# Privacy-Preserving Weighted Similarity Query over Encrypted Healthcare Data

by

Guojun Tang

**Bachelor of Computer Science, South China Normal University, 2019**

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF**

**Master of Computer Science**

In the Graduate Academic Unit of Computer Science

| | |
|---|---|
| Supervisor: | Rongxing Lu, Ph.D., Faculty of Computer Science |
| Examining Board: | Saqib Hakak, Ph.D., Faculty of Computer Science, Chair |
| | Sajjad Dadkhah, Ph.D., Faculty of Computer Science |
| | Zhen Lei, Ph.D., Department of Civil Engineering |

This thesis is accepted by the
Dean of Graduate Studies

**THE UNIVERSITY OF NEW BRUNSWICK**

**August 2022**

# Abstract

The development of smart applications in the healthcare area has aroused the exponential growth of healthcare data. Similarity range query, which is purposed to search for similar objects in a dataset with particular metrics, has been widely applied in a variety of practical applications, including the disease diagnosis scenario. As a result, the data owner of those healthcare data tends to outsource them to powerful cloud servers and the latter can provide query services for the doctors with similarity range query. However, for privacy concerns, the data owner may outsource the encrypted data instead of plaintexts to the cloud servers. Meanwhile, when using the data query service, sometimes query users may search the data based on their preference. To address the aforementioned issues, in this thesis, we propose WeightedSim, an efficient and privacy-preserving weighted similarity range query scheme for outsourced healthcare data. Specifically, we first develop an encrypted R-Tree index by utilizing the Symmetric Homomorphic Encryption (SHE) technique and then employ it to perform a weighted similarity range query under the two cloud servers model. We analyze the security of our scheme to be selectively secure when the SHE is semantically secure against CPA and also conduct extensive experiments to validate the scheme's efficacy.

# Dedication

To my family and teachers without whom, I would have achieved nothing.

# Acknowledgements

This work comes from countless people's support and guidance. First of all, I would like to express my sincere gratitude to my supervisor, Dr. Rongxing Lu. His kind personality, professional guidance, and wealth of knowledge have impacted me a lot. It is he that leads me into the academia and guide my journey on University of New Brunswick.

Also, I am immensely grateful to Dr. Mohammad Mamun, my supervisor in NRC, and Dr. Yandong Zheng, my labmate in UNB. They can always give me precious suggestions and meticulous comments on my work as well as my studies.

Thanks to my labmates, Dr. Yandong Zheng, Ms. Jiacheng Jin, Ms. Ellen Zhang, Mr. Yunguo Guan, Mr. Jiaqi Zhao, and Mr. Songnian Zhang. Their guidance and support helped me complete my studies.

Most importantly, I would like to show my deepest gratitude to my parents and sister, who have given me a decent life and unconditional love, as well as encouraged and supported me to chase my career.

Again, thanks to all. Without your support, this thesis would not have been possible.

# Table of Contents

# List of Tables

# List of Figures

# List of Symbols, Nomenclature or Abbreviations

| | |
|---|---|
| HC | Healthcare Center |
| IoT | Internet of Things |
| HE | Homomorphic Encryption |
| RLWE | Ring Learning with Errors |
| PHE | Partially Homomorphic Encryption |
| DoS | Denial of Service |
| SWHE | Somewhat Homomorphic Encryption |
| FHE | Fully Homomorphic Encryption |
| SHE | Symmetric Homomorphic Encryption |

# Chapter 1

# Introduction

The state-of-the-art technologies from IoT [1] and Artificial Intelligence [2] have led to an evolution of healthcare applications. Benefitting from those innovations in IoT and Artificial Intelligence, many healthcare scenarios have dramatic improvements, such as online medical primary diagnosis, electronic healthcare records, health monitoring, and precision medicine [1, 2, 3]. As a result, there has been a significant increase in healthcare data, requiring a solution for managing tremendous data. At the same time, the data owners of healthcare data, usually healthcare centers, are supposed to provide the data query service for doctors for various applications. Similarity query, which retrieves similar data with specific metrics, is one of the most popular data query service options. In the majority of instances, however, the healthcare center lacks the computing resources and storage capacity to accommodate such a high volume of query requests. To address the computing capabilities shortcoming, healthcare centers typically outsource their healthcare data to third-party cloud servers, which contain robust computing power and may provide a more feasible and reliable data query service. Nevertheless, due to the sensitivity of healthcare data privacy, it raises a new issue of how to prevent private information leakage from cloud servers, which may not be fully trusted. Therefore, the privacy-preserving similarity

range query over the healthcare data has become a notable topic. In this chapter, we will present the introduction to the similarity range query, cloud computing, and our problem statement, followed by our main contributions to this thesis and the organization of this thesis.

## 1.1 Similarity Query

Similarity query, which retrieves similar data with specific metrics, is one of the most popular data query service options. It has been widely used in a variety of real-world applications, including healthcare [4] and the smart grid [5]. Given a query object $q$ and specific data metrics, it can return a query result consisting of similar objects to $q$ in a dataset. In this section, we will introduce two primary types of similarity query: similarity range query and nearest neighbour query.

### 1.1.1 Similarity Range Query

As a fundamental type of similarity query, similarity range query is a data search technique to figure out all the data points less than or equal to a distance constraint in the database. Specifically, given a query point $q$, query threshold $\tau$, dataset $X$, and the data distance function $d(*,*)$ which calculates the distance between two data points in a specific data metric such as Euclidean distance or edit distance, we may formally define the similarity range query as follow [6]:

$$R(q,\tau) = \{x_i | x_i \in X; d(x_i, q) \leq \tau\}$$

An example of a Euclidean distance-based similarity range query on a two-dimensional platform is shown in Figure 1.1. The similarity range query may be applied to a variety of practical applications, such as the location-based service. In the two-dimensional platform, for instance, the data points may be represented as buildings.

When we seek to identify all buildings within a certain distance of our current position, we can set our location as the query point and use the similarity range query.

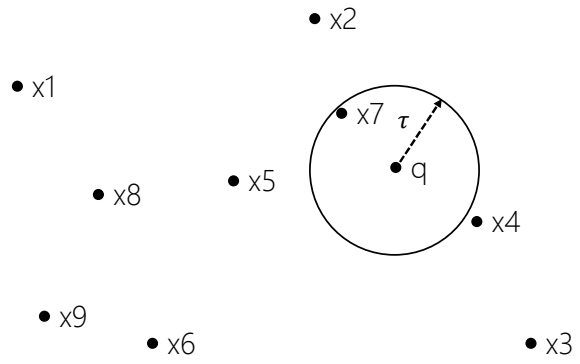


Figure 1.1: Example of Similarity Range Query

### 1.1.2 Nearest Neighbour Query

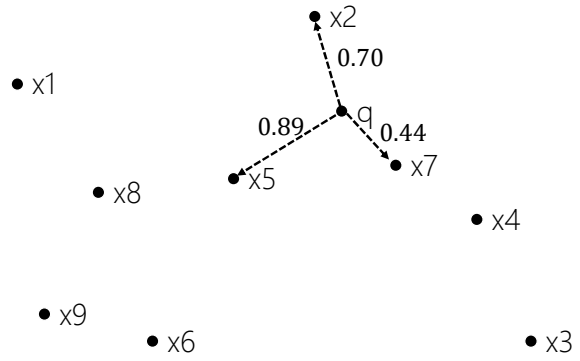


Figure 1.2: Example of 2NN query

The nearest neighbour query is the other typical form of similarity query, finding out the closest data point to a given data point. In practice, it can be extended to the k Nearest Neighbour ($kNN(q)$) query, which returns the k data points in a dataset that are closest to the query point. With the given query point $q$, dataset $X$, and the data distance function $d(*,*)$, we can define $kNN(q)$ as follow [6]:

$$kNN(q) = \{x_i | x_i \in S, S \subseteq X, |S| = k, \forall x_j \in X - S : d(q, x_i) \leq d(q, x_j)\}$$

An example about of Euclidean distance-based 2NN in a two-dimensional platform is shown in Figure 1.2. The kNN query may also be used for various practical applications. In the previous example, when we intend to identify the k-nearest buildings around our current position, we can set our location as the query point and apply the kNN query.

## 1.2 Cloud Computing

Cloud computing is an evolving paradigm that can provide more flexible and sustainable computing resources to users compared with the traditional computing model. As a result of its powerful computing power and huge storage capacity, it has become a viable solution to manage the booming data in various healthcare applications such as online diagnosis [3]. The definition of cloud computing from the National Institute of Standards and Technology (NIST) is shown in Definition 1.2.1.

**Definition 1.2.1** (Cloud Computing)**.** *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [7].*

Also, NIST outlines critical aspects of cloud computing, including five essential characteristics, three service models, and four deployment models (shown in Table 1.1).

- **Essential Characteristics**: There are five essential characteristics of cloud computing: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. On-demand self-service denotes that a consumer may use the cloud facility unilaterally without requiring human interaction with each service provider. Broad network access means that the

consumer can access the cloud facility through the network by using heterogeneous clients such as mobile phones or laptops. For the resource pooling characteristic, the provider may pool computing resources flexibly to serve multiple consumers by dynamically assigning and reassigning different physical and virtual resources depending on the consumer demand. Also, cloud computing demands rapid elasticity so that the cloud capability can be set up and released elastically to scale rapidly outward and inward commensurate with demand. And the cloud computing needs to satisfy the measured service requirement in which the cloud provider offers a metering capability for monitoring, controlling, and reporting the computing resource's utilization and provides transparency for both the provider and the consumer.

- **Service Models**: The service models of cloud computing mainly consist of three components: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS permits the consumer to use the provider's applications without managing or controlling the cloud infrastructure. PaaS allows the consumer the capability to deploy consumer-created or acquired applications by provider's programming tools. The consumer still does not have the ability to control or manage the cloud infrastructure but may configure the application. Compared with the previous two models, IaaS may permit the consumer to partly control computing resources such as storage and networks to deploy and run arbitrary software, including operating systems and applications.

- **Deployment Models**: The deployment models are primarily categorized into four models: private cloud, community cloud, public cloud, and hybrid cloud. The private cloud is provisioned for exclusive use by a single organization. Community cloud provides the cloud service for exclusive use by a specific community of consumers from organizations that have shared concerns, while

the public cloud is for the open use by the general public. And the hybrid cloud is the cloud infrastructure consisting of two or more deployment models (private, community, or public).

Table 1.1: Aspects of Cloud Computing

| | |
|---|---|
| Essential Characteristics | On-demand self-service<br>Broad network access<br>Resource pooling<br>Rapid elasticity<br>Measured service |
| Service Models | Software as a Service (SaaS)<br>Platform as a Service (PaaS)<br>Infrastructure as a Service (IaaS) |
| Deployment Models | Private cloud<br>Community cloud<br>Public cloud<br>Hybrid cloud |

## 1.3 Problem Statement

Due to the attractive computing resources in cloud computing, data owners seek to outsource their healthcare data to those cloud facilities that offer query services to other users. However, it inevitably raises new issues about privacy, such as preventing the data from leaking to the cloud server.

The privacy concern in this scenario mainly consists of the following two parts:

- **Privacy of Data owner's Dataset**: The outsourced data are the private property of the data owners and may contain sensitive information, such as healthcare records; A leak of these data may endanger the affected parties, such as patients. Therefore, the outsourced datasets must be protected against unauthorized individuals and entities. However, the cloud server is usually

managed by a third party and cannot be fully trusted. It is often considered to be honest-but-curious, a security assumption that the cloud server may follow our protocols for providing query services to users, but to be curious about the private details of datasets and query requests at the same time. In this instance, outsourcing the similarity range query to the cloud server may result in the disclosure of datasets, evoking privacy concerns from the data owner.

- **Privacy of Query**: The query requests involving query records and even distance thresholds reflect users' behaviors and preferences to some extent, and disclosing such information to unauthorized individuals and organizations poses a threat to users' property and lives. Using a cloud server that can't be trusted to handle similarity queries will make people worry about their privacy of the query result.

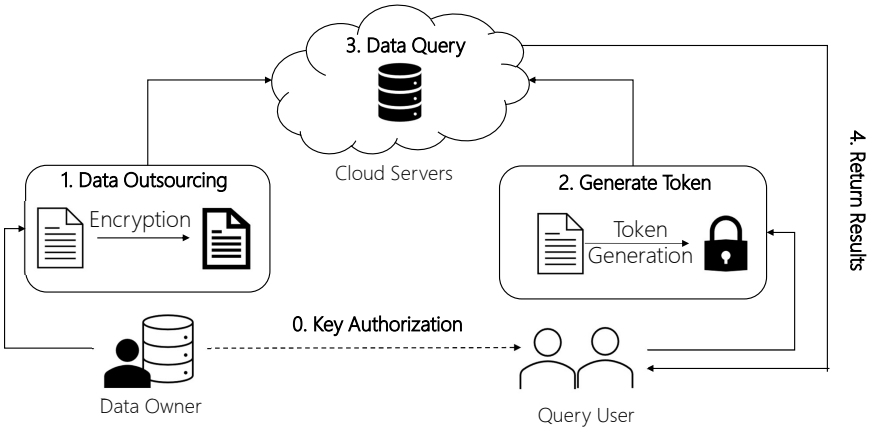


Figure 1.3: Similarity Query in the Cloud Server

To address the aforementioned issue, generally, the data owner will outsource the encrypted healthcare data and design a data query scheme over the encrypted data. Later, users who are authorized by the data owner may launch the data query request by sending the query token to the cloud server in order to retrieve the data from it. The overview is shown in Figure 1.3. In the literature, many privacy-preserving

similarity range schemes have been proposed for various data metrics (data models), such as key-value model [8], spatial data [9], string edit distance [10], time-series model [11] and Euclidean distance [4]. Zheng et al. [4] designed a privacy-preserving healthcare data query scheme based on Euclidean distance. However, this scheme did not account for the preference on certain dimensions. To improve this situation, in our scheme, we consider the similarity range query with the weighted Euclidean distance [12], which enables the query user to specify weighted values for each data dimension, hence making the similarity range query more specific. Additionally, to achieve a more effective and secure query scheme, some schemes leverage two cloud servers model instead of a single cloud server. In the literature, Zheng et al. [11] proposed a similarity range query scheme over the time series based on two semi-trusted cloud servers model. Zhang et al. [13] employed the SHE [14] to implement the privacy-preserving dynamic skyline query scheme for the online medical diagnosis system, which was also based on the two cloud servers model.

