Fall 2015

Privacy-preserving query processing over encrypted data in cloud

Yousef M. Elmehdwi

Recommended Citation

Elmehdwi, Yousef M., "Privacy-preserving query processing over encrypted data in cloud" (2015). *Doctoral Dissertations*. 2442.
https://scholarsmine.mst.edu/doctoral_dissertations/2442

PRIVACY-PRESERVING QUERY PROCESSING OVER ENCRYPTED DATA IN CLOUD

by

YOUSEF M. ELMEHDWI

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2015

Approved by

Dr. Wei Jiang, Advisor
Dr. Ali Hurson
Dr. Dan Lin
Dr. Xuerong (Meggie) Wen
Dr. Zhaozheng Yin

ABSTRACT

The query processing of relational data has been studied extensively throughout the past decade. A number of theoretical and practical solutions to query processing have been proposed under various scenarios. With the recent popularity of cloud computing, data owners now have the opportunity to outsource not only their data but also data processing functionalities to the cloud. Because of data security and personal privacy concerns, sensitive data (e.g., medical records) should be encrypted before being outsourced to a cloud, and the cloud should perform query processing tasks on the encrypted data only. These tasks are termed as Privacy-Preserving Query Processing (PPQP) over encrypted data. Based on the concept of Secure Multiparty Computation (SMC), SMC-based distributed protocols were developed to allow the cloud to perform queries directly over encrypted data. These protocols protect the confidentiality of the stored data, user queries, and data access patterns from cloud service providers and other unauthorized users. Several queries were considered in an attempt to create a well-defined scope. These queries included the k-Nearest Neighbor (kNN) query, advanced analytical query, and correlated range query. The proposed protocols utilize an additive homomorphic cryptosystem and/or a garbled circuit technique at different stages of query processing to achieve the best performance. In addition, by adopting a multi-cloud computing paradigm, all computations can be done on the encrypted data without using very expensive fully homomorphic encryptions. The proposed protocols' security was analyzed theoretically, and its practicality was evaluated through extensive empirical results.

ACKNOWLEDGMENTS

I would like to express my gratitude to all those who have helped me with this thesis. First, I would like to extend my sincere thanks to my advisor, Dr. Wei Jiang, for his continuous suggestions, encouragement, and motivation, which greatly helped me carry through difficult times. I consider him more than just an advisor, but a dear friend.

I would also like to thank the members of my Ph.D. committee, Prof. Ali Hurson, Dr. Dan Lin, Dr. Xuerong (Meggie) Wen, and Dr. Zhaozheng Yin, for their constructive and valuable suggestions.

I appreciate the advice and support of Gerry Howser, Hu Chun, Bharath Samanthula, Amir Bahmani, and Michael Howard. Thanks should be extended also to the staff members of the computer department and to Amy Ketterer, in the graduate editing services, for their great assistance.

Lastly, but most importantly, I owe a great debt to my father Mohammed, to whom I would like to dedicate this work- for his undying love and faith in me-, my mother Salma, my dear wife Azza, my lovely kids, Sammy and Mohammed, and all other family members. With their support over the years, I have overcome the difficulties in my study and life. Thank you all.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

1. INTRODUCTION

Cloud computing [51, 66] enables an entity to outsource both its database and its data processing functionalities. The cloud provides access mechanisms for not only querying but also managing the hosted database. When data is outsourced, the data owner can enjoy the benefits of reduced data management costs, reduced data storage overhead, and improved service. Unfortunately, the cloud cannot be fully trusted; preserving data confidentiality and query privacy is a challenging task. This is because it is difficult for the cloud to actually guarantee the confidentiality of sensitive data. The challenge arises due to several well-documented security risks faced by existing cloud service providers [31, 47, 81, 82, 94, 103]. One such risk, for instance, is that when a breach occurs in the cloud, any sensitive data stored in the clear (i.e., not encrypted) can be easily exposed to the attacker.

One straightforward way to protect the confidentiality of outsourced data from the cloud as well as from unauthorized users involves the data owner encrypting the data before it is outsourced [1, 61, 77]. This encryption guarantees the confidentiality of data, even when a cloud is compromised due to external threats such as hacking. A potentially curious, untrustworthy, or even malicious cloud operator can track down the user's queries and infer what the user is looking for. Thus, a user's query privacy may be compromised while searching through hosted data within the cloud. To help preserve query privacy, authorized users demand that their queries become encrypted before they are sent to the cloud for evaluation. Furthermore, the cloud can derive useful and sensitive information about the actual data items by observing the data access patterns during query processing, even if the data and the query are encrypted [27, 97]. The term "data access patterns" refers to the relationships between the encrypted data that can be observed by the cloud during the query processing, such as the outcome of the search (i.e., which records have been retrieved).

During query processing, the need is to keep not only the data private from the cloud but also the users' input queries. The question to ask is "how can the cloud execute queries over encrypted data without ever decrypting them or compromising the user's privacy?" Such a question has resulted in a specific research area known as *query processing over encrypted data.*

A trivial solution is to encrypt the data with symmetric key encryption schemes (e.g., Advanced Encryption Standard (AES)) and then outsource it to a cloud. At first, it seems to be a better approach. Data encryption by symmetric key encryption schemes, however, restricts a cloud's ability to perform fundamental data processing functionality (e.g., query processing) without disclosing the original data and user queries. Under this case, the only option for the user is to download the whole encrypted data from the cloud, decrypt the data, and then perform queries on the plaintext data locally. However, this is clearly impractical, especially for mobile users and large data.

Following the aforementioned discussions, it is clear that there is a strong need to develop protocols for query processing over encrypted data that can guarantee the following: (1) maintaining the confidentiality of encrypted data, (2) ensuring the privacy of a user's query, and (3) hiding data access patterns.

This work was conducted to address the challenge of performing queries on encrypted outsourced data in a cloud without compromising either the data's confidentiality or the user's queries. This work focuses on developing secure protocols over encrypted data under the semi-honest model for the following types of queries: the k-Nearest Neighbor (kNN) query, advanced analytical query, and correlated range query. The k-Nearest Neighbor query identifies the top k closest records to the query input record at the database. The advanced analytical query performs any data mining task (e.g., classification). Given a specific threshold t, the correlated range query aims to find all records in the dataset whose distances with the query lie either below or above t, depending on the application.

1.1. DEFINING THE PROBLEM

This work considers the following scenarios. There are three different distributed parties including the data owner (here referred to as Alice), the cloud, and the data authorized consumer/user (here referred to as Bob). Let T denote Alice's database with n records, denoted by $\langle t_1, \ldots, t_n \rangle$, and m attributes. Let $t_{i,j}$ denote the j^{th} attribute value of tuple t_i for $1 \leq i \leq n$ and $1 \leq j \leq m$. Assume that Alice encrypts her database (T) attribute-wise using her public key (pk) and outsources both the encrypted database and all future query processing services to a cloud. This work assumed that Alice's public/secret keys are generated using a semantically secure and an Additive Homomorphic Public Key Encryption (AH-Enc) scheme (such as the Paillier cryptosystem [75]). Semantic security means that ciphertexts should be computationally indistinguishable from the cloud's perspective. If the encryption scheme is semantically secure, then the ciphertexts are random numbers from the cloud's perspective. That ensures that the cloud cannot distinguish an encryption of one message from another [45]. More details are given in Section 3.3. By encrypting the data using an AH-Enc scheme, Alice makes it possible for the cloud to conduct certain mathematical operations directly over encrypted data.

Let T' denote Alice's encrypted database. Suppose Alice allows Bob to securely retrieve data from T' in the cloud. Suppose also that at some future time, Bob wants to execute a query $\big(q = \langle q_1, \ldots, q_m \rangle\big)$ on T' in the cloud in a privacy-preserving manner. During this process, neither Bob's query (q) nor the database's contents (T) should be revealed to the cloud. Access patterns to the data should also be protected from the cloud and other unauthorized users. This process is referred to as Privacy-Preserving Query Processing (PPQP) over encrypted data in the cloud. Let q_{out} denote the set of records that satisfies q. The PPQP protocol can now be formally defined as:

$$\text{PPQP}(T', q) \rightarrow q_{out}$$

At the end of the PPQP protocol, the output q_{out} should be revealed only to Bob (the authorized user who initiated the query).

1.2. OBJECTIVES AND CONTRIBUTIONS

This thesis addresses the problem of performing queries on encrypted data stored on a cloud. PPQP protocols are proposed to facilitate different types of queries, namely, the k-Nearest Neighbor query, the advanced analytical query, and the correlated range query that protect the confidentiality of the stored data, user queries, and data access patterns from cloud service providers and other unauthorized users. In the proposed protocols, once Alice outsources her encrypted data to the cloud, she stops participating in the query processing task. Therefore, no information is revealed to Alice. The protocols proposed in this thesis meet the following privacy requirements:

- Data confidentiality - During the query processing, neither the contents (T) nor any intermediate results are disclosed to the cloud.
- Query privacy - At any point in time, Bob's input query (q) should not be disclosed to the cloud or to Alice.
- Hiding data access patterns - Access patterns to the data (e.g., the records corresponding to q) should not be revealed to either Alice or the cloud to prevent any inference attacks. Access patterns related to any intermediate computations should also be hidden from the cloud, thereby preserving the semantic security of the encrypted data.
- Output security - At the end of the protocol, the output q_{out} should be revealed only to Bob; no information should be revealed to the cloud. Additionally, no information other than q_{out} should be revealed to Bob.

The intermediate results observed by the cloud in these protocols are either newly generated, randomized encryptions or random numbers. Thus, the data records that correspond to query q are unknown to the cloud.

The desirable privacy requirements can be relaxed to provide a practical, more efficient protocol. For example, a protocol can be allowed to leak a specified amount of information (e.g., data access patterns) to the cloud to improve efficiency.

Additionally, the proposed protocols achieve the following properties:

- Efficiency - These protocols incur a low computational overhead (negligible computation cost) on the end-user. After Bob sends his encrypted query to the cloud, he stops involving in any computations (less workload at Bob's local machine). Hence, data access patterns are further protected from Bob. He only performs a very small number of encryption/decryption operations (bounded by the number of attributes).
- Correctness - The output q_{out} should be computed accurately.

1.3. ORGANIZATION

The thesis is organized as follows. Chapter 2 presents an overview of some existing work that is closely related to the research being proposed. Chapter 3 provides basic information regarding SMC and the AH-Enc scheme. This chapter also introduces a set of privacy-preserving primitives, along with their implementations and security analysis, under the semi-honest model.

Chapter 4 considers the kNN query over relational data. In particular, this chapter focuses on solving the secure processing of kNN query over encrypted relational data. More specifically, this chapter presents two novel Secure k-Nearest Neighbor (SkNN) protocols [29]. Data confidentiality and query privacy of both protocols are completely protected. The first protocol (which acts as a basic solution) leaks a specified amount of information (data access patterns) to the cloud. The second protocol is more secure (provides a better security guarantee) as it hides the data access patterns. The second protocol, however, is more expensive than the first protocol.

Chapter 5 addresses the advanced analytical query over relational data. In particular, this chapter focuses on solving the classification problem over encrypted data. A novel Privacy-Preserving k-Nearest Neighbor (PPkNN) protocol over semantically secure encrypted data is developed [85]. This protocol protects not only the confidentiality of the original data but also the user query from the cloud. It also hides the data access patterns and the classification result. The proposed protocol's performance under different parameter settings is evaluated in this chapter.

The correlated range query over image data is discussed in Chapter 6. More specifically, Content-Based Image Retrieval (CBIR) is considered. A desirable image was retrieved from a large amount of image data based on the similarity of some common attributes, known as features. More specifically, this chapter proposes a secure Outsourceable and Privacy-Preserving Biometric Authentication (PPBA$_O$) protocol, which is a special case of CBIR [21]. At a high level, a hybrid approach is adopted to implement PPBA$_O$ in order to take advantage of both homomorphic encryption and garbled circuit-based approaches to achieve the best performance while simultaneously protecting both the confidentiality of the biometric data and the user's input query, in addition to hiding the data access patterns.

In this chapter, the protocol's security is analyzed and it's performance under different parameter settings is evaluated. The proposed PPBA$_O$ was also modified slightly to produce an Outsourceable and Privacy-Preserving Biometric Identification (PPBI$_O$) protocol. The PPBI$_O$ protocol completely protects the confidentiality of the stored biometric data, user queries, and data access patterns from cloud service providers and other unauthorized users.

Both the contributions of this work as well as the potential for future studies are discussed in Chapter 7.

2. RELATED WORK

This chapter reviews the existing work related to the Secure k-Nearest Neighbor [29] techniques, Privacy-Preserving Data Mining [85], and Privacy-Preser- ving Biometric Authentication/Identification [21] protocols.

2.1. SECURE k-NEAREST NEIGHBOR TECHNIQUES

Retrieving the k-Nearest Neighbors to a given query (q) is one of the most fundamental problems in many application domains such as similarity search, pattern recognition, and data mining. In the literature, many techniques have been proposed to address the SkNN problem, which can be classified into two categories based on whether the data are encrypted or not: *centralized* and *distributed.*

Centralized Methods: In the centralized methods, the data owner is assumed to outsource his/her database and DBMS functionalities (e.g., kNN query) to an untrusted external service provider, which manages the data on behalf of the data owner, where only the trusted users are allowed to query the hosted data. By outsourcing data to an untrusted server, many security issues arise such as data privacy (protecting the confidentiality of the data from both the server and the query issuer). To achieve data privacy, the data owner is required to use data anonymization models (e.g., k-anonymity) or cryptographic (e.g., encryption and data perturbation) techniques over his/her data before outsourcing them to the server.

Encryption is a traditional technique used to protect the confidentiality of sensitive data such as medical records. Due to data encryption, the process of query evaluation over encrypted data becomes challenging. Along this direction, various techniques have been proposed for processing range queries [49, 50, 91] and aggregation queries [46, 68] over encrypted data. This work, however, restricts the discussion to secure evaluation of the kNN query.

In the past few years, researchers have proposed different methods [51, 98, 99, 105] to address the SkNN problem. Wong et al. [98] proposed a new encryption scheme called

Asymmetric Scalar-Product-preserving Encryption (ASPE) that preserves the scalar product between the query vector (q) and any tuple vector (t_i) from database (T) for distance comparison, which is sufficient to find kNN. Both the data and query are encrypted using slightly different encryption schemes before outsourcing to the server, and all the query users know the decryption key. As an improvement, Zhu et al. [105] proposed a novel SkNN method in which the key of the data owner is not disclosed to the user. However, their architecture requires the participation of the data owner during query encryption. As an alternative, Hu et al. [51] proposed a method based on a provably secure privacy homomorphism encryption scheme from a provably secure additive and multiplicative privacy homomorphism [28] that supports modular addition, subtraction, and multiplication over encrypted data. They addressed the SkNN problem under the following setting: the client has the ciphertexts of all data points in database T and the encryption function of T, whereas the server has the decryption function of T and some auxiliary information regarding each data point. Both methods [51, 98], however, are not secure because they are vulnerable to chosen-plaintext attacks. All the above methods also leak data access patterns to the server.

Recently, Yao et al. [99] proposed a new SkNN method based on partition-based a Secure Voronoi Diagram (SVD). Instead of asking the cloud to retrieve the exact kNN, they required the cloud to retrieve a relevant encrypted partition $\big(E_{pk}(G)$ for $E_{pk}(T)\big)$ such that G is guaranteed to contain the k-nearest neighbors of q. This work, however, solves the SkNN problem accurately by letting the cloud retrieve the exact k-Nearest Neighbors of q (in encrypted form). Additionally, most of the computations during the query processing step in [51, 99, 105] are performed locally by the end-user. That conflicts with the purpose of outsourcing the DataBase Management System (DBMS) functionalities to the cloud. Furthermore, the protocol in secure nearest neighbor revisited [99] leaks data access patterns, such as the partition ID corresponding to a user query, to the cloud.

Data Distribution Methods: In the data distributed methods, data are assumed to be partitioned either vertically or horizontally and distributed among a set of independent, non-colluding parties. The data distributed methods rely on SMC techniques that enable multiple parties to securely evaluate a function using their respective private inputs without disclosing the input of one party to the others. Many efforts have been made to address the problem of the kNN query in a distributed environment. Shaneck et al. [89] proposed a privacy-preserving algorithm to perform the k-Nearest Neighbor search. The protocol in privacy preserving nearest neighbor search [89] is based on Secure Multiparty Computation for privately computing kNN points in a horizontally partitioned dataset. Qi et al. [79] proposed a single-step kNN search protocol that is provably secure with linear computation and communication complexities. Vaidya et al. [92] studied privacy-preserving top-k queries in which the data are vertically partitioned. Ghinita et al. [39] proposed a Private Information Retrieval (PIR) based framework for answering kNN queries in location-based services. Their solution, however, protects only the query privacy (i.e., it does not address data confidentiality and access pattern issues). Note that, in private queries in location based services [39], the data residing in the server are in plaintext format. However, if the data are encrypted to ensure data confidentiality, it is not clear how a user can obliviously retrieve the output records because he/she does not know the indices that match his/her input query. Nevertheless, even if a user can retrieve the records using PIR, the user still needs to perform local computations to identify the k-Nearest Neighbors. However, in this study, the user's computation is completely outsourced to a cloud.

In summary, the above data distribution methods are not applicable to perform kNN queries over encrypted data for two reasons: (1) This work deals with an encrypted form of the data and query, which is not the case in the above methods. (2) The data in this work are assumed to be encrypted and stored in the cloud, whereas, in the above methods, they are partitioned (in plaintext format) among different parties.

2.2. PRIVACY-PRESERVING DATA MINING

Privacy-Preserving Data Mining (PPDM) is defined as the process of extracting/deriving knowledge about data without compromising the privacy of the data [3, 64, 80]. In the past decade, a number of PPDM techniques have been proposed to facilitate users in performing data mining tasks in privacy-sensitive environments. Agrawal and Srikant [3], as well as Lindell and Pinkas [63], were the first to introduce the notion of privacy-preserving under data mining applications. Existing PPDM techniques can be classified into two broad categories: *data perturbation* and *data distribution.*

Data Perturbation Methods: With these methods, values of individual data records are perturbed by adding random noise in such a way that the distribution of the perturbed data look very different from that of the actual data. After such a transformation, the perturbed data is sent to the Miner to perform the desired data mining tasks. Agrawal and Srikant [3] proposed the first data perturbation technique that could be used to build a decision-tree classifier. A number of randomization-based methods were later proposed [6, 33, 34, 73, 104]. Data perturbation techniques are not, however, applicable to semantically-secure encrypted data. They also fail to produce accurate data mining results due to the addition of statistical noises to the data.

Data Distribution Methods: These methods assume that the dataset is partitioned either horizontally or vertically and distributed across different parties. The parties can later collaborate to securely mine the combined data and learn the global data mining results. During this process, data owned by individual parties is not revealed to other parties. This approach was first introduced by Lindell and Pinkas [63] who proposed a decision tree classifier under a two-party setting. A number of studies have since used SMC techniques [2, 22, 51, 57, 100].

Classification is one important task in many applications of data mining, including health-care and business. Recently, performing data mining in the cloud attracted significant attention. In cloud computing, the data owner outsources his/her data to the cloud. However, from the user's perspective, privacy becomes an important issue when sensitive data needs to be outsourced to the cloud. The direct way to guard the outsourced data is to apply encryption on the data before outsourcing.

Existing privacy-preserving classification techniques are not sufficient and applicable in this work for the following reasons: (*i*) In the existing methods, data are partitioned (in plaintext format) among different parties, whereas in this work, they are assumed to be encrypted and stored in the cloud. (*ii*) They fail to produce accurate data mining results because some amount of information is lost due to the addition of statistical noises (in order to hide the sensitive attributes). (*iii*) Data access patterns can be leaked. The cloud can easily derive useful and sensitive information about users' data items by simply observing the data access patterns.

2.3. PRIVACY-PRESERVING BIOMETRIC AUTHENTICATION/ IDENTIFICATION

Biometric authentication or identification is a special case of Content-Based Image Retrieval (CBIR). In a CBIR system, a desirable image is retrieved from a large image database based on the similarity of some common attributes, which are called features. There are two kinds of features: global and local. Global features usually include color, texture, and shape, which are well-known and studied in image retrieval. Local features are usually used in a localized circumstance, which needs to recognize objects more precisely.

The formulation of Privacy-Preserving Biometric Identification (PPBI) is slightly different from Privacy-Preserving Biometric Authentication (PPBA). The PPBI protocol generally returns the profile of any person whose biometric data record (stored on the server) matches the user's input biometric data record. In contrast, the PPBA protocol only returns a single bit to indicate if there is a match or not. Existing PPBI protocols can be easily modified to satisfy the PPBA problem statement definition. From a technical perspective, the difference between PPBI and PPBA is negligible. Since there is a lack of existing work directly related to PPBA, works mainly related to PPBI are presented. In both PPBA and PPBI, the involved biometric data is never disclosed to the participating parties, except for their own data.

Erkin et al. [30] proposed the first privacy-preserving biometric face recognition protocol based on the standard Eigenfaces recognition algorithm. The protocol is a secure two-party computation protocol where one party (client) wants to learn whether its candidate biometric reading matches for one or more records in the other party's (server's) database without disclosing any information except the final result. The protocol computes Euclidean distances between face image feature vectors from the client and server's face image database, and it returns the profile information associated with the facial image data record that has the smallest distance to the client's input biometric data. In the end, the client only knows the profile information without knowing other biometric data and their associated profiles stored at the server, and the server knows nothing, even regarding the profile information returned to the client. The data exchanged during an execution of the protocol are encrypted by an AH-Enc scheme, but the data stored at the sever are not encrypted. Therefore, the server knows the contents of its biometric database.

Sadeghi et al. [83] developed a hybrid privacy-preserving face recognition protocol that improved the efficiency of Erkin's work. This protocol follows the problem setting of Erkin's work and also uses an AH-Enc (the Paillier cryptosystem) to securely compute Euclidean distances. On the other hand, garbled circuits [53, 102] are used to find out the minimum distance. The authors also proposed a method of packing multiple values together into a single ciphertext before blinding. This can save significant communication costs.

Huang et al. [32] developed an efficient Privacy-Preserving Biometric Identification protocol for fingerprint recognition. The basic algorithm for fingerprint identification is based on *FingerCode* [55]. In the main system, the server and a client jointly perform the protocol to retrieve the identity record from the server's database whose fingerprint best matches the client's input fingerprint reading. The protocol provides the same security guarantee as the previously mentioned protocols. In addition, similar to Sadeghi's work, this protocol also combines AH-Enc with garbled circuits. However, the protocol improves efficiency for both the distance-computing phase and matching phase compared to Sadeghi's work. The protocol separates the retrieval step from the matching phase, and it uses the by-product of evaluating the garbled circuit in the matching phase to perform the oblivious retrieval efficiently. In the retrieval phase, the protocol utilizes a backtracking tree to obliviously and efficiently recover the profile corresponding to the closest matching vector.

Blanton and Gasti [11] developed a Privacy-Preserving Biometric Identification protocol for iris codes based on Hamming distance. In their protocol, the client who possesses an iris reading would also like to learn whether his/her reading matches one or more records in an iris database managed by the server. The contents of the server's database are never disclosed to the client, but the client learns the comparison results between his/her input iris record and every record in the server's database. The implementation of the protocol also uses both AH-Enc and garbled circuits, and it reduces the complexity of the circuits used for comparison based on an optimization that permits XOR gates to be evaluated for free. The proposed techniques can be applied in protocols where *FingerCode* and Euclidean distances are adopted to improve the performance compared to the existing Euclidean distance-based solutions.

SCiFI [74] is a realization of a privacy-preserving face identification system that uses a component-based face identification technique that builds a binary index into a vocabulary representation. The adopted image representation technique is robust against different viewing conditions, such as illumination, occlusions, and changes in appearance (like wearing glasses). The protocol utilizes Hamming distance to measure image similarity, and its implementation was based on AH-Enc and oblivious transfer [69].

Although the aforementioned privacy-preserving biometric identification/authentication protocols protect the confidentiality of both the server's and client's biometric data, in this study, the problem setting is quite different from these protocols. This study deals with the encrypted form of the biometric database that is outsourced to a server (or a cloud), and the server does not have the key to decrypt the data. Thus, the server does not know anything regarding the original biometric database. Since the previously proposed solutions require the server to perform computations on the original biometric data, these solutions cannot be applied to solve this outsourced biometric authentication problem.

Recently, Blanton and Aliasgari [10] developed a secure approach to outsourcing the computations of matching iris biometric data records. The setting of their work is very similar to this study. Two protocols were proposed for either a single-server setting or a multiple-server setting. In their single-server setting, the protocol uses a predicated encryption scheme [59, 90] that allows the server to perform non-interactive computations. The predicate encryption scheme is not as secure as AH-Enc. As a result, the protocol proposed in this study offers much better protection of the biometric data's confidentiality. Under the multiple-server setting, the protocol adopts a secret sharing scheme (e.g., Shamir [88]) to "encrypt" the outsourced biometric database. The protocol, however, requires at least three independent servers to perform the intermediate computations, whereas this study only requires two; thus, it is more practical. Additionally, the protocol mainly focuses on calculating Hamming distance, whereas this work allows Euclidean and Hamming distances to be calculated. Since both distances are commonly used to retrieve images according to different kinds of biometric data, this work offers more generality. Another difference is that, at the end of protocol execution, the server can know which encrypted biometric data record matches the client's input. This access pattern leakage can violate the security guarantee of the underlying encryption scheme [54]. Because this work does not disclose this information to any participating party, it offers the same security protection as the encryption scheme used to encrypt the outsourced biometric data.

3. SECURITY DEFINITIONS AND BASIC SECURITY PRIMITIVES

This chapter presents basic information about Secure Multiparty Computation along with the security threat or adversary models that best match in this study. Then, it summarizes the homomorphic properties of the encryption scheme used in this study as a background. Finally, it introduces a set of sub-protocols [21, 29, 85] that were used as basic primitives when constructing the proposed PPQP protocols, along with their possible implementations, security, and complexity analysis. For ease of presentation, some common notations that are used extensively throughout this chapter are summarized in Table 3.1.

3.1. SECURE MULTIPARTY COMPUTATION

Secure Multiparty Computation (SMC) was first introduced by Yao's Millionaire Problem [101, 102], where Alice and Bob want to know who is richer without disclosing their actual wealth to each other. Suppose there are n parties ($P_1, \ldots, P_n$) who hold private inputs ($a_1, \ldots, a_n$). An SMC protocol allows $P_1, \ldots, P_n$ to collaboratively compute a function f on inputs $a_1, \ldots, a_n$ without disclosing a_i to P_j, where $1 \leq i, j \leq n$ and $i \neq j$. To achieve that, the participating parties have to exchange messages and perform some local computations until all the parties get the desired output. More formally, SMC allows the evaluation of the function $f(a_1, \ldots, a_n) = (r_1, \ldots, r_n)$ such that the output (r_i) is known only to party P_i and the privacy of each party's input (a_i) is preserved.

The first general and provably secure solution for a two-party case was developed by Yao, and it demonstrated that any function that can be described by a polynomial size boolean circuit of logarithm depth can be solved securely [101, 102]. This work was extended to multiparty computations by Goldreich et al. [42]. It was proved in [42] that any computation that can be done in polynomial time by a single party can also be done securely by multiple parties. Since then, much work has been published for the multiparty case [7, 8, 18, 19, 23, 40, 58, 62].

Table 3.1: Common notations

P_1, P_2	Two non-colluding semi-honest parties
$\langle E_{pk}, D_{sk} \rangle$	A pair of Paillier's encryption and decryption functions with (pk, sk) as public-secret key pair
X , Y	m-dimensional vectors, and the content of the individual dimension is represented by x_j (y_j), where $1 \leq j \leq m$
$E_{pk}(X)$, $E_{pk}(Y)$	Attribute-wise encryption of X, Y in which $E_{pk}(X) = \big\langle E_{pk}(x_1), \ldots, E_{pk}(x_m) \big\rangle$ and $E_{pk}(X) = \big\langle E_{pk}(y_1), \ldots, E_{pk}(y_m) \big\rangle$

3.2. THREAT MODEL

In general, there is a conceptual difference between privacy and security. This work, however, will not differentiate the two terms. Regarding a distributed protocol, security is generally related to the amount of information leaked during the protocol execution. The goal is to ensure no information, other than what they can deduce from their own outputs, is leaked to the involved parties. There are many ways to define information disclosure. To maximize privacy, or minimize information disclosure, security definitions that appear in the literature of SMC were adopted for this study.

SMC-based secure protocols generally assume three basic adversarial models: semi-honest (also referred to as honest but curious), covert, and malicious. An adversarial model generally specifies what an adversary or attacker is allowed to do during an execution of a secure protocol. In the semi-honest model, an attacker (i.e., one of the participating parties) is expected to follow the prescribed steps of a protocol. The attacker, however, can compute any additional information based on his/her private input, output, and messages received during an execution of the secure protocol. As a result, whatever can be inferred from the private input and output of an attacker is not considered as a privacy violation. An adversary in the semi-honest model can be treated as a passive attacker, whereas an adversary in the malicious model can be treated as an active attacker who can arbitrarily diverge from the normal execution of a protocol. In contrast, the covert adversary model [5] lies between the semi-honest and malicious models. More specifically, an adversary under the covert model may deviate arbitrarily from the rules of a protocol in the case of cheating. The honest party is guaranteed to detect this cheating with good probability. In this study, to develop secure, efficient protocols, all the participating parties were assumed to be semi-honest.

