that in theory, keeping your data on such a server is safe, it encourages people to do so even if the underlying implementation does not achieve the privacy/security standards they should demand. This thesis does not take a stand on any privacy related issues, but we do note that the existence of homomorphic encryption schemes may encourage people to be comfortable

having their data stored on a third-party, which may not be as practically secure as they believe.

4.2 Yao's Garbled Circuits and Oblivious Transfer

In this section and the next, we outline two possible alternatives to homomorphic encryption schemes: Yao's Garbled Circuits and Functional Encryption. While these protocols aren't exact substitutes for homomorphic encryption schemes, there are certain cases where they are more suitable for the task at hand.

4.2.1 Motivation & Overview

Yao's garbled circuits provide us with a method to achieve secure computation across multiple parties. In particular, each party provides some inputs to the computation and the goal is for none of the parties to learn anything more about the inputs of the others than is absolutely necessary from knowing the result of the computation.

For simplicity, suppose we are considering 2-party computation between Alice and Bob. The overarching idea is that the desired computation can be modeled as a function, and in particular as a boolean circuit, i.e. if Alice has inputs $a_1, ...a_m$ and Bob has inputs $b_1, ..., b_n$, then this function is $f(a_1, ..., a_n, b_1, ..., b_n)$. On a high level, Yao's garbled circuits works by first having Alice hard code her inputs into the function, creating another function $g(b_1, ..., b_n) = f(A_1, ..., A_m, b_1, ..., b_n)$ which she will send to Bob in the form of a set of keys, of which Bob will select a subset, then evaluate some function of the keys to get an "output key". Then, Bob sends this "output key" back to Alice and she will determine the output of the shared computation.

Before we formally present the protocol, we must first discuss the notion of oblivious transfer, because this is how Bob will specify his inputs without Alice knowing.

4.2.2 Oblivious Transfer Protocols

Suppose we have two parties Alice and Bob, and that Alice is allowing Bob to acquire exactly one of two messages m_0 and m_1. The goal of oblivious transfer protocols is to allow this to occur under the following constraints:

- Alice cannot know which of m_0 or m_1 Bob chose to receive
- Bob cannot know anything about the message he did not choose (i.e. if he chose m_i, he cannot learn anything about m_{1-i} for $i \in \{0, 1\}$)

The first oblivious transfer scheme was constructed by Rabin in 1981 [Rab81], which allows one to send a message to a receiver with probability 1/2. However, this is not exactly what we want in this case, which is to allow Alice to send exactly 1 of 2 messages to Bob. To do this, we use an oblivious transfer protocol by Evan, Goldreich and Lempel [EGL85]. It involves using the RSA cryptosystem to encode Bob's choice of message, and a modified form of both messages m_0 and m_1. The example above describes a situation in which a 1-of-2 oblivious transfer protocol would be desired, since Bob is choosing one of

two options. However, these 1-of-2 oblivious transfer protocols have been generalized to 1-of-n and even k-of-n protocols. Furthermore, even though Rabin's construction is based on RSA, new protocols are being constructed based on other cryptographic primitives such as elliptic curve cryptography [Par06].

For the sake of concreteness, we present Even, Goldreich and Lempel's construction of a 1-of-2 oblivious transfer protocol below, using RSA as the public key encryption scheme:

- Alice wants Bob to receive 1 of 2 messages m_0, m_1
- Alice generates the RSA key (N, e, d) and sends the public keys (N, e) to Bob
- Alice generates two random messages x_0, x_1 and sends them to Bob
- Bob receives x_0, x_1, selects a bit $b \in \{0, 1\}$ and a random k
- Bob computes $v = x_b + k^e \bmod N$ and sends this to Alice
- Alice computes $k_0 = (v - x_0)^d \bmod N$ and $k_1 = (v - x_1)^d \bmod N$. One of these will equal Bob's chosen k but Alice will not know which
- Alice computes $m_0^{'} = m_0 + k_0$ and $m_1^{'} = m_1 + k + 1$ and sends these to Bob
- Bob decrypts $m_b = m_b^{'} - k$ with the bit he chose earlier. Note that if he tries to decrypt the other one, he will not recover m_{1-b}

4.2.3 The Protocol

Now that we've seen an example of oblivious transfer and can be convinced that it is possible, we can resume the presentation of Yao's garbled circuits. Once Alice has hardcoded her inputs into the function, we have the function $g(b_1, ..., b_n)$. We can represent this function as a Boolean circuit where Bob's inputs are input wires.