In summary, this thesis focuses on the problem of how to perform an efficient and privacy-preserving weighted similarity range query based on weighted Euclidean distance over healthcare data under a two cloud servers model. Specifically, our scheme has the following characteristics:

- **Privacy-Preservation**: The primary requirement of our proposed scheme is to achieve privacy-preservation. The sensitive and private information of both the outsourced data and query requests should be kept secret from the cloud server.

- **Efficiency**: While the cloud server provides a privacy-preserving data query service, it must also be optimized to improve the performance of data queries and reduce their computational cost.

## 1.4 Main Contributions

In this thesis, we propose WeightedSim, an efficient and privacy-preserving weighted similarity range query scheme for healthcare data outsourcing based on Symmetric Homomorphic Encryption (SHE) [14] and R-Tree under a two cloud servers model. Specifically, our main contributions are shown as follows.

- We construct efficient building blocks for the weighted similarity range query using R-Tree to index encrypted healthcare data. The healthcare data in our scheme are processed as multi-dimensional vectors.
- Based on the SHE technique [14], we present a set of privacy-preserving protocols and compare them on SHE ciphertexts. With those protocols, we design an encrypted R-Tree to preserve the privacy of the data and offer an efficient method to apply the weighted similarity range query over the ciphertext.
- Finally, we analyze the security of our scheme and conduct the simulation to evaluate its performance. The result indicates that our scheme is privacy-preserving and achieves selectively secure [15] when the SHE is IND-CPA secure. Also, it shows that our scheme is efficient on multi-dimensional data queries.

## 1.5 Thesis Organization

The remainder of this thesis is organized as follows: A literature review about the cryptography and privacy-preserving similarity query schemes is presented in Chapter 2. In Chapter 3, we propose the system model and an overview of our scheme. And then we describe the necessary preliminaries of our project in Chapter 4. After that, we present the details of our proposed scheme in Chapter 5, followed by the

security analysis and experiments in Chapter 6 and Chapter 7 respectively. Finally, we draw a conclusion about our thesis and discuss the future work in Chapter 8.

# Chapter 2

# Literature Review

## 2.1 Cryptography

Modern cryptography addresses a wide range of problems, including ensuring the security of communication across an insecure medium. During communication, it is inevitable that enemies will attempt to eavesdrop on the communication channel in order to get information. The fundamental purpose of cryptography is to provide such communicative parties with a means to imbue their communications with security properties akin to those provided by the ideal channel [16]. Specifically, cryptography is used to achieve security goals as below [17]:

- **Confidentiality**: Confidentiality is the ultimate target of encryption that confirms that only the cipher-key owner receives the message.
- **Authenticity**: Authentication is the process of establishing a person's identity in order to access a restricted resource using keys.
- **Data Integrity**: Data Integrity is the operation that has access to modify the database of a particular group or individual.
- **Non-Repudiation**: Non-Repudiation guarantees that both the sender and

recipient acknowledge the report's delivery.

- **Access Control**: Access Control verifies that only the authorised group is permitted to access the delivered message.

In order to achieve security goals, cryptography supplies the sender and the receiver with a protocol, which is a collection of programs such as software and algorithms, and contains a cryptographic key to ensure security. Based on the type of cryptographic key, cryptography techniques can be primarily categorized into Symmetric Encryption and Asymmetric Encryption.

### 2.1.1 Symmetric Encryption

Symmetric Encryption is a form of cryptography where the encryption and decryption require the same secret key. The primary components of Symmetric Encryption methods are plaintext, encryption algorithm, ciphertext, and decryption algorithm [18]. An overview of the Symmetric Encryption model is shown in Figure 2.1. Using the encryption algorithm and secret key, users can convert the intelligible message or data (plaintext) into a seemingly random and unintelligible stream. Only an entity that also possesses the same secret key may recover the original data from the ciphertext. However, due to the fact that encryption and decryption utilize the same secret key, a secure channel or mechanism is required to share the same key among communicating entities.

Generally, Symmetric Encryption can be further sorted into two categories: Block Cipher and Stream Cipher.

- **Block Cipher**: Block Cipher is a cryptography scheme which applies a fixed-length block as the input of encryption and decryption procedures. Most block cipher algorithms introduce diffusion and confusion techniques [19] into the

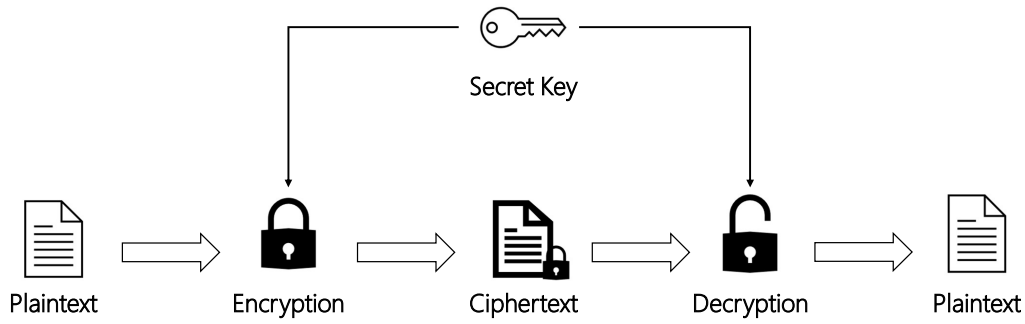


Figure 2.1: Overview of Symmetric Encryption

building blocks. Examples of Block Ciphers include DES [20], Blowfish [21], AES [22], PRESENT [23] and IDEA [24].

- **Stream Cipher**: Stream Cipher is a cryptographic technique that encrypts the data stream one bit or one byte at a time. Examples of Stream Ciphers include ChaCha [25] and RC4 [26].

### 2.1.2 Asymmetric Encryption

Asymmetric Encryption, also known as public-key encryption, is a cryptographic scheme that permits users to encrypt and decrypt data with separate keys. In contrast to Symmetric Encryption, entities may launch communication privately without sharing the secret key in advance. The Asymmetric Encryption is composed of similar components as the Symmetric Encryption except for the encryption key and decryption key. The overview of Asymmetric Encryption is shown in Figure 2.2. During the communication, the receiver first generates the key pair of public key and private key. After that, the receiver publicly announces the public key to the other entities while keeping the private key secret. When someone (a sender) intends to send a message to the receiver, he/she will encrypt the message with the public key and then send it to the receiver. The ciphertext in Asymmetric Encryption can only

be decrypted correctly by the associated private key. The Asymmetric Encryption not only can be appropriate for encrypting the message in communication but also is applicable for the other applications such as digital signature [27]. Examples of popular Asymmetric Encryption algorithms include RSA [28], ElGamal [29].

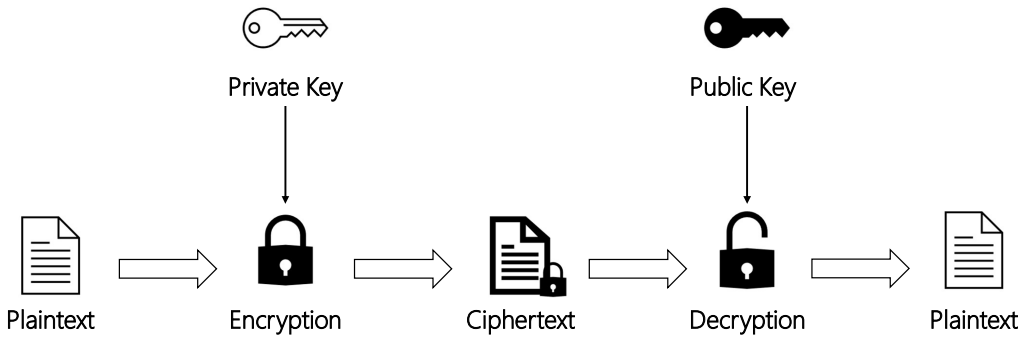


Figure 2.2: Overview of Asymmetric Encryption

### 2.1.3 Comparison between Symmetric Encryption and Asymmetric Encryption

There is no doubt that the most distinct difference between Symmetric Encryption and Asymmetric Encryption is that the encryption and decryption of symmetric encryption require a single secret key, while the latter uses a public key for encryption and a private key for decryption [30]. Therefore, the communication parties of Symmetric Encryption are supposed to share the same secret key via a secure channel, whereas the public key may be sent from one party to the other over a public channel without compromising security. Due to the asymmetric key, the roles of sender and receiver in Asymmetric Encryption are interchangeable. That is, it only allows one-direction communication from the sender to the receiver. However, the single secret key in the Symmetric Encryption allows bidirectional communication between two entities. In addition, the efficiency of Asymmetric Encryption is roughly two

to three orders of magnitude slower than Symmetric Encryption. The comparison table is shown in Table 2.1.

Table 2.1: Comparison Table of Symmetric Encryption and Asymmetric Encryption

| Symmetric Encryption | Asymmetric Encryption |
|---|---|
| The same single secret key for encryption and decryption. | A key pair of public key and private key for encryption and decryption respectively. |
| Require secure channel for key sharing. | Announce the public key on public channel. |
| Bidirectional communication. | One-way communication from sender to receiver. |
| More efficient. | Roughly 2 to 3 orders of magnitude slower than symmetric encryption. |
| Examples:<br>DES, Blowfish, AES, PRESENT, IDEA, ChaCha, RC4 | Examples:<br>RSA, ElGamal |

Based on the aforementioned advantages and disadvantages, in practical communication, we can design communicative protocols combining Asymmetric Encryption with Symmetric Encryption. For instance, the Asymmetric Encryption may create a secure channel for sharing a secret key, which may be employed for Symmetric Encryption and work as the session key for further communication [31].

### 2.1.4 Homomorphic Encryption

Homomorphic Encryption (HE) is an encryption scheme with homomorphic properties which may allow the third party to perform certain functions on ciphertexts while preserving the features of the function and format of ciphertexts [32]. The homomorphic property originates from the idea of group homomorphism (See Definition 2.1.1). Given two messages $m_1$ and $m_2$, and their corresponding ciphertexts $E(m_1)$ and $E(m_2)$, the homomorphic property of homomorphic encryption can be defined as $E(m_1) \odot E(m_2) = E(m_1 \star m_2)$. Due to the homomorphic property, HE

techniques are widely used in privacy-preservation in which we can outsource the encrypted data to a powerful third party (e.g., cloud servers) while providing various computations and query services on the ciphertexts.

**Definition 2.1.1** (Homomorphism [33]). *Suppose $G$ is an Abelian group with operation $\odot$ and $H$ is a group with group operation $\star$. A homomorphism from $(G, \odot)$ to $(H, \star)$ is a mapping $f : G \mapsto H$ that satisfies $f(x_1) \odot f(x_2) = f(x_1 \star x_2), \forall x_1, x_2 \in G$.*

The HE scheme generally comprises four components: $KeyGen$, $Enc$, $Dec$ and $Eval$. $KeyGen$ is the function for generating the key for encryption and decryption. In the asymmetric HE, $KeyGen$ generates the key pair of public key and private key while a single secret key in the symmetric HE. $Enc$ and $Dec$ are the encryption and decryption functions, respectively. $Eval$ is the homomorphic operation in HE. Specifically, without the leakage of plaintexts, $Eval$ applies the computable function over the ciphertexts $\{c_1, c_2\}$ and then outputs a evaluated ciphertext which maintains the corresponding relation of this function over the original plaintext.

Table 2.2: Taxonomy of Homomorphic Encryption

| Category | Homomorphic operation | Number of Homomorphic operations | Related Works |
|---|---|---|---|
| Partially Homomorphic Encryption (PHE) | Addition **OR** multiplication | Arbitrary | [28, 29, 34, 35, 36] |
| Somewhat Homomorphic Encryption (SWHE) | Addition **AND** multiplication | Limited | [37, 14] |
| Fully Homomorphic Encryption (FHE) | Addition **AND** multiplication | Arbitrary | [38, 39, 40, 41] |

Based on supported homomorphic operations, existing HE schemes can be sorted into three categories: Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWHE) and Fully Homomorphic Encryption (FHE). The

taxonomy of HE is shown in Table 2.2. PHE only provides either homomorphic multiplication or homomorphic addition. For example, RSA [28] and ElGamal [29] only provide the homomorphic multiplication while GM [34], Benaloh [35] and Paillier [36] only support the homomorphic addition. On the other hand, SWHE can provide both homomorphic multiplication and homomorphic addition, but the number of homomorphic operations is limited, e.g., BGN [37] can provide an unlimited number of operations in homomorphic addition and one-time multiplication, Symmetric Homomorphic Encryption (SHE) [14] provides an arbitrary number of homomorphic addition but a limited depth of homomorphic multiplication. FHE, which is more flexible and preferred compared with PHE and SWHE, can support an arbitrary number of homomorphic multiplication and addition. In 2009, Gentry [38] proposed an ideal lattice-based FHE scheme which edified other cryptography researchers and led to a breakthrough in this area. Currently, these FHE schemes can be mainly categorized into four families: ideal lattice-based [38], over integers [39], RLWE-based [40] and NTRU-like [41]. However, due to its efficiency, FHE may not be applicable in certain real-world circumstances.

#### 2.1.4.1 SHE Technique

Symmetric Homomorphic Encryption (SHE) can provide efficient homomorphic addition and multiplication for data encryption [14]. The SHE technique primarily consists of three algorithms, including $SHEKeyGen$, $SHEEnc$ and $SHEDec$. The definition of these algorithms is as follows.

- **SHEKeyGen($k_0$,$k_1$,$k_2$):** Given security parameters $k_0$, $k_1$, $k_2$ satisfying $\frac{k_0}{2} > k_2 \gg k_1$, the key generation algorithm sets up a number $N = pq$ with two large prime numbers $p$,$q$ where $|p| = |q| = k_0$ and a random number $L$ where $|L| = k_2$. And then the algorithm outputs the public parameter $pb \leftarrow (k_0, k_1, k_2, N)$, the secret key $sk \leftarrow (p, L)$ and the message space $M \leftarrow \{m|m \in [-2^{k_1-1}, 2^{k_1-1})\}$.

- **SHEEnc($sk$, $m$):** Given the secret key $sk$ and a message $m \in M$, the encryption algorithm outputs the ciphertext of $m$ as $E(m) = (rL+m)(1+r'p)\ mod\ N$, where $r$ and $r'$ are two random numbers satisfying $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$.