Detailed security definitions and models can be found in other studies [40, 41]. The following definition briefly captures the previous discussion regarding a secure protocol under a semi-honest model:

Definition 1. *Let a_i be the input of party P_i, $\Pi_i(\pi)$ be the party P_i's execution image of the protocol π, and r_i be the result computed from π for the party P_i. Then, π is secure if $\Pi_i(\pi)$ can be simulated from (a_i, r_i) such that the distribution of a simulated image is computationally indistinguishable from $\Pi_i(\pi)$.*

A formal way to prove the security of a protocol under the semi-honest model is to use the simulation approach [43]. The execution image in Definition 1 typically includes the input, the output, and the messages communicated during the execution of a protocol. To prove a protocol is secure under the semi-honest model, it is required to show that the execution image of a protocol does not leak any information regarding the private inputs of participating parties [40].

Definition 2. *Composition Theorem [41]: If a protocol consists of sub-protocols, the protocol is secure as long as the sub-protocols are secure and all the intermediate results are random or pseudo-random.*

In this work, the proposed PPQP protocols are constructed based on a sequential composition of sub-protocols. Thus, the security of each sub-protocol needed to be proved before the PPQP's security could be proved. In other words, to formally prove the security of the proposed PPQP protocols under the semi-honest model, according to the composition theorem given in Definition 2, one needed to show that the simulated image of each sub-protocol was computationally indistinguishable from the actual execution image and it produced random shares or pseudo-random as intermediate results.

3.2.1. Justification of Use of Semi-Honest Model. By semi-honest model, this work implicitly assumes that the cloud service providers (or other participating users) utilized in the protocols proposed in this study do not collude. This model may not be appropriate for situations in which background knowledge on the parties is missing (e.g., P_1 and P_2). There are two main reasons to adopt the semi-honest adversary model in this study. First, as mentioned in "Faster secure two-party computation using garbled circuits" [53], developing protocols under the semi-honest setting is an important first step towards constructing protocols with stronger security guarantees. Almost all practical SMC-based protocols (e.g., [48, 52, 53, 72]) are secure under the semi-honest model. Using zero-knowledge proofs [43], these protocols can be transformed into secure protocols under the malicious model. Second, both P_1 and P_2 were assumed to be two cloud service providers. Today, cloud service providers in the market are legitimate, well-known companies (e.g., Amazon, Google, and Microsoft). These companies maintain reputations that are invaluable assets that need to be protected at all costs. Thus, a collusion between them is highly unlikely as it will damage their reputation, which, in turn, affects their revenues. Therefore, P_1 and P_2 can safely be assumed to be semi-honest. As a consequence, in this study, it is very realistic to assume that the participating parties are semi-honest.

3.3. ADDITIVE HOMOMORPHIC ENCRYPTION

Homomorphic encryption is a special type of encryption that allows operating on ciphertexts without decrypting them [67]. This work adopted an Additive Homomorphic Public Key Encryption (AH-Enc) scheme as the building block. Let E_{pk} and D_{sk} be the encryption and decryption functions, respectively, in an AH-Enc scheme with public key pk and private key sk. Without sk, no one can discover x from $E_{pk}(x)$ in polynomial time. For any given two plaintexts $x, y \in \mathbb{Z}_N$, an AH-Enc scheme exhibits the following properties:

a. Homomorphic Addition - The encryption function is additive homomorphic. The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts.

$$D_{sk}\big(E_{pk}(x+y)\big) = D_{sk}\big(E_{pk}(x) * E_{pk}(y) \bmod N^2\big)$$

b. Homomorphic Multiplication - An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts. Given a constant c and ciphertext $E_{pk}(x)$:

$$D_{sk}\big(E_{pk}(c * x)\big) = D_{sk}\big(E_{pk}(x)^c \bmod N^2\big)$$

c. Probablistic - Let $c_1 = E_{pk}(x)$ and c_2 =$E_{pk}(y)$,

$$Probability\ for\ c_1 \neq c_2\ is\ very\ high\ even\ if\ x = y$$

d. Semantic Security - The encryption scheme is semantically secure, as defined in “The Foundations of Cryptography” and “The knowledge complexity of interactive proof systems” [41, 44]. Briefly, given a set of ciphertexts, an encryption scheme is semantically secure if an adversary cannot deduce/learn any information about the plaintext(s) with polynomial-bounded computing power. In other words, the encryption scheme is secure against a chosen-plaintext attack.

Any Additive Homomorphic Public Key Encryption (AH-Enc) system is applicable, but the Paillier encryption scheme [75] was adopted in this study for its implementation simplicity. Informally speaking, the public key in the system is (N,g), where N is the result of multiplying two large prime numbers of similar bit length and $g \in \mathbb{Z}^*_{N^2}$ is a randomly chosen generator. For succinctness, the mod N^2 term that occurs during homomorphic operations has been dropped from the remainder of this work. Note that, for any given $x \in \mathbb{Z}_{\mathbb{N}}$, “$N - x$” is equivalent to “$-x$” under $\mathbb{Z}_N$. Hereafter, the notation $r \in_R \mathbb{Z}_N$ is used to denote r as a random number in $\mathbb{Z}_N$.

Example 1. *Let E_{pk} be the encryption function with public key pk for any given two plaintexts $x, y \in \mathbb{Z}_N$, and $r \in_R Z$. According to the additive homomorphic property of the encryption scheme, $E_{pk}(x - y + r)$ can be computed as follows:*

$$E_{pk}(x - y + r) \leftarrow E_{pk}(x) * E_{pk}(y)^{N-1} * E_{pk}(r) \qquad \square$$

3.4. DISTANCE COMPUTATION

This work adopted the two most common distance metrics to implement the proposed PPQP protocols: Euclidean distance and Hamming distance. Suppose X and Y are m-dimensional vectors and the content of dimension j is represented by x_j (y_j), where $1 \leq j \leq m$.

3.4.1. Euclidean Distance. The Euclidean distance between X and Y can be computed as follows:

$$\text{Euclidean\_Dist}(X, Y) = \sqrt{\sum_{i=1}^{m}(x_j - y_j)^2} \tag{3.1}$$

When comparing two distances, the square root does not make any difference. Thus, in this work, the square root was dropped to compute the square of the Euclidean distance between X and Y:

$$\text{Euclidean\_Dist}^2(X, Y) = |X - Y|^2 = \sum_{i=1}^{m}(x_j - y_j)^2 \tag{3.2}$$

where $|X - Y|$ denotes the Euclidean distance between vectors X and Y.

3.4.2. Hamming Distance. When X and Y are binary vectors, Hamming distance can be used to determine how close X and Y are. At a high level, the Hamming distance can be computed as follows:

$$\text{Hamming\_Dist}(X, Y) = m - \sum_{j=1}^{m} x_j * y_j \tag{3.3}$$

3.5. BASIC SECURITY PRIMITIVES

A set of generic sub-protocols that are used as basic primitives when constructing the proposed Privacy-Preserving Query Processing (PPQP) protocols are presented here [21, 29, 85]. All of the following protocols are considered under a two-party semi-honest setting. In particular, assume the existence of two semi-honest parties (P_1 and P_2) such that Paillier's secret key (sk) is known only to P_2; the pk is public. Also, let X and

Y be m-dimensional vectors in which $E_{pk}(X) = \big\langle E_{pk}(x_1), \dots, E_{pk}(x_m)\big\rangle$ and $E_{pk}(Y) = \big\langle E_{pk}(y_1), \dots, E_{pk}(y_m)\big\rangle$.

The term random shares means that the actual value is divided into two additive random shares, each of which is independently and uniformly distributed in a group $\big($e.g., $\mathbb{Z}_N = \{0, 1, \dots, N-1\}\big)$. The actual value can be reconstructed by adding the two shares' modulo N.

As mentioned in Section 3.2, to formally prove that a protocol is secure under the semi-honest model, one must show that the simulated execution image of that protocol is computationally indistinguishable from its actual execution image [41]. In other words, an execution image of that protocol does not leak any information regarding the private inputs of participating parties. An execution image generally includes the messages exchanged and the information computed from these messages.

Next, each of these protocols are discussed in detail along with their complexity and security analysis. This work either proposed a new solution or referred to the most efficient known implementation to each of these protocols.

3.5.1. Secure Multiplication. Consider a party (P_1) with input $\big(E_{pk}(a), E_{pk}(b)\big)$ and a party (P_2) with the secret key (sk). The goal of the Secure Multiplication (SM) protocol is to return the encryption of $a*b$ (i.e., $E_{pk}(a*b)$ as output to P_1). No information regarding either a or b is revealed either to P_1 or P_2 during this protocol. The basic idea of the SM protocol is based on the following property, which holds for any given $a, b \in \mathbb{Z}_N$:

$$a * b = (a + r_a) * (b + r_b) - a * r_b - b * r_a - r_a * r_b \tag{3.4}$$

where all the arithmetic operations are performed under $\mathbb{Z}_N$. The overall steps in SM are shown in Algorithm 1. Briefly, P_1 initially randomizes both a and b by computing $a' = E_{pk}(a) * E_{pk}(r_a)$ and $b' = E_{pk}(b) * E_{pk}(r_b)$ and sends them to P_2. Here, r_a and r_b are random numbers in $\mathbb{Z}_N$ and known only to P_1. Upon receiving, P_2 decrypts and multiplies them to get $h = (a + r_a) * (b + r_b) \bmod N$. Then, P_2 encrypts h and sends it to P_1. After this, P_1 removes extra random factors from $h' = E_{pk}\big((a + r_a) * (b + r_b)\big)$ based on Equation 3.4 to get $E_{pk}(a * b)$.

Algorithm 1 SM$(E_{pk}(a), E_{pk}(b)) \rightarrow E_{pk}(a * b)$

```
Require: $P_1$ has $E_{pk}(a)$ and $E_{pk}(b)$; $P_2$ has $sk$
1: $P_1$:
    (a). Pick two random numbers $r_a, r_b \in \mathbb{Z}_N$
    (b). $a' \leftarrow E_{pk}(a) * E_{pk}(r_a)$
    (c). $b' \leftarrow E_{pk}(b) * E_{pk}(r_b)$
    (d). Send $a', b'$ to $P_2$
2: $P_2$:
    (a). Receive $a'$ and $b'$ from $P_1$
    (b). $h_a \leftarrow D_{sk}(a')$;  $h_b \leftarrow D_{sk}(b')$
    (c). $h \leftarrow h_a * h_b \bmod N$
    (d). $h' \leftarrow E_{pk}(h)$
    (e). Send $h'$ to $P_1$
3: $P_1$:
    (a). Receive $h'$ from $P_2$
    (b). $s \leftarrow h' * E_{pk}(a)^{N-r_b}$
    (c). $s' \leftarrow s * E_{pk}(b)^{N-r_a}$
    (d). $E_{pk}(a * b) \leftarrow s' * E_{pk}(N - r_a * r_b)$
```

Example 2. *Assume that* $a = 59$ *and* $b = 58$. *For simplicity, let* $r_a = 1$ *and* $r_b = 3$. *Initially,* P_1 *computes* $a' = E_{pk}(60) = E_{pk}(a) * E_{pk}(r_a)$ *and* $b' = E_{pk}(61) = E_{pk}(b) * E_{pk}(r_b)$ *and sends them to* P_2. *Then,* P_2 *decrypts and multiplies them to get* $h = 3660$. *After this,* P_2 *encrypts* h *to get* $h' = E_{pk}(3660)$ *and sends it to* P_1. *Upon receiving* h', P_1 *computes* $s = E_{pk}(3483) = E_{pk}(3660 - a * r_b)$ *and* $s' = E_{pk}(3425) = E_{pk}(3483 - b * r_a)$. *Finally,* P_1 *computes* $E_{pk}(a * b) = E_{pk}(3422) = E_{pk}(3425 - r_a * r_b)$. □

Security Analysis: According to Algorithm 1, the execution image of P_2 is denoted by $\Pi_{P_2}(\text{SM})$, where

$$\Pi_{P_2}(\text{SM}) = \left\{\langle a', h_a\rangle, \langle b', h_b\rangle\right\}$$

Observe that $h_a = a + r_a \bmod N$ and $h_b = b + r_b \bmod N$ are derived upon decrypting a' and b', respectively. Note that both h_a and h_b are random numbers in $\mathbb{Z}_N$. Suppose that the simulated image of P_2 is $\Pi^S_{P_2}(\mathrm{SM})$, where

$$\Pi^S_{P_2}(\mathrm{SM}) = \left\{\langle a^*, r'_a\rangle, \langle b^*, r'_b\rangle\right\}$$

Here, a^* and b'_* are randomly generated from $\mathbb{Z}_{N^2}$, whereas r'_a and r'_b are randomly generated from $\mathbb{Z}_N$. Because E_{pk} is a semantically secure encryption scheme with a resulting ciphertext size that is less than N^2, a' and b' are computationally indistinguishable from a^* and b^*, respectively. Similarly, as r_a and r_b are randomly chosen from $\mathbb{Z}_N$, h_a and h_b are computationally indistinguishable from r'_a and r'_b, respectively. As a result, $\Pi_{P_2}(\mathrm{SM})$ is computationally indistinguishable from $\Pi^S_{P_2}(\mathrm{SM})$.

Similarly, assume that the execution image of P_1 is denoted by $\Pi_{P_1}(\mathrm{SM})$, where

$$\Pi_{P_1}(\mathrm{SM}) = \{h'\}$$

Here, h' is an encrypted value. Let the simulated image of P_1 be denoted by $\Pi^S_{P_1}(\mathrm{SM})$, where

$$\Pi^S_{P_1}(\mathrm{SM}) = \{h^*\}$$

where h^* is randomly chosen from $\mathbb{Z}_{N^2}$. Since E_{pk} is a semantically secure encryption scheme with a resulting ciphertext size of less than N^2, h' is computationally indistinguishable from h^*. As a result, $\Pi_{P_1}(\mathrm{SM})$ is computationally indistinguishable from $\Pi^S_{P_1}(\mathrm{SM})$. Putting the above results together, and following from Definition 1, it was concluded that the SM was secure under the semi-honest model.

Complexity Analysis: Each step of the SM protocol needed to be examined before the upper bound could be derived. The SM protocol performed 4 encryptions and 2 multiplications at Step 1. Step 2(b) involved performing 2 decryptions. A decryption has a cost that is similar to an encryption (encryption and decryption times are almost the same under Paillier's scheme). Thus, 2 encryptions were counted for Step 2(a). Step 2(c,d) involved performing 1 multiplication and 1 encryption. Step 3(b-d) involved performing 2 encryptions and 3 multiplications. Although it was slightly less expensive than an encryption operation, the exponentiation operation was treated as one encryption operation. Thus, the two exponentiations $\big(E_{pk}(a)^{N-r_b}$ and $E_{pk}(b)^{N-r_a}\big)$ were counted as two encryptions. Therefore, the total number of multiplications and encryptions for the Secure Multiplication protocol was 9 encryptions and 6 multiplications. Thus, the computation cost of Secure Multiplication was bounded by a (small) constant number of encryptions and multiplications.

3.5.2. Secure Squared Euclidean Distance. Suppose P_1 holds two encrypted vectors $\big(E_{pk}(X), E_{pk}(Y)\big)$ and P_2 holds the secret key (sk). The goal of the Secure Squared Euclidean Distance (SSED) protocol is to securely compute the encryption of the squared Euclidean distance between vectors X and Y. During this protocol, no information regarding X and Y is revealed to either P_1 or P_2. The output of this protocol returns $E_{pk}\big(|X-Y|^2\big)$, which is known only to P_1. According to Equation 3.2, the main challenge is to compute x_j^2, y_j^2, and $x_j * y_j$ from $E_{pk}(x_j)$ and $E_{pk}(y_j)$, where $E_{pk}(x_j)$ and $E_{pk}(y_j)$ are encryptions of x_j and y_j.

The main steps involved in the SSED protocol are shown in Algorithm 2. Briefly, for $1 \le j \le m$, P_1 initially computes $E_{pk}(x_j - y_j)$ by using the homomorphic properties. Then, P_1 and P_2 jointly compute $E_{pk}\big((x_j-y_j)^2\big)$, using the SM protocol, for $1 \le j \le m$. Note that the outputs of SM are known only to P_1. Finally, by applying homomorphic properties on $E_{pk}\big((x_j-y_j)^2\big)$, P_1 can compute $E_{pk}\big(|X-Y|^2\big)$ locally based on Equation 3.2.

Example 3. *Suppose that P_1 holds the encrypted data records of X and Y, given by* $E_{pk}(X) = \big\langle E_{pk}(63), E_{pk}(1),\ E_{pk}(1), E_{pk}(145), E_{pk}(233), E_{pk}(1), E_{pk}(3), E_{pk}(0), E_{pk}(6),\ E_{pk}(0)\ \big\rangle$ *and* $E_{pk}(Y) = \ \big\langle E_{pk}(56), E_{pk}(1), E_{pk}(3), E_{pk}(130),\ E_{pk}(256), E_{pk}(1), E_{pk}(2),$ $E_{pk}(1), E_{pk}(6), E_{pk}(2)\big\rangle$. *During the* SSED *protocol, P_1 initially computes* $E_{pk}(x_1-y_1) = E_{pk}(7), \ldots, E_{pk}(x_{10}-y_{10}) = E_{pk}(-2)$. *Then, P_1 and P_2 jointly compute* $E_{pk}\big((x_1-y_1)^2\big) = E_{pk}(49) = \mathrm{SM}\big(E_{pk}(7), E_{pk}(7)\big), \ldots, E_{pk}\big((x_{10}-y_{10})^2\big) = \mathrm{SM}\big(E_{pk}(-2)\ , E_{pk}(-2)\big) = E_{pk}(4)$. *$P_1$ locally computes* $E_{pk}\big(|X-Y|^2\big) = E_{pk}\big(\sum_{i=1}^{10}(x_i-y_i)^2\big) = E_{pk}(813)$. □

Security Analysis: The security of the SSED protocol directly follows from the SM protocol, which is used as the fundamental building block in the SSED protocol. This is because, apart from the SM protocol, the rest of the steps in the SSED protocol are non-interactive. More specifically, as shown in Algorithm 2, P_1 and P_2 jointly compute $E_{pk}\big((x_i-y_i)^2\big)$

Algorithm 2 SSED$\big(E_{pk}(X), E_{pk}(Y)\big) \rightarrow E_{pk}\big(|X-Y|^2\big)$

```
Require: $P_1$ has $E_{pk}(X)$ and $E_{pk}(Y)$; $P_2$ has $sk$
1: $P_1$, for $1 \leq j \leq m$ do:
     (a). $E_{pk}(x_j - y_j) \leftarrow E_{pk}(x_j) * E_{pk}(y_j)^{N-1}$
2: $P_1$ and $P_2$, for $1 \leq j \leq m$ do:
     (a). Compute $E_{pk}\big((x_j - y_j)^2\big)$ using the SM protocol
3: $P_1$:
     (a). $E_{pk}\big(|X-Y|^2\big) \leftarrow \prod_{j=1}^{m} E_{pk}\big((x_j - y_j)^2\big)$
```

using the SM protocol for $1 \leq i \leq m$. P_1 then performs homomorphic operations on $E_{pk}\big((x_i - y_i)^2\big)$ locally (i.e., with no interaction between P_1 and P_2).

Complexity Analysis: Each step of the protocol needed to be examined before the upper bound could be derived. The protocol performed m encryptions and m multiplications at Step 1. Although it was slightly less expensive than an encryption operation, the $E_{pk}(y_j)^{N-1}$ was treated as one encryption. Since the SSED protocol utilized the SM protocol as a sub-routine, Step 2 consisted of $7m$ multiplications and $10m$ encryptions (a detailed complexity analysis of SM is discussed in Section 3.5.1). Step 3 required $m-1$ multiplications. Therefore, the total number of multiplications and encryptions for the SSED protocol was bounded by $O(m)$. Additionally, because an encryption operation is generally several orders of magnitude more expensive than a multiplication, this work can claim that the computation complexity of the SSED protocol was bounded by $O(m)$ encryptions, where m is the vector size.

3.5.3. Secure Squared Euclidean Distance-Random Share. Suppose that P_1 holds two encrypted vectors $\big(E_{pk}(X), E_{pk}(Y)\big)$ and P_2 holds the secret key sk. The goal of the Secure Squared Euclidean Distance-Random Share (SSED$_R$) protocol is to securely compute the encryption of squared Euclidean distance between vectors X and Y, denoted by $E_{pk}\big(|X-Y|^2\big)$. During the computation, both X and Y are never decrypted to preserve the confidentiality of X and Y. Let d_e denote the squared Euclidean distance between X and Y. The output of this protocol returns two random shares (d_e' and d_e''), one for each

Algorithm 3 SSED$_R\big(E_{pk}(X), E_{pk}(Y)\big) \rightarrow (d'_e, d''_e)$

```
Require: $P_1$ has $E_{pk}(X)$ and $E_{pk}(Y)$; $P_2$ has the decryption key
1: $P_1$:
    (a) $E_{pk}(t_j) \leftarrow E_{pk}(x_j) - E_{pk}(y_j) + E_{pk}(r_j)$, where $r_j \in_R \mathbb{Z}_N$ and $1 \leq j \leq m$
    (b) Send $E_{pk}(t_1), \ldots, E_{pk}(t_m)$ to $P_2$
2: $P_2$:
    (a) Perform decryption to get $t_1, \ldots, t_m$
    (b) Compute $E_{pk}(t_1^2), \ldots, E_{pk}(t_m^2)$, and send them to $P_1$
3: $P_1$:
    (a) Obtain $E_{pk}\big((x_j - y_j)^2\big)$ by eliminating $r_j^2$ and $2r_j(x_j - y_j)$ from $E_{pk}(t_j^2)$, for $1 \leq j \leq m$
    (b) $E_{pk}(d_e) \leftarrow E_{pk}\big(\sum_{j=1}^{m} (x_j - y_j)^2\big)$
    (c) Compute $E_{pk}(d_e + r)$, where $r \in_R \mathbb{Z}_N$
    (d) Set $d'_e = N - r$ and send $E_{pk}(d_e + r)$ to $P_2$
4: $P_2$:
    (a) Perform decryption on $E_{pk}(d_e + r)$ to obtain $d_e + r$
    (b) Set $d''_e = d_e + r$
```

party, such that $d'_e + d''_e \mod N = d_e$. Note that the random share d'_e is known only to P_1, whereas the random share d''_e is known only to P_2. The key steps to securely compute the Euclidean distance between the two vectors, X and Y, based on Equation 3.2 are given by Algorithm 3, where the computations are only based on either encrypted or randomized data. At Step 1(a) of the algorithm, t_j is a randomized difference between x_j and y_j, and r_j is randomly chosen to hide the actual difference. According to the additive homomorphic property of the encryption scheme, $E_{pk}(t_j)$ can be computed as follows:

$$E_{pk}(x_j - y_j + r_j) \leftarrow E_{pk}(x_j) * E_{pk}(y_j)^{N-1} * E_{pk}(r_j)$$

P_2 can decrypt $E_{pk}(t_j)$ to obtain t_j because it has the decryption key. P_2 then computes the square of t_j and sends the encrypted results (e.g., $E_{pk}(t_1), \ldots, E_{pk}(t_m)$) to P_1.

Because P_1 knows r_j and $E_{pk}(x_j - y_j)$, it can easily compute $E_{pk}(r_j^2)$ and $E_{pk}\big(2r_j(x_j - y_j)\big)$. Since $E_{pk}(t_j^2) = E_{pk}\big((x_j - y_j)^2 + 2r_j(x_j - y_j) + r_j^2\big)$, P_1 can obtain $E_{pk}\big((x_j - y_j)^2\big)$ by:

$$E_{pk}\big((x_j - y_j)^2\big) \leftarrow E_{pk}(t_j^2) * \big(E_{pk}(x_j - y_j)^{2r_j} * E_{pk}(r_j^2)\big)^{N-1}$$

From here, $E_{pk}(d_e)$ can be directly derived:

$$E_{pk}(d_e) \leftarrow E_{pk}\big((x_1 - y_1)^2\big) * \cdots * E_{pk}\big((x_m - y_m)^2\big)$$

The rest of the steps are straightforward. All of the above computations are based on the additive homomorphic property of the encryption schemes.

Security Analysis: The security of the SSED$_R$ protocol was dependent on the encryption key's size. In general, N needed to be at least 1024 bits long. One can then say that the Paillier encryption scheme possessed semantic security. Thus, N in this proof was assumed to be either 1024-bit or larger. The security guarantee in the semi-honest model was quite easy to prove because the computations were performed on either encrypted data or randomized data. The following steps were used to ensure the data are either protected or never disclosed.

The computations are performed directly on each pair of $E_{pk}(x_j)$ and $E_{pk}(y_j)$ at Step 1(a) of Algorithm 3. They were never decrypted, nor was information leaked regarding X and Y, as long as the encryption scheme was semantically secure. The original data could only be disclosed at Step 2(a) because decryption operations are performed on the intermediate computation results. However, t_j was uniformly distributed in $\mathbb{Z}_N$ from the viewpoint of P_2 because $t_j = x_j - y_j + r_j \mod N$ and r_j was randomly chosen and known only to P_1. Therefore, t_j did not leak any information regarding X and Y to P_2. The remaining computations were based on either encrypted or randomized data. As a consequence, if P_1 and P_2 did not collude, no information related to either X or Y was ever disclosed to the two parties. (Note that collusion was prohibited under the semi-honest model).

Complexity Analysis: The number of multiplications and encryptions was bounded by O(m), where m is the vector size for the SSED$_R$ protocol. Each step of the protocol needed to be examined before the upper bound could be derived. The protocol performed $2m$ encryptions and $2m$ multiplications at Step 1. Although it was slightly less expensive than an encryption operation, the $E_{pk}(y_j)^{N-1}$ was treated as one encryption. Step 2(a) involved performing m decryptions. A decryption has a cost that is similar to an encryption. Thus, m encryptions were counted for Step 2(a). Step 2(b) involved performing m encryptions and m multiplications. Step 3(a) involved performing $3m$ encryptions and $2m$ multiplications. The two exponentiations were counted as two encryptions. Step 3(b) required $m-1$ multiplications. The computations for the remaining steps were constant. Therefore, the total number of multiplications and encryptions for this sub-protocol was bounded by $O(m)$. Additionally, because an encryption operation is generally several orders of magnitude more expensive than a multiplication, this work can claim that the computation complexity of the SSED$_R$ protocol was bounded by $O(m)$ encryptions.

3.5.4. Secure Hamming Distance-Random Share. Suppose that P_1 holds two encrypted binary vectors $\big(E_{pk}(X), E_{pk}(Y)\big)$ and P_2 holds the secret key (sk). Here, X and Y are two m-dimensional binary vectors. The goal of the Secure Hamming Distance-Random Share (SHD$_R$) protocol is to securely compute the encryption of Hamming distance between the two binary vectors, X and Y. During this protocol, no information regarding X and Y is revealed to either P_1 or P_2. Let d_h denote the Hamming distance between X and Y. The output of this protocol returns two random shares (d'_h and d''_h) such that $d'_h + d''_h \bmod N = d_h$. Note that the random share d'_h is known only to P_1, whereas the random share d''_h is known only to P_2.