Then for every wire (input, intermediate, and output) w_i, Alice will come up with two keys K_i^0 and K_i^1 which correspond to the wire inputting a value of 0 or 1 respectively. Then Alice will construct a (very) large truth table for the Boolean circuit and encrypt the output for a particular set of inputs with the keys corresponding to those inputs. For instance, for an OR gate where Bob happens to choose both inputs w_i and w_j to get output w_k, Alice will send Bob the following 4 codes, because there are 4 entries in the truth table for an OR gate:

$$X_{0,0} = \mathtt{Enc}_{K_i^0}\left(\mathtt{Enc}_{K_j^0}(K_k^0)\right)$$
$$X_{0,1} = \mathtt{Enc}_{K_i^0}\left(\mathtt{Enc}_{K_j^1}(K_k^1)\right)$$
$$X_{1,0} = \mathtt{Enc}_{K_i^1}\left(\mathtt{Enc}_{K_j^0}(K_k^1)\right)$$
$$X_{1,1} = \mathtt{Enc}_{K_i^1}\left(\mathtt{Enc}_{K_j^1}(K_k^1)\right)$$

Then, Bob chooses keys corresponding to his input values using the a 1-of-2 oblivious transfer protocol. For the sake of concreteness, say Bob chooses his inputs to be $w_i = w_j = 0$, and so gets keys K_i^0 and K_j^0.

Then for each of the four X values he was given, he will try to decrypt using those keys, i.e. compute

$$\texttt{Dec}_{K_i^0}\left(\texttt{Dec}_{K_j^0}(X)\right)$$

The only one that will decrypt to a proper key is $X_{0,0}$ which will decrypt to K_k^0. Bob can then send this key back to Alice, who can determine the result of the computation. Of course, Bob cannot know which of the four decryptions he will obtain is the correct one. Thus, Alice must also supply Bob with authentication codes $\texttt{auth}(K_k^0)$ and $\texttt{auth}(K_k^1)$ to check which one is correct.

4.2.4 Comparison to Homomorphic Encryption

In this section, we consider some of the tradeoffs between using homomorphic encryption vs. Yao's garbled circuits. One complaint against Yao's garbled circuits is they are not reusable, so you may only evaluate one function on one set of inputs. Even though developments on these circuits have made them reusable [GKP+13], these developments are still based on homomorphic encryption.

An even more significant problem with Yao's garbled circuits is that the number of keys that need to be generated is proportional to the size of the circuit representation of the function to be evaluated. This is highly undesirable for a secure multiparty computation protocol. As Gentry outlines in his thesis [Gen09], if we wanted to search over some encrypted data, the user would have to send a circuit proportional to the size of his data. Therefore, when we are looking at applications over large data sets, such as doing computations over a database, fully homomorphic encryption may be preferable to Yao's garbled circuits.

That said, there are definitely applications where the size of the data is not so much the bottleneck as is the computation required. While it is difficult to generalize about the set of homomorphic encryption algorithms, most that currently exist suffer from performance issues. For instance, those based on the Learning With Errors assumption suffer from having to refresh the noise in the ciphertext to ensure that the ciphertext can still be decrypted. On the other hand, although Yao's garbled circuits involves potentially transmitting a lot of data, they tend to be less intensive since all we need to do is evaluate the decryption algorithm, albeit having to do it $O(ng)$ times where n is the number of inputs and g is the number of gates in the circuit. Thus, if our system is such that computation on the server is a bottleneck, but communication between client and server is not, then Yao's garbled circuits may be an alternative worth considering.

Finally, security is another reason for us to possibly prefer Yao's garbled circuits. Recall earlier that we characterized homomorphic encryption schemes as malleable, which put an inherent limitation on their security, i.e. they cannot be secure against adaptive chosen ciphertext attacks. Furthermore, in the case where we do not trust the third-party server, Yao's garbled circuits provides us with the ability to verify outputs, which FHE does not. As a result, one can argue that because Yao's garbled circuits preserves circuit privacy and is verifiable, there are situations where they provide more security guarantees than a homomorphic encryption scheme can.