- **SHEDec($sk$, $E(m)$):** Given the secret key $sk$ and the ciphertext $E(m)$, the decryption algorithm recovers the message $m' = (E(m)\ mod\ p)\ mod\ L$. If $m' < \frac{L}{2}$, it indicates $m \geq 0$ and $m = m'$. Otherwise, $m < 0$ and $m = m' - L$.

The SHE technique has the homomorphic properties as follows and the proof of correctness is in Appendix A.

- **Homomorphic addition-I:** Two ciphertexts $E(m_1)$ and $E(m_2)$ satisfy $E(m_1)+E(m_2)\ mod\ N \rightarrow E(m_1+m_2)$.

- **Homomorphic multiplication-I:** Two ciphertexts $E(m_1)$ and $E(m_2)$ satisfy $E(m_1) * E(m_2)\ mod\ N \rightarrow E(m_1 * m_2)$.

- **Homomorphic addition-II:** A ciphertext $E(m_1)$ and a plaintext message $m_2$ satisfy $E(m_1) + m_2\ mod\ N \rightarrow E(m_1+m_2)$.

- **Homomorphic multiplication-II:** A ciphertext $E(m_1)$ and a plaintext message $m_2 > 0$ satisfy $E(m_1) * m_2\ mod\ N \rightarrow E(m_1 * m_2)$.

The SHE technique is homomorphic encryption that supports both homomorphic addition and homomorphic multiplication, but it can only provide a limited round of calculation in homomorphic multiplication-I [11]. The detailed proof of the decryption and the maximum multiplicative depth of this homomorphic property is shown in Appendix A.2.2. To avoid incorrect decryption results, we make sure the depth of homomorphic multiplication-I should be within $\theta = \lfloor \frac{k_0}{2k_2} - 1 \rfloor$.
According to the homomorphic properties, we can encrypt the message with the public key $pk \leftarrow \{E(0)_1, E(0)_2\}$, where $E(0)_1$ and $E(0)_2$ are two ciphertexts of 0

with different random numbers. When we apply the public key encryption on the message, we have $E(m) = m + r_1E(0)_1 + r_2E(0)_2 \ mod \ N$, where $r_1$ and $r_2$ are two random numbers satisfying $r_1, r_2 \in \{0,1\}^{k_2}$. In later sections, we denote the public key encryption as $SHEEnc(pk, m)$.

### 2.1.5 Security Models in Cryptography

#### 2.1.5.1 Security Models on Encryption Schemes

The security model of symmetric and asymmetric encryption usually refers to which level of attack the encryption algorithm can resist. These attacks, which mainly focus on systematically recovering the plaintext from ciphertext or deducing the decryption key, can be divided into four categories: Cipher-Only Attack (COA), Known-Plaintext Attack (KPA), Chosen-Plaintext Attack (CPA), and Chosen-Ciphertext Attack (CCA) [42]. The summary table is shown in Table 2.3. COA is a security attack where the adversary can only deduce the decryption key or plaintext by observing the ciphertext. Because COA is the weakest attack, we consider an encryption algorithm completely insecure if it can not resist against this attack. In KPA, the adversary may be allowed to hold a quantity of plaintexts and the corresponding ciphertexts for analysis. However, these plaintexts can not be chosen by the adversary. In CPA, the adversary can select the plaintext and then be given the corresponding ciphertext. Similarly, CCA is a security attack where the adversary may select the ciphertext and then be given the corresponding plaintext. In addition, there are special categories of security attacks, including Adaptive Chosen Plaintext Attack and Adaptive Chosen Ciphertext Attack. Compared with the original CPA or CCA, the adaptive attack allows the adversary to select the plaintext or ciphertext according to the results received from previous requests.

Table 2.3: Security Attacks of Encryption Schemes

| Security Attack | Attack Capability |
|---|---|
| Cipher-Only Attack (COA) | The adversary deduce the decryption key or plaintext only by observing the ciphertext. |
| Known-Plaintext Attack (KPA) | The adversary may be allowed to hold a quantity of plaintexts and corresponding ciphertexts but can not select plaintexts. |
| Chosen-Plaintext Attack (CPA) | The adversary can choose the plaintext and then is given the corresponding ciphertext. |
| Chosen-Ciphertext Attack (CCA) | The adversary can select the ciphertext and then is given the corresponding plaintext. |

#### 2.1.5.2 Indistinguishability Under Chosen-Plaintext Attack and Semantic Security

The basic idea of indistinguishably under chosen-plaintext attack (IND-CPA) is to consider multiple rounds of security experiments with an adversary and an oracle. For each round, the adversary chooses a pair of messages and submits it to the oracle, which will later encrypt one of the messages from the pair and return the ciphertext to the adversary. If the adversary has difficulty distinguishing which of the two messages is the encrypted one, we may consider the encryption scheme is secure against chosen-plaintext attack [16].

Generally, IND-CPA is regarded as equivalent to be semantic security [34], an alternative notation of encryption schemes security. The SHE technique, the primary cryptography technique used in our scheme, is proved to be semantically secure against CPA [11].

## 2.2 Review of Privacy-Preserving Similarity Range Query

Similarity range query, one of the most fundamental data query services, is widely used in numerous applications, including the healthcare data outsourcing scenario. As individuals become more conscious of privacy issues regarding their data, there are a number of efficient and privacy-preserving similarity range query schemes proposed [13, 4, 12]. In the following sections, we will conduct a literature review of the privacy-preserving similarity range query in the healthcare application, followed by a survey of the other applications, such as IoT and the location-based service.

### 2.2.1 Similarity Query over Healthcare Data

Based on SHE technique building blocks, Zhang et al. [13] proposed a privacy-preserving dynamic skyline query scheme for the secure online medical diagnosis system. In their scheme, they designed a series of SHE privacy-preserving protocols for two cloud servers model as secure primitives. They implemented the secure dynamic dominance (SSD) protocol and the secure minimum of n data (SMINn) protocol for the secure dynamic skyline query. Similarly, in [11], the authors presented a privacy-preserving similarity range query scheme over the encrypted time series data with the k-d tree based on the SHE technique and two cloud servers model. To avoid exceeding the multiplicative depth of homomorphic multiplication in SHE, they designed a bootstrap protocol to refresh the ciphertexts. However, the bootstrap protocol will cause additional computation and reduce the efficiency of the data query.

In [4], Zheng et al. designed a Euclidean distance-based privacy-preserving similarity range query for healthcare data. In their scheme, these healthcare data were encrypted with a predicate encryption named modified asymmetric-scalar-product

encryption (MASPE) and a quadsector tree was created as the index. They applied the filtration strategy to prune the unnecessary tree searching paths and the refinement strategy to validate whether the data satisfied the similarity range query. However, this scheme did not consider the preference of some specific data dimensions when querying. Later, Zheng et al. [12] first introduced the weighted similarity range query on the healthcare data. In their scheme, they defined the negative infinity norm (NIND) distance as the lower bound of the weighted Euclidean distance. When building an index, they selected reference points from the data and applied the triangle inequality of NIND to build up a sorted list as the index, in which MASPE also protected these data. However, NIND doesn't satisfy the triangle inequality and the query results are incorrect.

Wang et al. [43], adopting a predicate encryption scheme as the main cryptographic primitive and processing the genetic sequence with a suffix tree, proposed a privacy-preserving genomic testing scheme based on the cloud server model. As the authors analyzed, this scheme had $O(n)$, $O(m)$ and $O(mn)$ computation complexity on the sequence encryption, token generation, and data search, respectively, where $n$ is the length of the DNA sequence and $m$ is the length of the query sequence. Also, according to the authors' analysis, it was provably secure to protect the privacy of the data, search pattern, and testing results. Also focusing on the privacy issue of genomic data, Zhu et al. [44] designed a privacy-preserving similar patients query (SPQ) scheme which aimed to find the genomic data similar to a patient and assist the physician in making a precise and optimal therapy for the patient. In their scheme, the authors introduced the BK-tree data structure, which enhanced the performance of the similarity query on edit distance. By integrating the data structure with their privacy-preserving data query scheme, the physicians could request the genomic data of their patients without disclosing confidential information to the cloud server. The experiment demonstrated that this scheme may provide an effective and

highly accurate query service while guaranteeing privacy-preservation.

Wang et al. [45], in order to address the privacy issue of the patients' personal data in the Mobile Healthcare Monitoring Network, developed an Efficient Privacy-preserving Sensor data Monitoring and online Diagnosis (EPSMD) system for the healthcare data outsourcing application. They first proposed an improved Multidimensional Range Query Technique (MRQT) for the range query of multi-dimensional healthcare data. To maximize the potential of MRQT, based on this technique, the authors designed a privacy-preserving Bayesian classifier for the data mining and the online diagnosis as well as a real-time healthcare monitoring scheme.

Similarly, Zheng et al. [46] proposed a privacy-preserving multi-dimensional range query (PMRQ) scheme that launched a range query over multi-dimensional healthcare data in a single cloud server setting. In their scheme, the authors first introduced a homomorphic encoding technique and matrix encryption to protect the data. To improve the efficiency of range queries, they built an R-tree to index the encrypted database. Based on the homomorphic encoding and the matrix encryption technique, an encoding-based multi-dimensional range intersection algorithm and a multi-dimensional intersection predicate encryption (MRIPE) were introduced into the proposed scheme so that we could employ the range query over the encrypted multi-dimensional healthcare data while preserving privacy. This scheme was shown to be efficient in their experiments and was proved to be selectively secure under the given leakage.

Xu et al. [47] designed two privacy-preserving e-healthcare schemes where users could employ a top-k disease matching over the patients' diagnosis files without disclosing the private information. In the first scheme, the authors implemented the disease matching system by relying on a weighted-Euclidean-distance-based secure kNN query, while the second scheme applied the top-k disease matching by introducing the Euclidean distance under the modified Paillier homomorphic encryption.

### 2.2.2 Privacy-Preserving Data Query in Other Applications

Advanced communication technologies such as 5G and 6G boost the development of IoT and lead to an exponential expansion of the data from IoT systems. Therefore, outsourcing the data to a cloud platform is a feasible solution that optimizes the data management and enables the data to be further analyzed. However, similar to healthcare data, the generated data from the IoT system is also sensitive and inevitable to meet the privacy concerns of data outsourcing. To address this issue, integrating the blockchain technique and cloud platform, Rahman [48] designed a privacy-preserving verifiable data query framework for the Industrial Internet-of-Things (IIoT). The IIoT data, encrypted by symmetric encryption, was stored on blockchain and cloud servers. The scheme constructed multi-keyword range search indexes based on B+-Tree [49] for blockchain and cloud servers respectively and encrypted them with order-preserving encryption (OPE) [50]. However, this scheme relied on the OPE, which leaked the order relation of the index and weakened the security. To decrypt query results, the user should also keep the secret key, necessitating a greater security level for the management of users to prevent the secret from leakage. Guan et al. [51] proposed a scheme to solve the privacy concern in cybertwin [52], an innovative technique to create digital representations for physical objects to implement various functionalities in the 6G era. Similar to [11, 13], the authors designed a privacy-preserving data query scheme for both spatial, temporal, and keyword query criteria that is encrypted based on the SHE technique under the two cloud servers model.

The smart grid is the next generation of power grid that enables the service provider to construct an automated and distributed advanced energy delivery network by collecting and analyzing electricity usage data from users. As a viable solution, the service provider relies on cloud server technology to fulfil these duties, which may also cause privacy concerns about users' data. Jiang et al. [5] proposed a Locality

Sensitive Hashing based similarity query scheme for the multidimensional metering data in the smart grid. In this scheme, the users may encrypt and outsource their metering data to the semi-trusted cloud server and then authorize the electricity service provider to obtain their metering data. Based-on the fog computing and a double trapdoor decryption cryptosystem [53], Liu et al. [54] designed a privacy-preserving scheme for the smart grid system. Fog nodes in the system aggregated the electricity usage data and outsourced the encrypted data to the cloud server platform. After that, the user and service provider launched a privacy-preserving function query, a data query protocol that allows the query requester to obtain the data by the results of an arithmetic circuit without leaking the private information.

# Chapter 3

# Models and Design Goals

In this chapter, we formalize our system model, security model, and identify our design goals.

## 3.1 System Model

In the system model, we propose a weighted similarity range query among the healthcare center (HC), a cloud with two servers $\{S_1, S_2\}$, and multiple query doctors (query users) $U = \{U_1, U_2, \dots\}$. The overview of our scheme is shown in Figure 3.1.

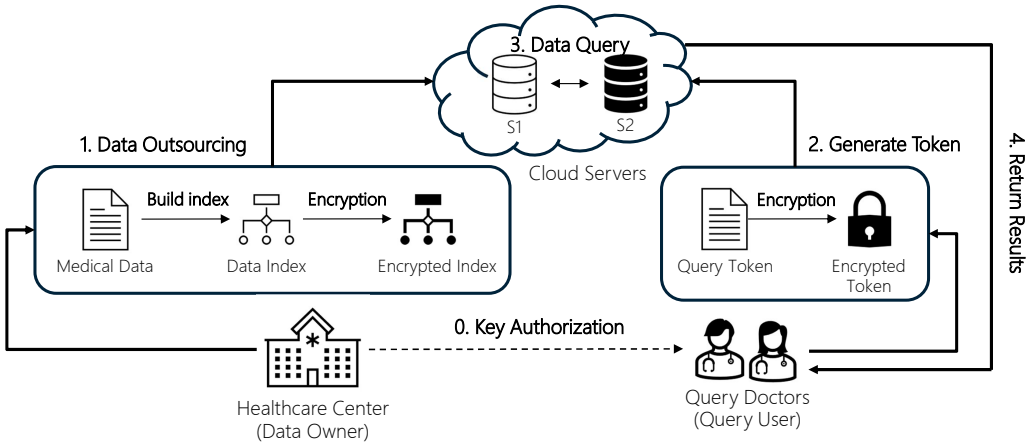


Figure 3.1: Overview of WeightedSim

- **Healthcare Center(HC):** The HC is treated as the data owner of a large number of healthcare data. HC will offer a weighted similarity range query of database $X$ to the query user. Due to the constrained computing and storage resources, HC will outsource the database to the powerful cloud server. For the privacy concern, the outsourced data should be encrypted, denoted by $E(X)$, in the cloud server.