According to Equation 3.3, the main challenge is to compute $x_j * y_j$ from $E_{pk}(x_j)$ and $E_{pk}(y_j)$, where $E_{pk}(x_j)$ and $E_{pk}(y_j)$ are encryptions of x_j and y_j. The key steps to securely compute the Hamming distance between the two binary vectors, X and Y, are given by Algorithm 4, where the computations are only based on either encrypted or randomized data. At Step 1(a) of the algorithm, t_{j_1} and t_{j_2} are randomizations of x_j and y_j, respectively, and r_{j_1} and r_{j_2} are randomly chosen to hide the actual values of x_j and y_j. According to the additive homomorphic property of the encryption scheme, $E_{pk}(t_{j_1})$ and $E_{pk}(t_{j_2})$ can be computed as follows:

$$E_{pk}(x_j + r_{j_1}) \leftarrow E_{pk}(x_j) * E_{pk}(r_{j_1})$$

Algorithm 4 $\text{SHD}_R\big(E_{pk}(X), E_{pk}(Y)\big) \rightarrow (d'_h, d''_h)$

```
Require: $P_1$ has $E_{pk}(X)$ and $E_{pk}(Y)$; $P_2$ has the decryption key
1: $P_1$:
    (a) $E_{pk}(t_{j_1}) \leftarrow E_{pk}(x_j + r_{j_1})$ and $E_{pk}(t_{j_2}) \leftarrow E_{pk}(y_j + r_{j_2})$, where $r_{j_1}, r_{j_2} \in_R \mathbb{Z}_N$ and $1 \leq j \leq m$
    (b) Send $\big(E_{pk}(t_{1_1}), E_{pk}(t_{1_2})\big), \ldots, \big(E_{pk}(t_{m_1}), E_{pk}(t_{m_2})\big)$ to $P_2$
2: $P_2$:
    (a) Perform decryption operation to get $(t_{1_1}, t_{1_2})$, $\ldots$, $(t_{m_1}, t_{m_2})$
    (b) Compute $E_{pk}(t_{1_1} * t_{1_2}), \ldots, E_{pk}(t_{m_1} * t_{m_2})$, and send them to $P_1$
3: $P_1$:
    (a) Obtain $E_{pk}(x_j \cdot y_j)$ by eliminating $r_{j_1} * r_{j_2}$, $r_{j_2} * x_j$, and $r_{j_1} * y_j$ from $E_{pk}(t_{j_1} * t_{j_2})$, for $1 \leq j \leq m$
    (b) $E_{pk}(d_h) \leftarrow E_{pk}\big(m - \sum_{j=1}^{m} x_j * y_j\big)$
    (c) Compute $E_{pk}(d_h + r)$, where $r \in_R \mathbb{Z}_N$
    (d) Set $d'_h = N - r$ and send $E_{pk}(d_h + r)$ to $P_2$
4: $P_2$:
    (a) Perform decryption on $E_{pk}(d_h + r)$ to obtain $d_h + r$
    (b) Set $d''_h = d_h + r$
```

$$E_{pk}(y_j + r_{j_2}) \leftarrow E_{pk}(y_j * E_{pk}(r_{j_2})$$

P_2 can decrypt $\big(E_{pk}(t_{j_1}), E_{pk}(t_{j_2})\big)$ to obtain (t_{j_1}, t_{j_2}) at Step 2(a) because it has the decryption key. P_2 computes the multiplication of $t_{j_1} * t_{j_2}$. It sends then the encrypted results to P_1. Because P_1 knows r_{j_1}, r_{j_2}, $E_{pk}(x_j)$, and $E_{pk}(y_j)$, it can easily compute $E_{pk}(r_{j_1} * r_{j_2})$, $r_{j_2} * x_j$, and $r_{j_1} * y_j$. In Step 3(a), since $E_{pk}(t_{j_1} * t_{j_2}) = E_{pk}(x_j * y_j + r_{j_1} * r_{j_2} + r_{j_2} * x_j + r_{j_1} * y_j)$, P_1 can obtain $E_{pk}(x_j * y_j)$ by:

$$E_{pk}(x_j * y_j) \leftarrow E_{pk}(t_{j_1} * t_{j_2}) * (E_{pk}(r_{j_1} * r_{j_2}) * E_{pk}(r_{j_2} * x_j) * E_{pk}(r_{j_1} * y_j))^{N-1}$$

From here, $E_{pk}(d_h)$ can be directly derived at Step 3(b):

$$E_{pk}(d_h) \leftarrow E_{pk}(m) * \big(E_{pk}(x_j \cdot y_j) * \cdots * E_{pk}(x_m \cdot y_m)\big)^{N-1}$$

The rest of the steps are fairly obvious. All of the above computations are based on the additive homomorphic property of the encryption schemes.

Security Analysis: The SHD$_R$ protocol has the same structure as the SSED$_R$ protocol. Therefore, it is secure under the semi-honest model as well.

Complexity Analysis: This work can claim that the computation complexity of SHD$_R$ was bounded by $O(m)$ encryptions. This asymptotic bound can be derived by following the same steps as discuses in Section 3.5.3.

Justification for using AH-Enc to securely implement a distance: The SM protocol is the main building block and the performance bottleneck for securely computing the distance in both Euclidean and Hamming metrics. Therefore, the computation costs of two different implementations of the SM protocol are analyzed and compared: homomorphic encryption approach (SM-HE) and garbled circuit approach (SM-GC). The running times of the two implementations were compared for varying number of bits required to perform multiplication. The running time of each implementation was independent of the input size, and the computation cost of SM-GC was significantly higher than that of SM-HE (see Table 3.2). For example, the computation time when SM-GC was used to multiply two 10-bit numbers was 1.752 seconds. While in contrast was 0.02984 second when SM-HE is used. Therefore, the homomorphic encryption approach was adopted to implement the distance in both Euclidean and Hamming metrics.

3.5.5. Secure Bit-Decomposition. Assume that P_1 has $E_{pk}(z)$ and P_2 has sk, where z is not known to both parties and $0 \leq z < 2^l$, where l is referred to as the domain size (in bits) of z. Given $E_{pk}(z)$, the Secure Bit-Decomposition (SBD) protocol [86, 87] protocol is used primarily to compute the encryptions of the individual bits of binary representation of z. The output $[z] = \langle E_{pk}(z_1), \ldots, E_{pk}(z_l) \rangle$ is known only to P_1 at the end, where z_1 and z_l denote the most and least significant bits of z, respectively. During this process, neither the value of z nor any z_i's is revealed either to P_1 or P_2. Since the goal of this work is not to investigate existing SBD protocols, the most efficient SBD scheme that was recently proposed in [86] is used in the proposed protocols.

Share size	Distance size	SM-GC	SM-HE
1024	10	1.752s	0.02984s
1024	20	1.792s	0.02985s

Table 3.2: Garbled circuit vs. homomorphic-based secure multiplication

Example 4. *Assume* $z = 55$ *and* $l = 6$. *Then, the* SBD *protocol in [86] with input* $E_{pk}(55)$ *returns* $[55] = \langle E_{pk}(1), E_{pk}(1), E_{pk}(0),\ E_{pk}(1), E_{pk}(1), E_{pk}(1) \rangle$ *as the output to* P_1. □

Security Analysis: The SBD protocol in "An efficient and probabilistic secure bit- decomposition" [86] is secure under the semi-honest model.

Complexity Analysis: The computation complexity of the SBD protocol proposed in "An efficient and probabilistic secure bit-decomposition" [86] was bounded by $O(l)$ encryptions (under the assumption that encryption and decryption operations based on the Paillier cryptosystem [75] take similar amount of time) .

3.5.6. Secure Bit-OR. In the Secure Bit-OR (SBOR) protocol, both P_1, with input $\big(E_{pk}(o_1), E_{pk}(o_2)\big)$ and P_2 with (sk) securely compute $E_{pk}(o_1 \vee o_2)$, where o_1 and o_2 are two bits. The output $E_{pk}(o_1 \vee o_2)$ is known only to P_1.

Security Analysis: The security of the SBOR depends solely on the underlying SM protocol. This is because, the only step at which P_1 and P_2 interact in the SBOR protocol is during the SM protocol. Since the SM is secure under the semi-honest model, this work claims that the SBOR protocol is also secure under the semi-honest model.

Complexity Analysis: Since the SBOR protocol utilized the Secure Multiplication (SM) protocol as a sub-routine, the computation cost of the Secure Bit-OR protocol was bounded by (small) constant number of encryptions and multiplications.

3.5.7. Secure Minimum. Assume that in the Secure Minimum (SMIN) protocol, P_1 holds private input (u', v') and P_2 holds sk, where $u' = \big([u], E_{pk}(s_u)\big)$ and $v' = \big([v], E_{pk}(s_v)\big)$. Here, $[u] = \big\langle E_{pk}(u_1), \ldots, E_{pk}(u_l)\big\rangle$ and $[v] = \big\langle E_{pk}(v_1), \ldots, E_{pk}(v_l)\big\rangle$ where u_1 (resp., v_1) and u_l (resp., v_l) are the most and least significant bits of u (resp., v), respectively. Here, s_u (resp., s_v) denotes the secret corresponding to u (resp., v). The goal of the SMIN is for P_1 and P_2 to jointly compute the encryptions of the individual bits of minimum number between u and v, denoted by $\big([\min(u,v)]\big)$. In addition, they compute $E_{pk}\big(s_{\min(u,v)}\big)$, the encryption of the secret corresponding to the minimum value of u and v. That is, the output is $\big([\min(u,v)], E_{pk}\big(s_{\min(u,v)}\big)\big)$, which will be known only to P_1. No information regarding the contents of u,v,s_u, or s_v is revealed to either P_1 or P_2 during this protocol.

The basic concept of the proposed SMIN protocol is that P_1 randomly chooses the functionality F by flipping a coin, where F is either $u > v$ or $v > u$ and obliviously executes F with P_2. The result of the functionality F is oblivious to P_2 because F is randomly chosen and known only to P_1. Based on the comparison result and chosen F, P_1 computes $\big[\min(u,v)\big]$ and $E_{pk}\big(s_{\min(u,v)}\big)$ locally using homomorphic properties.

The overall steps involved in the SMIN protocol are shown in Algorithm 5. P_1 initially chooses the functionality F as either $u > v$ or $v > u$ randomly. P_1 then uses the SM protocol, with the help of P_2, to compute $E_{pk}(u_i * v_i)$ for $1 \leq i \leq l$. P_1 then follows several steps for $1 \leq i \leq l$:

- Use $T_i = E_{pk}(u_i) * E_{pk}(v_i) * E_{pk}(u_i * v_i)^{N-2}$ to compute the encrypted bit-wise XOR between the bits u_i and v_i. In general, for any two given bits (o_1 and o_2), the property $o_1 \oplus o_2 = o_1 + o_2 - 2(o_1 * o_2)$ always holds.

- Compute an encrypted vector H by preserving the first occurrence of $E_{pk}(1)$ (if there exists one) in T by initializing $H_0 = E_{pk}(0)$. The remaining H entries are computed as $H_i = H_{i-1}^{r_i} * T_i$. At most one of the entries in H is $E_{pk}(1)$, and the remaining entries are encryptions of either 0 or a random number.

- P_1 then computes $\Phi_i = E_{pk}(-1) * H_i$. Note that, "$-1$" is equivalent to "$N-1$" under $\mathbb{Z}_N$. Here, $\Phi_i = E_{pk}(0)$ occurs at most once because H_i is equal to $E_{pk}(1)$ occurs at most once. Additionally, if $\Phi_j = E_{pk}(0)$, then index j is the position at which the bits of u and v differ first (beginning at the most significant bit position).

P_1 creates two encrypted vectors (W and Γ) for $1 \leq i \leq l$ as follows:

Algorithm 5 SMIN$(u', v') \rightarrow \big[\min(u,v)\big], E_{pk}\big(s_{\min(u,v)}\big)$

```
Require: $P_1$ has $u' = \big([u], E_{pk}(s_u)\big)$ and $v' = \big([v], E_{pk}(s_v)\big)$, where $0 \le u, v < 2^l$; $P_2$ has $sk$

1: $P_1$:
    (a). Randomly choose the functionality $F$
    (b). for $i = 1$ to $l$ do:
        • $E_{pk}(u_i * v_i) \leftarrow \text{SM}\big(E_{pk}(u_i), E_{pk}(v_i)\big)$
        • $T_i \leftarrow E_{pk}(u_i \oplus v_i)$
        • $H_i \leftarrow H_{i-1}^{r_i} * T_i$; $r_i \in_R \mathbb{Z}_N$ and $H_0 = E_{pk}(0)$
        • $\Phi_i \leftarrow E_{pk}(-1) * H_i$
        • if $F : u > v$ then:
          $W_i \leftarrow E_{pk}(u_i) * E_{pk}(u_i * v_i)^{N-1}$ and $\Gamma_i \leftarrow E_{pk}(v_i - u_i) * E_{pk}(\hat{r}_i)$; $\hat{r}_i \in_R \mathbb{Z}_N$
          else $W_i \leftarrow E_{pk}(v_i) * E_{pk}(u_i * v_i)^{N-1}$  $\Gamma_i \leftarrow E_{pk}(u_i - v_i) * E_{pk}(\hat{r}_i)$; $\hat{r}_i \in_R \mathbb{Z}_N$
        • $L_i \leftarrow W_i * \Phi_i^{r'_i}$; $r'_i \in_R \mathbb{Z}_N$
    (c). if $E_{pk}(s_u)$ and $E_{pk}(s_u) \neq Nil$ then
        • if $F : u > v$ then: $\delta \leftarrow E_{pk}(s_v - s_u) * E_{pk}(\bar{r})$
        • else $\delta \leftarrow E_{pk}(s_u - s_v) * E_{pk}(\bar{r})$, where $\bar{r} \in_R \mathbb{Z}_N$
    (d). $\Gamma' \leftarrow \pi_1(\Gamma)$ and $L' \leftarrow \pi_2(L)$
    (e). Send $\delta, \Gamma'$ and $L'$ to $P_2$

2: $P_2$:
    (a). Receive $\delta, \Gamma'$ and $L'$ from $P_1$
    (b). Decryption: $M_i \leftarrow D_{sk}(L'_i)$, for $1 \le i \le l$
    (c). if $\exists\ j$ such that $M_j = 1$ then $\alpha \leftarrow 1$   else $\alpha \leftarrow 0$
    (d). if $\alpha = 0$ then:
        • $M'_i \leftarrow E_{pk}(0)$, for $1 \le i \le l$
        • if $\delta' \neq nil$ then $\delta' \leftarrow E_{pk}(0)$
      else
        • $M'_i \leftarrow \Gamma_i'^{\alpha} * r^N$, where $r \in_R \mathbb{Z}_N$ and is different for $1 \le i \le l$
        • if $\delta' \neq nil$ then $\delta' \leftarrow \delta * r_\delta^N$, where $r_\delta \in_R \mathbb{Z}_N$
    (e). Send $M', E_{pk}(\alpha)$ and $\delta'$ to $P_1$
3: $P_1$:
    (a). Receive $M', E_{pk}(\alpha)$ and $\delta'$ from $P_2$
    (b). $\widetilde{M} \leftarrow \pi_1^{-1}(M')$ and $\theta \leftarrow \delta' * E_{pk}(\alpha)^{N-\bar{r}}$
    (c). $\lambda_i \leftarrow \widetilde{M}_i * E_{pk}(\alpha)^{N-\hat{r}_i}$, for $1 \le i \le l$
    (d). if $F : u > v$ then:
        • $E_{pk}\big(\min(u,v)_i\big) \leftarrow E_{pk}(u_i) * \lambda_i$, for $1 \le i \le l$
        • if $E_{pk}(s_u) \neq nil$ then $E_{pk}(s_{\min(u,v)}) \leftarrow E_{pk}(s_u) * \theta$ else $E_{pk}(s_{\min(u,v)}) \leftarrow nil$
      else
        • $E_{pk}\big(\min(u,v)_i\big) \leftarrow E_{pk}(v_i) * \lambda_i$, for $1 \le i \le l$
        • if $E_{pk}(s_v) \neq nil$ then $E_{pk}\big(s_{\min(u,v)}\big) \leftarrow E_{pk}(s_v) * \theta$ else $E_{pk}(s_{\min(u,v)}) \leftarrow nil$
```

- If $F : u > v$,

$$\begin{aligned} W_i &= E_{pk}(u_i) * E_{pk}(u_i * v_i)^{N-1} \\ &= E_{pk}\big(u_i * (1 - v_i)\big) \\ \Gamma_i &= E_{pk}(v_i - u_i) * E_{pk}(\hat{r}_i) \\ &= E_{pk}(v_i - u_i + \hat{r}_i) \end{aligned}$$

- If $F : v > u$,

$$\begin{aligned} W_i &= E_{pk}(v_i) * E_{pk}(u_i * v_i)^{N-1} \\ &= E_{pk}\big(v_i * (1 - u_i)\big) \\ \Gamma_i &= E_{pk}(u_i - v_i) * E_{pk}(\hat{r}_i) \\ &= E_{pk}(u_i - v_i + \hat{r}_i) \end{aligned}$$

where $\hat{r}_i$ is a random number (hereafter denoted by $\in_R$) in $\mathbb{Z}_N$. Observe that if $F : u > v$, then $W_i = E_{pk}(1)$ iff $u_i > v_i$. Otherwise, $W_i = E_{pk}(0)$. Similarly, when $F : v > u$, then $W_i = E_{pk}(1)$ iff $v_i > u_i$. Otherwise $W_i = E_{pk}(0)$. Also dependent on F, Γ_i stores the encryption of the randomized difference between u_i and v_i, which is used in later computations.

P_1 computes L by combining Φ with W. More precisely, P_1 computes $L_i = W_i * \Phi_i^{r'_i}$, where r'_i is a random number in $\mathbb{Z}_N$. The observation here is if $\exists$ an index j such that $\Phi_j = E_{pk}(0)$, denoting the first flip in the bits of u and v, then W_j stores the corresponding desired information (i.e., whether $u_j > v_j$ or $v_j > u_j$) in encrypted form. Dependent on F, P_1 also computes the encryption of the randomized difference between s_u and s_v, storing it in δ. More specifically, if $F : u > v$, then $\delta = E_{pk}(s_v - s_u + \bar{r})$. Otherwise, $\delta = E_{pk}(s_u - s_v + \bar{r})$, where $\bar{r} \in_R \mathbb{Z}_N$.

P_1 next uses two random permutation functions functions (π_1 and π_2) to permute the encrypted vectors Γ and L. More specifically, he/she computes $\Gamma' = \pi_1(\Gamma)$ and $L' = \pi_2(L)$ and then sends them, along with δ, to P_2. Upon receiving, P_2 decrypts L' component-wise to obtain $M_i = D_{sk}(L'_i)$ for $1 \leq i \leq l$ and checks for index j. That is, if $M_j = 1$, then P_2 sets α to 1. Otherwise, he/she sets α to 0. P_2 also computes a new encrypted vector (M')

depending on the value of α. Precisely, If $\alpha = 0$, then $M'_i = E_{pk}(0)$ for $1 \le i \le l$. Here, $E_{pk}(0)$ is different for each i. In contrast, when $\alpha = 1$, P_2 sets M'_i to the re-randomized value of Γ'_i. That is, $M'_i = \Gamma'^{\alpha}_i * r^N$, where the term r^N comes from the re-randomization and $r \in_R \mathbb{Z}_N$ should be different for each i. Re-randomization property means that any one can alter a ciphertext for a plaintext into a new ciphertext in an unlinkable way, such that both ciphertexts will decrypt to the same original plaintext [78]. Furthermore, P_2 computes $\delta' = E_{pk}(0)$ if $\alpha = 0$. However, when $\alpha = 1$, P_2 sets δ' to $\delta * r_\delta^N$, where r_δ is a random number in $\mathbb{Z}_N$. P_2 then sends $M', E_{pk}(\alpha)$ and δ' to P_1. After receiving $M', E_{pk}(\alpha)$, and δ', P_1 computes the inverse permutation of M' as $\widetilde{M} = \pi_1^{-1}(M')$. P_1 then performs the following homomorphic operations to compute the encryption of the i^{th} bit of $\min(u, v)$ $\big($i.e., $E_{pk}\big(\min(u, v)_i\big)\big)$ for $1 \le i \le l$:

- Remove the randomness from $\widetilde{M}_i$ by computing $\lambda_i = \widetilde{M}_i * E_{pk}(\alpha)^{N-\hat{r}_i}$
- If $F : u > v$, compute $E_{pk}\big(\min(u, v)_i\big) = E_{pk}(u_i) * \lambda_i = E_{pk}\big(u_i + \alpha * (v_i - u_i)\big)$. Otherwise, compute $E_{pk}\big(\min(u, v)_i\big) = E_{pk}(v_i) * \lambda_i = E_{pk}\big(v_i + \alpha * (u_i - v_i)\big)$.

P_1 also computes $E_{pk}\big(s_{\min(u,v)}\big)$ as follows: If $F : u > v$, P_1 computes $E_{pk}\big(s_{\min(u,v)}\big) = E_{pk}(s_u) * \theta$, where $\theta = \delta' * E_{pk}(\alpha)^{N-\bar{r}}$. Otherwise, P_1 computes $E_{pk}\big(s_{\min(u,v)}\big) = E_{pk}(s_v) * \theta$.

One main observation in the SMIN protocol, (upon which the correctness of the final output can be justify) is that, if $F : u > v$, then $\min(u, v)_i = (1 - \alpha) * u_i + \alpha * v_i$ always holds for $1 \le i \le l$. In contrast, if $F : v > u$, then $\min(u, v)_i = \alpha * u_i + (1 - \alpha) * v_i$ always holds. Similar conclusions can be drawn for $s_{\min(u,v)}$. Similar formulations can also be used to design a Secure Maximum (SMAX) protocol to compute both $\big[\max(u, v)\big]$ and $E_{pk}\big(s_{\max(u,v)}\big)$. Multiple secrets of u and v can also be fed as input (in encrypted form) to SMIN and SMAX. For example, let s_u^1 and s_u^2 (resp., s_v^1 and s_v^2) be two secrets associated with u (resp., v). The SMIN protocol then takes $\big([u], E_{pk}(s_u^1), E_{pk}(s_u^2)\big)$ and $\big([v], E_{pk}(s_v^1), E_{pk}(s_v^2)\big)$ as P_1's input. It outputs $\big[\min(u, v)\big], E_{pk}\big(s^1_{\min(u,v)}\big)$ and $E_{pk}\big(s^2_{\min(u,v)}\big)$ to P_1.

Example 5. *For simplicity, consider that $u = 55$, $v = 58$, and $l = 6$. Suppose s_u and s_v are the secrets associated with u and v, respectively. Assume that P_1 holds $\big([55], E_{pk}(s_u)\big)$ $\big([58], E_{pk}(s_v)\big)$. Additionally, assume that P_1's random permutation functions are as given in Table 3.3. Suppose P_1 chooses the functionality $F : v > u$ without loss of generality.*

Table 3.3: P_1's random permutation functions

i	$=$	1	2	3	4	5	6
		$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
$\pi_1(i)$	$=$	6	5	4	3	2	1
$\pi_2(i)$	$=$	2	1	5	6	3	4

Then, various intermediate results based on the SMIN *protocol are as given in Table 3.4. These results lead to the following observations':*

- *One of the entries in H is , at most $E_{pk}(1)$, namely, H_3 and the remaining entries are encryptions of either 0 or a random number in $\mathbb{Z}_N$.*
- *Index $j = 3$ is the first position at which the corresponding bits of u and v differ.*
- *Because H_3 is equal to $E_{pk}(1)$, $\Phi_3 = E_{pk}(0)$. Additionally, because $M_5 = 1$, P_2 sets α to 1.*
- $E_{pk}(s_{\min(u,v)}) = E_{pk}\big(\alpha * s_u + (1-\alpha) * s_v\big) = E_{pk}(s_u)$.

At the end, only P_1 knows $\big[\min(u,v)\big] = [u] = [55]$ and $E_{pk}\big(s_{\min(u,v)}\big) = E_{pk}(s_u)$. □

Security Analysis: According to Algorithm 5, the execution image of P_2 is denoted by $\Pi_{P_2}(\text{SMIN})$, where

$$\Pi_{P_2}(\text{SMIN}) = \big\{\langle \delta, s + \bar{r} \bmod N\rangle, \langle \Gamma'_i, \mu_i + \hat{r}_i \bmod N\rangle, \langle L'_i, \alpha\rangle\big\}$$

Observe that $s + \bar{r} \bmod N$ and $\mu_i + \hat{r}_i \bmod N$ are derived upon decrypting δ and Γ'_i for $1 \le i \le l$, respectively. Note that the modulo operator is implicit in the decryption function. Also, P_2 receives L' from P_1. Let α denote the (oblivious) comparison result computed from L'. Without loss of generality, suppose the simulated image of P_2 be denoted by $\Pi^S_{P_2}(\text{SMIN})$, where

$$\Pi^S_{P_2}(\text{SMIN}) = \big\{\langle \delta^*, r^*\rangle, \langle s'_{1,i}, s'_{2,i}\rangle, \langle s'_{3,i}, \alpha'\rangle \mid \text{for } 1 \le i \le l\big\}$$

Here, $\delta^*, s'_{1,i}$ and $s'_{3,i}$ are randomly generated from $\mathbb{Z}_{N^2}$, whereas r^* and $s'_{2,i}$ are randomly generated from $\mathbb{Z}_N$. Additionally, α' denotes a random bit. Since E_{pk} is a semantically secure

Table 3.4: SMIN example: P_1 chooses F as $v > u$, where $u = 55$ and $v = 58$

$[u]$	$[v]$	W_i	T_i	G_i	H_i	Φ_i	L_i	Γ'_i	L'_i	M_i	λ_i	$\min_i$
1	1	0	r	0	0	-1	r	$1+r$	r	r	0	1
1	1	0	r	0	0	-1	r	r	r	r	0	1
0	1	1	$-1+r$	1	1	0	1	$1+r$	r	r	-1	0
1	0	0	$1+r$	1	r	r	r	$-1+r$	r	r	1	1
1	1	0	r	0	r	r	r	r	1	1	0	1
1	0	0	$1+r$	1	r	r	r	r	r	r	1	1

*All column values except those in M_i column are in encrypted form. Additionally, $r \in_R \mathbb{Z}_N$ which is different for each row and column.

encryption scheme a with resulting ciphertext size that is less than N^2, δ is computationally indistinguishable from δ^*. Similarly, Γ'_i and L'_i are computationally indistinguishable from $s'_{1,i}$ and $s'_{3,i}$, respectively. Also, as $\bar{r}$ and $\hat{r}_i$ are randomly generated from $\mathbb{Z}_N$, both $s + \bar{r} \bmod N$ and $\mu_i + \hat{r}_i \mod N$ are computationally indistinguishable from r^* and $s'_{2,i}$, respectively. Furthermore, because the functionality is randomly chosen by P_1 (at Step 1(a) of Algorithm 5), α is either 0 or 1 with equal probability. Thus, α is computationally indistinguishable from α'. Putting the above results together, and following from Definition 1, it was concluded that $\Pi_{P_2}(\text{SMIN})$ was computationally indistinguishable from $\Pi^S_{P_2}(\text{SMIN})$. This implies that, during the execution of the SMIN protocol, P_2 does not learn any information regarding either u, v, s_u, s_v, or the actual comparison result. Intuitively speaking, the information that P_2 has during an execution of the SMIN protocol is either random or pseudo-random, so this information does not disclose anything regarding u, v, s_u, or s_v. Additionally, as F is known only to P_1, the actual comparison result is oblivious to P_2.

In contrast, the execution image of P_1, denoted by $\Pi_{P_1}(\text{SMIN})$, is given by

$$\Pi_{P_1}(\text{SMIN}) = \left\{M'_i, E_{pk}(\alpha), \delta' \mid \text{for } 1 \leq i \leq l\right\}$$

Here, M'_i, and δ' are encrypted values, which are random in $\mathbb{Z}_{N^2}$, received from P_2 (at Step 3(a) of Algorithm 5). Let the simulated image of P_1 be denoted by $\Pi^S_{P_1}(\text{SMIN})$, where

$$\Pi^S_{P_1}(\text{SMIN}) = \{s'_{4,i}, b', b'' \mid \text{for } 1 \leq i \leq l\}$$

Values $s'_{4,i}, b'$ and b'' are randomly generated from $\mathbb{Z}_{N^2}$. Since E_{pk} is a semantically secure encryption scheme with a resulting ciphertext size of less than N^2, it implies that both $M'_i, E_{pk}(\alpha)$ and δ' are computationally indistinguishable from $s_{4,i}, b'$ and b'', respectively. Therefore, Π_{P_1}(SMIN) is computationally indistinguishable from $\Pi^S_{P_1}$(SMIN) based on Definition 1. As a result, P_1 cannot learn any information regarding u, v, s_u, s_v, or the comparison result during the execution of the SMIN. Putting everything together, and following from Definition 1, it was concluded that the SMIN protocol was secure under the semi-honest model.

Complexity Analysis: Recall that, under Paillier's encryption scheme, a decryption operation had almost the same cost as an encryption operation and an exponentiation operation treated as an encryption operation. Additionally, because an encryption operation is generally several orders of magnitude more expensive than a multiplication, it was concluded that the computation complexity of the Secure Minimum protocol was bounded by O(l) encryptions. This asymptotic bound can be derived by following the same techniques used to prove the previous sub-protocols.

3.5.8. Secure Minimum out of n Numbers. Consider P_1 with n encrypted vectors $\big([d_1], \ldots, [d_n]\big)$ along with their encrypted secrets and P_2 with sk, where $0 \leq d_i < 2^l$ for $1 \leq i \leq n$. Here, $[d_i] = \big\langle E_{pk}(d_{i,1}), \ldots, E_{pk}(d_{i,l})\big\rangle$, where $d_{i,1}$ and $d_{i,l}$ are the most and least significant bits of integer d_i, respectively, for $1 \leq i \leq n$. Here, the secret of d_i is denoted by s_{d_i} for $1 \leq i \leq n$. The primary goal of the Secure Minimum Out of n Numbers (SMIN$_n$) protocol is to compute $\big[\min(d_1, \ldots, d_n)\big] = \big[d_{\min}\big]$ along with the encryption of the secret that corresponds to the global minimum, denoted by $E_{pk}\big(s_{\min(d_1,\ldots,d_n)}\big) = E_{pk}\big(s_{d_{\min}}\big)$ without revealing any information about d_i's and their secrets to either P_1 or P_2. . This work constructed a new SMIN$_n$ protocol by utilizing the SMIN protocol as the building block. The proposed SMIN$_n$ protocol is an iterative approach that computes the desired output in a hierarchical fashion. In each iteration, both the minimum between a pair of values and the secret that corresponds to the minimum value are computed (in encrypted form) and fed as input to the next iteration. That will generate a binary execution tree in a bottom-up fashion. At the end, only P_1 knows the final results $[d_{\min}]$ and $E_{pk}(s_{d_{\min}})$. The overall steps involved in the proposed SMIN$_n$ protocol are highlighted in Algorithm 6.