4.3 Functional Encryption

4.3.1 Motivation & Overview

On a broad level, the idea behind a functional encryption scheme is to have the decryption function, instead of decrypting the ciphertext to get the plaintext m, decrypt to some function of the plaintext $f(m)$ without allowing the client (thought of here as the adversary) to learn anything about the plaintext m. Following the exposition in [GKP+13], we can define a functional encryption scheme more formally as

Definiton 4.3.1: A functional encryption scheme for a class of functions $\mathcal{F}$ is a tuple of four probabilistic polynomial time algorithms $(\texttt{Setup}, \texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ such that

- $\texttt{Setup}(1^\lambda)$ takes in the security parameter 1^λ and outputs a master public key mpk and master secret key msk
- $\texttt{KeyGen}(msk, f)$ takes in the master secret key and a function $f \in \mathcal{F}$, and outputs a function specific secret key sk_f
- $\texttt{Enc}(mpk, m)$ takes in the master public key and a message $m \in \{0,1\}^*$ and outputs a ciphertext c
- $\texttt{Dec}(sk_f, c)$ takes in a function specific secret key sk_f and a ciphertext c and outputs a value x

As we can see, we can think of a functional encryption scheme as a regular public-key encryption scheme, except that there a multiple secret decryption keys: one corresponding to each permissible function in the encryption scheme. For correctness, we require that for every message m and sufficiently large security parameter λ:

$$\Pr[\texttt{Dec}(sk_f, \texttt{Enc}(m)) \neq f(m)] \leq \texttt{negl}(n)$$

where the probability is taken over all the randomness in $\texttt{Enc}, \texttt{Dec}$.

Just as a brief note, the security definitions for functional encryption schemes are slightly different from those of regular encryption schemes. In particular, the security requirement is that the client does not learn anything about the message m other than the function $f(m)$. For more details on the security definitions of functional encryption schemes, we refer the reader to [GKP+13].

Since functional encryption schemes allow secure computation over encrypted data, we can see why functional encryption schemes could be an alternative to homomorphic encryption schemes. In [GKP+13], they construct a secure functional encryption scheme using a fully homomorphic encryption scheme, an attribute-based encryption scheme, and Yao's garbled circuits, demonstrating that they are attainable conditioning on fully homomorphic encryption and attribute-based encryption.

While it may seem questionable that we are comparing fully homomorphic encryption to something that employs fully homomorphic encryption in its construction, note that the usage of fully homomorphic encryption is unique to this particular construction. Alwen et al. show that under some assumptions, a Functional Encryption scheme that supports evaluation on two ciphertexts implies Fully Homomorphic Encryption, suggesting that functional encryption might be able to be instantiated without fully homomorphic encryption [ABF+13].

4.3.2 Comparison to Homomorphic Encryption

While functional encryption and homomorphic encryption share many similarities, they are subtly different in how they are designed to be used. This creates situations where functional encryption could be a preferable scheme to fully homomorphic encryption (even if the functional encryption scheme makes use of fully homomorphic encryption as a building block).

As an example, take a shared patient database across several hospitals. In particular, this database contains patient data from each of the participating hospitals, all encrypted under the same encryption scheme. If this were a homomorphic encryption scheme, any hospital with the decryption key would be able to evaluate any of the functions allowed by the encryption scheme (any function if it were a fully homomorphic encryption scheme). More specifically, every participating hospital will be allowed to evaluate the same functions on the encrypted data.

However, there are definitely situations wherein we don't want this to be the case. Perhaps one hospital is doing research and so should only be allowed to perform aggregate computations on the data in order to protect patient privacy, whereas another hospital needs patient information on the individual level to care for patients and handle administrative matters. Even within the same hospital, the functions that health practitioners need to compute differ from those that administrative and finance staff need to compute. A vanilla homomorphic encryption scheme would not support this functionality, but a functional encryption scheme would by selectively distribution function specific secret keys to the appropriate parties.

Of course, one can argue that restricting access to certain functions/methods should not be handled by the encryption scheme but rather by some permissions scheme on the database. Even so, sometimes different parties need to evaluate different functions on the same sets of data points, and the fact that functional encryption schemes can handle this make them seem more appropriate for this use case.

Chapter 5

Conclusion

From the topics explored in this thesis, we can come to the conclusion that fully homomorphic encryption presents some exciting possibilities for machine learning. The ability to perform these learning algorithms on encrypted data has applications in several areas: medicine and social networks to name a few. However, even though many homomorphic encryption schemes have been constructed and the area is widely researched, there are still many hurdles to be overcome in terms of improving computational efficiency. Exploring the use of homomorphic encryption outside of the honest-but-curious model is also another important avenue for the practical utility of this scheme.