- **Cloud Server:** The cloud server model consists of two cloud servers $\{S_1, S_2\}$. Both of them can provide powerful computing power and abundant storage. $S_1$ is responsible for the storage of the encrypted database $E(X)$. $S_1$ and $S_2$ will cooperate to offer the weighted similarity range query service to query users. When query users obtain the data from the cloud server, they will construct a query request $(q, w, \tau)$ and encrypt it into the corresponding query token. Once the cloud server receives the query token from the query user, it will search the encrypted database $E(X)$ for the data satisfying $d_w(x_i, q) = \sqrt{\sum_{j=1}^{l} w_j(x_{i,j} - q_j)^2} \leq \tau$ and returns the ciphertext of those data to the query user.

- **Query User:** The query users $U = \{U_1, U_2, \dots\}$, once authorized by the healthcare center, will obtain the authorized key from it. When the query user applies the weighted similarity range query on the encrypted data, it will first generate a query token with the authorized key and send the query token to the cloud server. After receiving the encrypted data from the cloud server, the query user may recover the original healthcare data.

## 3.2 Security Model

In our security model, the healthcare center works as the data owner of healthcare data and is considered as fully trusted. Query users, once authorized by the health-

care center, are also regarded as honest entities who faithfully follow our proposed design to generate query tokens and launch weighted similarity range query requests to the cloud servers. However, the cloud servers $\{S_1, S_2\}$ are considered to be honest-but-curious. They will follow the proposed design and provide a weighted similarity range query on the encrypted database $E(X)$ honestly, but at the same time, cloud servers are curious about the ciphertext and try to obtain the private information of the data. In addition, we assume that there is no collusion between $S_1$ and $S_2$. Since our scheme focuses on privacy preservation, we will not discuss attacks out of this scope, e.g., Denial of Service (DoS) attacks.

## 3.3 Design Goals

In this work, we aim to design an efficient and privacy-preserving weighted similarity range query scheme and have the following two objectives.

- **Privacy Preservation:** Plaintexts of the healthcare data and query requests should be secret to the cloud server under the honest-but-curious model.
- **Efficiency:** We also intend to cut down the computational cost of weighted similarity range query among the query user and cloud server and should improve the query efficiency as much as possible.

# Chapter 4

# Preliminaries

In this section, we will introduce the basic definition of weighted similarity range query, R-Tree and the SHE-based privacy-preserving protocols.

## 4.1 Weighted Similarity Range Query

Let $X = \{x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,k}) | i = 1, 2, \ldots, n\}$ be the healthcare database with which there are $n$ data records totally in $X$ and $k$ dimensions in each data record. Let $(q, w, \tau)$ be the query request, where $q = (q_1, q_2, \ldots, q_k)$ is a k-dimensional query vector, $w = (w_1, w_2, \ldots, w_k)$ is a k-dimensional weighted vector satisfying $\sum_{j=1}^{k} w_j = 1 \wedge w_j > 0$, and $\tau$ is the distance threshold. With a given query request $(q, w, \tau)$ and data record $x_i$, we define the weighted Euclidean distance as $d_w(x_i, q) = \sqrt{\sum_{j=1}^{k} w_j (x_{i,j} - q_j)^2}$ and the weighted similarity range query is to figure out all the data records in $X$ which satisfy $d_w(x_i, q) \leq \tau$. Because it is hard to implement the root squared computation in homomorphic encryption and $d_w(x_i, q)$ and $\tau$ are non-negative values, we will only consider $d_w(x_i, q)^2 \leq \tau^2$ in later sections. In addition, homomorphic properties are only applicable to integers, thus we must transform real numbers to integers by multiplying them by the number of multiples of ten.

## 4.2 R-Tree

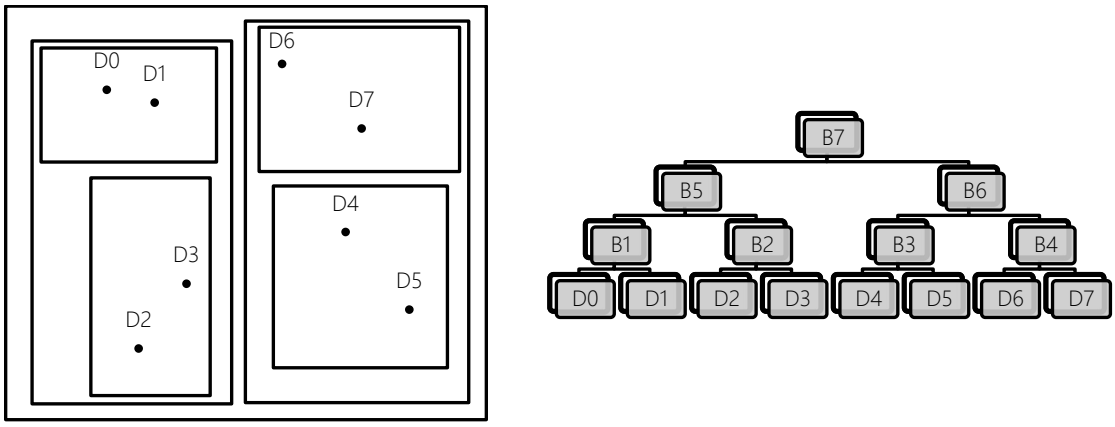


Figure 4.1: Construction of R-Tree

R-Tree [55] is a spatial data structure that is friendly to multi-dimensional data and can improve the efficiency of the range query. The main idea behind the R-Tree's construction is to group nearby objects at the same level space with a minimum bounding rectangle in the tree's upper level. An example is presented in Figure 4.1.

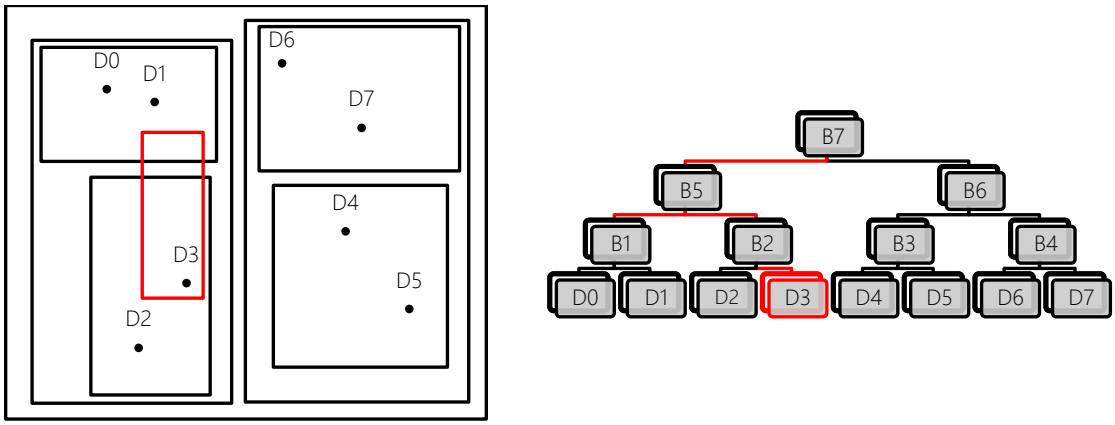


Figure 4.2: Range Query on R-Tree

The R-Tree is defined as $T \leftarrow \{B_1, B_2, \ldots, B_m, D_1, D_2, \ldots, D_n\}$. $D = \{D_1, \ldots, D_n\}$ is the data record and stored in the leaf-node of R-Tree, where $n$ is the number of data and $m$ is the number of internal nodes. $D_i = (d_{i,1}, d_{i,2}, \ldots, d_{i,k})$ is a k-dimensional

vector. $B = \{B_1, B_2, \ldots, B_m\}$ is the set of the internal node in R-Tree with $k$ elements, where $B_j = \{(l_{j,1}, u_{j,1}), (l_{j,2}, u_{j,2}), \ldots, (l_{j,k}, u_{j,k})\}$ is the k-dimensional rectangle with which $(l_{j,\alpha}, u_{j,\alpha})$ is the lower bound and upper bound of the rectangle in the $\alpha^{th}$ dimension. $B_j$ is the parent or ancestor node of $D_j$ as long as $l_{j,\alpha} \leq d_{i,\alpha} \leq u_{j,\alpha}$ for $1 \leq \alpha \leq k$. In other words, a rectangle and a data point satisfy $l_{0,k} > d_{0,k} || u_{0,k} < d_{0,k}$ indicating that this data point is not involved within this rectangle. The R-Tree range query algorithms are shown at Algorithm 1, 2 and 3.

The range query algorithm is shown in Figure 4.2. When the user applies a query on R-Tree, they will first generate a query rectangle to show the range of the query [56]. And then it will check whether the query rectangle overlaps with the object in the R-Tree. The query result will return all the data involved within this query rectangle.

**Algorithm 1** rangeQuery($B_0$, $T$)

**Input:** Query rectangle $B_0$, R-Tree node $T$
**Output:** Candidate set $C$

```
for each t ∈ T.Children do
    if t.isLeadNode() == false then
        if intersect(B_0, t) then
            rangeQuery(B_0, t)
        end if
    else
        if inside(B_0, t) then
            C ← R ∪ {t}
        end if
    end if
end for
```

## 4.3 SHE-based Privacy-Preserving Protocols

Based on the SHE-based privacy-preserving protocol in [13], this section will introduce some atomic privacy-preserving protocols.

**SLESSE:** Cloud server $S_1$, $S_2$ work together to calculate the result of whether

**Algorithm 2** inside($B_0$, $D_0$)

**Input:** Rectangle $B_0$, data point $D_0$
**Output:** Return true if $D_0 \in B_0$; Otherwise return false.

```
for i ← 1 to k do
    if (l_{0,k} > d_{0,k} || u_{0,k} < d_{0,k}) then
        return false
    end if
end for
return true
```

**Algorithm 3** intersect($B_0$, $B_1$)

**Input:** Rectangle $B_0$, $B_1$
**Output:** Return true if $B_0$ intersects with $B_1$; Otherwise return false.

```
for i ← 1 to k do
    if (l_{0,k} > u_{1,k} || u_{0,k} < l_{1,k}) then
        return false
    end if
end for
return true
```

$m_1 \leq m_2$ with $E(m_1)$ and $E(m_2)$ while the original information of $m_1$ and $m_2$ will be in secret. In this protocol, $S_1$ holds the public parameter $pb$ and ciphertexts $\{E(m_1), E(m_2), E(-1)\}$, where $E(-1)$ is utilized to assist in the calculation of the protocol, while $S_2$ holds the public parameter $pb$ and the secret key $sk$. In the later section, we denote it as $SLESSE(m_1, m_2)$. The detail of this protocol is as follows.

- **Step 1:** $S_1$ calculates the ciphertext $E(x') = E(m_1 - m_2) = E(m_1) + E(-1) * E(m_2)$ and, according to properties of homomorphic addition-II and homomorphic multiplication-II,then figures out $E(x) = E(s * r_1 * x' + s * r_2) = s * r_1 * E(x') - s * r_2$ where $r_1, r_2 \in \{0,1\}^{k_1}$ are two random numbers satisfying $r_1 > r_2 > 0$ and $s \in \{-1, 1\}$ is a random number generated by coin flipping by $S_1$. $S_1$ sends the $E(x)$ to $S_2$.

- **Step 2:** After receiving the $E(x)$, $S_2$ utilizes the secret key $sk$ to recover $x$. $S_2$ returns a encrypted value $E(b')$ to $S_1$. If $x \leq 0$, $E(b') = E(1)$ otherwise $E(b') = E(0)$.

- **Step 3:** $S_1$ receives $E(b')$ from $S_2$ and outputs the result $E(b)$. If $s = 1$, $E(b) = E(b')$. If $s = -1$ and $E(b) = E(1) + E(s) * E(b') = E(1 - b')$. $E(b)$ outputs $E(b) = E(1)$ if $m_1 \leq m_2$, otherwise $E(b) = E(0)$.

$$E(b) = \begin{cases} E(b') & s = 1 \\ E(1) + E(s) * E(b') = E(1 - b') & s = -1 \end{cases}$$

- **Correctness:** The protocol is correct iff when $m_1 \leq m_2$ it outputs $E(b) = E(1)$ otherwise $E(b) = E(0)$. When $s = 1$, if $m_1 \leq m_2$, because of $r_1 > r_2 > 0$ and $m_1 - m_2 \leq 0$, we may figure out that $x = s * r_1 * (m_1 - m_2) - s * r_2 \leq 0$ and the $S_2$ will return $E(b') = 1$ to $S_1$. Due to $s = 1$, the protocol will return $E(b) = E(b') = E(1)$. When $s = -1$, If $m_1 \leq m_2$, we have $x \geq 0$ and the $S_2$ will return $E(b') = 0$ to $S_1$. And the $S_1$ will compute the result $E(b) = E(1 - b') = E(1)$. We can use the similar method to prove that the protocol will return $E(b) = E(0)$ if $m_1 > m_2$. Therefore, the Secure Less Equal Than protocol is correct.

**DLESS:** Cloud server $S_1$, $S_2$ work together to calculate the result of whether $m_1 < m_2$ with $E(m_1)$ and $E(m_2)$. However, $S_1$ gets the result directly while it still keeps secret to $S_2$. The other private information such as plaintexts should still be in secret. In this protocol, $S_1$ holds the public parameter $pb$ and ciphertexts $\{E(m_1), E(m_2), E(-1)\}$, where $E(-1)$ is utilized to assist in the calculation of the protocol, while $S_2$ holds the public parameter $pb$ and the secret key $sk$. In the later section, we denote it as $DLESS(m_1, m_2)$. The detail of this protocol is as follows.

- **Step 1:** $S_1$ calculates the ciphertext $E(x') = E(m_1 - m_2) = E(m_1) + E(-1) * E(m_2)$ and, according to properties of homomorphic addition-II and homomorphic multiplication-II, then figures out $E(x) = E(s * r_1 * x' + s * r_2) = s * r_1 * E(x') + s * r_2$ where $r_1, r_2 \in \{0, 1\}^{k_1}$ are two random numbers satisfying

$r_1 > r_2 > 0$ and $s \in \{-1, 1\}$ is a random number generated by coin flipping by $S_1$. $S_1$ sends $E(x)$ to $S_2$.

- **Step 2:** After receiving the $E(x)$, $S_2$ utilizes the secret key $sk$ to recover $x$. $S_2$ returns the result $b'$ directly to $S_1$. If $x < 0$, $b' = true$ otherwise $b' = false$.