Algorithm 6 $\text{SMIN}_n\Big(\big([d_1], E_{pk}(s_{d_1})\big), .., \big([d_n], E_{pk}(s_{d_n})\big)\Big) \rightarrow \big([d_{\min}], E_{pk}(s_{d_{\min}})\big)$

```
Require: $P_1$ has $\Big(\big([d_1], E_{pk}(s_{d_1})\big), \ldots, \big([d_n], E_{pk}(s_{d_n})\big)\Big)$; $P_2$ has $sk$
$P_1$:
    (a). $[d'_i] \leftarrow [d_i]$ and $s'_i \leftarrow E_{pk}(s_{d_i})$, for $1 \leq i \leq n$
    (b). $num \leftarrow n$
for $i = 1$ to $\lceil \log_2 n \rceil$:
    (a). for $1 \leq j \leq \lfloor \frac{num}{2} \rfloor$:
        • if $i = 1$ then:
            – $\big([d'_{2j-1}], s'_{2j-1}\big) \leftarrow \text{SMIN}(x, y)$, where $x = \big([d'_{2j-1}], s'_{2j-1}\big)$ and $y = \big([d'_{2j}], s'_{2j}\big)$
            – $[d'_{2j}] \leftarrow 0$ and $s'_{2j} \leftarrow 0$
          else
            – $\big([d'_{2i(j-1)+1}], s'_{2i(j-1)+1}\big) \leftarrow \text{SMIN}(x, y)$, where $x = \big([d'_{2i(j-1)+1}], s'_{2i(j-1)+1}\big)$ and $y = \big([d'_{2ij-1}], s'_{2ij-1}\big)$
            – $[d'_{2ij-1}] \leftarrow 0$ and $s'_{2ij-1} \leftarrow 0$
    (b). $num \leftarrow \lceil \frac{num}{2} \rceil$
$P_1$: $[d_{\min}] \leftarrow [d'_1]$ and $E_{pk}\big(s_{d_{\min}}\big) \leftarrow s'_1$
```

Initially, P_1 assigns both $[d_i]$ and $E_{pk}(s_{d_i})$ to a temporary vector $[d'_i]$ and variable s'_i for $1 \leq i \leq n$, respectively. Also, he/she creates a global variable (num) and initializes it to n, where num represents the number of (non-zero) vectors involved in each iteration. Since the SMIN_n protocol executes in a binary tree hierarchy (bottom-up fashion), it has $\lceil \log_2 n \rceil$ iterations. In each iteration, the number of vectors involved varies. In the first iteration (i.e., $i = 1$), P_1 with private input $\Big(\big([d'_{2j-1}], s'_{2j-1}\big), \big([d'_{2j}], s'_{2j}\big)\Big)$, and P_2 with sk involve in the SMIN protocol, for $1 \leq j \leq \lfloor \frac{num}{2} \rfloor$. At the end of the first iteration, only P_1 knows $\big[\min(d'_{2j-1}, d'_{2j})\big]$ and $s'_{\min(d'_{2j-1}, d'_{2j})}$, and nothing is revealed to P_2 for $1 \leq j \leq \lfloor \frac{num}{2} \rfloor$. Also, P_1 stores the result $\big[\min(d'_{2j-1}, d'_{2j})\big]$ and $s'_{\min(d'_{2j-1}, d'_{2j})}$ in $[d'_{2j-1}]$ and s'_{2j-1}, respectively. Additionally, P_1 updates the values of $[d'_{2j}]$, s'_{2j} to 0 and num to $\lceil \frac{num}{2} \rceil$, respectively.

During the i^{th} iteration, only the non-zero vectors (along with the corresponding encrypted secrets) are involved in the SMIN protocol for $2 \leq i \leq \lceil \log_2 n \rceil$. For example, during the second iteration (i.e., $i = 2$), only $\big([d'_1], s'_1\big), \big([d'_3], s'_3\big)$, and so on are involved. Note that in each iteration, the output is revealed only to P_1 and num is updated to $\lceil \frac{num}{2} \rceil$. At the end of the SMIN$_n$ protocol, P_1 assigns the final encrypted binary vector of global minimum value $\big($i.e., $\big[\min(d_1, \ldots, d_n)\big]\big)$, which is stored in $[d'_1]$, to $[d_{\min}]$. P_1 also assigns s'_1 to $E_{pk}(s_{d_{\min}})$.

Example 6. *Suppose that P_1 holds $\big\langle [d_1], \ldots, [d_6] \big\rangle$ (i.e., $n = 6$). For simplicity, it is assuming that there are no secrets associated with d_i's. Then, based on* SMIN$_n$ *protocol, the binary execution tree (in a bottom-up fashion) to compute $\big[\min(d_1, \ldots, d_6)\big]$ is shown in Figure 3.1. Note that, $[d'_i]$ is initially set to $[d_i]$ for $1 \leq i \leq 6$.* □

Security Analysis: According to Algorithm 6, it is clear that the SMIN$_n$ protocol uses SMIN protocol as a building block in an iterative manner. As proved in Section 3.5.7, the SMIN protocol is secure under the semi-honest model. Also, the output of SMIN which is passed as input to the next iteration in the SMIN$_n$ protocol is in encrypted format. Note that the SMIN$_n$ protocol is solely based on SMIN and there are no other interactive steps between P_1 and P_2. Hence, by composition theorem [40], it was concluded that the sequential combination of the SMIN routines led to the proposed SMIN$_n$ protocol that guarantees security under the semi-honest model.

Complexity Analysis: Recall that, under Paillier's encryption scheme [75], a decryption operation had almost the same cost as an encryption operation and an exponentiation operation treated as an encryption operation. Additionally, because an encryption operation is generally several orders of magnitude more expensive than a multiplication, it was concluded that the computation complexity of the Secure Minimum Out of n Numbers protocol was bounded by $O(l * n * \log_2 n)$ encryptions, where l referred as the domain size (in bits). $\log_2 n$ because the SMIN$_n$ protocol executed in a binary tree hierarchy (bottom-up fashion). This asymptotic bound can be derived by following the same techniques used to prove the previous sub-protocols.

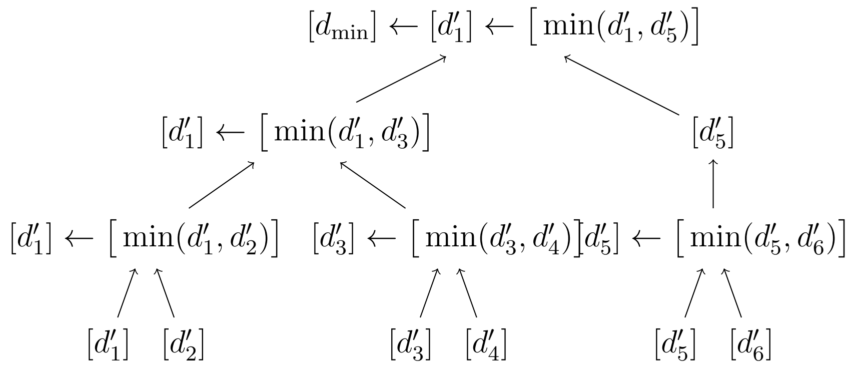

$[d_{\min}] \leftarrow [d'_1] \leftarrow [\min(d'_1, d'_5)]$
$[d'_1] \leftarrow [\min(d'_1, d'_3)]$
$[d'_5]$
$[d'_1] \leftarrow [\min(d'_1, d'_2)]$
$[d'_3] \leftarrow [\min(d'_3, d'_4)]$
$[d'_5] \leftarrow [\min(d'_5, d'_6)]$
$[d'_1]$ $[d'_2]$
$[d'_3]$ $[d'_4]$
$[d'_5]$ $[d'_6]$

Figure 3.1: Binary execution tree for $n = 6$ based on the SMIN$_n$

3.5.9. Secure Frequency. Consider a situation where P_1 holds private input $\Big(\big\langle E_{pk}(c_1), \ldots, E_{pk}(c_w)\big\rangle, \big\langle E_{pk}(c'_1), \ldots, E_{pk}(c'_k)\big\rangle\Big)$ and P_2 holds the secret key sk. The Secure Frequency (SF) protocol is used primarily to securely compute the encryption of the frequency of c_j, denoted by $E_{pk}\big(f(c_j)\big)$, in the list $\langle c'_1, \ldots, c'_k\rangle$, for $1 \leq j \leq w$. Here, $f(c_j)$ denotes the number of times element c_j occurs (i.e., frequency) in the list $\langle c'_1, \ldots, c'_k\rangle$. Here, it is explicitly assumed that c_j's are unique and $c'_i \in \{c_1, \ldots, c_w\}$ for $1 \leq i \leq k$. The output $\Big\langle E_{pk}\big(f(c_1)\big), \ldots, E_{pk}\big(f(c_w)\big)\Big\rangle$ is revealed only to P_1. During the SF protocol, neither c'_i nor c_j is revealed to either P_1 or P_2. Additionally, $f(c_j)$ is kept private from both P_1 and P_2 for $1 \leq i \leq k$ and $1 \leq j \leq w$.

The overall steps involved in the proposed SF protocol are presented in Algorithm 7. P_1 initially computes an encrypted vector S_i such that $S_{i,j} = E_{pk}(c_j - c'_i)$ for $1 \leq j \leq w$. P_1 then randomizes S_i component-wise to obtain $S'_{i,j} = E_{pk}\big(r_{i,j} * (c_j - c'_i)\big)$ where $r_{i,j}$ is a random number in $\mathbb{Z}_N$. P_1 next uses a random permutation function π_i (known only to P_1) to randomly permutes S'_i component-wise for $1 \leq i \leq k$. The output $Z_i \leftarrow \pi_i(S'_i)$ is sent to P_2. Upon receiving, P_2 decrypts Z_i component-wise, computes a vector u_i, and proceeds as follows:

- If $D_{sk}(Z_{i,j}) = 0$, then $u_{i,j}$ is set to 1. Otherwise, $u_{i,j}$ is set to 0.
- The observation is, since $c'_i \in \{c_1, \ldots, c_w\}$, that exactly one of the entries in vector Z_i is an encryption of 0 and the rest are encryptions of random numbers. This further implies that exactly one of the decrypted values of Z_i is 0 and the rest are random numbers. Precisely, if $u_{i,j} = 1$, then $c'_i = c_{\pi^{-1}(j)}$.

Algorithm 7 SF$(\Lambda, \Lambda') \rightarrow \left\langle E_{pk}\big(f(c_1)\big), \ldots, E_{pk}\big(f(c_w)\big) \right\rangle$

```
Require: $P_1$ has $\Lambda = \langle E_{pk}(c_1), \ldots, E_{pk}(c_w) \rangle$, $\Lambda' = \langle E_{pk}(c'_1), \ldots, E_{pk}(c'_k) \rangle$ and $\langle \pi_1, \ldots, \pi_k \rangle$;
    $P_2$ has $sk$
1: $P_1$:
    (a). for $i = 1$ to $k$ do:
        • $T_i \leftarrow E_{pk}(c'_i)^{N-1}$
        • for $j = 1$ to $w$ do:
            – $S_{i,j} \leftarrow E_{pk}(c_j) * T_i$
            – $S'_{i,j} \leftarrow {S_{i,j}}^{r_{i,j}}$, where $r_{i,j} \in_R \mathbb{Z}_N$
        • $Z_i \leftarrow \pi_i(S'_i)$
    (b). Send $Z$ to $P_2$
2: $P_2$:
    (a). Receive $Z$ from $P_1$
    (b). for $i = 1$ to $k$ do
        • for $j = 1$ to $w$ do:
            – if $D_{sk}(Z_{i,j}) = 0$ then $u_{i,j} \leftarrow 1$
              else $u_{i,j} \leftarrow 0$
            – $U_{i,j} \leftarrow E_{pk}(u_{i,j})$
    (c). Send $U$ to $P_1$
3: $P_1$:
    (a). Receive $U$ from $P_2$
    (b). $V_i \leftarrow \pi_i^{-1}(U_i)$, for $1 \leq i \leq k$
    (c). $E_{pk}\big(f(c_j)\big) \leftarrow \prod_{i=1}^{k} V_{i,j}$, for $1 \leq j \leq w$
```

- Compute $U_{i,j} = E_{pk}(u_{i,j})$ and send it to P_1, for $1 \leq i \leq k$ and $1 \leq j \leq w$.

P_1 then performs row-wise inverse permutation on it to obtain $V_i = \pi_i^{-1}(U_i)$ for $1 \leq i \leq k$. Finally, P_1 computes $E_{pk}(f\big(c_j)\big) = \prod_{i=1}^{k} V_{i,j}$ locally for $1 \leq j \leq w$.

Example 7. *Suppose that P_1 holds $\Lambda = \langle E_{pk}(2), E_{pk}(3), E_{pk}(5) \rangle$ and $\Lambda' = \langle E_{pk}(3), E_{pk}$ $(2), E_{pk}(3), E_{pk}(2), E_{pk}(5), E_{pk}(2) \rangle$ (i.e., $w = 3$ and $k = 6$). For simplicity, it is assuming that P_1 has only one random function which is as given in Table 3.5. Then, based on* SF

Table 3.5: SF example: P_1's random permutation function

i	$=$	1	2	3
		$\downarrow$	$\downarrow$	$\downarrow$
$\pi(i)$	$=$	3	1	2

protocol, the various intermediate results based on the SF *protocol are given in Tables 3.6 and 3.7. By applying the Homomorphic properties,* $E_{pk}\big(f(c_1)\big) = E_{pk}\big(f(2)\big) = E_{pk}(0) * E_{pk}(1) * E_{pk}(0) * E_{pk}(1) * E_{pk}(0) * E_{pk}(1) = E_{pk}(3)$, $E_{pk}\big(f(c_2)\big) = E_{pk}\big(f(3)\big) = E_{pk}(1) * E_{pk}(0) * E_{pk}(1) * E_{pk}(0) * E_{pk}(0) * E_{pk}(0) = E_{pk}(2)$, *and* $E_{pk}\big(f(c_3)\big) = E_{pk}\big(f(5)\big) = E_{pk}(0) * E_{pk}(0) * E_{pk}(0) * E_{pk}(0) * E_{pk}(1) * E_{pk}(0) = E_{pk}(1)$. *Thus, the output of the* SF *to* P_1 *is* $\langle E_{pk}(3), E_{pk}(2), E_{pk}(1)\rangle$ □

Security Analysis: Let the execution image of the SF for P_2 is denoted by $\Pi_{P_2}(\text{SF})$, and is given as (according to Algorithm 7)

$$\Pi_{P_2}(\text{SF}) = \{Z_{i,j}, u_{i,j} \mid \text{for } 1 \leq j \leq w\}$$

where $u_{i,j}$ is derived upon decrypting $Z_{i,j}$ (at step 2(b) of Algorithm 7). Suppose that the simulated image of P_2 is denoted by $\Pi^S_{P_2}(\text{SF})$ which can be given by

$$\Pi^S_{P_2}(\text{SF}) = \{Z^*_{i,j}, u^*_{i,j} \mid \text{for } 1 \leq j \leq w\}$$

Here, $Z^*_{i,j}$ is randomly generated from $\mathbb{Z}_{N^2}$. Also, u^*_i is a vector generated at random such that exactly one of them is 0, and the rest are random numbers in $\mathbb{Z}_N$. Since E_{pk} is a semantically secure encryption scheme with a resulting ciphertext size that is less than N^2, $Z_{i,j}$ is computationally indistinguishable from $Z^*_{i,j}$. Also, since π_i is a random permutation function known only to P_1, u_i will be a vector with exactly one zero (at random location) and the rest are random numbers in $\mathbb{Z}_N$. Hence, u_i is computationally indistinguishable from u^*_i. Thus, it was concluded that $\Pi_{P_2}(\text{SF})$ was computationally indistinguishable from $\Pi^S_{P_2}(\text{SF})$.

Table 3.6: SF example: Vectors $S_{i,j}$ and $Z_{i,j}$, for $1 \leq i \leq k = 6$ and $1 \leq j \leq w = 3$

		c'_i					
	$S_{i,j}$	3	2	3	2	5	2
	2	-1	0	-1	0	-3	0
c_j	3	0	1	0	1	-2	1
	5	2	3	2	3	0	3

	$Z_{i,j}$	3	2	3	2	5	2
	5	0	$r_{2,2}$	0	$r_{2,4}$	$r_{2,5}$	$r_{2,6}$
c_j	1	$r_{3,1}$	$r_{3,2}$	$r_{3,3}$	$r_{3,4}$	0	$r_{3,6}$
	3	$r_{1,1}$	0	$r_{1,3}$	0	$r_{1,5}$	0

Table 3.7: SF example: Vector $V_{i,j}$, for $1 \leq i \leq k = 6$ and $1 \leq j \leq w = 3$

					c'_i		
	$V_{i,j}$	3	2	3	2	5	2
	2	$E_{pk}(0)$	$E_{pk}(1)$	$E_{pk}(0)$	$E_{pk}(1)$	$E_{pk}(0)$	$E_{pk}(1)$
c_j	3	$E_{pk}(1)$	$E_{pk}(0)$	$E_{pk}(1)$	$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(0)$
	5	$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(1)$	$E_{pk}(0)$

In contrast, let the execution image of P_1 is denoted by $\Pi_{P_1}(\text{SF})$, and is given by

$$\Pi_{P_1}(\text{SF}) = \{U_{i,j} \mid \text{for } 1 \leq i \leq k \text{ and } 1 \leq j \leq w\}$$

Here, $U_{i,j}$ is an encrypted value sent by P_2 at step 2(c) of Algorithm 7. Suppose that the simulated image of P_1 is given by

$$\Pi^S_{P_1}(\text{SF}) = \{U^*_{i,j} \mid \text{for } 1 \leq i \leq k \text{ and } 1 \leq j \leq w\}$$

where $U^*_{i,j}$ is a random number in $\mathbb{Z}_{N^2}$. Since E_{pk} is a semantically secure encryption scheme with a resulting ciphertext size of less than N^2, $U_{i,j}$ is computationally indistinguishable from $U^*_{i,j}$. As a result, $\Pi_{P_1}(\text{SF})$ is computationally indistinguishable from $\Pi^S_{P_1}(\text{SF})$. Combining all the above results, and following from Definition 1, it was concluded that the SF protocol was secure under the semi-honest model.

Complexity Analysis: This work can claim that the computation complexity of the SF protocol was bounded by $O(k * w)$ encryptions. This upper bound can be derived the same way that used in the previous sub-routines.

3.5.10. Secure Comparison with a Threshold. In the Secure Comparison with a Threshold (SCT) protocol, P_1 holds a private input (d') and P_2 holds a private input (d''). Here, d' and d'' are two random shares for d such that $d' + d'' \mod N = d$. Given a threshold t as a publicly known parameter, the goal of the SCT protocol is to securely evaluate the condition $d < t$. The output of this protocol returns two random shares (b' and b'') for b such that $b' + b'' \mod N = b$, where b denotes the comparison result. More specifically, $b = 0$ if $d < t$, otherwise, $b = 0$. Note that the random share b' is revealed only to P_1, whereas the random share b'' is revealed only to P_2.

The inputs received from most existing, secure, comparison protocols [9, 25, 26, 35, 71] are either non-encrypted or non-random shares. Thus, they are not directly applicable to this problem domain. In contrast, garbled circuit has been known for its efficiency in securely evaluating simple functionalities (e.g., comparison). Additionally, the garbled circuit requires only one round of communication. As a result, it can easily be modified to fit this problem domain. Therefore, Garbled Circuit Parser tool (GCParser) [65], an interpreter for garbled circuits intermediate language, was adopted in this study to implement the SCT protocol. Technical details on how to produce and evaluate a garbled circuit have been well documented in "Faster secure two-party computation using garbled circuits" [53].

Security Analysis: The SCT protocol was implemented using a garbled circuit, which was also secure under the semi-honest model [53].

Complexity Analysis: Inputs to the SCT protocol were two random shares with a size that bounded by N. Thus, the initial phase of the garbled circuit needed $O(\log N)$ gates to add the shares. The resulting distance value was much smaller than N in the problem domain. Therefore, the total number of gates in the garbled circuit was bound by $O(\log N)$. Since the fast garbled circuit evaluation method proposed in "Faster secure two-party computation using garbled circuits" [53] is very efficient and use a symmetric key encryption to garble the circuit. One can expect that the total computation cost of the SCT protocol is about performing several homomorphic encryption operations. Thus, $O(m)$ encryptions provided an appropriate upper bound for the SCT protocol.

3.6. OVERVIEW OF THE PROPOSED PRIVACY-PRESERVING QUERY PROCESSING PROTOCOLS

According to the number of participants, an SMC protocol can be classified into either two-party or multi-party protocol. In this work, it is more cost effective to use two cloud service providers, so one will focus on developing two-party Privacy-Preserving Query Processing (PPQP) protocol.

Let C_1 denote a cloud service provider who stores Alice's encrypted data (T') and performs query processing based on T' and a user query (q) on behalf of her. When outsourced data (T') are encrypted with a fully homomorphic encryption scheme [13, 14, 36, 38, 93], C_1 can perform any possible computations over the encrypted data. Fully homomorphic encryptions, however, have yet to be practical due to their extremely high computation costs. The research community is still looking for more efficient constructions of fully homomorphic encryptions. Therefore, in addition to C_1, this work will utilize another independent cloud service provider (C_2) and use an Additive Homomorphic Public Key Encryption (AH-Enc) scheme to encrypt each records in T and q component-wise to produce T' and $E_{pk}(q)$. There are several AH-Enc schemes, and without loss of generality, this work adopted the Paillier cryptosystem [75] for its simple implementation and being semantically secure. Both C_1 and C_2 are assumed to be semi-honest cloud service providers. Such an assumption is not new and has been commonly used in the recent related works (e.g., [15, 95]). The intuition behind such an assumption is as follows: Most of the cloud service providers in the market are legitimate, well-known IT companies (e.g., Google, Amazon, and Microsoft). These companies maintain reputations that are invaluable assets that need to be protected at all costs. Therefore, a collusion between them is highly unlikely as it will damage their reputations which in turn can affect their revenues and profits.

Under this setting, Alice outsources her encrypted database (T') to C_1, and the secret key (sk) to C_2. Here, it is possible for Alice to replace C_2 with her private server. However, if Alice has a private server, one can argue that there is no need for data outsourcing from Alice's point of view. The main purpose of using C_2 can be motivated by the following two reasons: (i) With limited computing resource and technical expertise, it is in the best interest of Alice to completely outsource her data management and operational tasks to a cloud. For example, Alice may want to access her data and analytical results using either a smart phone or any device with very limited computing capability. (*ii*) If Alice uses a private server, she has to perform computations assumed by C_2 under which the very purpose of outsourcing the encrypted data to C_1 is negated. Moreover, she might be able to track down Bob's queries and infer what he is looking for. Thus, Bob's query privacy and access patterns may be compromised while searching through hosted data within the cloud and Alice' private server.

In general, whether Alice uses either a private server or a cloud service provider (C_2) actually depends on her resources. This work preferred to use a multi-cloud computing framework as this would avoid the above-mentioned disadvantages (i.e., in case of Alice using a private server) altogether.

Note that Bob sends only his encrypted query and never participates in any intermediate computations. Thus, how the user behaves is irrelevant to the protocol's security.

4. k-NEAREST NEIGHBOR QUERY

Using encryption as a way to achieve data confidentiality may cause another issue during the query processing step in the cloud. In general, it is very difficult to process encrypted data without ever having to decrypt it. The question here is how the cloud can execute the queries over encrypted data while the data stored at the cloud are encrypted at all times. In the literature, various techniques related to query processing over encrypted data have been proposed, including range queries [49, 50, 91] and other aggregate queries [46, 68]. These techniques, however, are either not applicable or inefficient to solve advanced queries such as the kNN query.

In this chapter, the problem of secure processing of kNN (SkNN) query over encrypted relational data in a cloud was addressed [29]. That is, given a user's encrypted query record (q), the objective of the SkNN problem is to securely identify the top k-nearest (closest) records to q using the encrypted database (T') in the cloud without allowing the cloud to learn anything regarding either the actual contents of the database (T) or q. More specifically, an effective SkNN protocol needs to satisfy the following properties:

- Preserve the confidentiality of T and q at all times
- Hiding data access patterns from the cloud
- Accurately compute the k-nearest neighbors of query q
- Incur low computation overhead on the end-user

Various techniques related to solving the SkNN problem have been proposed [51, 98, 99, 105]. The existing SkNN methods violate, however, at least one of the above-mentioned desirable properties of a SkNN protocol. On one hand, the methods in [51, 98] are insecure because they are vulnerable to chosen and known plaintext attacks. On the other hand, the recent method in [99] returns a non-accurate kNN result to the end-user. More precisely, in [99], the cloud retrieves the relevant encrypted partition instead of finding the encrypted exact k-nearest neighbors. Furthermore, in [51, 99, 105], the end-user involves in heavy computations during the query processing step. By doing so, these methods utilize cloud as just a storage medium, i.e., no significant work is done on the cloud side. Additionally, the existing SkNN methods do not protect data access patterns from the cloud. More details regarding the existing SkNN methods were provided in Section 2.1.

This chapter presented first a basic scheme to solve SkNN problem and demonstrated that such a naive solution was not secure. To provide better security, a novel maximum SkNN protocol was also proposed that protected both the data confidentiality and the access patterns [29]. For ease of presentation, some common notations that are used extensively throughout this chapter are summarized in Table 4.1.

4.1. DEFINING THE PROBLEM

Suppose that Alice owns a database T of n records $t_1, \ldots, t_n$ and m attributes. Let $t_{i,j}$ denote the j^{th} attribute value of record t_i. Also, assume that Alice generates a pair of public-secret key pair (pk, sk) based on an Additive Homomorphic Public Key Encryption (AH-Enc) cryptosystem that is semantically secure (e.g., Paillier cryptosystem [75]). Initially, Alice encrypts her database (T). Let $T' = \{E_{pk}(t_1), \ldots, E_{pk}(t_n)\}$ denote the encryption of T, where each $E_{pk}(t_i)$ is encrypted component-wise. That is, she computes $E_{pk}(t_{i,j})$ for $1 \leq i \leq n$ and $1 \leq j \leq m$, where E_{pk} denotes the encryption function of a public-key cryptosystem that is semantically secure [75]. Assume that Alice outsources both T' and the future querying processing services to the cloud.

Consider Bob who wants to ask the cloud for k-neighbor records that are closest to his input query $q = \langle q_1, \ldots, q_m \rangle$ based on T', where m denotes the number of attributes. During this process, neither Bob's query (q) nor the database's contents (T) should be revealed to the cloud. Access patterns for the data should also be protected from the cloud. This process is referred to here as a Secure k-Nearest Neighbor (SkNN) query over encrypted relational data in the cloud [29]. Let $\{t'_1, \ldots, t'_k\}$ denote the k-nearest records to q. The SkNN protocol can now be formally defined.

$$\text{S}k\text{NN}\big(T', q\big) \rightarrow \{t'_1, \ldots, t'_k\}$$

At the end of the SkNN protocol, the output $\{t'_1, \ldots, t'_k\}$ should be revealed only to Bob. The following example presented a real-life application of the SkNN protocol.

Table 4.1: Common notations used in the SkNN protocols

Alice	The data owner of relational database T
Bob	An authorized user who can access T' in the cloud
C_1, C_2	Two non-colluding semi-honest cloud service providers
$\langle E_{pk}(.), D_{sk}(.)\rangle$	A pair of Paillier's encryption and decryption functions with (pk, sk) as public-secret key pair
$E_{pk}(x)$	Component-wise encryption of x: $E_{pk}(x_1), \ldots, E_{pk}(x_m)$
T	A relational database with n records: $t_1, \ldots, t_n$
T'	An encryption of T: $E_{pk}(t_1), \ldots, E_{pk}(t_n)$
n	Number of data records in T
m	Number of attributes in T
t_i	i^{th} record in T
q	Bob's input query record
t'_i	i^{th} nearest record to q based on T

Example 8. *Consider a physician who wants to know the risk factor of heart disease in a specific patient. Let T denote the sample heart disease dataset with attributes record-id, age, sex, cp, trestbps, chol, fbs, slope, ca, thal, and num as shown in Table 4.2. The description and range for each of these attributes are shown in Table 4.3. The heart disease dataset given in Table 4.2 is obtained from the UCI machine learning repository [56]. Initially, the data owner (hospital) encrypts T attribute-wise, outsources the encrypted database T' to the cloud for easy management. Additionally, the data owner delegates the future query processing services to the cloud. Now, we consider a doctor working at the hospital, say Bob, who would like to know the risk factor of heart disease in a specific patient based on T. Let the patient medical information be $q = \langle 58, 1, 4, 133, 196, 1, 2, 1, \ 6\rangle$. In the SkNN protocol, Bob first need to encrypt q (to preserve the privacy of his query) and send it to the cloud. The cloud then searches on the encrypted database T' to figure out the k-Nearest Neighbors to the user's request. For simplicity, let us assume $k = 2$. Under this case, the* 2*-nearest neighbors to q are t_4 and t_5 (by using Euclidean distance as the similarity metric). The cloud then sends both t_4 and t_5 (in encrypted form) to Bob. Here, the cloud should identify the nearest neighbors of q in an oblivious manner without knowing any sensitive information, i.e., all the computations have to be carried over encrypted records. Finally, Bob receives both t_4 and t_5 that will help him to make medical decisions.* □

Table 4.2: Sample heart disease dataset T

record-id	age	sex	cp	trestbps	chol	fbs	slope	ca	thal	num
t_1	63	1	1	145	233	1	3	0	6	0
t_2	56	1	3	130	256	1	2	1	6	2
t_3	57	0	3	140	241	0	2	0	7	1
t_4	59	1	4	144	200	1	2	2	6	3
t_5	55	0	4	128	205	0	2	1	7	3

4.2. MAIN CONTRIBUTIONS

A novel SkNN protocol to facilitate the kNN search over semantically encrypted relational data is proposed [29]. Withing this protocol, Alice does not participate in any computations once the encrypted data are outsourced to the cloud. Therefore, no information is revealed to Alice. This protocol also meets the desired requirements discussed in Section 1.2:

- Neither the contents of T nor any intermediate results should be revealed to the cloud.
- Bob's query (q) should not be revealed to the cloud.
- The output $\{t'_1, \ldots, t'_k\}$ should be computed accurately and revealed only to Bob. Additionally, no information other than $\{t'_1, \ldots, t'_k\}$ should be revealed to Bob.
- Incurs a low computational overhead on Bob after the encrypted query record is sent to the cloud.
- Access patterns for the data (e.g., the records corresponding to the k-Nearest Neighbors of q) should not be revealed to either Alice or the cloud to prevent any inference attacks.