Furthermore, there are also instances where fully homomorphic encryption may not be the most appropriate scheme for the use cases at hand. For instance, functional encryption presents an alternative that may be more suitable when different parties have different needs in terms of the functions they wish to evaluate. That said, as a theoretical construct, homomorphic encryption has given us a method to compute over encrypted data: something that on a first glance wouldn't even seem possible, and that in itself is quite remarkable.

Bibliography

[ABF+13] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. *Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, chapter On the Relationship between Functional Encryption, Obfuscation, and Fully Homomorphic Encryption, pages 65–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[AHPW15] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Fast and secure linear regression and biometric authentication with security update. *IACR Cryptology ePrint Archive*, 2015:692, 2015.

[Axl95] Sheldon Axler. *Linear Algebra Done Right*. 1995.

[BGV12a] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. In *In Innovations in Theoretical Computer Science*, 2012.

[BGV12b] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 478–492, Washington, DC, USA, 2013. IEEE Computer Society.

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2007.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 575–584, New York, NY, USA, 2013. ACM.

[BPTG14] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. Cryptology ePrint Archive, Report 2014/331, 2014. `http://eprint.iacr.org/`.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *In Advances in Cryptology - Crypto 2012, volume 7417 of Lecture*, 2012.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, pages 1–12, New York, NY, USA, 2014. ACM.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil Vadhan. *Advances in Cryptology – CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, chapter Improved Delegation of Computation Using Fully Homomorphic Encryption, pages 483–501. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[Cra50] Gabriel Cramer. Introduction à l'analyse des lignes courbes algébriques. 1750.

[CS04] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2004.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *SIAM Journal on Computing*, pages 542–552, 2000.

[Dod67] C.L. Dodgson. Condensation of determinants, being a new and brief method for computing their arithmetical values. *Proceedings of the Royal Society of London*, 15:150–155, 1867.

[DP97] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29(2-3):103–130, 1997.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6), 1985.

[Gen09] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, 2009.

[Gen10] Craig Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010.

[GG98] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 1–9, New York, NY, USA, 1998. ACM.

[GKP+13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 555–564. ACM, 2013.

[GLN12] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology – ICISC 2012, Lecture Notes in Computer Science, to appear*. Springer Verlag, December 2012.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. Cryptology ePrint Archive, Report 2013/340, 2013. `http://eprint.iacr.org/`.

[JLMS15] Christine Jost, Ha Lam, Alexander Maximov, and Ben J. M. Smeets. Encryption performance improvements of the paillier cryptosystem. *IACR Cryptology ePrint Archive*, 2015:864, 2015.

[Jol02] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics, 2002.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[KSS09] V. Kolesnikov, A.R. Sadeghi, and T. Schneider. How to combine homomorphic encryption and garbled circuits - improved circuits and computing the minimum distance efficiently. *1st International Workshop on Signal Processing in the EncrypEd Domain (SPEED '09)*, 2009.

[LJMA15] Chris C. Homes Louis J. M. Aslett, Pedro M. Esperanca. A review of homomorphic encryption and software tools for encrypted statistical machine learning. 2015.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LNV11] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, 2011.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. *EUROCRYPT*, pages 700–718, 2012.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

[Mur12] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. 2012.

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457, 2001.

[PA15] Hilder V.L. Pereira and Diego F. Aranha. Principal component analysis over encrypted data using homomorphic encryption. 2015.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptography*, EUROCRYPT '99, pages 223–228, 1999.

[Par06] Abhishek Parakh. Oblivious transfer using elliptic curves. In *Proceedings of the 15th International Conference on Computing*, pages 323–328. IEEE, 2006.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer, 1981.

[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, pages 84–93. ACM, 2005.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *In Proceedings of the 8th Theory of Cryptography Conference, TCC '11*, pages 219–234, 2011.

[Ski98] Steven S. Skiena. *The Algorithm Design Manual*. Springer-Verlag, 1998.

[TRMP12] Sebastian Tschiatschek, Peter Reinprecht, Manfred Mücke, and Franz Pernkopf. Bayesian network classifiers with reduced precision parameters. *Machine Learning and Knowledge Discovery in Databases*, pages 74–89, 2012.

[WH12] David Wu and Jacob Haven. Using homomorphic encryption for large scale statistical analysis. 2012.