- **Step 3:** $S_1$ receives $b'$ from $S_2$ and outputs the result $b$. If $s = 1$, $b = b'$. If $s = -1$ and $b = \neg b'$. The protocol outputs $b = true$ if $m_1 < m_2$, otherwise $b = false$.

$$E(b) = \begin{cases} b' & s = 1 \\ \neg b' & s = -1 \end{cases}$$

- **Correctness:** We can prove the correctness of this protocol in a similar method with SLESSE.

**DWITHIN:** Given three ciphertexts $E(m_1)$,$E(m_2)$,$E(m_3)$ and assumed that $m_1 < m_3$, this protocol is to check whether $m_1 \leq m_2 \leq m_3$. In this protocol, $S_1$ holds the public parameter $pb$ and ciphertexts $\{E(m_1), E(m_2), E(m_3), E(-1)\}$, where $E(-1)$ is utilized to assist in the calculation of the protocol, while $S_2$ holds the public parameter $pb$ and the secret key $sk$. Similar to the DLESS, $S_1$ will figure out the result directly while the result is still in secret in $S_2$. In the later section, we denote it as $DWITHIN(m_1, m_2, m_3)$. The detail instruction is as follow.

- **Step 1:** $S_1$ calculates the ciphertext $E(x') = E((m_1 - m_2) * (m_3 - m_2)) = (E(m_1) + E(-1) * E(m_2)) * (E(m_3) + E(-1) * E(m_2))$. Similar to $SLESSE$, $S_1$ flips a coin $s \in \{-1, 1\}$ and chooses two random numbers $r_1, r_2 \in \{0, 1\}^{k_1}$ which satisfy $r_1 > r_2 > 0$ to compute $E(x) = E(s * r_1 * x' + s * r_2) = s * r_1 * E(x') - s * r_2$ and then sends $E(x)$ to $S_2$.

- **Step 2:** When $S_2$ receives the $E(x)$, it recovers $x$ with the secret key $sk$. If $x \leq 0$, $S_2$ returns a plaintext flag $b' = true$ to $S_1$ otherwise returns $b' = false$.

- **Step 3:** If $s = 1$, $S_1$ computes the result bit b as $b = b'$. If $s = -1$, $S_1$ returns the flipped flag $b = \neg b'$ as result. If $m_1$,$m_2$ and $m_3$ satisfy $m_1 \leq m_2 \leq m_3$, the protocol output $b = true$, otherwise $b = false$.

$$b = \begin{cases} b' & s = 1 \\ \neg b' & s = -1 \end{cases}$$

- **Correctness:** If $m_1 \leq m_2 \leq m_3$ and $m_1 < m_3$, we may deduce the condition to $(m_1 - m_2) * (m_3 - m_2) <= 0$. And then we can use the similar method to prove the DWITHIN.

# Chapter 5

# Proposed Scheme

## 5.1 Weighted Similarity Range Query over Plaintexts

We first build up the R-Tree index over the database. To reduce the calculation of the weighted Euclidean distance, we apply the range query on R-Tree to filter the data points which are impossible to include within the weighted Euclidean distance with the query point. Given a query request $(q, w, \tau)$, for a data point $x_i$, if it exists a dimension $j$ that satisfies $\sqrt{w_j(x_{i,j} - q_j)^2} > \tau$, we can consider this data point is impossible to be involved within the weighted similarity range query and preclude it from the result set. Therefore, we set up a query rectangle $B_0 = \{(q_j - \frac{\tau}{\sqrt{w_j}}, q_j + \frac{\tau}{\sqrt{w_j}})|j = 1, ..., k\}$. Because homomorphic properties are only applicable to integers, we must multiply real numbers in the data and the query rectangle by the number of multiples of ten to convert them to integers. Once we apply this query rectangle to the R-Tree, we can acquire those data points which may probably satisfy the weighted similarity range query and add them to a candidate set. Finally, we check whether the weighted euclidean distance between each candidate set element and the query point is less than or equal to the $\tau$.

## 5.2 Description of Our Scheme

This section will introduce the detailed description of our privacy-preserving weighted similarity range query on healthcare data, including the system initialization, healthcare data outsourcing and privacy-preserving weighted similarity range query.

### 5.2.1 System Initialization

The system initialization starts at the healthcare center. The HC first selects the proper security parameters $k_0, k_1, k_2$ before invoking the $SHEKeyGen(k_0, k_1, k_2)$ to generate the public parameter and secret key $\{pb, sk\}$ for the encryption. After that, HC generates the public key $pk \leftarrow \{E(0)_1, E(0)_2\}$ at the Section 2.1.4.1. Because the query users may have a lower level of security, we delegate the public key to them instead of the secret key. HC delivers $pb$ to $S_1$ and $sk$ to $S_2$. And then HC generates a ciphertext $E(-1)$ leveraged for the filtration stage and refinement stage in the weighted range query and sends the ciphertexts to $S_1$. HC delegates the query users $U = \{U_1, U_2, \dots\}$ in the system and sends $pk$ to them. When the new query user registers on the system and is authorized by HC, it will also receive the keys from HC.

### 5.2.2 Healthcare Data Outsourcing

Given the healthcare database $X = \{x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,k}) | i = 1, 2, \dots, n\}$ , the healthcare center will construct the encrypted R-Tree over $X$ and then outsource it to $S_1$. The detail of data outsourcing is as follows.

**Step 1:** HC builds up a k-dimensional R-Tree $T \leftarrow \{B_1, B_2, \dots, B_m, D_1, D_2, \dots, D_n\}$ on the dataset $X$. As stated in Section 4.1, if the numbers in the database are not integers, they must first be preprocessed and converted into integers.

**Step 2:** HC encrypts the the tree into $E(T) \leftarrow \{E(B_1), \dots, E(B_m), E(D_1), \dots,$

$E(D_n)\}$, where $B_j = \{(E(l_{j,1}), E(u_{j,1})) \ldots, (E(l_{j,k}), E(u_{j,k}))\}$ and $E(D_i) = \{E(x_i)\}$ with $SHEEnc(sk, m)$. After that, the healthcare center outsources $E(T)$ on $S_1$.

### 5.2.3 Weighted Similarity Range Query Over Ciphertexts

After receiving the encrypted tree $E(T)$, $S_1$ will cooperate with $S_2$ to offer the weighted similarity range query for query users. The query user may launch the similarity query by submitting the query request $(q, w, \tau)$. The overview of the range query is shown in Figure 5.1.

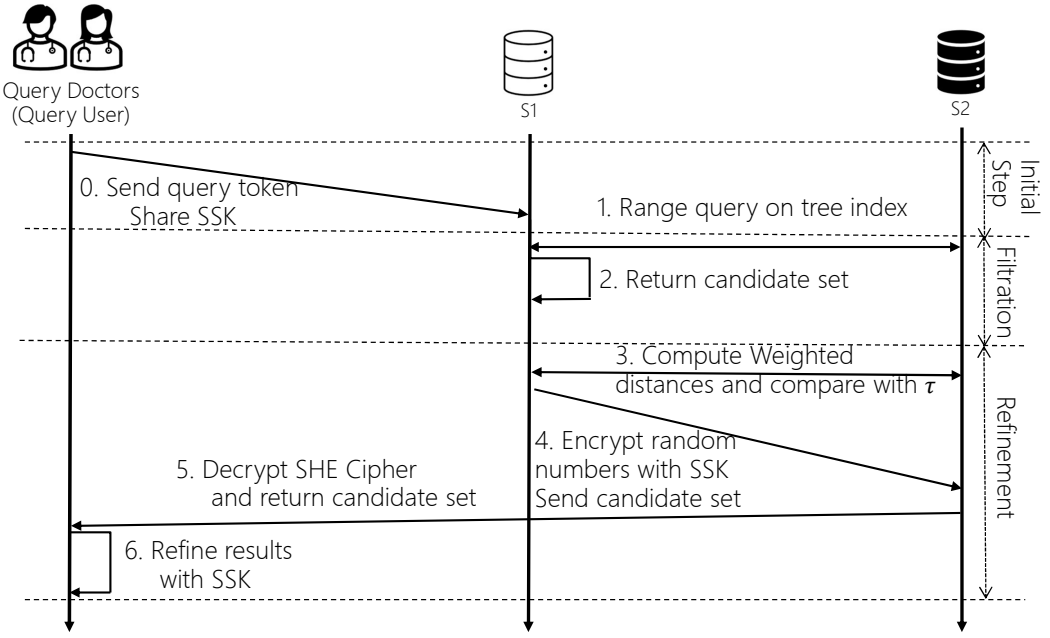


Figure 5.1: Overview of Range Query in WeightedSim

**Initial Step:** The query user constructs a query request $(q,\ w,\ \tau)$ and generates the refinement token $(E(q), E(w), E(\tau))$ with $SHEEnc(pk, m)$. And then the query user constructs the query rectangle $B_0 = \{(q_j - \frac{\tau}{\sqrt{w_j}}, q_j + \frac{\tau}{\sqrt{w_j}}) | j = 1, ..., k\}$ as the instructions in Section 5.1 and encrypted it into the filtration token. As stated in Section 4.1, we should convert the real numbers in the query rectangle into integers before generating the filtration token. After that, the query user chooses a session key $ssk$ which is used for recovering the query result and sends the session key $ssk$

and query token $\{E(q), E(w), E(\tau), E(B_0)\}$ to $S_1$.

**Filtration Step:** To launch the R-Tree range query on the encrypted data, we design the privacy-preserving R-Tree-based range query based on SHE privacy-preserving protocols. We first design the top level function as Algorithm 4 based on Algorithm 1. To check whether the encrypted data point is inside the encrypted rectangle, because the statement $l_{0,k} > d_{0,k} || u_{0,k} < d_{0,k}$ is equivalent to $\neg(l_{0,k} \leq d_{0,k} \leq u_{0,k})$, we may design the $secureInside(E(B_0),\ E(D_0))$ as Algorithm 5 based on Algorithm 2 by implementing the conditional statement with DWITHIN protocol in Section 4.3. Correspondingly, we can implement $secureIntersect(E(B_1), E(B_0))$ as Algorithm 6 based on Algorithm 3 to check whether two encrypted rectangles intersect with DLESS. Eventually, The algorithm outputs a candidate set $C = \{E(x_i) | x_i$ may probably satisfy $d_w(x_i, q) \leq \tau\}$.

**Algorithm 4** secureRangeQuery($E(B_0)$, $E(T)$)

**Input:** Encrypted query rectangle $E(B_0)$, Encrypted R-Tree node $E(T)$
**Output:** Candidate set $C$

```
for each E(t) ∈ T.Children do
    if E(t).isLeadNode() == false then
        if secureIntersect(E(B_0), E(t)) then
            secureRangeQuery(E(B_0), E(t))
        end if
    else
        if secureInside(E(B_0), E(t)) then
            E(x_i) ← E(t).getData()
            R ← R ∪ {E(x_i)}
        end if
    end if
end for
```

**Refinement Step:** For $x_i \in C$, $S_1$ computes the weighted Euclidean distance between $x_i$ and $q$ according to the SHE homomorphic properties. Because it is difficult to apply square root in the homomorphic encryption, $S_1$ calculates $E(d_w(x_i, q)^2) = E(\sum_{j=1}^{l} w_j(x_{i,j} - q_j)^2)$ in the practical. After that, $S_1$ and $S_2$ cooperate to launch the SLESSE in Section 4.3 to compare $E(d_w(x_i, q)^2)$ and $E(\tau^2)$. If $d_w(x_i, q)^2 \leq \tau^2$,

**Algorithm 5** secureInside($E(B_0)$, $E(D_0)$)

**Input:** Encrypted rectangle $E(B_0)$, encrypted data point $E(D_0)$
**Output:** Return true if $D_0 \in B_0$; Otherwise return false.

```
for i ← 1 to k do
    if ¬(DWITHIN(l_{0,k}, d_{0,k}, u_{0,k})) then
        return false
    end if
end for
return true
```

**Algorithm 6** secureIntersect($E(B_0)$, $E(B_1)$)

**Input:** Encrypted rectangle $E(B_0)$, $E(B_1)$
**Output:** Return true if $B_0$ intersects with $B_1$; Otherwise return false.

```
for i ← 1 to k do
    if (DLESS(u_{1,k}, l_{0,k})||DLESS(u_{0,k}, l_{1,k})) then
        return false
    end if
end for
return true
```

in other words, $x_i$ satisfies the similarity query, the protocol returns a encrypted result bit $E(b_i) = E(1)$, otherwise $E(b_i) = E(0)$. Next, $S_1$ generates a k-dimensional random vectors $\Gamma_i = \{\gamma_{i,1}, \ldots, \gamma_{i,k}\}$ and random value $\gamma_{i,0}$ for each candidate $x_i$ and calculates $E(x_i+\Gamma_i) = E(x_i)+\Gamma_i$ and $E(b_i+\gamma_{i,0}) = E(b_i)+\gamma_{i,0}$ to obfuscate plaintexts. Those random numbers belong to the SHE message space $\{m|m \in [-2^{k_1-1}, 2^{k_1-1})\}$. $S_1$ encrypts each random vector and random value with symmetric encryption and the session key $ssk$ generated from the query. In our scheme, we leverage AES as the encryption algorithm, denoted as $AES_{ssk}(\Gamma_i)||AES_{ssk}(\gamma_{i,0})$. Finally, $S_1$ construct a result set $R = \{(E(x_i+\Gamma_i), E(b_i+\gamma_{i,0}), AES_{ssk}(\Gamma_i)||AES_{ssk}(\gamma_{i,0}))|x_i \in C\}$ and sends to $S_2$.

Once $S_2$ receives the result set, it recovers the SHE plaintexts by the private key $sk$ and transfers the new result set $\{x_i + \Gamma_i, b_i + \gamma_{i,0}, AES_{ssk}(\Gamma_i)||AES_{ssk}(\gamma_{i,0})\}$ to the query user.

Because the query user also owns the session key $ssk$, it can remove random values

easily. Firstly, it decrypts the $AES_{ssk}(\gamma_{i,0})$ to obtain $b_i$ by removing the random value $\gamma_{i,0}$. If $b_i = 1$, it means that $x_i$ is one of the results of the range query, and then the query user continues to decrypt $AES_{ssk}(\Gamma_i)$ and deduces the random vector $\Gamma_i$ to obtain the original data $x_i$.

# Chapter 6

# Security Analysis

In this chapter, firstly we will present the security analysis of the SHE-based privacy-preserving protocol, including SLESSE and DWITHIN. And then we will analyse the security of our scheme.