The intermediate results observed by the clouds in this protocol are either newly generated, randomized encryptions or random numbers. Thus, the data records correspond to the k-Nearest Neighbors of q are unknown to the cloud. Also, after Bob sends his encrypted query record to the cloud, he stops involving in computations. Hence, data access patterns are further protected from Bob. More details are given in Section 4.3.2.

Table 4.3: Attribute description of heart disease dataset T

sex	1=male, 0=female
cp	chest pain type: 1=typical angina, 2=atypical angina, 3=non-anginal pain, 4=asymptomatic
trestbps	resting blood pressure (mm Hg)
chol	serum cholesterol in mg/dl
fbs	fasting blood sugar > 120 mg/dl (1=true; 0=false)
slope	slope of the peak exercise ST segment (1=upsloping, 2=flat, 3=downsloping)
ca	number of major vessels (0-3) colored by fluoroscopy
thal	3=normal, 6=fixed defect, 7=reversible defect
num	diagnosis of heart disease from 0 (no presence) to 4

4.3. THE PROPOSED SECURE k-NEAREST NEIGHBOR PROTOCOLS

A basic version of the SkNN protocol is first presented and demonstrated why such a simple solution was not secure. Then, the second approach that achieved better security under the semi-honest model is presented. Both protocols were constructed based on the set of sub-components presented in Section 3.5 as building blocks.

In the SkNN protocols, the existence of two non-colluding semi-honest cloud service providers, denoted by C_1 and C_2, which together form a federated cloud was assumed. Such an assumption is not new and has been commonly used in the related problem domains [16, 96]. More details are given in Section 3.6.

Recall that Alice's database consists of n records, denoted by $T = \{t_1, \ldots, t_n\}$, and m attributes, where $t_{i,j}$ denotes the j^{th} attribute value of record t_i. Initially, Alice encrypts her database attribute-wise using here public key pk. Let the encrypted database be denoted by T'. Assume that all attribute values and their Euclidean distances lie in $[0, 2^l)$.

Under this setting, Alice outsources her encrypted database (T') to C_1, and the secret key (sk) to C_2. The goal of the proposed SkNN protocols is to retrieve the top k records that are closest to a user's query in an efficient, secure manner. Briefly, consider Bob who wants to find k records that are closest to his query record $q = \langle q_1, \ldots, q_m \rangle$ based on T' in C_1. Bob initially sends his query q (in encrypted form) to C_1. After this, both C_1 and C_2 involve in a set of sub-protocols to securely retrieve (in encrypted form) the set of k records corresponding to the k-nearest neighbors of the input query q. At the end of the proposed protocols, only Bob will receive the k-Nearest Neighbors to q as the output.

4.3.1. Basic Secure k-Nearest Neighbor Protocol. In the basic protocol, denoted by SkNN$_b$, the desirable properties discussed in Section 1.2 are relaxed to produce an efficient protocol (More details on this are given in the latter part of this section). The overall steps involved in the SkNN$_b$ protocol are presented in Algorithm 8. Initially, Bob encrypts his query (q) attribute-wise, that is, he computes $E_{pk}(q) = \big\langle E_{pk}(q_1), \ldots, E_{pk}(q_m)\big\rangle$. He then sends $E_{pk}(q)$ to C_1. Upon receiving $E_{pk}(q)$ from Bob, both C_1 with private input $\big(E_{pk}(q), E_{pk}(t_i)\big)$, and C_2 with the secret key sk jointly involve in the Secure Squared Euclidean Distance (SSED) protocol, where $E_{pk}(t_i) = \big\langle E_{pk}(t_{i,1}), \ldots, E_{pk}(t_{i,m})\big\rangle$ for $1 \leq i \leq n$. The output of this step, denoted by $E_{pk}(d_i)$, is the encryption of squared Euclidean distance between q and t_i, i.e., $d_i = |q - t_i|^2$. Recall that, $E_{pk}(d_i)$ is known only to C_1 for $1 \leq i \leq n$. Note that computation of the exact Euclidean distance between encrypted vectors is hard to achieve as it involves square root. In the kNN, however, it is sufficient to compare the squared Euclidean distances as it preserves relative ordering. After this, C_1 sends $\Big\{\big\langle 1, E_{pk}(d_1)\big\rangle, \ldots, \big\langle n, E_{pk}(d_n)\big\rangle\Big\}$ to C_2, where the entry $\big\langle i, E_{pk}(d_i)\big\rangle$ corresponds to the data record t_i for $1 \leq i \leq n$. Upon receiving $\big\langle 1, E_{pk}(d_1)\big\rangle, \ldots, \big\langle n, E_{pk}(d_n)\big\rangle$, C_2 decrypts the encrypted distance in each entry to obtain $d_i = D_{sk}\big(E_{pk}(d_i)\big)$. C_2 then generates an index list $\delta = \langle i_1, \ldots, i_k\rangle$ such that $\langle d_{i_1}, \ldots\ d_{i_k}\rangle$ are the top k smallest distances among $\langle d_1, \ldots, d_n\rangle$. After this, C_2 sends δ to C_1. Upon receiving δ, C_1 proceeds as follows:

- Select the encrypted records $E_{pk}(t_{i_1}), \ldots, E_{pk}(t_{i_k})$ as the k-nearest records to q and randomize them attribute-wise. More specifically, C_1 computes $E_{pk}(\gamma_{j,h}) = E_{pk}(t_{i_j,h}) * E_{pk}(r_{j,h})$ for $1 \leq j \leq k$ and $1 \leq h \leq m$. Here, $r_{j,h}$ is a random number in $\mathbb{Z}_N$ and $t_{i_j,h}$ denotes the column h attribute value of the data record t_{i_j}. C_1 then send $\gamma_{j,h}$ to C_2, and $r_{j,h}$ to Bob for $1 \leq j \leq k$ and $1 \leq h \leq m$.

Upon receiving $\gamma_{j,h}$, for $1 \leq j \leq k$ and $1 \leq h \leq m$, C_2 decrypts it to obtain $\gamma'_{j,h} = D_{sk}(\gamma_{j,h})$ and sends them to Bob. Note that due to randomization by C_1, $\gamma'_{j,h}$ is always a random number in $\mathbb{Z}_N$.

Algorithm 8 SkNN$_b\big(T', q\big) \rightarrow \langle t'_1, \ldots, t'_k\rangle$

```
Require: $C_1$ has $T'$; $C_2$ has $sk$; Bob has $q$
Bob:
    (a). Compute $E_{pk}(q_j)$, for $1 \leq j \leq m$
    (b). Send $E_{pk}(q) = \big\langle E_{pk}(q_1), \ldots, E_{pk}(q_m)\big\rangle$ to $C_1$
$C_1$ and $C_2$:
    (a). $C_1$ receives $E_{pk}(q)$ from Bob
    (b). for $i = 1$ to $n$ do:
        • $E_{pk}(d_i) \leftarrow \mathrm{SSED}\big(E_{pk}(q), E_{pk}(t_i)\big)$
    (c). Send $\Big\{\big\langle 1, E_{pk}(d_1)\big\rangle, \ldots, \big\langle n, E_{pk}(d_n)\big\rangle\Big\}$ to $C_2$
$C_2$:
    (a). Receive $\Big\{\big\langle 1, E_{pk}(d_1)\big\rangle, \ldots, \big\langle n, E_{pk}(d_n)\big\rangle\Big\}$ from $C_1$
    (b). $d_i \leftarrow D_{sk}\big(E_{pk}(d_i)\big)$, for $1 \leq i \leq n$
    (c). Generate $\delta \leftarrow \langle i_1, \ldots, i_k\rangle$, such that $\langle d_{i_1}, \ldots, d_{i_k}\rangle$ are the top $k$ smallest distances
         among $\langle d_1, \ldots, d_n\rangle$
    (d). Send $\delta$ to $C_1$
$C_1$:
    (a). Receive $\delta$ from $C_2$
    (b). for $1 \leq j \leq k$ and $1 \leq h \leq m$ do:
        • $\gamma_{j,h} \leftarrow E_{pk}(t_{i_j,h}) * E_{pk}(r_{j,h})$, where $r_{j,h} \in_R \mathbb{Z}_N$
        • Send $\gamma_{j,h}$ to $C_2$ and $r_{j,h}$ to Bob
$C_2$:
    (a). for $1 \leq j \leq k$ and $1 \leq h \leq m$ do:
        • Receive $\gamma_{j,h}$ from $C_1$
        • $\gamma'_{j,h} \leftarrow D_{sk}(\gamma_{j,h})$; send $\gamma'_{j,h}$ to Bob
Bob:
    (a). for $1 \leq j \leq k$ and $1 \leq h \leq m$ do:
        • Receive $r_{j,h}$ from $C_1$ and $\gamma'_{j,h}$ from $C_2$
        • $t'_{j,h} \leftarrow \gamma'_{j,h} - r_{j,h} \mod N$
```

Finally, upon receiving $r_{j,h}$ from C_1 and $\gamma'_{j,h}$ from C_2, Bob computes the attribute values of j^{th} nearest neighbor to q as $t'_{j,h} = \gamma'_{j,h} - r_{j,h} \mod N$ for $1 \leq j \leq k$ and $1 \leq h \leq m$.

4.3.2. Maximally Secure k-Nearest Neighbor Protocol. The Basic Secure k-Nearest Neighbor (SkNN$_b$) protocol completely protected both the data confidentiality and the user's query privacy because the computations were performed on either encrypted data or randomized data. It did reveal, however, the data access patterns to C_1 and C_2. That is, both C_1 and C_2 knew which data records correspond to the k-Nearest Neighbors for any given q. It also did reveal d_i values to C_2. Leakage of such information, however, may not be acceptable in privacy-sensitive applications (e.g., medical data).

Along with this direction, this work proposed a Maximally Secure k-Nearest Neighbor (SkNN$_m$) protocol under the semi-honest model, where m stands for maximally secure. The desirable properties discussed in Section 1.2 are completely preserved in the SkNN$_m$ protocol.

The overall steps involved in the proposed SkNN$_m$ protocol are presented in Algorithm 10. Bob initially sends his attribute-wise encrypted query q, that is, $E_{pk}(q) = \big\langle E_{pk}(q_1), \ldots, E_{pk}(q_m)\big\rangle$ to C_1. Upon receiving $E_{pk}(q)$, both C_1 with input $\big(E_{pk}(q), E_{pk}(t_i)\big)$ and C_2 with the secret key sk jointly involve in the SSED protocol to compute $E_{pk}\big(|q-t_i|^2\big)$ for $1 \leq i \leq n$. The output $E_{pk}(d_i) = E_{pk}\big(|q-t_i|^2\big)$ will be known only to C_1 for $1 \leq i \leq n$. Using the Secure Bit-Decomposition (SBD) protocol, both C_1 with input $E_{pk}(d_i)$, and C_2 with sk securely compute $[d_i]$, the encryptions of the individual bits of d_i, for $1 \leq i \leq n$.

The output $[d_i] = \big\langle E_{pk}(d_{i,1}), \ldots, E_{pk}(d_{i,l})\big\rangle$ for $1 \leq i \leq n$ will be known only to C_1, where $d_{i,1}$ and $d_{i,l}$ are the most and least significant bits of d_i, respectively. Note that $0 \leq d_i < 2^l$ for $1 \leq i \leq n$.

After this, both C_1 and C_2 compute the top k (in encrypted form) records that are closest to q in an iterative manner. More specifically, they compute $E_{pk}(t'_1)$ in the first iteration, $E_{pk}(t'_2)$ in the second iteration, and so on. Here, t'_s denote the s^{th} nearest neighbor to q for $1 \leq s \leq k$. At the end of k iterations, only C_1 knows $\big\langle E_{pk}(t'_1), \ldots, E_{pk}(t'_k)\big\rangle$. To start with, in the first iteration, both C_1 and C_2 jointly compute the encryptions of the individual bits of the minimum value among $d_1, \ldots, d_n$ using the Secure Minimum Out of n Numbers (SMIN$_n$) protocol. That is, C_1, with input $(\theta_1, \ldots, \theta_n)$, and C_2, with sk compute $\big([d_{\min}], nil\big)$, where $\theta_i = \big([d_i], nil\big)$ for $1 \leq i \leq n$. Here, $d_{\min}$ denotes the minimum value

Algorithm 9 SkNN$_m\big(T', q\big) \rightarrow \big\langle t'_1, \ldots, t'_k \big\rangle$

```
Require: $C_1$ has $T'$ and $\pi$; $C_2$ has $sk$; Bob has $q$
1: Bob sends $E_{pk}(q) = \big\langle E_{pk}(q_1), \ldots, E_{pk}(q_m) \big\rangle$ to $C_1$
2: $C_1$ and $C_2$:
    (a). $C_1$ receives $E_{pk}(q)$ from Bob
    (b). for $i = 1$ to $n$ do:
        • $E_{pk}(d_i) \leftarrow \mathrm{SSED}\big(E_{pk}(q), E_{pk}(t_i)\big)$
        • $[d_i] \leftarrow \mathrm{SBD}\big(E_{pk}(d_i)\big)$
3: for $s = 1$ to $k$ do:
    (a). $C_1$ and $C_2$:
        • $[d_{\min}], nil \leftarrow \mathrm{SMIN}_n(\theta_1, \ldots, \theta_n)$, where $\theta_i = \big([d_i], nil\big)$
    (b). $C_1$:
        • $E_{pk}(d_{\min}) \leftarrow \prod_{\gamma=0}^{l-1} E_{pk}(d_{\min,\gamma+1})^{2^{l-\gamma-1}}$
        • if $s \neq 1$ then, for $1 \leq i \leq n$
            – $E_{pk}(d_i) \leftarrow \prod_{\gamma=0}^{l-1} E_{pk}(d_{i,\gamma+1})^{2^{l-\gamma-1}}$
        • for $i = 1$ to $n$ do:
            – $\tau_i \leftarrow E_{pk}(d_{\min}) * E_{pk}(d_i)^{N-1}$
            – $\tau'_i \leftarrow \tau_i^{r_i}$, where $r_i \in_R \mathbb{Z}_N$
        • $\beta \leftarrow \pi(\tau')$; send $\beta$ to $C_2$
    (c). $C_2$:
        • Receive $\beta$ from $C_1$
        • $\beta'_i \leftarrow D_{sk}(\beta_i)$, for $1 \leq i \leq n$
        • Compute $U$, for $1 \leq i \leq n$:
            – if $\beta'_i = 0$ then $U_i = E_{pk}(1)$
            – else $U_i = E_{pk}(0)$
        • Send $U$ to $C_1$
    (d). $C_1$:
        • Receive $U$ from $C_2$ and compute $V \leftarrow \pi^{-1}(U)$
        • $V'_{i,j} \leftarrow \mathrm{SM}\big(V_i, E_{pk}(t_{i,j})\big)$, for $1 \leq i \leq n$ and $1 \leq j \leq m$
        • $E_{pk}(t'_{s,j}) \leftarrow \prod_{i=1}^{n} V'_{i,j}$, for $1 \leq j \leq m$
        • $E_{pk}(t'_s) = \big\langle E_{pk}(t'_{s,1}), \ldots, E_{pk}(t'_{s,m}) \big\rangle$
    (e). $C_1$ and $C_2$, for $1 \leq i \leq n$:
        • $E_{pk}(d_{i,\gamma}) \leftarrow \mathrm{SBOR}\big(V_i, E_{pk}(d_{i,\gamma})\big)$, for $1 \leq \gamma \leq l$
The rest of the steps are similar to steps 4-6 of S$k$NN$_b$
```

among $d_1, \ldots, d_n$. In this chapter, during the process of the SMIN$_n$ protocol, the secret information associated with each tuple does not involve in the computation. Therefore, the corresponding parameter for the secret information in the SMIN$_n$ protocol is set to *nil*. The output $\big([d_{\min}], nil\big)$ is known only to C_1. C_1 then follows several steps:

- Compute the encryption of $d_{\min}$ from its encrypted individual bits as follows:

$$\begin{aligned} E_{pk}(d_{\min}) &= \prod_{\gamma=0}^{l-1} E_{pk}(d_{\min,\gamma+1})^{2^{l-\gamma-1}} \\ &= E_{pk}(d_{\min,1} * 2^{l-1} + \cdots + d_{\min,l}) \end{aligned}$$

  Where $d_{\min,1}$ and $d_{\min,l}$ are the most and least significant bits of $d_{\min}$, respectively.

- Compute the encryption of difference between $d_{\min}$ and each d_i. That is, C_1 computes $\tau_i = E_{pk}(d_{\min}) * E_{pk}(d_i)^{N-1} = E_{pk}(d_{\min} - d_i)$ for $1 \leq i \leq n$.

- Randomize τ_i to obtain $\tau_i' = \tau_i^{r_i} = E_{pk}\big(r_i * (d_{\min} - d_i)\big)$, where r_i is a random number in $\mathbb{Z}_N$. Note that τ_i' is an encryption of either 0 or a random number for $1 \leq i \leq n$. Next, use a random permutation function π (known only to C_1) to permute τ' such that $\beta = \pi(\tau')$. Send β to C_2.

Upon receiving β, C_2 decrypts it component-wise to obtain $\beta_i' = D_{sk}(\beta_i)$ for $1 \leq i \leq n$. It then computes an encrypted vector U of length n. If $\beta_i' = 0$, then $U_i = E_{pk}(1)$; otherwise, $U_i = E_{pk}(0)$. It is assumed here that exactly one of the entries in β equals to zero and the remaining entries are random. This further implies that exactly one of the entries in U is an encryption of 1 and the remaining entries are encryption of 0's. However, if β' has more than one 0's, then C_2 can randomly pick one of those indexes and set $E_{pk}(1)$ to the corresponding index of U, and $E_{pk}(0)$ to the rest. C_2 then sends U to C_1. After receiving U, C_1 performs inverse permutation on it to obtain $V = \pi^{-1}(U)$. Note that exactly one of the entries in V is $E_{pk}(1)$ and the remaining entries are encryption of 0's. Additionally, if $V_i = E_{pk}(1)$, then t_i is the closest record to q. Both C_1 and C_2, however, do not know which entry in V is corresponding to $E_{pk}(1)$.

C_1 computes $E_{pk}(t_1')$, the encryption of the closest record to q, and updates the distance vectors as follows:

- Both C_1 and C_2 jointly involve in the Secure Multiplication (SM) protocol to compute $V'_{i,j} = V_i * E_{pk}(t_{i,j})$ for $1 \leq i \leq n$ and $1 \leq j \leq m$. The output V' from the SM protocol is known only to C_1. After this, by using homomorphic properties, C_1 computes the encrypted record $E_{pk}(t'_1) = \big\langle E_{pk}(t_{1,1}), \ldots, E_{pk}(t_{1,m})\big\rangle$ locally, $E_{pk}(t'_{1,j}) = \prod_{i=1}^{n} V'_{i,j}$, where $1 \leq j \leq m$. Note that, $t'_{1,j}$ denotes the j^{th} attribute value of record t'_1.

- It is important to note that the first nearest record to q should be obliviously excluded from further computations. However, because C_1 does not know the record corresponding to $E_{pk}(t'_1)$, one need to obliviously eliminate the possibility of choosing this record again in next iterations. For this, C_1 obliviously updates the distance corresponding to $E_{pk}(t'_1)$ to the maximum value, i.e., $2^l - 1$. More specifically, C_1 updates the distance vectors with the help of C_2 using the Secure Bit-OR (SBOR) protocol as following for $1 \leq i \leq n$ and $1 \leq \gamma \leq l$.

$$E_{pk}(d_{i,\gamma}) = \text{SBOR}\big(V_i, E_{pk}(d_{i,\gamma})\big)$$

  Note that, when $V_i = E_{pk}(1)$, the corresponding distance vector d_i is set to the maximum value. That is, under this case, $[d_i] = \big\langle E_{pk}(1), \ldots, E_{pk}(1)\big\rangle$. However, when $V_i = E_{pk}(0)$, the OR operation has no effect on d_i.

The above process is repeated for k iterations. In each iteration, $[d_i]$, corresponding to the currently chosen record, is set to the maximum value. However, because C_1 and C_2 do not know which $[d_i]$ is updated, they have to re-compute $E_{pk}(d_i)$ in each iteration for $1 \leq i \leq n$. In iteration s, $E_{pk}(t'_s)$ is known only to C_1. At the end of the iterative step (Step 3 of Algorithm 10), C_1 has $\big\{E_{pk}(t'_1), \ldots, E_{pk}(t'_k)\big\}$, the list of encrypted records of k-Nearest Neighbors to q.

The rest of the process is similar to steps 4 to 6 of Algorithm 8. Briefly, C_1 randomizes $E_{pk}(t'_j)$ attribute-wise to obtain $\gamma_{j,h} = E_{pk}(t'_{j,h}) * E_{pk}(r_{j,h})$. It sends $\gamma_{j,h}$ to C_2 and $r_{j,h}$ to Bob for $1 \leq j \leq k$ and $1 \leq h \leq m$. Here, $r_{j,h}$ is a random number in $\mathbb{Z}_N$. Upon receiving $\gamma_{j,h}$'s, C_2 decrypts them to obtain the randomized k-nearest records as $\gamma'_{j,h} = D_{sk}(\gamma_{j,h})$ and sends them to Bob for $1 \leq j \leq k$ and $1 \leq h \leq m$. Finally, upon receiving $r_{j,h}$ from C_1 and $\gamma'_{j,h}$ from C_2, Bob computes the j^{th} nearest neighboring record to q, as $t'_{j,h} = \gamma'_{j,h} - r_{j,h} \bmod N$ for $1 \leq j \leq k$ and $1 \leq h \leq m$.

4.4. SECURITY ANALYSIS

First, due to the encryption of q and by semantic security of the Paillier cryptosystem, Bob's input query q was protected from Alice, C_1, and C_2 in both protocols. In the SkNN$_b$ protocol, the decryption operations at step 3(b) of Algorithm 8 did reveal d_i values to C_2. Additionally, since C_2 generates the top k index list (at step 3(c) of Algorithm 8) and sends it to C_1, the data access patterns are revealed to both C_1 and C_2. Therefore, the SkNN$_b$ protocol was secure under the assumption that d_i values can be revealed to C_2 and data access patterns can be revealed to both C_1 and C_2.

In contrast, the security analysis of the SkNN$_m$ protocol is as follows. At step 2 of Algorithm 10, the outputs of the SSED and SBD protocols were in encrypted format and were known only to C_1. Additionally, all the intermediate results decrypted by C_2 in the SSED protocol were uniformly random in $\mathbb{Z}_N$. Also, as discussed in [86], the SBD protocol was secure. Thus, no information was revealed during step 2 of Algorithm 10. In each iteration, the output of the SMIN$_n$ protocol was known only to C_1 and no information was revealed to C_2. Also, both C_1 and C_2 did not know which record belongs to the current global minimum. Thus, data access patterns were protected from both C_1 and C_2. At step 3(c) of Algorithm 10, a component-wise decryption of β did reveal the tuples that satisfy the current global minimum distance to C_2. Due to permutation by C_1, however, C_2 could not trace back to the corresponding data records. Note that, the decryption of β gave either encryptions of 0's or random numbers in $\mathbb{Z}_N$. Similarly, because U is an encrypted vector, C_1 could not know which record that corresponds to the current global minimum distance. Thus, the data access patterns were further protected at this step from C_1. Additionally, the update process at step 3(e) of Algorithm 10 did not leak any information to either C_1 or C_2. In summary, both C_1 and C_2 did not know which data records that correspond to the output set $\langle t'_1, \ldots, t'_k \rangle$.

Following from the previous discussions, it was clear that the SkNN$_m$ protocol protected the data confidentiality, the privacy of a user's input query, and hide the data access patterns from C_1 and C_2. Also, since all these sub-protocols of the SkNN$_m$ protocol produced pseudo-random values as intermediate results, according to the Composition Theorem [41] given in Definition 2, the SkNN$_m$ protocol is also secure under the semi-honest model.

4.5. COMPLEXITY ANALYSIS

The proposed SkNN$_b$ and SkNN$_m$ protocols' computation complexity were analyzed under the assumption that encryption and decryption operations based on Paillier cryptosystem take a similar amount of time, an exponentiation operation was treated as an encryption operation, and an encryption operation is generally several orders of magnitude more expensive than a multiplication.

The computation complexity of the SkNN$_b$ protocol was bounded by $O(n * m + k)$ encryptions. In practice, $k \ll n * m$; therefore, the computation complexity of the SkNN$_b$ protocol could be bounded by $O(n * m)$ encryptions.

In contrast, the computation complexity of the SkNN$_m$ protocol was bounded by $O(n)$ instantiations of the SBD and the SSED protocols, $O(k)$ instantiations of the SMIN$_n$ protocol , and $O(n * l)$ instantiations of the SBOR protocol. Note that, the computation complexity of the SBD protocol proposed in [86] was bounded by $O(l)$ encryptions. Also, the computation complexity of the SSED protocol was bounded by $O(m)$ encryptions. In addition, the computation complexity of the SMIN$_n$ protocol was bounded by $O\big((l * n * \log_2 n)\big)$ encryptions. Since the SBOR protocol utilized the SM protocol as a sub-routine, the computation cost of the SBOR protocol was bounded by (small) constant number of encryptions. Based on the previous analysis, the total computation complexity of the SkNN$_m$ protocol was bounded by $O\big(n * (l + m + k * l * \log_2 n)\big)$ encryptions. More details regarding the complexity analysis of sub-protocols are given in Section 3.5.

Depending on the encryption key size, the overall computation cost of the proposed SkNN$_m$ protocol (more expensive than SkNN$_b$) is between 2 and 3 orders of magnitude higher than the non-crypto cases. This is the cost we need to pay to maximize data confidentiality. However, on the user or client side, the running time is comparable to the non-crypto case since the user only performs a very small number of encryption operations (bounded by the number of attributes) which was done in less than a second as will be shown in the experiments. The goal here is to outsource all or most computations to the cloud so that the user can issue queries using any mobile device with limited storage and computing capability. Note that data confidentiality is fully protected under the SkNN$_m$ protocol.

4.6. PERFORMANCE EVALUATION

The proposed protocols were implemented using Paillier cryptosystem [75] in C language on top of the GNU multiple precision arithmetic library (http://gmblib.org/). Their performance was also measured across a range of inputs.

Synthetic datasets were randomly generated according to the parameter values being considered because it is difficult to control the parameters in a real dataset. The advantage of using these synthetic datasets is that a more elaborated analysis could be performed on the computation costs of the proposed protocols under different parameter settings. These datasets were encrypted attribute-wise by the Paillier cryptosystem [75] with varied key sizes. The encrypted data were stored on a local machine. A Linux machine with an Intel® Xeon® Six-Core$^{\mathrm{TM}}$ CPU 3.07 GHz processor and 12GB RAM running Ubuntu 10.04 LTS was used to conduct all the experiments performed. The proposed protocols were performed according to these dataset. Then, a random query was chosen and executed over the encrypted data based on the proposed protocol.

For the rest of this section, the performance of Alice was not discussed as she was a one-time cost, and she did not participate in computations. Instead, the evaluations were based on the performance of both SkNN$_b$ and SkNN$_m$ protocols separately. Additionally, the computation costs of the two proposed protocols were compared. In the experiments, the Paillier encryption key size K was either 512 or 1024 bits.

4.6.1. Performance of the Basic Secure k-Nearest Neighbor Protocol. The computation costs of the SkNN$_b$ protocol were evaluated by varying the number of data records (n), number of attributes (m), number of nearest neighbors (k), and encryption key size (K). Note that the SkNN$_b$ protocol is independent of the domain size of attributes (l).

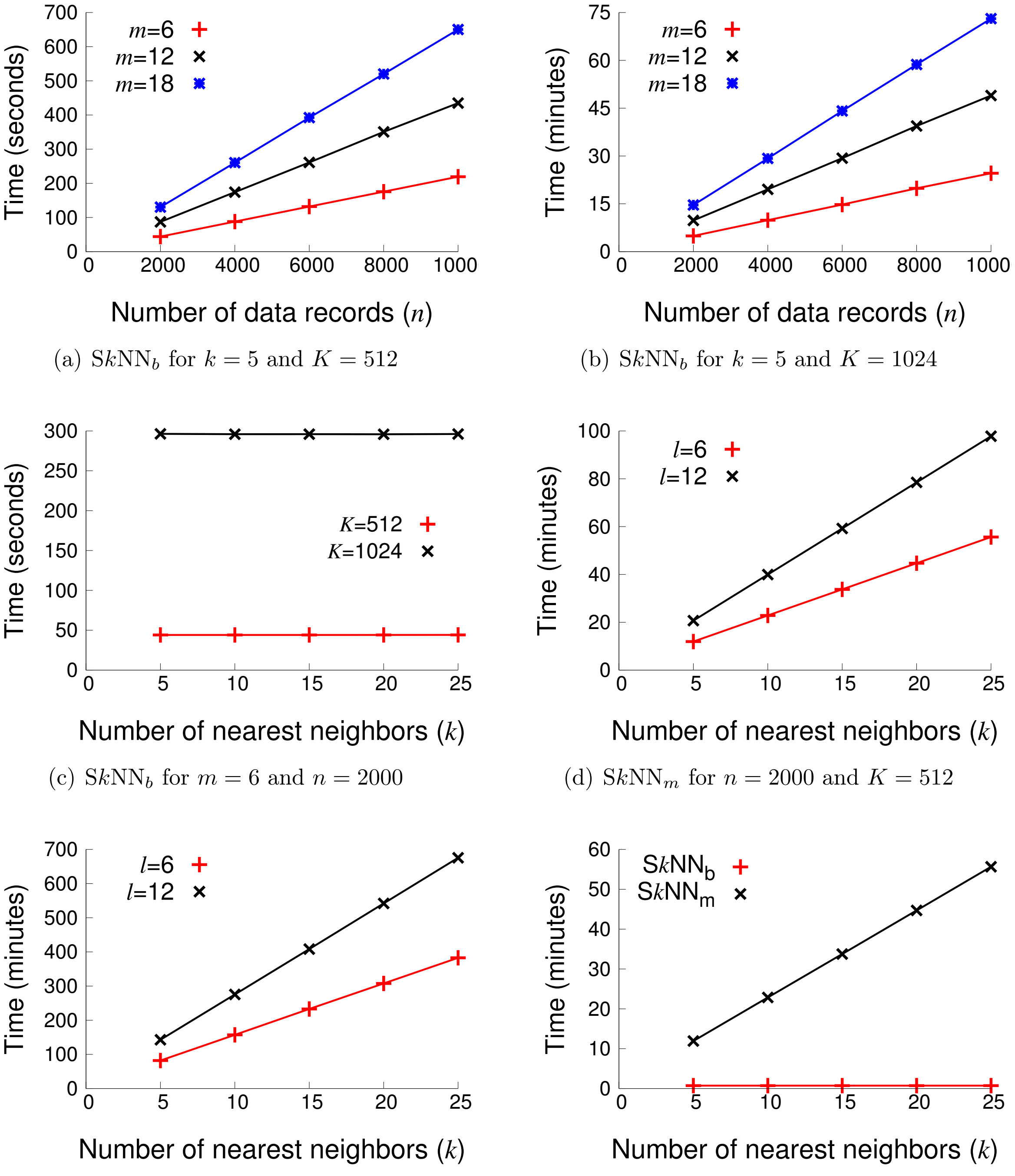
The computation costs of the SkNN$_b$ protocol were evaluated first by fixing $k = 5$ and $K = 512$ and varying values of n and m. The computation cost grew linearly with n and m (see Figure 4.1(a)). For example, when $m = 6$, the computation time of the SkNN$_b$ protocol increased from 44.08 to 87.91 seconds when n changed from 2000 to 4000. A similar trend could be observed for $K = 1024$ (see Figure 4.1(b)). For any fixed parameters, the computation time of the SkNN$_b$ increased by almost a factor of 7 whenever K doubled.

The computation costs of the SkNN$_b$ protocol were analyzed next by fixing $m = 6$ and $n = 2000$ and varying values of k and K (see Figure 4.1(c)). Irrespective of K, the computation time of the SkNN$_b$ protocol did not change much with varying k. This is because the computation time for the SkNN$_b$ protocol was dominated by the SSED protocol which was independent of k. For example, when $K = 512$ bits, the computation time of the SkNN$_b$ protocol changed from 44.08 to 44.14 seconds when k changed from 5 to 25. The computation costs of the SkNN$_b$ protocol mainly depended on (or grew linearly with) n and m. This finding is consistent with the upper bound derived in Section 4.5.

4.6.2. Performance of the Maximally Secure k-Nearest Neighbor Protocol.

The computation costs of the SkNN$_m$ protocol were analyzed by varying values of k, l, and K. Throughout this evaluation, the values of m and n were fixed to 6 and 2000, respectively. However, the running time of the SkNN$_m$ protocol grew almost linearly with n and m. For $K = 512$ bits, the computation costs of the SkNN$_m$ protocol for varying k and l are given in Figure 4.1(d). For $l = 6$, the running time of the SkNN$_m$ protocol varied from 11.93 to 55.65 minutes when k was changed from 5 to 25, respectively. Also, for $l = 12$, the running time of the SkNN$_m$ protocol varied from 20.68 to 97.8 minutes when k varied from 5 to 25, respectively. In either case, the cost of the SkNN$_m$ protocol grew almost linearly with k and l. A similar trend can be observed for $K = 1024$ (see Figure 4.1(e)). In particular, for any given fixed parameters, the computation cost of the SkNN$_m$ protocol increased by almost a factor of 7 when K is doubled. For example, when $k = 10$, the SkNN$_m$ protocol took 22.85 and 157.17 minutes to generate the 10 nearest neighbors of q under $K = 512$ and K=1024 bits, respectively. Furthermore, when $k = 5$, around 69.7% of the cost in SkNN$_m$ is accounted due to the SMIN$_n$ protocol which is initiated k times in SkNN$_m$ (once in each iteration). Also, the cost incurred due to the SMIN$_n$ protocol increased from 69.7% to, at least, 75% when k was increased from 5 to 25.

Additionally, the running times of both protocols were compared by fixing $n = 2000$, $m = 6$, $l = 6$, and $K = 512$ and varying values of k. The running time of the SkNN$_b$ protocol remained to be constant at 0.73 minutes because it was almost independent of k. The running time of the SkNN$_m$ protocol , however, changed from 11.93 to 55.65 minutes when k increased from 5 to 25 (see Figure 4.1(f)).



(a) SkNN$_b$ for $k = 5$ and $K = 512$

(b) SkNN$_b$ for $k = 5$ and $K = 1024$

(c) SkNN$_b$ for $m = 6$ and $n = 2000$

(d) SkNN$_m$ for $n = 2000$ and $K = 512$

(e) SkNN$_m$ for $n = 2000$ and $K = 1024$

(f) Time Complexity: SkNN$_b$ vs. SkNN$_m$

Figure 4.1: Time complexities of both SkNN$_b$ and SkNN$_b$ for varying values of n, m, l, k, and encryption key size K

Putting the above results together, it was concluded that the computation costs of the the SkNN$_m$ were significantly higher than that of the SkNN$_b$ protocol. However, the SkNN$_m$ protocol is more secure than the SkNN$_b$ protocol; therefore, the two protocols acts as a trade-off between security and efficiency. Note that Bob's computation cost is mainly due to the encryption of his input query record. As an example, for $m = 6$, Bob's computation costs are 4 and 17 milliseconds when K is 512 and 1024 bits, respectively. In the proposed protocols, it is worth pointing out that users do not involve in any computations. Therefore, the proposed protocols are very efficient from the end-user's perspective.

4.6.3. Performance Improvement. At first, it seems that the proposed protocols are costly and may not scale well for large datasets. However, in both protocols, the computations involved in each data record are independent of others. Therefore, the operations on data records can be parallelized for efficiency purpose. To empirically evaluate this claim, a parallel version of the SkNN$_b$ protocol was implemented using OpenMP programming [24] and compared its computation costs with its serial version. Recall that the machine used in experiments had 6 cores which could be used to perform parallel operations on 6 threads. For $m = 6$, $k = 5$, and $K = 512$ bits, the comparison results are given in Figure 4.2.

The parallel version of the SkNN$_b$ protocol was roughly 6 times more efficient than its serial version because of the fact that the parallel version could execute operations on 6 data records at a time (i.e., on 6 threads in parallel). For example, the running times of parallel and serial versions of the SkNN$_b$ protocol for $n = 10000$ were 40 and 215.59 seconds, respectively. Similar efficiency gains could be achieved by parallelizing the operations in the SkNN$_m$ protocol.

Based on the above discussions, it was concluded that the scalability issue of the proposed protocols can be eliminated or mitigated especially in a cloud computing environment, where high-performance parallel processing can easily be achieved. Additionally, using the existing map-reduce techniques, one can drastically improve the performance further by executing parallel operations on multiple nodes. Following from the previous empirical analysis, it is clear that SMIN$_n$ protocol is the most costly sub-routine utilized in the SkNN$_m$ protocol. Therefore, by improving the efficiency of the SMIN$_n$ protocol, that can improve the overall computation cost of the SkNN$_m$ protocol.

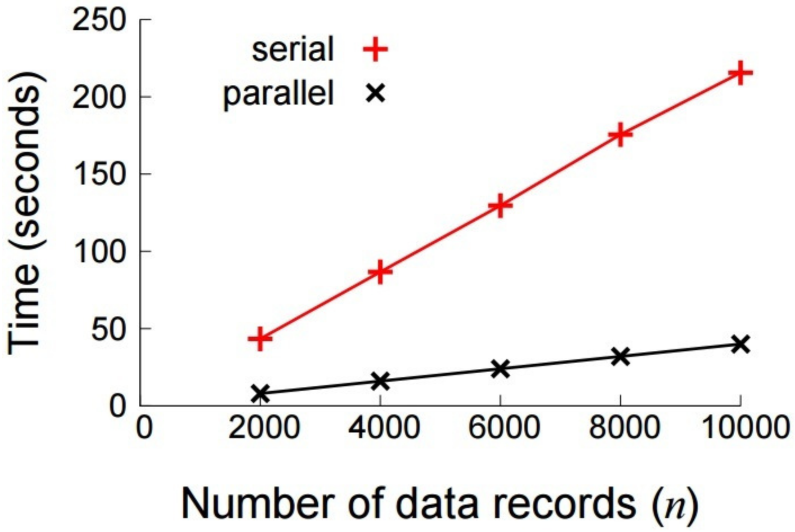


Figure 4.2: Parallel vs. serial versions of the SkNN$_b$ protocol for $m = 6$, $k = 5$, and $K = 512$

5. ADVANCED ANALYTICAL QUERY

Data Mining has wide applications in many areas including banking, medicine, scientific research, and government agencies. Classification is a very important task in various data mining applications. Performing data mining tasks by a cloud has recently attracted significant attentions. In general, performing any data mining tasks becomes challenging without ever decrypting the data irrespective of the underlying encryption scheme [76, 84]. Existing privacy-preserving classification techniques are not applicable because the data on the cloud is in encrypted form. Additional privacy concerns are demonstrated by the following example.

Example 9. *Suppose an insurance company outsourced its encrypted customers' database and relevant data mining tasks to a cloud. When an agent from the company wants to determine the risk level of a potential new customer, he/she can use a classification method to determine the customer's risk level. First, the agent needs to generate a data record (q) for the customer. This record contains certain personal information of the customer, e.g., credit score, age, marital status, etc. This record can then be sent to the cloud, and the cloud will compute the class label for q. Nevertheless, to protect the customer's privacy, q should be encrypted before it is sent to the cloud because it contains sensitive information.*
□

The above example illustrates that Data Mining over Encrypted Data (denoted by DMED) on a cloud also needs to protect a user's record when the record is part of a data mining process. A cloud can also derive useful, sensitive information about the actual data items by observing the data access patterns, even if the data are encrypted [27, 97]. Therefore, the privacy/security requirements of the DMED problem on a cloud are threefold: (1) confidentiality of the encrypted data, (2) confidentiality of a user's query record, and (3) hiding data access patterns.

Existing work on Privacy-Preserving Data Mining (PPDM) (either perturbation or a SMC-based approach) cannot solve the DMED problem. Perturbed data do not possess semantic security, so data perturbation techniques cannot be used to encrypt highly sensitive data. Additionally, perturbed data do not produce accurate data mining results. The SMC-based approach assumes that data are distributed and not encrypted at each participating party. Additionally, many intermediate computations are performed based on non-encrypted data.

Fully homomorphic cryptosystems (e.g., [36]) can solve the DMED problem because they allow a third-party (that hosts the encrypted data) to execute arbitrary functions over encrypted data without ever decrypting them. Such techniques, however, are very expensive, and their usage in practical applications has yet to be explored. For example, it was shown in [37] that even for weak security parameters one "bootstrapping" operation of the homomorphic operation would take at least 30 seconds on a high- performance machine. As a result, in this chapter, a method to effectively solve the DMED problem on encrypted relational data outsourced to a cloud is proposed.

This chapter specifically was focused on the classification problem over encrypted data because the classification is one of the most common data mining tasks. Each classification technique has its own advantage. In particular, this work was concentrated on executing the k-Nearest Neighbor (kNN) classification method over encrypted cloud data. More specifically, a Secure k-Nearest Neighbor (SkNN) classifier over encrypted relational data in the cloud was proposed [85]. Briefly, given a user's encrypted query record q, the goal is for a cloud to securely returns the encrypted class label for q based on the kNN classification method. This protocol protects not only confidentiality of the original data but also the user query from the cloud. It also hides the data access patterns and the classification result. This work was the first to develop a SkNN classifier over encrypted data under the semi-honest model [85]. For ease of presentation, some common notations that are used extensively throughout this chapter were summarized in Table 5.1.

5.1. DEFINING THE PROBLEM

Suppose Alice owns a database T of n records $t_1, \ldots, t_n$ and $m+1$ attributes. Let $t_{i,j}$ denote the j^{th} attribute value of record t_i. Initially, Alice encrypts her database attribute-wise. That is, she computes $E_{pk}(t_{i,j})$ for $1 \leq i \leq n$ and $1 \leq j \leq m+1$, where the column $(m+1)$ contains the class labels. Assume that the underlying encryption scheme

Table 5.1: Common notations used in the PPkNN protocol

Alice	The data owner holding database T
Bob	An authorized user who can access T' in the cloud
C_1 or C_2	Two non-colluding semi-honest cloud service providers
$\langle E_{pk}, D_{sk} \rangle$	A pair of Paillier's encryption and decryption functions with (pk, sk) as public-secret key pair
$E_{pk}(x)$	Component-wise encryption of x: $E_{pk}(x_1), \ldots, E_{pk}(x_m)$
T	A relational database with n records: $t_1, \ldots, t_n$
T'	An encryption of T: $E_{pk}(t_1), \ldots, E_{pk}(t_n)$
q	Bob's input query (m-dimensional vector): $\langle q_1, \ldots, q_m \rangle$
w	Number of unique class labels in T
l	Domain size (in bits)
$\langle z_1, z_l \rangle$	The least and most significant bits of integer z
$[z]$	Vector of encryptions of the individual bits of z
c_q	The class label corresponding to q based on T

is semantically secure [75]. Let the encrypted database be denoted by T'. Assume Alice outsources both T' and the future classification process to a cloud.

Let Bob be an authorized user who wants to classify his input record $q = \langle q_1, \ldots, q_m \rangle$ by applying the kNN classification method based on T'. Such a process is referred to as a Privacy-Preserving k-Nearest Neighbor (PPkNN) classification over encrypted relational data in the cloud. Formally, the PPkNN protocol is defined as:

$$\text{PP}k\text{NN}(T', q) \rightarrow c_q$$

where c_q denotes the class label for q after the kNN classification method is applied to both T' and q.

5.2. MAIN CONTRIBUTIONS

A novel PPkNN protocol, a secure kNN classifier over semantically secure encrypted data, is proposed [85]. Within this protocol, Alice does not participate in any computations once the encrypted data are outsourced to the cloud. Therefore, no information is revealed to Alice. This protocol also meets all the desired requirements discussed in Section 1.2:

- Neither the contents of T nor any intermediate results should be revealed to the cloud.
- Bob's input query q should not be revealed to the cloud.
- The output c_q should be computed accurately and revealed only to Bob; no other information should be revealed to Bob.
- Incur low computation overhead on the Bob after the encrypted query record is sent to the cloud.
- Data access patterns, such as the records corresponds to the k-Nearest Neighbors of q, should not be revealed to either Bob or the cloud (thereby preventing any inference attacks).

The intermediate results that the cloud can see as part of this protocol are either newly generated, randomized encryptions or random numbers. Thus, the cloud does not know which data records correspond to the k-Nearest Neighbors and the output class label. Bob does not involve in any computations after he sends his encrypted query record to the cloud. Hence, the data access patterns are further protected from Bob.

5.3. THE PROPOSED PRIVACY-PRESERVING k-NEAREST NEIGHBOR CLASSIFICATION PROTOCOL

A novel PPkNN classification protocol is proposed[85]. This protocol was constructed based on the set of sub-components presented in Section 3.5 as building blocks [21, 29, 85]. In the PPkNN protocol, the existence of two non-colluding semi-honest cloud service providers, denoted by C_1 and C_2, which together form a federated cloud was assumed.

Recall that Alice's database consists of n records, denoted by $T = \{t_1, \ldots, t_n\}$, and $m + 1$ attributes, where $t_{i,j}$ denotes the j^{th} attribute value of record t_i and the column $(m + 1)$ contains the class labels. Initially, Alice encrypts her database attribute-wise using her public key (pk). Let the encrypted database be denoted by T'. Without loss of generality, assume also that all attribute values and their Euclidean distances lie in $[0, 2^l)$. Additionally, let w denote the number of unique class labels in T.

Under this setting, Alice outsources her encrypted database (T') to C_1, and the secret key (sk) to C_2. In this work, after outsourcing encrypted data to the cloud, Alice does not participate in any future computations.

The goal of the PPkNN protocol is to classify users' query records using T' in a privacy-preserving manner. Consider Bob who wants to classify his query record $q = \langle q_1, \ldots, q_m \rangle$ based on T' in C_1. The proposed PPkNN protocol mainly consists of the following two stages [85]:

- Stage 1 - Secure Retrieval of k-Nearest Neighbors (SRkNN):

  In this stage, Bob initially sends his query q (in encrypted form) to C_1. Both C_1 and C_2 then involve in a set of sub-protocols to securely retrieve (in encrypted form) the class labels corresponding to the k-Nearest Neighbors of the input query q. At the end of this step, encrypted class labels of the k-Nearest Neighbors are known only to C_1.

- Stage 2 - Secure Computation of Majority Class (SCMC$_k$):

  Following from Stage 1, both C_1 and C_2 jointly compute the class label with a majority voting among the k-Nearest Neighbors of q. At the end of this step, only Bob knows the class label that corresponds to his input query record q.

The main steps involved in the proposed PPkNN protocol are presented in Algorithm 10. Next, each of the two stages in the PPkNN protocol is explained in detail.

5.3.1. Stage 1: Secure Retrieval of k-Nearest Neighbors. During Stage 1 (SRkNN), Bob initially encrypts his query q attribute-wise. That is, he computes $E_{pk}(q) = \big\langle E_{pk}(q_1), \ldots, E_{pk}(q_m) \big\rangle$. He then sends $E_{pk}(q)$ to C_1. The main steps involved in Stage 1 are shown as the steps 1 to 3 in Algorithm 10. Upon receiving $E_{pk}(q)$, C_1 with private input $\big(E_{pk}(q), E_{pk}(t_i)\big)$, and C_2 with the secret key sk jointly involve in the Secure Squared Euclidean Distance (SSED) protocol. Here, $E_{pk}(t_i) = \big\langle E_{pk}(t_{i,1}), \ldots, E_{pk}(t_{i,m}) \big\rangle$ for $1 \leq i \leq n$. The output of this step, denoted by $E_{pk}(d_i)$, is the encryption of squared Euclidean distance between q and t_i $\big($i.e., $d_i = |q - t_i|^2\big)$. Recall that $E_{pk}(d_i)$ is known only to C_1 for $1 \leq i \leq n$. Note that the computation of exact Euclidean distance between encrypted vectors is hard to achieve as it involves square root. In this study, however, it is sufficient to compare the squared Euclidean distances as it preserves relative ordering. Both C_1 with input $E_{pk}(d_i)$, and C_2 then securely compute the encryptions of the individual bits of d_i using the Secure Bit-Decomposition (SBD) protocol. Note that, the output $[d_i] = \langle E_{pk}(d_{i,1}), \ldots, E_{pk}(d_{i,l}) \rangle$ is known only to C_1, where $d_{i,1}$ and $d_{i,l}$ are the most and least significant bits of d_i, for $1 \leq i \leq n$, respectively.

Both C_1 and C_2 then compute the encryptions of class labels that correspond to the k-nearest neighbors of q in an iterative manner. More specifically, they compute $E_{pk}(c'_1)$ in the first iteration, $E_{pk}(c'_2)$ in the second iteration, and so on. Here, c'_s denotes the class label of s^{th} nearest neighbor to q for $1 \leq s \leq k$. At the end of k iterations, only C_1 knows $\langle E_{pk}(c'_1), \ldots, E_{pk}(c'_k) \rangle$. To start with, consider the first iteration. Both C_1 and C_2 jointly compute the encryptions of the individual bits of the minimum value among $d_1, \ldots, d_n$ and encryptions of both the location and class label corresponding to $d_{\min}$ using the Secure Minimum Out of n Numbers (SMIN$_n$) protocol. That is, both C_1, with input $(\theta_1, \ldots, \theta_n)$, and C_2, with sk, compute $\big([d_{\min}], E_{pk}(I), E_{pk}(c')\big)$, where $\theta_i = \big([d_i], E_{pk}(I_{t_i}), E_{pk}(t_{i,m+1})\big)$ for $1 \leq i \leq n$. Here, $d_{\min}$ denotes the minimum value among $d_1, \ldots, d_n$; I_{t_i} and $t_{i,m+1}$ denote the unique identifier and the class label corresponding to the data record t_i, respectively. Specifically, $\big(I_{t_i}, t_{i,m+1}\big)$ is the secret information associated with t_i. For simplicity, this work assumes that $I_{t_i} = i$. In the output, I and c' denote the index and the class label corresponding to $d_{\min}$. The output $\big([d_{\min}], E_{pk}(I), E_{pk}(c)\big)$ is known only to C_1. Now, C_1 performs the following operations locally:

- Assign $E_{pk}(c')$ to $E_{pk}(c'_1)$. Remember that, according to the SMIN$_n$ protocol, c' is equivalent to the class label of the data record that corresponds to $d_{\min}$. Thus, it is same as the class label of the most nearest neighbor to q.

- Compute the encryption of difference between I and i, where $1 \leq i \leq n$. That is, C_1 computes $\tau_i = E_{pk}(i) * E_{pk}(I)^{N-1} = E_{pk}(i - I)$ for $1 \leq i \leq n$.

- Randomize τ_i to obtain $\tau'_i = \tau_i^{r_i} = E_{pk}\big(r_i * (i - I)\big)$, where r_i is a random number in $\mathbb{Z}_N$. Note that τ'_i is an encryption of either 0 or a random number for $1 \leq i \leq n$. Also, it is worth noting that exactly one of the entries in τ' is an encryption of 0 (which happens iff $i = I$) and the remaining entries are encryptions of random numbers. Permute τ'

Algorithm 10 PPkNN$(T', q) \rightarrow c_q$

```
Require: $C_1$ has $T'$ and $\pi$; $C_2$ has $sk$; Bob has $q$
1: Bob:
    (a). Compute $E_{pk}(q_j)$, for $1 \leq j \leq m$
    (b). Send $E_{pk}(q) = \langle E_{pk}(q_1), \ldots, E_{pk}(q_m) \rangle$ to $C_1$
2: $C_1$ and $C_2$:
    (a). $C_1$ receives $E_{pk}(q)$ from Bob
    (b). for $i = 1$ to $n$ do:
        – $E_{pk}(d_i) \leftarrow \mathrm{SSED}\big(E_{pk}(q), E_{pk}(t_i)\big)$
        – $[d_i] \leftarrow \mathrm{SBD}\big(E_{pk}(d_i)\big)$
3: for $s = 1$ to $k$ do:
    (a). $C_1$ and $C_2$:
        – $\big([d_{\min}], E_{pk}(I), E_{pk}(c')\big) \leftarrow \mathrm{SMIN}_n(\theta_1, \ldots, \theta_n)$,
          where $\theta_i = \big([d_i], E_{pk}(I_{t_i}), E_{pk}(t_{i,m+1})\big)$
        – $E_{pk}(c'_s) \leftarrow E_{pk}(c')$
    (b). $C_1$:
        – $\Delta \leftarrow E_{pk}(I)^{N-1}$
        – for $i = 1$ to $n$ do:
            * $\tau_i \leftarrow E_{pk}(i) * \Delta$
            * $\tau'_i \leftarrow \tau_i^{r_i}$, where $r_i \in_R \mathbb{Z}_N$
        – $\beta \leftarrow \pi(\tau')$; send $\beta$ to $C_2$
    (c). $C_2$:
        – Receive $\beta$ from $C_1$
        – $\beta'_i \leftarrow D_{sk}(\beta_i)$, for $1 \leq i \leq n$
        – Compute $U'$, for $1 \leq i \leq n$: if $\beta'_i = 0$ then $U'_i = E_{pk}(1)$ else $U'_i = E_{pk}(0)$
        – Send $U'$ to $C_1$
    (d). $C_1$:
        – Receive $U'$ from $C_2$ and compute $V \leftarrow \pi^{-1}(U')$
    (e). $C_1$ and $C_2$, for $1 \leq i \leq n$ and $1 \leq \gamma \leq l$: $E_{pk}(d_{i,\gamma}) \leftarrow \mathrm{SBOR}\big(V_i, E_{pk}(d_{i,\gamma})\big)$
4: $\mathrm{SCMC}_k\big(E_{pk}(c'_1), \ldots, E_{pk}(c'_k)\big)$
```

using a random permutation function π (known only to C_1) to obtain $\beta = \pi(\tau')$ and send it to C_2.

Upon receiving β, C_2 decrypts it component-wise to obtain $\beta_i' = D_{sk}(\beta_i)$ for $1 \leq i \leq n$. After this, C_2 computes an encrypted vector U' of length n. If $\beta_i' = 0$, then $U_i = E_{pk}(1)$; otherwise $U_i = E_{pk}(0)$. Because exactly one of the entries in τ' is an encryption of 0, this further implies that exactly one of the entries in U' is an encryption of 1 and the remaining entries are encryptions of 0's. It is important to note that if $\beta_k' = 0$, then $\pi^{-1}(k)$ is the index of the data record that corresponds to $d_{\min}$. C_2 then sends U' to C_1. After receiving U', C_1 performs inverse permutation on it to obtain $V = \pi^{-1}(U')$. Note that exactly one of the entry in V is $E_{pk}(1)$ and the remaining are encryptions of 0's. Additionally, if $V_i = E_{pk}(1)$, then t_i is the most nearest tuple to q. Both C_1 and C_2, however, do not know which entry in V is corresponding to $E_{pk}(1)$.

Finally, C_1 updates the distance vectors $[d_i]$ as follows:

- It is important to note that, the first nearest tuple to q should be obliviously excluded from further computations. Since C_1, however, does not know the record corresponding to $E_{pk}(c_1')$, one need to obliviously eliminate the possibility of choosing this record again in next iterations. For this, C_1 obliviously updates the distance corresponding to $E_{pk}(c_1')$ to the maximum value (i.e., $2^l - 1$). More specifically, C_1 updates the distance vectors with the help of C_2 using the Secure Bit-OR (SBOR) protocol for $1 \leq i \leq n$ and $1 \leq \gamma \leq l$ as follows:

$$E_{pk}(d_{i,\gamma}) = \text{SBOR}\big(V_i, E_{pk}(d_{i,\gamma})\big)$$

Observe that, when $V_i = E_{pk}(1)$, the corresponding distance vector d_i is set to the maximum value. That is, under this case, $[d_i] = \big\langle E_{pk}(1), \ldots, E_{pk}(1)\big\rangle$. In contrast, when $V_i = E_{pk}(0)$, the OR operation has no effect on the corresponding encrypted distance vector.

The above process is repeated for k iterations. In each iteration, $[d_i]$ corresponding to the current chosen label is set to the maximum value. Both C_1 and C_2, however, do not know which $[d_i]$ is updated. In iteration s, $E_{pk}(c_s')$ is returned only to C_1. At the end of Stage 1, C_1

has $\big\langle E_{pk}(c'_1), \ldots, E_{pk}(c'_k)\big\rangle$, the list of the encrypted class labels of the k-Nearest Neighbors to q.

5.3.2. Stage 2: Secure Computation of Majority Class. Without loss of generality, suppose Alice's dataset T consists of w unique class labels $\big($denoted by $c = \langle c_1, \ldots, c_w\rangle\big)$. Assume that Alice outsources her list of encrypted classes to C_1. That is, Alice outsources $\big\langle E_{pk}(c_1), \ldots, E_{pk}(c_w)\big\rangle$ to C_1 along with her encrypted database T' during the data outsourcing step. Note that, for security reasons, Alice may add dummy categories into the list to protect the number of class labels (i.e., w) from both C_1 and C_2. For simplicity, this work, however, assumes that Alice does not add any dummy categories to c.

During Stage 2 (SCMC$_k$), C_1 with private inputs $\Lambda = \big\langle E_{pk}(c_1), \ldots, E_{pk}(c_w)\big\rangle$ and $\Lambda' = \big\langle E_{pk}(c'_1), \ldots, E_{pk}(c'_k)\big\rangle$, and C_2 with sk securely compute $E_{pk}(c_q)$. Here, c_q denotes the majority class label among $c'_1, \ldots, c'_k$. At the end of the stage 2, only Bob knows the class label c_q.