## 6.1 Security of Privacy-Preserving Protocol

With the given ciphertexts {E($m_1$), E($m_2$)},the SLESSE protocol outputs the result of whether $m_1 < m_2$ while $S_1$ and $S_2$ can not obtain the information of plaintexts. In this protocol, $S_1$ holds the encrypted inputs (ciphertexts {E($m_1$), E($m_2$)}) and the encrypted output (E($b$)). Those messages are encrypted by SHE, which has been proven to be semantically secure against CPA [11]. Therefore, without the secret key $sk$, $S_1$ can not obtain any information of those messages from the ciphertexts. On the other side, $S_2$ can recover the plaintext $x = s * r_1 * (m_1 - m_2) + s * r_2$ with the $sk$. However, the random numbers $r_1$ and $r_2$ prevent the $S_2$ from obtaining the original information of $(m_1 - m_2)$. Besides, the coin flipping number $s$ obfuscates the result and $S_2$ can not figure out the relationship between $m_1$ and $m_2$ without the coin flipping number $s$.

We can use a similar way to analyze DLESS and DWITHIN protocols. However, $S_1$

directly knows the result of the protocol while $S_2$ can not figure it out due to the coin-flipping.

## 6.2 Security of Weighted Similarity Range Query

We will prove that our scheme is selectively secure by the real/ideal world model. Firstly, we will define the leakage function from the perspectives of $S_1$ and $S_2$. After that, we will present the definitions of the real-world model and ideal world model and prove that the probability of the adversaries in our security definition distinguishing the view of the real/ideal world model is negligible.

**Leakage Function:** The leakage function $L(\cdot)$ denotes the leak information to an entity. Here, we define the leak information to $S_1$ and $S_2$.

- $L(S_1)$**:** The information leakage to $S_1$ involves the public parameter $pb$, the encrypted R-Tree $E(T)$, the encrypted query token $(E(q), E(w), E(\tau), E(B_0)$, and ciphertext $\{E(-1)\}$. Also, because this is a tree-based search scheme, it is inevitable to leak the access pattern and search pattern [15]. Due to the R-Tree structure and DWITHIN and DLESS protocols, the relations among some plaintexts will also be revealed.

- $L(S_2)$**:** The information leakage to $S_2$ involves the public parameter $pb$, the secret key $sk$, and the number of data records in the candidate set.

**Real World Model:** In the real world model, there are two probabilistic polynomial time (PPT) adversaries $A_1$ and $A_2$ (assume that these two adversaries will not collude with each other) and the challenge and they interact as follows.

- **Initialization Phrase:** $A_1$ sends a dataset $X = \{x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,k}) | i = 1, 2, \ldots, n\}$ to the challenger. And then, the challenger follows the instructions of Section 5.2.1 to generate the SHE keys $\{pb, sk\}$. The challenger sends $pb$

to $A_1$ and $sk$ to $A_2$. As Section 5.2.2, the challenger builds up the R-Tree $T$ according to $\{X, P\}$ and encrypts it into $E(T)$ with SHE key.

- **Token Phrase 1:** $A_1$ submits $c_0$ weighted similarity range queries, denoted as $\{q_i, w_i, \tau_i | 1 \leq i \leq c_0\}$, where $c_0$ is a polynomial number, and sends to the challenger. The challenger applies the token generation algorithm in Section 5.2.3 and generates the query tokens $\{E(q_i), E(w_i), E(\tau_i), E(B_i) | 1 \leq i \leq c_0\}$. After that, the challenger returns these query tokens to $A_1$.

- **Challenge Phrase:** The challenger returns $E(T)$ to $A_1$.

- **Token Phrase 2:** $A_1$ submits $c_1 - c_0$ weighted similarity range queries, denoted as $\{q_i, w_i, \tau_i | c_0 + 1 \leq i \leq c_1\}$, where $c_1$ is also a polynomial number, and gets the query tokens $\{E(q_i), E(w_i), E(\tau_i), E(B_i) | c_0 + 1 \leq i \leq c_1\}$ from challenger.

- **Query Processing:** With these query tokens $\{E(q_i),$ $E(w_i), E(\tau_i), E(B_i) | 1 \leq i \leq c_1\}$, $A_1$ and $A_2$ cooperate to apply the weighted similarity range query on $E(T)$ to obtain the query result.

We denote the view of $A_1$ in the real world as $realView(A_1)$ which involves the query tokens, SHE ciphertexts from $E(T)$ and SLESSE protocol access pattern and search pattern of $E(T)$, the plaintexts relation from DWITHIN and DLESS protocols and R-Tree. Similarly, we denote the view of $A_2$ in the real world as $realView(A_2)$ which includes the AES ciphertexts in the result set, plaintexts in the result set and some plaintexts from protocols.

**Ideal World Model:** In the ideal world model, there are two probabilistic polynomial time (PPT) adversaries $A_1$ and $A_2$ (assume that these two adversaries will not collude with each other) and a simulator based on $L(S_1)$ and $L(S_2)$. They interact as follows.

- **Initialization Phrase:** $A_1$ sends a dataset $X = \{x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,k}) | i = 1, 2, \ldots, n\}$ to the simulator. And then, the simulator delivers $pb \in L(S_1)$ to $A_1$ and $sk \in L(S_2)$ to $A_2$. According to the $E(T) \in L(S_1)$, the simulator builds up a encrypted R-Tree $E'(T)$ which is isomorphic to $E(T)$. The main idea of constructing $E'(T)$ is to apply SHE self-binding [11]. Firstly, the simulator leverages $sk \in L(S_2)$ to compute two different ciphertexts of 0 $E_1(0)$ and $E_2(0)$. After that, for each ciphertext $E(x)$ in each node, the simulator calculates $E'(x) = E(x) + r_1 * E_1(0) + r_2 * E_2(0)$, where $r_1$ and $r_2$ are random numbers.

- **Token Phrase 1:** $A_1$ submits $c_0$ weighted similarity range queries, denoted as $\{q_i, w_i, \tau_i | 1 \leq i \leq c_0\}$ and sends to the simulator. The simulator generates the query tokens $\{E(q_i), E(w_i), E(\tau_i), E(B_i) | 1 \leq i \leq c_0\}$ with the SHE key in $L(S_1)$ and converts the query token into $\{E'(q_i), E'(w_i), E'(\tau_i), E'(B_i) | 1 \leq i \leq c_0\}$ by self-binding. After that, the simulator returns these query tokens to $A_1$.

- **Challenge Phrase:** The simulator returns $E'(T)$ to $A_1$.

- **Token Phrase 2:** $A_1$ submits $c_1 - c_0$ weighted similarity range queries and gets the query tokens $\{E'(q_i), E'(w_i), E'(\tau_i), E'(B_i) | c_0 + 1 \leq i \leq c_1\}$ from simulator.

- **Query Processing:** With these query tokens $\{E'(q_i), E'(w_i), E'(\tau_i), E'(B_i) | 1 \leq i \leq c_1\}$, $A_1$ and $A_2$ cooperate to apply the weighted similarity range query on $E'(T)$ to obtain the query result.

We denote the view of $A_1$ in the ideal world as $idealView(A_1)$ which involves the query tokens, SHE ciphertexts from $E'(T)$ and SLESSE protocol access pattern and search pattern of $E'(T)$, the plaintexts relation from DWITHIN and DLESS protocols and R-Tree. In the similar way, we denote the view of $A_2$ in the ideal world as $idealView(A_2)$ which includes the AES ciphertexts in the candidate set, some plaintexts received from protocols and candidate set.

**Definition 6.2.1** (Selevtive Security of Our Scheme)**.** *our scheme is selectively secure with $L(S_1)$ and $L(S_2)$ iff, for any two PPT adversaries $A_1$ and $A_2$, there exists a simulator that the probability of $A_1$ and $A_2$ to distinguish the real world model and ideal world model is negligible.*

**Theorem 6.2.1.** *Our scheme is selectively secure with the leakage $L(S_1)$ and $L(S_2)$ iff the SHE technique is semantically secure against CPA.*

**Proof:** According to Definition 6.2.1, to prove the selective security of our scheme, we will show that $A_1$ and $A_2$ can not distinguish between the view of the real world and the ideal world respectively, as follows.

- **View of $A_1$:** Both $idealView(A_1)$ and $realView(A_1)$ include the query tokens, SHE-encrypted R-tree, the access pattern and search pattern of R-Tree. As for the $E(T)$ and $E'(T)$, because $E(T)$ is isomorphic to $E'(T)$, $A_1$ can not distinguish these two views from the tree structures. The internal nodes and leaf nodes in $E'(T)$ are generated by the self-binding from $E(T)$, also these nodes are encrypted by SHE which has been proven to IND-CPA [11], so $A_1$ can not distinguish the nodes in $E(T)$ and $E'(T)$. Correspondingly, $A_1$ can not distinguish the ciphertexts of SLESSE protocol and query token from $idealView(A_1)$ and $realView(A_1)$ because they also are encrypted by SHE. Because of the self-binding in $E'(T)$, the ciphertexts still hold up the same plaintext relation. As a consequence, the same query token will have the same DLESS and DWITHIN results as well as the search path on the tree structure. Therefore, $A_1$ can not distinguish the access pattern and search pattern of $E(T)$ and $E'(T)$. Therefore, $A_1$ can not distinguishing $idealView(A_1)$ and $realView(A_1)$.

- **View of $A_2$:** Both $idealView(A_2)$ and $realView(A_2)$ include the AES ciphertexts in the candidate set, some plaintexts received from privacy-preserving

protocols and candidate set. Because the AES ciphertexts are encrypted from random numbers, $A_2$ can not distinguish the ciphertexts from $idealView(A_2)$ and $realView(A_2)$. As for the plaintexts from protocols and candidate set, they are all garbled by random numbers, so $A_2$ can not distinguish the plaintexts from the real world and ideal world.

In conclusion, $A_1$ and $A_2$ can not distinguish between the view of the real world and the ideal world.

# Chapter 7

# Experiments

In this chapter, we conduct a simulation of WeightedSim to evaluate the performance of the weighted similarity range query. Specifically, to verify the efficiency of the filtration step of our scheme, we compare WeightedSim with the naive scheme that doesn't apply filtration in the query. The computational cost is correlated with three factors: the dataset size $N$, the data dimension $d$ and the query threshold $\tau$. In the following part, we will first introduce the experimental setting. After that, we will conduct the experiments around these three factors.

## 7.1 Experimental Setting

We implement the WeightedSim and naive scheme with Java (JDK 15 as execution environment) and execute on a machine with 16 GB memory, 2.90 GHz AMD Ryzen 7 4800H CPU and Win 11 OS. The Java implementation of WeightedSim is presented at Appendix B. In the simulation, we evaluate the performance on the EEG dataset [57], which contains 14980 rows of data with 15 attributes. As for the R-Tree, we leverage R*-Tree [58] as the data index construction strategy and implement the scheme based on an open-source R-Tree repository [59]. Also, the security parameters of SHE are set as $k_0 = 1200, k_1 = 30, k_2 = 80$. Therefore, we can figure out the

maximum multiplicative depth $\theta = 6$. All the SHE privacy-preserving protocols are involved within this maximum depth. Also, we may guarantee a strong enough security level and large enough a message space for our experimental data. The summary of the experimental environment is shown in Table 7.1.

Table 7.1: Experimental Environment

| **Parameter** | **Setting** |
|---|---|
| Evaluating CPU | 2.90 GHz AMD Ryzen 7 4800H |
| Evaluating RAM | 16GB |
| Programming Language | Java |
| Execution Environment | JDK 15 |
| Operating System | Win 11 |
| SHE Security Parameters | $k_0 = 1200, k_1 = 30, k_2 = 80$ |
| Dataset | [57] |

## 7.2 Experimental Results

### 7.2.1 Computational Cost versus $N$

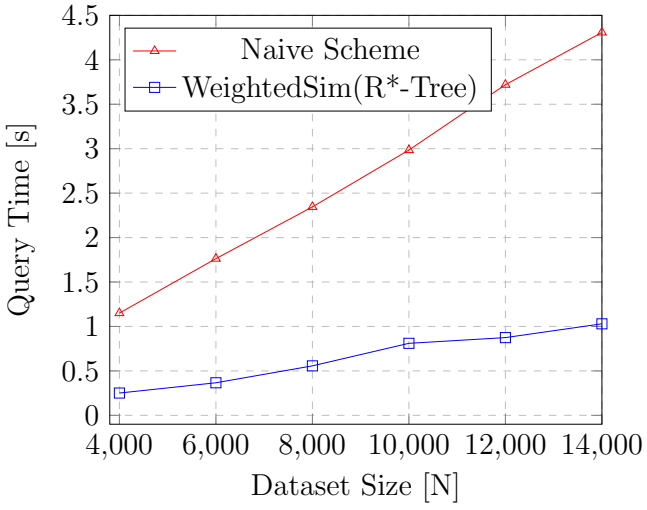


Figure 7.1: Computational Cost Varying with Dataset Size

In our scheme, the similarity query time increases correspondingly by the size of the

dataset. In this experiment, we set up the parameters as $N$ ranging from 4000 to 14000, $\tau = 6$ and $d = 5$. The Figure 7.1 indicates the computational cost of similarity query between our scheme and the naive scheme varying with the dataset size $N$. As the size of the dataset increases, so does the query time in the naive scheme. As the dataset size grows, WeightedSim must construct a larger index in the R-Tree, which causes the query time to increase. From the diagram, we can learn that after applying the filtration strategy, our scheme can considerably reduce the computation cost compared with the naive scheme. The reason is that it can preclude the data points that cannot satisfy the weighted similarity range query and reduce the time of calculating the weighted Euclidean distance. Besides, the increasing rate of our scheme is smaller than the naive scheme.