The overall steps involved in Stage 2 are presented in Algorithm 11. To start with, both C_1 and C_2 jointly compute the encrypted frequencies of each class label using the k-nearest set as input. That is, they compute $E_{pk}\big(f(c_i)\big)$ using (Λ, Λ') as C_1's input to the Secure Frequency protocol for $1 \leq i \leq w$. The output $\Big\langle E_{pk}\big(f(c_1)\big), \ldots, E_{pk}\big(f(c_w)\big)\Big\rangle$ is known only to C_1. Both C_1, with $E_{pk}\big(f(c_i)\big)$, and C_2, with sk, then involve in the SBD protocol to compute $\big[f(c_i)\big]$ for $1 \leq i \leq w$. Here, $\big[f(c_i)\big]$ denotes the vector of encryptions of the individual bits of $f(c_i)$ for $1 \leq i \leq w$. Both C_1 and C_2, then jointly involve in the Secure Maximum Out of w Numbers (SMAX$_w$) protocol. It's worth mentioning that by using the similar formulations used to design SMIN and SMIN$_n$ protocols, as discussed in Sections 3.5.7 and 3.5.8, one can also design SMAX and SMAX$_w$ protocols. Briefly, SMAX$_w$ utilizes the sub-routine SMAX to eventually compute $\big([f_{\max}], E_{pk}(c_q)\big)$ in an iterative fashion. Here, $[f_{\max}] = \Big[\max\big(f(c_1), \ldots, f(c_w)\big)\Big]$ and c_q denotes the majority class out of Λ'. At the end, the output $\big([f_{\max}], E_{pk}(c_q)\big)$ is known only to C_1. C_1 locally computes $\gamma_q = E_{pk}(c_q + r_q)$, where r_q is a random number in $\mathbb{Z}_N$ that known only to C_1. C_1 then sends γ_q to C_2 and r_q to Bob. Upon receiving γ_q, C_2 decrypts it to obtain the randomized majority class label $\gamma'_q = D_{sk}(\gamma_q)$ and sends it to Bob. Finally, upon receiving both r_q from C_1 and γ'_q from C_2, Bob computes the output class label that is corresponding to q as $c_q = \gamma'_q - r_q \bmod N$.

Algorithm 11 $\text{SCMC}_k\big(E_{pk}(c'_1), \ldots, E_{pk}(c'_k)\big) \rightarrow c_q$

```
Require: $\langle E_{pk}(c_1), \ldots, E_{pk}(c_w)\rangle$, $\langle E_{pk}(c'_1), \ldots, E_{pk}(c'_k)\rangle$ are known only to $C_1$; $sk$ is known
    only to $C_2$
1: $C_1$ and $C_2$:
    (a). $\Big\langle E_{pk}\big(f(c_1)\big), \ldots, E_{pk}\big(f(c_w)\big)\Big\rangle \leftarrow \text{SF}(\Lambda, \Lambda')$,
            where $\Lambda = \langle E_{pk}(c_1), \ldots, E_{pk}(c_w)\rangle$, $\Lambda' = \langle E_{pk}(c'_1), \ldots, E_{pk}(c'_k)\rangle$
    (b). for $i = 1$ to $w$ do:
        • $[f(c_i)] \leftarrow \text{SBD}\Big(E_{pk}\big(f(c_i)\big)\Big)$
    (c). $\big([f_{\max}], E_{pk}(c_q)\big) \leftarrow \text{SMAX}_w(\psi_1, \ldots, \psi_w)$, where $\psi_i = \Big(\big[f(c_i)\big], E_{pk}(c_i)\Big)$, for $1 \leq$
         $i \leq w$
2: $C_1$:
    (a). $\gamma_q \leftarrow E_{pk}(c_q) * E_{pk}(r_q)$, where $r_q \in_R \mathbb{Z}_N$
    (b). Send $\gamma_q$ to $C_2$ and $r_q$ to Bob
3: $C_2$:
    (a). Receive $\gamma_q$ from $C_1$
    (b). $\gamma'_q \leftarrow D_{sk}(\gamma_q)$; send $\gamma'_q$ to Bob
4: Bob:
    (a). Receive $r_q$ from $C_1$ and $\gamma'_q$ from $C_2$
    (b). $c_q \leftarrow \gamma'_q - r_q \bmod N$
```

5.4. SECURITY ANALYSIS

A formal security proof for the PPkNN protocol under the semi-honest model was provided [85]. First, Bob's input query (q) is protected from Alice, C_1 and C_2 due to the encryption of q and by semantic security of the Paillier cryptosystem [75]. Apart from guaranteeing query privacy, remember that, the goal of the PPkNN protocol is to protect data confidentiality and hide data access patterns.

In this study, to prove a protocol's security under the semi-honest model, the well-known security definitions from the literature of the Secure Multiparty Computation (SMC) was adopted. More specifically, the security proofs based on the standard simulation paradigm [41] was adopted (As discussed in Section 3.2). For presentation purpose, formal security proofs (under the semi-honest model) for Stages 1 and 2 of the PPkNN protocol separately were provided. Note that, the outputs returned by each sub-protocol were in encrypted form and would be known only to C_1.

5.4.1. Security Proof for Stage 1. Recall that, the computations involved in Stage 1 of the PPkNN were given as steps 1 to 3 in Algorithm 10. For ease of presentation, this work considered the messages exchanged between both C_1 and C_2 in a single iteration (however, similar analysis could be deduced for other iterations).

According to Algorithm 10, the execution image of C_2 is given by

$$\Pi_{C_2}(\mathrm{PP}k\mathrm{NN}) = \big\{\langle \beta_i, \beta'_i \rangle \mid \text{for } 1 \leq i \leq n\big\}$$

where β_i is an encrypted value which is a random in $\mathbb{Z}_{N^2}$. Also, β'_i is derived upon decrypting β_i by C_2. Remember that, exactly one entries in the β' is 0 and the remaining entries are random numbers in $\mathbb{Z}_N$. Without loss of generality, let the simulated image of C_2 be denoted by $\Pi^S_{C_2}(\mathrm{PP}k\mathrm{NN})$, where

$$\Pi^S_{C_2}(\mathrm{PP}k\mathrm{NN}) = \big\{\langle a'_{1,i}, a'_{2,i} \rangle \mid \text{for } 1 \leq i \leq n\big\}$$

where $a'_{1,i}$ is randomly generated from $\mathbb{Z}_{N^2}$ and the vector a'_2 is randomly generated in such a way that exactly one of the entries is 0 and the remaining entries are random numbers in $\mathbb{Z}_N$. Since E_{pk} is a semantically secure encryption scheme with a resulting ciphertext size that is less than $\mathbb{Z}_{N^2}$, β_i is computationally indistinguishable from $a'_{1,i}$. Additionally, since the random permutation function π is known only to C_1, β' is a random vector of exactly one 0 and random numbers in $\mathbb{Z}_N$. Thus, β' is computationally indistinguishable from a'_2. As a result, the $\Pi_{C_2}(\mathrm{PP}k\mathrm{NN})$ protocol is computationally indistinguishable from $\Pi^S_{C_2}(\mathrm{PP}k\mathrm{NN})$. This implied that, C_2 did not learn anything during the execution of Stage 1 in PPkNN.

Similarly, suppose the execution image of C_1 is denoted by $\Pi_{C_1}(\mathrm{PP}k\mathrm{NN})$, and is given by

$$\Pi_{C_1}(\mathrm{PP}k\mathrm{NN}) = \{U'\}$$

where U' is an encrypted value sent by C_2 (at Step 3(c) of Algorithm 10). Let the simulated image of C_1 in Stage 1 be denoted by $\Pi^S_{C_1}(\text{PP}k\text{NN})$, which is given as

$$\Pi^S_{C_1}(\text{PP}k\text{NN}) = \{a'\}$$

where a' is randomly generated from $\mathbb{Z}_{N^2}$. Since E_{pk} is a semantically secure encryption scheme with resulting ciphertexts in $\mathbb{Z}_{N^2}$, U' is computationally indistinguishable from a'. This implied that $\Pi_{C_1}(\text{PP}k\text{NN})$ is computationally indistinguishable from $\Pi^S_{C_1}(\text{PP}k\text{NN})$. Hence, C_1 could not learn anything during the execution of Stage 1 in PPkNN. Putting the above results together, it was concluded that Stage 1 of PPkNN was secure under the semi-honest model.

In each iteration, it is worth pointing out that both C_1 and C_2 did not know which data record belongs to the current global minimum. Thus, the data access patterns were protected from both C_1 and C_2. Informally speaking, at Step 3(c) of Algorithm 10, a component-wise decryption of β did reveal the tuple that satisfy the current global minimum distance to C_2. Due to the random permutation by C_1, however, C_2 could not trace back to the corresponding data record. Also, note that, decryption operations on vector β by C_2 would result in exactly one 0 and the remaining results were random numbers in $\mathbb{Z}_N$. Similarly, since U' was an encrypted vector, C_1 could not know which tuple corresponds to current global minimum distance.

5.4.2. Security Proof for Stage 2. Stage 2 of the PPkNN protocol was secure under the semi-honest model. Briefly, since the sub-protocols SF, SBD, and SMAX$_w$ are secure, no information was revealed to C_2. In contrast, the operations performed by C_1 were entirely on encrypted data. No information, therefore, was revealed to C_1.

Furthermore, the output data of Stage 1 which were passed as input to Stage 2 are in encrypted format. Therefore, the sequential composition of the two stages lead to our PPkNN protocol and one could claim it to be secure under the semi-honest model according to the Composition Theorem [41] given in Definition 2. In particular, based on the above discussions, it was clear that the proposed PPkNN protocol protected the data confidentiality, user's input query, and also hide data access patterns from Alice, C_1, and C_2. Note that, Alice did not participate in any computations of the PPkNN protocol.

5.5. COMPLEXITY ANALYSIS

The proposed PPkNN protocol's computation was analyzed. The computation complexity for each sub-protocol was analyzed first under the assumptions that encryption and decryption operations based on Paillier cryptosystem [75] take a similar amount of time, an exponentiation operation was treated as an encryption operation, and an encryption operation is generally several orders of magnitude more expensive than a multiplication. The computation complexity of Stage 1 in PPkNN was bounded by $O(n)$ instantiations of the SBD and the SSED protocols, $O(k)$ instantiations of SMIN$_n$ protocol, and $O(n * k * l)$ instantiations of the SBOR protocol. The computation complexity of the SBD protocol proposed in [86] was bounded by $O(l)$ encryptions. Also, the computation complexity of the SSED protocol was bounded by $O(m)$ encryptions. Additionally, the computation complexity of the SMIN$_n$ protocol was bounded by $O(l * n * \log_2 n)$ encryptions. Since the SBOR protocol utilized the SM protocol as a sub-routine, the computation cost of the SBOR protocol was bounded by (small) constant number of encryptions and exponentiations. Based on the above analysis, the total computation complexity of Stage 1 was bounded by $O\big(n * (l + m + k * l * \log_2 n)\big)$ encryptions.

In contrast, the computation complexity of Stage 2 was bounded by $O(w)$ instantiations of the SBD protocol, and one instantiation of both the SF and the SMAX$_w$. Here, the computation complexity of the SF protocol was bounded by $O(k * w)$ encryptions and $O(k * w)$ exponentiations. More details regarding the complexity analysis of sub-protocols were given in Section 3.5. Therefore, the total computation complexity of Stage 2 was bounded by $O\big(w * (l + k + l * \log_2 w)\big)$ encryptions.

In general, $w \ll n$, therefore, the computation cost of Stage 1 should be significantly higher than that of Stage 2. This observation was further justified by the empirical results given in the next section.

5.6. PERFORMANCE EVALUATION

The proposed PPkNN protocol was implemented and its performance was measured across a range of inputs [85]. By using the Paillier cryptosystem[75] as the underlying additive homomorphic encryption scheme, the protocol was implemented in C. Various experiments were conducted on a Linux machine with an Intel® Xeon® Six-Core$^{\text{TM}}$ CPU 3.07 GHz processor and 12GB RAM running Ubuntu 12.04 LTS.

This work is the first effort to develop a secure k-Nearest Neighbor classifier under the semi-honest model. Thus, there was no existing work to compare with this work. Therefore, the performance of the PPkNN protocol is evaluated under different parameter settings.

For the experiments, the Car Evaluation dataset from the UCI KDD archive [12] had used. The dataset consisted of 1728 data records (i.e., $n = 1728$) with 6 input attributes (i.e., $m = 6$). Also, there is a separate class attribute. The dataset categorized into four different classes (i.e., $w = 4$) and encrypted attribute-wise, using the Paillier [75] encryption whose key size is varied in the experiments. The encrypted data stored on the above machine. Then, a random query was chosen and executed over the encrypted data based on the PPkNN protocol. For the rest of this section, the performance of Alice was not discussed as she was a one-time cost. Instead, the evaluation were based on the performances of the two stages in the PPkNN protocol separately.

The computation costs of Stage 1 in the PPkNN protocol were analyzed first by varying values of number of k-nearest neighbors (see Figure 5.1(a)). The Paillier encryption key size K was either 512 or 1024 bits. The computation cost of Stage 1 for K=512 bits varied from 9.98 to 46.16 minutes when k varied from 5 to 25, respectively. In contrast, the computation cost of Stage 1 for K=1024 bits varied from 66.97 to 309.98 minutes when k varied from 5 to 25, respectively. In either case, the computation time of Stage 1 grew almost linearly with k. Additionally, for any given k, the cost of Stage 1 increased by almost a factor of 7 whenever K doubled. For example, when k=10, Stage 1 required 19.06 and 127.72 minutes to generate the encrypted class labels of the 10 nearest neighbors under K=512 and K=1024 bits, respectively. Furthermore, when k=5, one could observe that around 66.29% of the cost in Stage 1 accounted due to SMIN$_n$ which initiated k times in the PPkNN protocol (once in each iteration). Also, the cost incurred due to SMIN$_n$ increased from 66.29% to 71.66% when k varied from 5 to 25.

The computation costs of Stage 2 were analyzed next by varying values of k and K (see Figure 5.1(b)). The computation time for Stage 2 varied from 0.118 to 0.285 seconds when k varied from 5 to 25. In contrast, for K=1024 bits, Stage 2 took 0.789 and 1.89 seconds when $k = 5$ and k=25, respectively. The low computation costs of Stage 2 were due to SMAX$_w$ which incurred significantly fewer computations than the SMIN$_n$ in Stage 1. This further justified the theoretical analysis given in Section 5.5. Note that, in the dataset, w=4 and n=1728. Like in Stage 1, for any given k, the computation time of Stage 2 increased by almost a factor of 7 whenever K doubled. For example, when k=10, the computation time of Stage 2 varied from 0.175 to 1.158 seconds when the encryption key size K changed from 512 to 1024 bits. A similar analysis could be observed for other values of k and K (see Figure 5.1(b)).

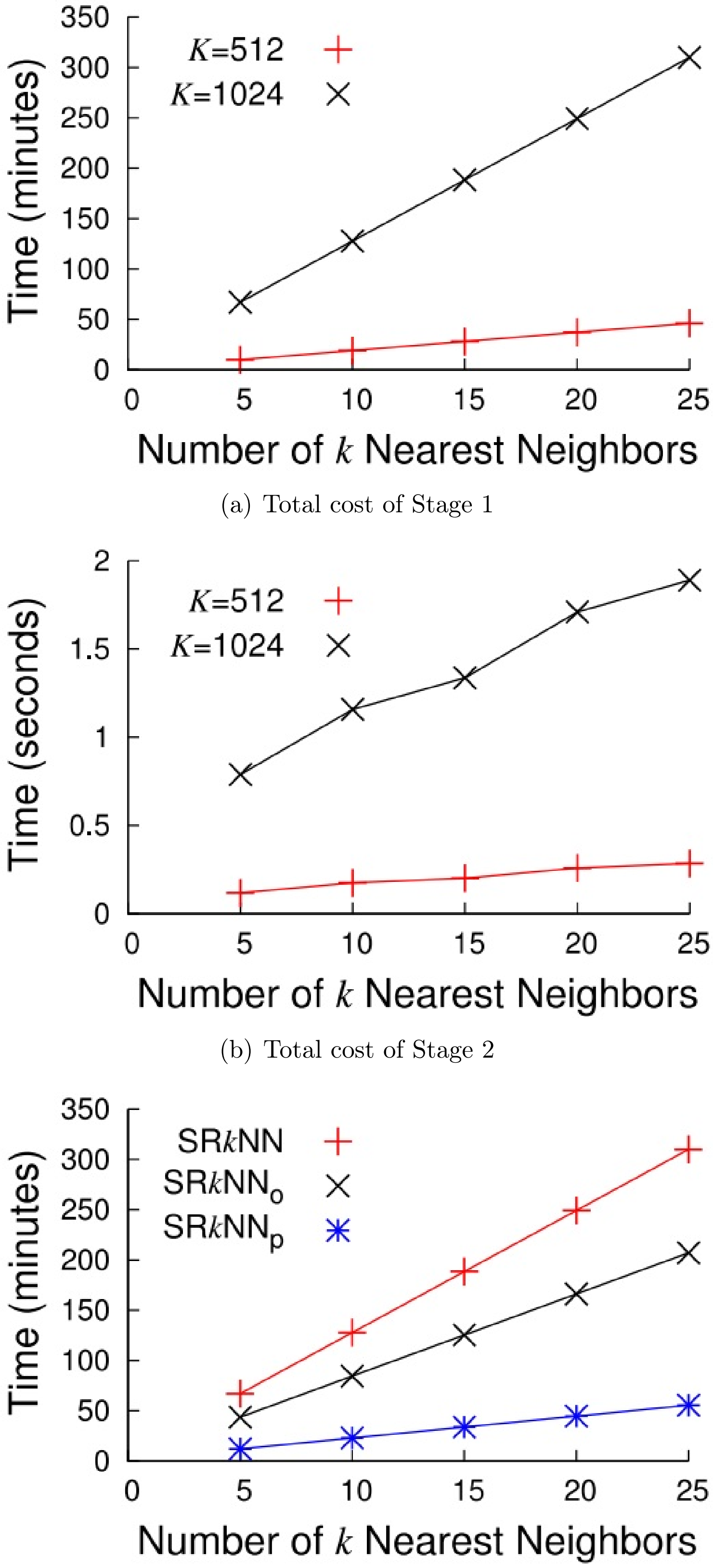
Based on the above results, it was clear that the computation cost of Stage 1 was significantly higher than that of Stage 2. More specifically, the computation time of Stage 1 accounted for at least 99% of the total time in the PPkNN protocol. For example, when $k = 10$ and K=512 bits, the computation costs of Stage 1 and 2 are 19.06 minutes and 0.175 seconds, respectively. Under this scenario, the cost of Stage 1 was 99.98% of the total cost of the PPkNN protocol. Putting the above together, it was concluded that the total computation time of the PPkNN protocol grew almost linearly with both n and k.

5.6.1. Performance Improvement. Two different approaches are proposed to boost the efficiency of Stage 1 (as the performance of the PPkNN protocol depended primarily on Stage 1).

The first approach to improving the performance the performance of Stage 1 is by pushing some computation offline. More specifically, some of the computations in Stage 1 could be pre-computed (pushed offline). For example, encryptions of random numbers, 0s and 1s could be pre-computed (by the corresponding parties) in the offline phase. As a result, the online computation cost of Stage 1 (denoted by SRkNN$_\text{o}$) is expected to be improved. To see the actual efficiency gains of such a strategy, the costs of SRkNN$_\text{o}$ were computed and compared with the costs of Stage 1 without an offline phase (simply denoted by SRkNN) and the results for $K = 1024$ bits are shown in Figure 5.1(c). Irrespective of the values of k, one could observe that SRkNN$_\text{o}$ was around 33% faster than SRkNN. For example, the computation costs of SRkNN$_\text{o}$ and SRkNN for $k = 10$ were 84.47 and 127.72 minutes, respectively (boosting the online running time of Stage 1 by 33.86%).

The second approach to improving the performance of Stage 1 is by using parallelism. Since operations on data records are independent of one another, the most computations in Stage 1 could be parallelized. To empirically evaluate this claim, a parallel version of Stage 1 (denoted by SRkNN$_\text{p}$) was implemented using OpenMP programming [24] and compared its cost with the costs of SRkNN (i.e., the serial version of Stage 1). The computation cost of SRkNN$_\text{p}$ for $K = 1024$ varied from 12.02 to 55.5 minutes when k changed from 5 to 25 (see Figure 5.1(c)). SRkNN$_\text{p}$ was almost 6 times more efficient than SRkNN. This was because the machine used for this experiments had 6 cores. Thus, the computations could be run in parallel on 6 separate threads. Based on the above discussions, it was clear that efficiency of Stage 1 could indeed be improved significantly using parallelism. Moreover, one could also use the existing map-reduce techniques to execute parallel operations on multiple nodes to drastically improve the performance further. Hence, the level of achievable performance in the PPkNN protocol actually depended on the implementation.

In contrast, Bob's computation cost in the PPkNN protocol was mainly due to the encryption of his input query. In the dataset, Bob's computation cost was 4 and 17 milliseconds when K was 512 and 1024 bits, respectively. It was apparent that PPkNN protocol was very efficient from Bob's computational perspective which was especially beneficial when he issued queries from a resource-constrained device (such as mobile phone and PDA).



(a) Total cost of Stage 1

(b) Total cost of Stage 2

(c) Efficiency gains of Stage 1 for $K = 1024$

Figure 5.1: Computation costs of the PPkNN protocol for varying number of kNNs and different encryption key sizes in bits (K)

6. CORRELATED RANGE QUERY

Recently, biometric authentication/identification has increasingly gained importance for various application domains. Those systems which include fingerprint-, face- and iris-authentication/identification systems are widely used in enterprise, civilian and law enforcement. Such systems typically consist of an entity whose databases hold biometric records and users/clients who send the entity their biometric recordings for authentication/identification [32]. During the process of biometric authentication, the systems need to identify whether candidate biometric readings from users match the records in the entity's biometric database. In contrast, during the process of biometric identification, the systems need to retrieve the profile of a person whose biometric data record in the entity's biometric database matches the user's input biometric data record.

There are potential privacy concerns regarding biometric authentication/ identification: biometric matching process would cause a leakage of user's biometric data especially running on untrusted servers. Biometric data are usually very sensitive because they could uniquely identify a person. Thus, the leakage of biometric data to malicious parties can lead to a violation of personal privacy. To avoid this privacy concern, Privacy-Preserving Biometric Authentication (PPBA) and Privacy-Preserving Biometric Identification (PPBI) protocols have been developed (e.g., [10, 11, 30, 32, 83]). A PPBI protocol generally returns the profile of a person whose biometric data record stored on the server matches the user's input biometric data record. Whereas a PPBA protocol only returns a single bit to indicate if there is a match or not. Under the existing PPBA and PPBI protocols, a user's biometric data record is never disclosed to the entity, and the entity's biometric database is never exposed to the user [21].

Due to the cost efficiency and operational flexibility of the cloud computing paradigm [4, 20, 60], an entity has the opportunity to outsource its data and their relevant computations to a cloud which can provide on-demand services. To outsource biometric authentication tasks, the entity's biometric database needs to be encrypted before outsourced to a cloud.

Recall that the goal of PPBA/PPBI protocols are to perform biometric authentication/identification without disclosing the involved biometric data to the participating parties, except for their own data. When the biometric data are encrypted and cannot be decrypted by a cloud, the existing PPBA/PPBI protocols are not applicable in the cloud computing environment where the cloud stores encrypted biometric data for authentication/identification purposes. Therefore, in this chapter, Outsourceable and Privacy-Preserving Biometric Authentication (PPBA$_O$) [21] and Outsourceable and Privacy-Preserving Biometric Identification (PPBI$_O$)) protocols over biometric data stored in the cloud were presented. For ease of presentation, some common notations that are used extensively throughout this chapter are summarized in Table 6.1.

6.1. OUTSOURCEABLE AND PRIVACY-PRESERVING BIOMETRIC AUTHENTICATION

This section focused on developing the PPBA$_O$ protocol. In this protocol, a user issues an encrypted biometric data query record to a cloud. The cloud then securely returns a single bit to indicate if the distance between any of the biometric data records stored in a cloud and the user's input query is below t. Here, t denotes a pre-defined threshold.

6.1.1. Defining the Problem. Let $D = \{v_1, \dots, v_n\}$ denote an entity's biometric image database with n biometric data records. Each v_i, $1 \leq i \leq n$ is an m-dimensional vector representation of a biometric data record. Let $D' = \big\{E_{pk}(v_1), \dots, E_{pk}(v_n)\big\}$ denote the encryption of D, where each $E_{pk}(v_i)$ is encrypted component-wise using an Additive Homomorphic Public Key Encryption (AH-Enc) scheme. Let Bob be a user who wants to be authenticated securely using his biometric image data denoted by u. Here, u is represented by the same method as the records in D. A PPBA$_O$ protocol can be defined as follows [21]:

$$\text{PPBA}_O\big(D', E_{pk}(u), t\big) \rightarrow b \tag{6.1}$$

A biometric data record (v_i) exists in D such that the distance between u and v_i is below t when $b = 1$, otherwise $b = 0$. t is defined by the underlying biometric authentication system. It varies from system to system. It also depends on the biometric data used for authentication. How to determine the best value for t is out of the scope of this work, so one

Table 6.1: Common notations used in the PPBA$_O$/PPBI$_O$ protocols

C_1 or C_2	Two non-colluding semi-honest cloud service providers
Bob	A user who wants to securely perform biometric authentication/identification
v_i or u	m-dimension feature vector representation of a biometric data image
p_i	s-dimension vector representation of an identity profile data
$E_{pk}(x)$	Component-wise encryption of x: $E_{pk}(x_1), \ldots, E_{pk}(x_m)$
D	A biometric database with n records: $v_1, \ldots, v_n$
D'	An encryption of D: $E_{pk}(v_1), \ldots, E_{pk}(v_n)$
P	An identity profile database with n records: $p_1, \ldots, p_n$
P'	An encryption of P: $E_{pk}(p_1), \ldots, E_{pk}(p_n)$
u	Bob's biometric image data record: $u_1, \ldots, u_m$
t	A Pre-defined threshold

can just assume t is publicly known parameter. Both D' and $E_{pk}(u)$ are never decrypted to maximize the confidentiality protection of both D and u during an execution of PPBA$_O$.

This work inherits the common structure of the existing PPBA systems which mainly consist of two phases: distance computation, and matching and retrieval [32]. In the distance computation phase, either Euclidean or Hamming distance between the feature vectors of biometric records v_i's in D and the user's biometric data record u are calculated pairwise. After that in matching and retrieval phase, those distances are compared with a pre-defined threshold t to decide whether u matches some v_i with t-distance apart. Some biometric authentication protocols such as face recognition [30] need a feature extraction phase to get the biometric feature vectors; however, this work assumed that the feature extraction phase as a pre-computation stage to produce the feature vectors in both D and u.

6.1.2. Main Contributions. A secure PPBA$_O$ protocol is proposed [21]. Within this protocol, the entity (data owner) does not participate in any computations once the encrypted data are outsourced to the cloud. Therefore, no information is revealed to the entity. This protocol also meets all the desired requirements discussed in Section 1.2:

- Neither the contents of D or any intermediate results should not be revealed to the cloud.
- Bob's biometric image data u should not be revealed to the cloud.

- The output b should not be revealed to the cloud.
- Incur low computation overhead at the end-user side because Bob stops involving in computations after he sends his encrypted biometric data.
- Data access patterns, such as the biometric data records corresponding to the matching or comparison results with u, should not be revealed to either Bob or the cloud.

It is worth pointing out that the intermediate results that the cloud can see as part of this protocol are either newly generated randomized encryptions or random numbers.

6.1.3. The Proposed Outsourceable and Privacy-Preserving Biometric Authentication Protocol. Let C_1 denote a cloud service provider who stores D' and performs biometric authentication based on D', $E_{pk}(u)$, and t. In addition to C_1, another independent cloud service provider (C_2) is utilized in the proposed PPBA$_O$ protocol. This protocol uses an AH-Enc scheme to encrypt each v_i and u component-wise to produce $E_{pk}(v_i)$ and $E_{pk}(u)$. There are several AH-Enc schemes, and, without loss of generality, this work adopts the Paillier cryptosystem [75] because it is simple to implement and semantically secure. D' is outsourced and stored at C_1, and only C_2 has the corresponding decryption key sk. Therefore, D' and $E_{pk}(u)$ cannot be decrypted by C_1 without accessing the decryption key. To solve the proposed PPBA$_O$ problem, one might suggest using garbled circuits [53] for every secure computation between C_1 and C_2. This is logical for simple tasks (e.g., secure comparison), where such an approach can be quite efficient. However, biometric authentication is a complex process consisting of several sub-components.

The best way to implement an efficient two-party secure protocol for a functionality as complex as biometric authentication is to combine a homomorphic encryption approach and a garbled circuit approach. This hybrid approach was adopted in [72] to produce a secure protocol that is more efficient than either the homomorphic encryption approach or the garbled circuit approach alone. The key challenges in utilizing the hybrid approach are:

- Breaking a complex functionality into a set of simpler sub-components.
- Determining which approach to use to implement each sub-component securely.

In the next section, this work dissects the PPBA$_O$ into sub-components and identifies the most efficient approach, between homomorphic encryption or garbled circuit, to implement each component [21].

6.1.3.1. Sub-Components of the PPBA$_O$ Protocol. To implement a PPBA$_O$ protocol, each biometric image first needs to be represented as a feature vector. For the rest of this work, assume the feature vector of each biometric image is given as part of the input to a PPBA$_O$ protocol.