### 7.2.2 Computational Cost versus $d$

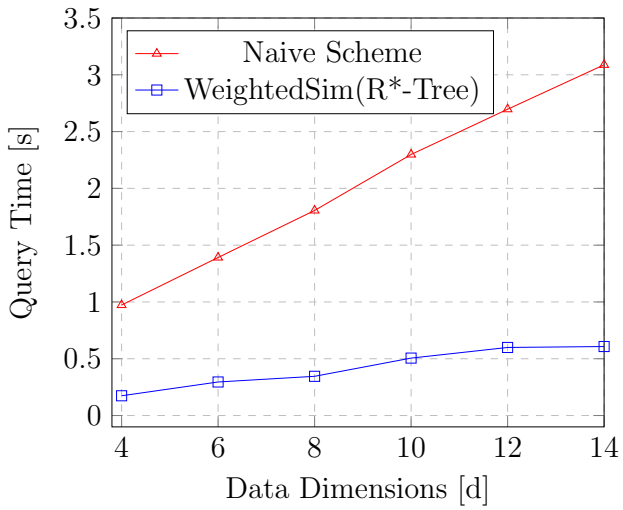


Figure 7.2: Computational Cost Varying from Data Dimension

In our scheme, the performance of the similarity query is also affected by the size of the dataset. In this experiment, we set up the parameters as $N = 4000$, $\tau = 6$ and $d$ ranging from 4 to 14 and the experiment result is as shown in the Figure 7.2. The increase of data dimension results in the increase of the computation time of

weighted distance in both naive scheme and WeightedSim and the range query in encrypted R-tree in WeightedSim. From the diagram, we can know that the query time will also increase with the increment of the data dimension in both schemes and our scheme is much more efficient than the naive scheme. Also, the increase of data dimension only impacts slightly on the performance in the WeightedSim compared with the naive scheme.

### 7.2.3 Computational Cost versus $\tau$

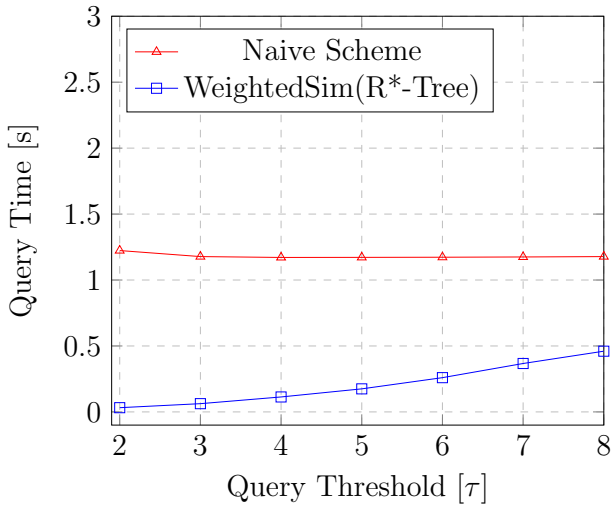


Figure 7.3: Computational Cost Varying with Query Threshold

In our scheme, the query threshold also impacts the query efficiency. With the increase of $\tau$, the size of the candidate set in the filtration step will become larger so that the query time will also increase correspondingly. In the experiment, we set up the parameters as $N = 4000$, $\tau$ ranging from 2 to 8 and $d = 5$ and plot the computational cost varying with $\tau$ in Figure 7.3. The chart shows that the query time of our scheme increases by the increase of $\tau$ while the computational cost of the naive scheme stays stable. The reason for this circumstance is that as the query threshold $\tau$ increases, the WeightedSim will collect a larger candidate set and spend more time calculating the weighted distance, while the naive scheme always

calculates the same data records regardless of the change in $\tau$.

# Chapter 8

# Conclusion and Future Work

In this chapter, we will first summarize our WeightedSim scheme. After that, we discuss some feasible extensions and directions for the future work.

## 8.1 Conclusion

In our scheme, we proposed an efficient and privacy-preserving weighted similarity range query scheme (WeightedSim) for the healthcare data outsourcing scenario under the two cloud servers model. Specifically, we designed an encrypted R-Tree as the encrypted data index for the healthcare data using the SHE technique, followed by the corresponding SHE privacy-preserving protocols on the two cloud servers model for the range query on the encrypted R-Tree. Using the provided encrypted R-Tree, we set up the filtration and refinement strategies to apply the weighted similarity range query over the ciphertexts. Finally, the security analysis shows that our scheme is selectively secure as long as SHE is IND-CPA and the experimental results demonstrate that our scheme is efficient. In the future, we intend to investigate a more effective data structure or method for enhancing the performance of the weighted similarity range query on healthcare data while preserving the privacy of the data.

## 8.2 Future Work

In this thesis, we present an efficient and privacy-preserving weighted similarity range query scheme for healthcare applications. However, there are still several open problems that can be explored in our future work. In this section, we discuss some research problems and their corresponding possible solutions.

- Our study is based on the honest-but-curious and no-collusion security assumptions, in which the semi-trusted cloud server will follow the instruction to offer the query service for the user but be curious and try to learn the private data from the encrypted data. The security analysis in Section 6 demonstrated that our scheme is selectively secure without leaking the extra private information under the honest-but-curious assumption. However, we can hardly guarantee that the cloud server will adhere to our security assumptions in practice. In our future work, we may attempt to improve the security level of our scheme with Trusted Execution Environment (TEE) [60] techniques such as Intel Software Guard Extensions (SGX) [61] to ensure that the server executes the instructions honestly and returns the desired output.

- Experiments in Section 7 indicate that our filtration and refinement strategies may improve the efficacy of weighted similarity range queries. The result in Figure 7.3 shows the data query time strongly related to the query threshold $\tau$. As $\tau$ expands, the improvement of our strategies will become less effective. In the future, we will adjust our scheme to be more expected and robust by introducing new methods that adapt our data query scheme to arbitrary query thresholds.

- Our scheme relies on the two cloud servers model. In reality, it is impossible to ignore the communication cost between two cloud servers. In our future work, we will also design an efficient and privacy-preserving weighted similarity range

query scheme under the single cloud server model based on suitable encryption techniques such as predicate encryption [4].

- In this thesis, we proposed a weighted Euclidean distance-based similarity range query scheme for healthcare data queries. However, a single data metric can not fulfil all the requirements of various healthcare applications in reality. Personalized similarity queries, whose query methods permit query users to specify their individual distance measures, are more desirable and versatile compared with similarity queries based on a limited data metric but also more challenging simultaneously. In the future, we will apply different data indexing and encryption techniques to implement the personalized similarity query.

# Bibliography

[1] Hadi Habibzadeh, Karthik Dinesh, Omid Rajabi Shishvan, Andrew Boggio-Dandry, Gaurav Sharma, and Tolga Soyata. A survey of healthcare internet of things (hiot): A clinical perspective. *IEEE Internet of Things Journal*, 7(1):53–71, 2020.

[2] Zeeshan Ahmed, Khalid Mohamed, Saman Zeeshan, and XinQi Dong. Artificial intelligence with multi-functional machine learning platform development for better healthcare and precision medicine. *Database*, 2020, 03 2020. baaa010.

[3] Jiafeng Hua, Hui Zhu, Fengwei Wang, Ximeng Liu, Rongxing Lu, Hao Li, and Yeping Zhang. Cinema: Efficient and privacy-preserving online medical primary diagnosis with skyline query. *IEEE Internet of Things Journal*, 6(2):1450–1461, 2019.

[4] Yandong Zheng, Rongxing Lu, Yunguo Guan, Jun Shao, and Hui Zhu. Efficient privacy-preserving similarity range query with quadsector tree in ehealthcare. *IEEE Transactions on Services Computing*, pages 1–1, 2021.

[5] Rong Jiang, Rongxing Lu, and Kim-Kwang Raymond Choo. Achieving high performance and privacy-preserving query over encrypted multidimensional big metering data. *Future Generation Computer Systems*, 78:392–401, 2018.

[6] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.

[7] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[8] Wanyu Lin, Helei Cui, Baochun Li, and Cong Wang. Privacy-preserving similarity search with efficient updates in distributed key-value stores. *IEEE Transactions on Parallel and Distributed Systems*, 32(5):1072–1084, 2021.

[9] Fuyuan Song, Zheng Qin, Liang Xue, Jixin Zhang, Xiaodong Lin, and Xuemin Shen. Privacy-preserving keyword similarity search over encrypted spatial data in cloud computing. *IEEE Internet of Things Journal*, 9(8):6184–6198, 2022.

[10] Jiacheng Jin, Yandong Zheng, and Pulei Xiong. Epsim-gs: Efficient and privacy-preserving similarity range query over genomic sequences. In *2021 18th International Conference on Privacy, Security and Trust (PST)*, pages 1–10, 2021.

[11] Yandong Zheng, Rongxing Lu, Yunguo Guan, Jun Shao, and Hui Zhu. Efficient and privacy-preserving similarity range query over encrypted time series data. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2021.

[12] Yandong Zheng, Rongxing Lu, and Songnian Zhang. Achieving privacy-preserving weighted similarity range query over outsourced ehealthcare data. *IEEE ICC'22, Seoul, South Korea*, 2022.

[13] Songnian Zhang, Suprio Ray, Rongxing Lu, Yandong Zheng, Yunguo Guan, and Jun Shao. Achieving efficient and privacy-preserving dynamic skyline query in online medical diagnosis. *IEEE Internet of Things Journal*, pages 1–1, 2021.

[14] Hassan Mahdikhani, Rongxing Lu, Yandong Zheng, Jun Shao, and Ali A. Ghorbani. Achieving o($\log^3$n) communication-efficient privacy-preserving range query in fog-based iot. *IEEE Internet of Things Journal*, 7(6):5220–5232, 2020.

[15] Boyang Wang, Yantian Hou, Ming Li, Haitao Wang, and Hui Li. Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 111–122, 2014.

[16] Mihir Bellare and Phillip Rogaway. Introduction to modern cryptography. *Ucsd Cse*, 207:207, 2005.

[17] Omar G Abood and Shawkat K Guirguis. A survey on cryptography algorithms. *International Journal of Scientific and Research Publications*, 8(7):495–516, 2018.

[18] William Stallings. *Cryptography and Network Security Principles and Practices*. 2005.

[19] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[20] National Institute of Standards and Technology. Data encryption standard (des). FIPS Publication 46-3, October 1999.

[21] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *International Workshop on Fast Software Encryption*, pages 191–204. Springer, 1993.

[22] Joan Daemen and Vincent Rijmen. Reijndael: The advanced encryption standard. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 26(3):137–139, 2001.

[23] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block

cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 450–466, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[24] Xuejia Lai and James L. Massey. A proposal for a new block encryption standard. pages 389–404. Springer-Verlag, 1991.

[25] Daniel J Bernstein et al. Chacha, a variant of salsa20. In *Workshop record of SASC*, volume 8, pages 3–5. Lausanne, Switzerland, 2008.

[26] Caroline Fontaine. *RC4*, pages 1031–1032. Springer US, Boston, MA, 2011.

[27] Don Johnson and Alfred Menezes. The elliptic curve digital signature algorithm (ecdsa). Technical report, 1999.

[28] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[29] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[30] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[31] Eric Rescorla and Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008.

[32] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), jul 2018.

[33] Douglas R Stinson. *Cryptography: theory and practice*. Chapman and Hall/CRC, 2005.

[34] Shafi Goldwasser and Silvio Micali. Probabilistic encryption; how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, page 365–377, New York, NY, USA, 1982. Association for Computing Machinery.

[35] Josh D. C. Benaloh. *Verifiable secret-ballot elections*. PhD thesis, 1987.

[36] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[37] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography*, pages 325–341, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[38] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[39] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[40] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 505–524, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[41] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption.

In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 1219–1234, New York, NY, USA, 2012. Association for Computing Machinery.

[42] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., USA, 1st edition, 1996.

[43] Bing Wang, Wei Song, Wenjing Lou, and Y. Thomas Hou. Privacy-preserving pattern matching over encrypted genetic data in cloud computing. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

[44] Dan Zhu, Hui Zhu, Xiangyu Wang, Rongxing Lu, and Dengguo Feng. Efficient and privacy-preserving similar patients query scheme over outsourced genomic data. *IEEE Transactions on Cloud Computing*, pages 1–1, 2021.

[45] Xiangyu Wang, Jianfeng Ma, Yinbin Miao, Ruikang Yang, and Yijia Chang. Epsmd: An efficient privacy-preserving sensor data monitoring and online diagnosis system. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 819–827, 2018.

[46] Yandong Zheng, Rongxing Lu, Songnian Zhang, Yunguo Guan, Jun Shao, Fengwei Wang, and Hui Zhu. Pmrq: Achieving efficient and privacy-preserving multi-dimensional range query in ehealthcare. *IEEE Internet of Things Journal*, pages 1–1, 2022.

[47] Chang Xu, Ningning Wang, Liehuang Zhu, Chuan Zhang, Kashif Sharif, and Huishu Wu. Reliable and privacy-preserving top-k disease matching schemes for e-healthcare systems. *IEEE Internet of Things Journal*, 9(7):5537–5547, 2022.

[48] Mohammad Saidur Rahman, Ibrahim Khalil, Nour Moustafa, Aditya Pribadi Kalapaaking, and Abdelaziz Bouras. A blockchain-enabled privacy-preserving

verifiable query framework for securing cloud-assisted industrial internet of things systems. *IEEE Transactions on Industrial Informatics*, 18(7):5007–5017, 2022.

[49] Sai Wu, Dawei Jiang, Beng Chin Ooi, and Kun-Lung Wu. Efficient b-tree based indexing for cloud data processing. *Proc. VLDB Endow.*, 3(1–2):1207–1218, sep 2010.

[50] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 224–241. Springer, 2009.

[51] Yunguo Guan, Rongxing Lu, Yandong Zheng, Songnian Zhang, Jun Shao, and Guiyi Wei. Toward privacy-preserving cybertwin-based spatiotemporal keyword query for its in 6g era. *IEEE Internet of Things Journal*, 8(22):16243–16255, 2021.

[52] Quan Yu, Jing Ren, Yinjin Fu, Ying Li, and Wei Zhang. Cybertwin: An origin of next generation network architecture. *IEEE Wireless Communications*, 26(6):111–117, 2019.

[53] Emmanuel Bresson, Dario Catalano, and David Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003*, pages 37–54, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[54] Jia-Nan Liu, Jian Weng, Anjia Yang, Yizhao Chen, and Xiaodong Lin. Enabling efficient and privacy-preserving aggregation communication and function query for fog computing-based smart grid. *IEEE Transactions on Smart Grid*, 11(1):247–257, 2020.

[55] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.

[56] Boyang Wang, Yantian Hou, and Ming Li. Practical and secure nearest neighbor search on encrypted large-scale data. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[57] EEG Eye State. `https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State`.

[58] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, may 1990.

[59] rtree multi. `https://github.com/davidmoten/rtree-multi`.

[60] Nir Drucker and Shay Gueron. Combining homomorphic encryption with trusted execution environment: a demonstration with paillier encryption and sgx. In *Proceedings of the 2017 international workshop on managing insider security threats*, pages 85–88, 2017.

[61] Intel. Intel software guard extensions. `https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html`.

[62] WeightedSim Github. `https://github.com/guojuntang/WeightedSim`.

# Appendix A

# The Correctness of SHE technique

In this chapter, we will prove the correctness of SHE technique[14], include the decryption and homomorphic properties.

## A.1 The Correctness of Decryption

A ciphertext $E(m) = (rL+m)(1+r'p)\ mod\ N$ can be decrypted to $m$ correctly with $sk \leftarrow (p, L)$, the proof is as follows:

$$\begin{aligned} SHEDec(sk, E(m)) &= (E(m)\ mod\ p)\ mod\ L \\ &= (((rL+m)(1+r'p)\ mod\ N)\ mod\ p)\ mod\ L \\ &= (rL+m)\ mod\ L\ (\because 2k_2 < k_0) \\ &= m\ (\because k_1 \ll k_2) \end{aligned}$$

## A.2 The Correctness of Homomorphic Properties

Given the public parameter $pb \leftarrow (k_0, k_1, k_2, N)$, we can apply the homomorphic properties in Section 2.1.4.1, and the proof of correctness is in the followed sections.

### A.2.1 The Correctness of Homomorphic Addition-I

Given $pb$ and two ciphertexts $E(m_1) = (r_1L + m_1)(1 + r'_1p)\ mod\ N$ and $E(m_2) = (r_2L + m_2)(1 + r'_2p)\ mod\ N$, we have $E(m_1) + E(m_2)\ mod\ N \rightarrow E(m_1 + m_2)$. The proof is shown as follows:

$$
\begin{aligned}
& E(m_1) + E(m_2)\ mod\ N \\
& = (r_1L + m_1)(1 + r'_1p) + (r_2L + m_2)(1 + r'_2p)\ mod\ N \\
& = ((r_1 + r_2)L + m_1 + m_2 + \alpha p)\ mod\ N
\end{aligned}
$$

where $\alpha = (r_1r'_1 + r_2r'_2)L + (m_1r'_1 + m_2r'_2)$.

When we try to decrypt $E(m_1) + E(m_2)\ mod\ N$, we have:

$$
\begin{aligned}
& SHEDec(sk, E(m_1) + E(m_2)) \\
& = ((((r_1 + r_2)L + m_1 + m_2 + \alpha p)\ mod\ N)\ mod\ p)\ mod\ L \\
& = ((r_1 + r_2)L + m_1 + m_2)\ mod\ L\ (\because 2k_2 < k_0) \\
& = m_1 + m_2\ (\because k_1 \ll k_2)
\end{aligned}
$$

We can decrypt it to $m_1 + m_2$. Therefore, $E(m_1) + E(m_2)\ mod\ N \rightarrow E(m_1 + m_2)$.

### A.2.2 The Correctness of Homomorphic Multiplication-I

Given $pb$ and two ciphertexts $E(m_1) = (r_1L + m_1)(1 + r'_1p)\ mod\ N$ and $E(m_2) = (r_2L + m_2)(1 + r'_2p)\ mod\ N$, when we apply the homomorphic multiplication:

$$\begin{aligned}
&E(m_1 * m_2)\ mod\ N \\
&= E(m_1) * E(m_2)\ mod\ N \\
&= (r_1L + m_1)(1 + r'_1p)(r_2L + m_1)(1 + r'_2p)\ mod\ N \\
&= (m_1 * m_2 + r_1r_2L^2 + r_1Lm_2 + r_2Lm_1 + \Delta * p)\ mod\ N
\end{aligned}$$

$\Delta$ is denotes as the coefficient of p. When we try to decrypt the ciphertext of $E(m_1 * m_2)$:

$$\begin{aligned}
&((E(m_1 * m_2)\ mod\ N)\ mod\ p)\ mod\ L) \\
&= ((((m_1 * m_2 + r_1r_2L^2 + r_1Lm_2 + r_2Lm_1 + \Delta * p)\ mod\ N)\ mod\ p)\ mod\ L) \\
&= ((((m_1 * m_2 + r_1r_2L^2 + r_1Lm_2 + r_2Lm_1)\ mod\ N)\ mod\ p)\ mod\ L)
\end{aligned}$$

If we want to decipher the ciphertext correctly, we should ensure $m_1 * m_2 + r_1r_2L^2 + r_1Lm_2 + r_2Lm_1) < p$. According to $m_1, m_2 \in \{0,1\}^{k_1}$, $r_1, r_2, L \in \{0,1\}^{k_2}$ and $k_2 \gg k_1$, we can conduct that $r_1r_2L^2$ is the largest noise term. To guarantee the correctness of decryption, we are supposed to ensure $r_1r_2L^2 < p$. Therefore, we set up the maximum depth of homomorphic multiplication-I, denoted as $\theta = \lfloor \frac{k_0}{2k_2} - 1 \rfloor$. Therefor, if the multiplicative depth is within $\theta$, we have:

$$
\begin{aligned}
&((((m_1 * m_2 + r_1 r_2 L^2 + r_1 L m_2 + r_2 L m_1) \ mod \ N) \ mod \ p) \ mod \ L) \\
&= ((r_1 r_2 L + r_1 m_2 + r_2 m_1) L + m_1 * m_2) \ mod \ L \\
&= m_1 * m_2 \ (\because k_1 \ll k_2)
\end{aligned}
$$

Homomorphic multiplication-I is established if the multiplicative depth is involved within $\theta$.

### A.2.3 The Correctness of Homomorphic Addition-II

Given $pb$ and a ciphertext $E(m_1) = (r_1 L + m_1)(1 + r'_1 p) \ mod \ N$ and a message $m_2$, we have $E(m_1) + m_2 \ mod \ N \rightarrow E(m_1 + m_2)$. The proof is shown as follows:

$$
\begin{aligned}
&E(m_1) + m_2 \ mod \ N \\
&= (r_1 L + m_1)(1 + r'_1 p) + m_2 \ mod \ N \\
&= (r_1 L + m_1 + m_2 + (r_1 L + m_1) r'_1 p) \ mod \ N
\end{aligned}
$$

When we try to decrypt $E(m_1) + m_2 \ mod \ N$, we have:

$$
\begin{aligned}
&SHEDec(sk, E(m_1) + m_2) \\
&= (((r_1 L + m_1 + m_2 + (r_1 L + m_1) r'_1 p) \ mod \ N) \ mod \ p) \ mod \ L \\
&= (r_1 L + m_1 + m_2) \ mod \ L \ (\because 2k_2 < k_0) \\
&= m_1 + m_2 \ (\because k_1 \ll k_2)
\end{aligned}
$$

We can decrypt it to $m_1 + m_2$. Therefore, $E(m_1) + m_2 \ mod \ N \rightarrow E(m_1 + m_2)$.

### A.2.4 The Correctness of Homomorphic Multiplication-II

Given $pb$ and a ciphertext $E(m_1) = (r_1 L + m_1)(1 + r'_1 p)\ mod\ N$ and a message $m_2 (m_2 > 0)$, we have $E(m_1) * m_2\ mod\ N \rightarrow E(m_1 * m_2)$. The proof is shown as follows:

$$
\begin{aligned}
& E(m_1) * m_2\ mod\ N \\
& = (r_1 L + m_1)(1 + r'_1 p) * m_2\ mod\ N \\
& = (r_1 m_2 L + m_1 * m_2 + (r_1 L + m_1) r'_1 p m_2)\ mod\ N
\end{aligned}
$$

When we try to decrypt $E(m_1) * m_2\ mod\ N$, we have:

$$
\begin{aligned}
& SHEDec(sk, E(m_1) + m_2) \\
& = (((r_1 m_2 L + m_1 * m_2 + (r_1 L + m_1) r'_1 p m_2)\ mod\ N)\ mod\ p)\ mod\ L \\
& = (r_1 m_2 L + m_1 * m_2)\ mod\ L\ (\because 2k_2 < k_0) \\
& = m_1 * m_2\ (\because k_1 \ll k_2)
\end{aligned}
$$

We can decrypt it to $m_1 * m_2$. Therefore, $E(m_1) * m_2\ mod\ N \rightarrow E(m_1 + m_2)$ if $m_2 > 0$.

# Appendix B

# Java Implementation of Core Modules

In this chapter, we will have a brief introduction of filtration and refinement strategies in our scheme as well as part of their Java implementations. The entire Java implementation is provided by [62].

## B.1 Implementation of Filtration

The main idea of the filtration strategy is to construct the query rectangle and apply a range query on the R-Tree by determining whether the query rectangle intersects with internal rectangles (nodes). Therefore, it can exclude the data points that are impossible to include within the weighted Euclidean distance from the query point. As for the tree search, we adopt a Breadth-first search (BFS) strategy to traverse the R-Tree. Because the data are encrypted by the SHE technique, we check the intersection by privacy-preserving protocols in Section 4.3. The detailed implementation is shown in Listing B.1.

Listing B.1: Java Implementation of Filtration

```
public List<EncryptedLeaf<T>> search(EncryptedRectangle r, DLESSProtocol
    dless, DWITHINProtocol dwithin){
        List<EncryptedLeaf<T>> result = new ArrayList<>();
        EncryptedNode node;
        EncryptedNonLeaf nonLeaf;
        int size;
        queue.add(root);
        while (!queue.isEmpty()){
            node = queue.poll();
            // leaf node, check if it is inside the rectangle
            if (node.isLeaf()){
                if (node.test(r, dwithin)){
                    result.add((EncryptedLeaf<T>) node);
                }
            }else{
                if (node.test(r, dless)){
                    nonLeaf = (EncryptedNonLeaf) node;
                    size = nonLeaf.getChildren().size();
                    for (int i = 0; i < size; i++) {
                        queue.add(nonLeaf.getChild(i));
                    }
                }
            }
        }
        return result;
    }
```

## B.2 Implementation of Refinement

In the refinement strategy, $S_1$ and $S_2$ will collaborate to return the accurate query result to the query user with disclosing the private information. Initially, $S_1$ calculates the weighted Euclidean distance by the SHE homomorphic properties. After that, $S_1$ masks the result with some random numbers and encrypts them with the session key shard with the query user. And $S_1$ delivers the result to $S_2$. The detailed implementation is shown in Listing B.2.

Listing B.2: Java Implementation of Refinement (S1)

```
private Pair<SHECipher, byte[]> maskResultBit(SHECipher b, SecretKey
    aesKey){
    BigInteger gamma = new BigInteger(pb.getK1(), rnd);
    SHECipher b_pi = SymHomEnc.hm_add(b, gamma, pb);
    byte[] gamma_cipher = AES.encrypt(gamma.toByteArray(), aesKey);
    return new Pair<>(b_pi, gamma_cipher);
}

public List<RefinementCandidate> refinement(EncryptedToken token,
    List<EncryptedLeaf<SHECipher[]>> candidate_set){
    List<RefinementCandidate> result = new ArrayList<>();
    SHECipher result_bit;
    SHECipher encryptedWeightedDis;
    for (EncryptedLeaf<SHECipher[]> x: candidate_set) {
        encryptedWeightedDis =
            calEncryptedWeightedEuclideanDisSquare(x.getValue(),
            token.getEncryptedQ(), token.getEncryptedW());
        result_bit = slesseProtocol.run(encryptedWeightedDis,
            token.getEncryptedTauSquare());
```

```
            result.add(new
                RefinementCandidate(maskEncryptedData(x.getValue(),
                token.getSsk()), maskResultBit(result_bit,
                token.getSsk())));
        }
        return result;
    }
```

Due to the fact that only $S_2$ possesses the SHE private key, the query result must be decrypted by $S_2$ prior to being returned to the query user. However, $S_2$ lacks knowledge of the session key, thus it cannot access the query result information. The detailed implementation is shown in Listing B.3.

Listing B.3: Java Implementation of Refinement (S2)

```
    public RefinementResult(RefinementCandidate candidate, SHEPrivateKey
        sk){
        SHECipher[] x_vector = candidate.getEncryptedData();
        int size = x_vector.length;
        this.x_pi = new BigInteger[size];
        // decrypt x_pi
        for (int i = 0; i < size; i++) {
            x_pi[i] = SymHomEnc.dec(x_vector[i], sk);
        }
        this.b_pi = SymHomEnc.dec(candidate.getResultBit(), sk);
        this.dataMasks = candidate.getDataMasks();
        this.resultMasks = candidate.getResultMasks();
    }

    public List<RefinementResult> refinement(List<RefinementCandidate>
```

```
    candidate){
    List<RefinementResult> result = new ArrayList<>();
    for (RefinementCandidate c:
         candidate) {
        result.add(new RefinementResult(c, sk));
    }
    return result;
}
```

Finally, with the shared session key, the query user is able to decrypt random numbers from $S_1$ and retrieve the query result by removing the random numbers. The result of the weighted similarity range query is the data record whose result bit is equal to 1. The detailed implementation is shown in Listing B.4.

Listing B.4: Java Implementation of Refinement (Query User)

```
private BigInteger[] decryptData(RefinementResult r, SecretKey ssk){
    BigInteger[] x_pi = r.getX_pi();
    List<byte[]> masks= r.getDataMasks();
    BigInteger gamma;
    int size = x_pi.length;

    BigInteger[] result = new BigInteger[size];
    for (int i = 0; i < size; i++) {
        gamma = new BigInteger(AES.decrypt(masks.get(i), ssk));
        result[i] = x_pi[i].subtract(gamma);
    }
    return result;
}
```

```
public List<BigInteger[]> refinement(SecretKey ssk,
    List<RefinementResult> set){
    List<BigInteger[]> result = new ArrayList<>();
    for (RefinementResult r:
         set) {
        if (checkResult(r, ssk)){
            result.add(decryptData(r, ssk));
        }
    }
    return result;
}
```

# Vita

**Candidate's full name**: Guojun Tang
**University attended**:

- Bachelor of Computer Science, South China Normal University, Guangdong, China, 2015-2019
- Master of Computer Science, University of New Brunswick, New Brunswick, Canada, 2021-2022

**Publications**:
Guojun Tang, Rongxing Lu, and Mohammad Mamun. WeightedSim: privacy-preserving weighted similarity query over encrypted healthcare data. (Under review)

**Presentations**:
Guojun Tang. WeightedSim: Privacy-preserving Weighted Similarity Query over Encrypted Healthcare Data. Cyber Security Research Group, Fredericton , NRC Digital Technology Research Center, on April 28, 2022.