According to Equation 6.1 and based on the description given in Section 6.1.1, a PPBA$_O$ protocol (with input $D' = \{E_{pk}(v_1), \ldots, E_{pk}(v_n)\}$, $E_{pk}(u)$ and t) consists of three main sub-components or protocols that need to be performed sequentially. According to the composition theorem [41], when a protocol is implemented based on a set of secure primitives, and for the protocol to be secure, the intermediate results produced from these primitives must be in the form of random shares. Both the description and the outcome of each sub-component are given as follows [21]:

(1) *Distance computation*: This component computes the distance between u and each v_i for $1 \leq i \leq n$. Note that the distance is between the actual feature vectors, but the input should be the encrypted feature vectors $E_{pk}(u)$ and $E_{pk}(v_i)$. During the computation, both $E_{pk}(u)$ and $E_{pk}(v_i)$ are never decrypted to preserve the confidentiality of both u and v_i. Let d_i denote the distance between u and v_i. The output of this component returns two random shares d'_i and d''_i for each d_i such that $d'_i + d''_i \mod N = d_i$.

(2) *Comparing with the threshold*: Once computed, these distances need to be compared with the threshold t to find out if u is one of $v_1, \ldots, v_n$. A distance less than t indicates that u matches some biometric record in D. Again, all these computations are based on encrypted data or random shares. More specifically, the (d'_i, d''_i) pairs and the threshold t are the input for this sub-component. Let b_i denote the comparison result between d_i and t. If $d_i < t$, then $b_i = 1$, otherwise $b_i = 0$. The output of this component returns two random shares b'_i and b''_i for each b_i such that $b'_i + b''_i \mod N = b_i$.

(3) *Combining the comparison results*: As long as there is a b_i equal to 1, one can conclude that u matches some v_i in D. Then the authentication succeeds. The (b'_i, b''_i) are the input for this sub-component. Let b_f denote the final authentication result. If $b_f = 1$,

then authentication succeeds. On the other hand, if $b_f = 0$, then the authentication fails. The output of this component returns two random shares b'_f and b''_f such that $b'_f + b''_f \mod N = b_f$.

Note that it is possible for the last component to return b_f directly, but both b'_f and b''_f are more useful in case b_f serves as an intermediate result for a more complex protocol. Based on their need, the participating parties can decide the appropriate output. Without loss of generality, this work adopts b'_f and b''_f as the final outcome of the proposed PPBA$_O$ protocol.

Depending on the specific functionality or computation, this work has identified the most efficient and secure method (either a homomorphic encryption based approach or garbled circuit) to implement each sub-component.

(1) Secure Distance Computation - Since biometric data have multiple types, and each type can be represented in various ways, this work implemented secure sub-protocols to compute the two most common distance metrics adopted in the existing PPBI work: Euclidean distance and Hamming distance (as discussed in Section 3.5). Both metrics have their advantages and disadvantages, and an entity can decide which metric to use for its PPBA$_O$ protocol. Two sub-protocols for implementing secure Euclidean and Hamming distances, namely Secure Squared Euclidean Distance-Random Share (SSED$_R$) and Secure Hamming Distance-Random Share (SHD$_R$), were implemented based on the homomorphic encryption approach. They were more efficient than garbled circuit-based solutions. A detailed analysis is provided in Sections 3.5.3 and 3.5.4. This chapter does not specify the distance metric (either Euclidean or Hamming). Depending on the type of biometric data and the feature vector, the entity who outsourced its biometric authentication to the cloud can decide the appropriate metric to use. For simplicity, the term $\textsc{Secure\_Distance}(a, b)$ is used to refer to the process of securely computing the distance between a and b.

(2) Securely Comparing with the Threshold - A secure comparison protocol is needed to implement the second sub-component securely. The secure comparison protocol, denoted by Secure Comparison with a Threshold (SCT) (as discussed in Section 3.5.10), takes random shares (i.e., d'_i and d''_i) and a threshold t as input and returns two random shares b'_i and b''_i. If $d_i \leq t$, then $b'_i + b''_i \mod N = 1$; otherwise $b'_i + b''_i \mod N = 0$.

It was assumed here that $\text{SCT}(d'_i, d''_i, t) \to (b'_i, b''_i)$ was the protocol used to implement this sub-component. A garbled circuit was used to construct it, where d'_i and d''_i are private input values from C_1 and C_2, respectively, and t is a publicly known parameter. The protocol returns b'_i to C_1 and b''_i to C_2.

(3) Securely Combining the Comparison Results - The third sub-component/ functionality was used to identify the existence of a comparison result that is equal to 1. The individual comparison results cannot be disclosed to maximize the security guarantee; otherwise, the first two sub-components would have been sufficient. The challenge now is to determine whether or not there is a comparison result of 1 without disclosing which encrypted biometric record $E_{pk}(v_i)$ matches $E_{pk}(u)$. Disclosing the matching result may be considered harmless because a server only knows if two encrypted biometric data records match. The matching result, however, must not be leaked to preserve the security guarantee of the underlying encryption scheme [54].

The protocol's design was based on the following observations to prevent disclosing either the matching or the comparison results to C_1 and C_2 [21]:

Observation 1. *Let $b_1, \ldots, b_n$ be the n actual comparison results from comparing $d_1, \ldots,$ d_n with the threshold t, where $d_1, \ldots, d_n$ are the distances between u and each of $v_1, \ldots, v_n$. Then, $b_f = 1$ if and only if $0 < \sum_{i=1}^{n} b_i$.*

Observation 2. *Let $\alpha = \sum_{i=1}^{n} b'_i$ and $\beta = \sum_{i=1}^{n} b''_i$. Then, $\alpha + \beta \mod N = \sum_{i=1}^{n} b_i$.*

These observations reveal that the same secure comparison protocol used to implement the second sub-component can be used here to implement this sub-component. More specifically, the protocol $\text{SCT}(\alpha, \beta, 0) \to (b'_f, b''_f)$ was used to implement the sub-component. A garbled circuit was used to conduct it, where α and β are private input values from C_1 and C_2, respectively, and 0 is a public parameter. The protocol returns b'_f to C_1 and b''_f to C_2 such that $b'_f + b''_f \mod N = b_f$.

6.1.3.2. The PPBA$_O$ Protocol: . A PPBA$_O$ protocol, directly based on these components, can be built once the three sub-components have been properly and securely implemented (as discussed in Section 3.5). The key steps in the PPBA$_O$ protocol are given in Algorithm 12 [21]. Here, a user's encrypted biometric data record $E_{pk}(u)$ is assumed to

Algorithm 12 $\text{PPBA}_O\big(D', E_{pk}(u), t\big) \rightarrow b$

```
Require: $C_1$ has $D'$ and $E_{pk}(u)$, $C_2$ has the decryption key $sk$, and $t$ is a public parameter
1: $C_1$ and $C_2$ jointly execute $\textsc{Secure\_Distance}\big(E_{pk}(v_i), E_{pk}(u)\big)$, for $i = 1$ to $n$
2: $C_1$ and $C_2$ jointly execute $\text{SCT}(d'_i, d''_i, t)$, for $i = 1$ to $n$
3: $C_1$: $\alpha \leftarrow \sum_{i=1}^{n} b'_i \mod N$
4: $C_2$: $\beta \leftarrow \sum_{i=1}^{n} b''_i \mod N$
5: $C_1$ and $C_2$ jointly execute $\text{SCT}(\alpha, \beta, 0)$
6: $b \leftarrow b'_f + b''_f \mod N$
```

have been received by C_1. The same public key that was used to encrypt D to produce D' was used here, where D' was encrypted by the entity that owns the biometric database D.

Step 1 of Algorithm 12 can be used to securely compute the distance between u and each of $v_1, \ldots, v_n$. Both C_1 and C_2 receive $d'_1, \ldots, d'_n$ and $d''_1, \ldots, d''_n$ as their private outputs, respectively. Note that the distance metric (neither Euclidean nor Hamming) is specified in the protocol. Depending on the type of biometric data and the feature vector, the entity who outsourced its biometric authentication to a cloud can decide the appropriate metric to use (either Euclidean nor Hamming). The SSED_R and SHD_R in Algorithm 12 are interchangeable without affecting the PPBA_O protocol's security and correctness.

The output from Step 1 serves as the input for Step 2. At the end of Step 2, C_1 receives $b'_1, \ldots, b'_n$, and C_2 receives $b''_1, \ldots, b''_n$. Both C_1 and C_2 compute α and β (each one an input for Step 5 of the algorithm) independently. Step 5 returns b'_f to C_1 and b''_f to C_2. C_2 can send d''_f to C_1, and b_f can be reconstructed by adding the two random shares modulo N to obtain the actual authentication result. Steps 2, 3, and 5 of Algorithm 12 correspond to the three sub-components. Their implementations are investigated in Section 3.5.

6.1.4. Security Analysis. The PPBA_O protocol was secure under the semi-honest adversary model of Secure Multiparty Computation (SMC). The protocol was a sequential composition of sub-protocols. Thus, the security of each sub-protocol needed to be proved before the PPBA_O's security could be proved. As discussed in Sections 3.5.3 and 3.5.4, both SSED_R and SHD_R protocols were proven secure under the semi-honest model because the computations were performed on either encrypted data or randomized data [21]. The SCT protocol was implemented using a garbled circuit, which was also secure under the semi-honest model [53]. All these sub-protocols of PPBA$_O$ produced random shares as intermediate results. Therefore, according to the sequential composition theorem [41] given in Definition 2, the PPBA$_O$ protocol was also secure under the semi-honest model.

6.1.5. Complexity Analysis. The proposed PPBA$_O$ protocol's computation complexity was analyzed under the assumptions that encryption and decryption operations based on the Paillier cryptosystem [75] take a similar amount of time, an exponentiation operation was treated as an encryption operation, and an encryption operation is generally several orders of magnitude more expensive than a multiplication. Recall that the computation complexity of the SSED$_R$ and the SHD$_R$ protocols were bounded by O(m) encryptions. Also, O(m) encryptions provided an appropriate upper bound for the SCT protocol. Since the PPBA$_O$ protocol executed the Secure_Distance protocol n times and the SCT protocol $n+1$ times, this work can claim that the total computation complexity of PPBA$_O$ was bound by O(mn) encryptions. More details regarding the complexity analysis of sub-protocols are given in Section 3.5.

6.1.6. Performance Evaluation. Recall that the proposed protocol is a sequential composition of three sub-protocols. Each sub-protocol was implemented, and its performance was measured across a range of inputs [21].

Synthetic datasets were randomly generated according to the parameter values being considered. The advantage of using these synthetic datasets is that a more elaborated analysis could be performed on the computation costs of the proposed protocols under different parameter settings. Biometrics feature vectors were randomly generated according to the parameter values being considered (e.g., n- number of record vectors and m-vector size). These feature vectors were encrypted component-wise by the Paillier cryptosystem [75] with a 1024-bit modulus and stored on a local machine. The PPBA$_O$ protocol was performed according to this dataset. A Linux machine with an Intel® Xeon® Six-Core$^{\text{TM}}$ CPU 3.07 GHz was used to conduct all of the experiments.

The computation costs of the SSED$_R$ sub-protocol (Step 1 of Algorithm 12) were analyzed first by varying m and n. The computation costs of the SCT sub-protocols (Steps 3 and 5 of Algorithm 12) were analyzed next by varying n because m is irrelevant to the secure comparison task.

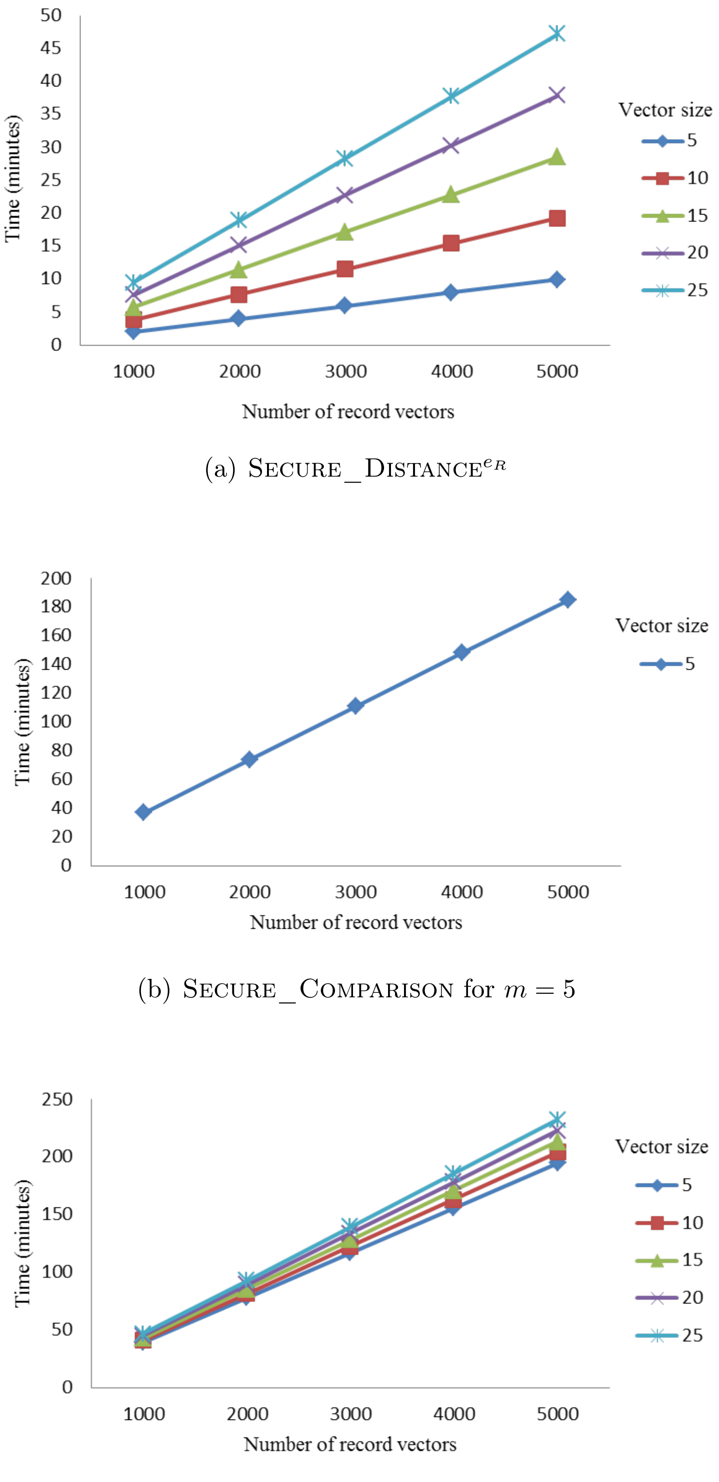
The SSED$_R$ was implemented by the Paillier encryption [75] in C language on top of the GNU multiple precision arithmetic library (https://gmplib.org/). The computation cost grew linearly with n and m (see Figure 6.1(a)). For example, when $m = 5$, the computation time of SSED$_R$ increased from 1.987 to 9.921 minutes when n was varied from 1000 to 5000.

The SCT sub-protocol was built on top of a GCParser framework [65], which is a modular intermediate level language for easily optimizing and executing garbled circuits. The m did not affect this stage's performance because the size of the inputs for this stage were fixed by 1024 bits modulus, which are random shares of a 1024 bit number. The computation costs of this SCT were evaluated by fixing $m = 5$ and varying values of n. The SCT's running time varied from 37.45 to 185.013 minutes when n changed from 1000 to 5000, respectively (see Figure 6.1(b)). Thus, the running time grew almost linearly with n.

The computation costs of the PPBA$_O$ protocol scaled almost linearly with n and m (see Figure 6.1(c)). For example, when $m = 5$, the computation time of the PPBA$_O$ protocol increased from 39.032 to 194.934 minutes when n varied from 1000 to 5000. Therefore, the computation time for the PPBA$_O$ protocol was dominated by the time required to compare the encrypted distance vectors. The run-time complexity of the SSED$_R$ protocol was, however, greater than the SCT protocol when m was sufficiently large (e.g., over 100). This finding is consistent with the upper bound derived in Section 6.1.5.

The communication cost was roughly 16MB for $n = 5000$, $m = 25$, and an encryption key size of 1024. The time needed to transmit 16MB of data was significantly less than the computation time. Therefore, this communication complexity was ignored in the proposed protocol.

6.1.6.1. Performance Improvement. The proposed PPBA$_O$ protocol is not practical if biometric authentication needs to be completed in real time, even though it is the best two-party protocol that is known. One primary advantage of the proposed protocol is that the computations of the sub-components can be parallelized. For example, at Steps 1 and 2 of Algorithm 12, the SECURE_DISTANCE computation and the SCT are independent and can be performed at the same time. The PPBA$_O$ protocol can take advantage of highly parallel computing capabilities because C_1 and C_2 are assumed to be cloud service



(a) SECURE_DISTANCE^{e_R}

(b) SECURE_COMPARISON for $m = 5$

(c) OPPBA execution time

Figure 6.1: Time complexities of a) SSED$_R$, b) SCT, and c) PPBA$_O$ by varying n and m

providers. If C_1 and C_2 have n nodes available to execute the PPBA$_O$ protocol, Step 1 can be performed within a second, and Step 2 can be performed in slightly more than 2 seconds. Thus, the total running time would be approximately 5 seconds. In general, SMC-based privacy-preserving protocols are very expensive. Utilizing the cloud is the only way to make real-time applications (e.g., biometric authentication) practical.

6.2. OUTSOURCEABLE AND PRIVACY-PRESERVING BIOMETRIC IDENTIFICATION

This section is focused on developing Outsourceable and Privacy-Preserving Biometric Identification (PPBI$_O$) protocol. In this protocol, a user issues an encrypted biometric data query record to a cloud. The cloud then securely returns all the identity profiles associated with the biometric image data records whose distances from the user's input query are below t. Here, t denotes a pre-defined threshold.

6.2.1. Defining the Problem. Let $D = \{v_1, \ldots, v_n\}$ denote an entity's biometric database with n biometric data records. Each v_i, $1 \leq i \leq n$ is an m-dimensional vector representation of a biometric image data record. Let $D' = \left\{E_{pk}(v_1), \ldots, E_{pk}(v_n)\right\}$ denote the encryption of D, where each $E_{pk}(v_i)$ is encrypted component-wise. Let $P = \{p_1, \ldots, p_n\}$ denote an identity profile database of n records and s attributes (e.g., SSN, name, age, and criminal record). Let $p_{i,j}$ denote the j^{th} attribute value of record p_i. Let $P' = \left\{E_{pk}(p_1), \ldots, E_{pk}(p_n)\right\}$ denote the encryption of P, where each $E_{pk}(p_i)$ is encrypted component-wise. Here, v_i denotes the biometric image data that corresponds to an identity profile p_i for $1 \leq i \leq n$. Suppose Bob is a user who wants to learn all the identity profile records associated with the biometric image data records whose distances from his input query u are below t. Here, u is represented by the same method as the records in D.

Briefly, the goal of the PPBI$_O$ protocol is to securely retrieve the set of identity profile records, denoted by $\mathcal{S}$, such that the following property holds:

$$\forall p_i \in \mathcal{S},\ b_i = 1,\ for\ 1 \leq i \leq n$$

where $b_1, \ldots, b_n$ is the n actual comparison result. If the distance between u and v_i is less than t then $b = 1$; otherwise, $b_i = 0$ for $1 \leq i \leq n$. The t is defined by the underlying biometric identification system. It varies from system to system. It also depends on the biometric data used for identification. More formally, the PPBI$_O$ protocol can be defined as follows, with D', P', and $E_{pk}(u)$ used as inputs:

$$\mathrm{PPBI}_O\big(D', P', E_{pk}(u), t\big) \rightarrow \mathcal{S} \tag{6.2}$$

The D', P', and $E_{pk}(u)$ are never decrypted to maximize the confidentiality protection of D, P, and u during an execution of the PPBI$_O$ protocol. This protocol also meets the desired requirements discussed in Section 1.2:

6.2.2. The Proposed Outsourceable and Privacy-Preserving Biometric Identification Protocol. The PPBI$_O$ protocol proposed here consists of three phases: distance computation, matching, and retrieval. Each biometric image needs to be represented as a feature vector before a PPBI$_O$ protocol can be implemented. The feature vector of each biometric image is assumed to be given as part of the input to a PPBI$_O$ protocol. According to Equation 6.2 and the above description, a PPBI$_O$ protocol consists of two main sub-components that need to be performed sequentially: distance computation and comparing with the threshold. A PPBI$_O$ protocol can be built based on the same components used to implement the PPBA$_O$ protocol.

The primary steps involved in the proposed PPBI$_O$ protocol are given in Algorithm 13. Here, Bob's encrypted biometric image data record $E_{pk}(u) = \big\{E_{pk}(u_1), \ldots, E_{pk}(u_m)\big\}$ is assumed to have been received by C_1. It is encrypted with the same public key as that used to encrypt D to produce D', where D' is encrypted by the entity that owns the biometric image database D. Step 1(a) of Algorithm 13 securely computes the distance between u and $v_1, \ldots, v_n$. C_1 and C_2 receive $d'_1, \ldots, d'_n$ and $d''_1, \ldots, d''_n$ as their private outputs, respectively. Note that, in this protocol, the distance metric (either Euclidean or Hamming) is not specified. The entity that outsourced its biometric identification to C_1 and C_2 can decide which metric to use. The SSED$_R$ and the SHD$_R$ protocols in Algorithm 13 are interchangeable without affecting the PPBI$_O$ protocol's security and correctness. The output from Step 1(a) serves as the input for Step 1(b). At the end of Step 1(b), C_1 receives $b'_1, \ldots, b'_n$, and C_2 receives $b''_1, \ldots, b''_n$, respectively, and then sends them to Bob $\big($step 1(c)$\big)$. Steps 1(a) and 1(b) of Algorithm 13 correspond to the two sub-components and their implementations investigated in Section 3.5.

Algorithm 13 $\text{PPBI}_O(D', P', E_{pk}(u), t) \rightarrow \mathcal{S}$

```
Require: $C_1$ has $D'$, $P'$ and $E_{pk}(u)$, $C_2$ has the decryption key, and $t$ is a public parameter
1: $C_1$ and $C_2$: for $1 \leq i \leq n$ do:
    (a). $(d'_i, d''_i) \leftarrow$ SECURE_DISTANCE$\big(E_{pk}(v_i), E_{pk}(u)\big)$
    (b). $(b'_i, b''_i) \leftarrow \text{SCT}(d'_i, d''_i, t)$
    (c). $C_1$ send $b'_i$ to Bob, $C_2$ send $b''_i$ to Bob
2: Bob: for $1 \leq i \leq n$ do:
    (a). Receive $b'_i$ from $C_1$ and $b''_i$ from $C_2$
    (b). Compute $b_i \leftarrow (b'_i, b''_i)$
    (c). if $\nexists\ b_i = 1$, for $1 \leq i \leq n$
        • NO MATCH
    (d). else
        • $\mathcal{S} \leftarrow \phi$
        • Retrieve the $i^{th}$ record from $C_1$ using OBLIVIOUS_TRANSFER$\binom{n}{1}$:
              $E_{pk}(p_i) \leftarrow \text{OT}^n_1(i)$
        • $E_{pk}(\gamma_{i,h}) \leftarrow E_{pk}(p_{i,h}) * E_{pk}(r_{i,h})$, where $r_{i,h} \in_R \mathbb{Z}_N$, for $1 \leq h \leq s$
        • Send $E_{pk}(\gamma_{i,h})$ to $C_2$, for $1 \leq h \leq s$
        • $C_2$: for $1 \leq h \leq s$
            – Receive $E_{pk}(\gamma_{i,h})$ from Bob
            – $\gamma'_{i,h} \leftarrow D_{sk}(E_{pk}\big(\gamma_{i,h})\big)$
            – Send $\gamma'_{i,h}$ to Bob
        • Bob: :
            – Receive $\gamma'_{i,h}$ from $C_2$, for $1 \leq h \leq s$
            – $p_{i,h} \leftarrow \gamma'_{i,h} - r_{i,h} \mod N$, for $1 \leq h \leq s$
            – $\mathcal{S} \leftarrow \mathcal{S} \cup p_i$
```

In Step 2, upon receiving the comparison results $b_1, \ldots, b_n$ from comparing $d_1, \ldots, d_n$ from C_1 and C_2, Bob proceeds as follows:

- Compute $b_i \leftarrow b'_i + b''_i \mod N$, for $1 \leq i \leq n$ and verify if there is a match or a “no match”. In case there is a match, Bob initially sets the output set $\mathcal{S}$ to ϕ and continues as follows:

- For each $b_i = 1$, Bob retrieves the i^{th} encrypted identity profile record $\big(E_{pk}(p_i)\big)$ whose corresponding comparison result (b_i) is equal to one by using any (1-N)-OBLIVIOUS_ TRANSFER protocol (e.g., [70]). He then randomizes it attribute-wise by computing

$E_{pk}(\gamma_{i,h}) = E_{pk}(p_{i,h}) * E_{pk}(r_{i,h})$ for $1 \leq h \leq s$. Here, $r_{i,h}$ is a random number in $\mathbb{Z}_N$, $p_{i,h}$ denotes the column h attribute value of the identity profile record p_i, $E_{pk}(r_{i,h})$ denotes the encryption of $r_{i,h}$, and $E_{pk}(\gamma_{i,h})$ denotes the encryption of $\gamma_{i,h}$. $E_{pk}(\gamma_{i,h})$ is sent to C_2 for $1 \leq h \leq s$.

C_2 decrypts each entry $E_{pk}(\gamma_{i,h})$ received from Bob to obtain $\gamma'_{i,h} = D_{sk}\big(E_{pk}(\gamma_{i,h})\big)$. It then sends them back to Bob. Note that, $\gamma'_{i,h}$ is always a random number in $\mathbb{Z}_N$ due to the randomization by Bob.

Bob then removes the randomness from $\gamma'_{i,h}$ to obtain the attribute values of the i^{th} identity profile as $p_{i,h} = \gamma'_{i,h} - r_{i,h} \mod N$ for each received entry $\{r_{i,h}, \gamma'_{i,h}\}$ for $1 \leq h \leq s$. Finally, Bob adds the identity profile record p_i to his output set ($\mathcal{S} = \mathcal{S} \cup p_i$).

6.2.3. Security Analysis. The PPBI$_O$ protocol was secure under the semi-honest adversary model of SMC. The protocol was a sequential composition of sub-protocols. Thus, the security of each sub-protocol needed to be proved before the PPBI$_O$'s security could be proved. A detailed analysis is provided in Sections 3.5.3 and 3.5.4. Both SSED$_R$ and SHD$_R$ protocols were proven secure under the semi-honest model because the computations were performed on either encrypted data or randomized data. The SCT protocol was implemented using a garbled circuit, which was also secure under the semi-honest model [53]. A detailed analysis is provided in Section 3.5.10. Also, the (1-N)-OBLIVIOUS_TRANSFER protocol (e.g., [70]) was proved secure under the semi-honest model. All these sub-protocols of PPBI$_O$ produced either random shares or pseudo-random as intermediate results. Therefore, according to the sequential composition theorem [41] given in Definition 2, the PPBI$_O$ protocol was also secure under the semi-honest model.

6.2.4. Complexity Analysis. The proposed PPBI$_O$ protocol's computation complexity was analyzed under the assumptions that encryption and decryption operations based on the Paillier cryptosystem [75] take a similar amount of time, an exponentiation operation was treated as an encryption operation, and an encryption operation is generally several orders of magnitude more expensive than a multiplication. The computation complexity of the SSED$_R$ protocol was bounded by O(m) encryptions (as discussed in Section 6.1.5). The same asymptotic bound for the SHD$_R$ can be derived in a similar manner. Additionally, the total computation cost of the SCT protocol is about that of performing several homomorphic encryption operations. Thus, O(m) encryptions provided an appropriate upper bound for the SCT (as discussed in Section 6.1.5). Also, the computational complexity of the (1-N)-OBLIVIOUS_TRANSFER protocol (e.g., [70]) was bounded by $O(n)$. The rest of the computation complexity was bounded by $O(s * n)$ encryption operations. In order to hide the data access patterns from C_1 and C_2, Bob, jointly with C_1, had to retrieve the i^{th} encrypted identity profile whose corresponding comparison result was $b_i = 1$ for $1 \leq i \leq n$. As a consequence, Bob's computation cost was not negligible. He did perform some expensive computational operations bounded by $O(s * n)$ encryptions. Therefore, this work can claim that the total computation complexity of PPBI$_O$ was bound by $O(s * n)$ encryptions.

7. CONCLUSIONS AND FUTURE DIRECTIONS

The cloud computing paradigm [17, 66] has recently revolutionized the organization's way of operating their data, particularly in the way they store, access, and process data. As an emerging computing paradigm, cloud computing attracts many organizations to consider a cloud's potential in terms of its cost-efficiency, flexibility, and offload of administrative overhead. Organizations often delegate their computational operations, in addition to their data, to a cloud; otherwise, there would be no point in outsourcing the data at the first place. Privacy and security issues in the cloud are preventing companies from utilizing those advantages despite the tremendous advantages that the cloud offers. Therefore, due to the rise of various privacy issues, sensitive data need to be encrypted before being outsourced to the cloud. Using encryption as a way to achieve data confidentiality may cause another issue at the cloud during the query evaluation. In general, it is very difficult to process encrypted data in a privacy-preserving manner without ever having to decrypt it. The question here is how the cloud can perform computations over encrypted data while the data stored in the cloud are encrypted at all times.

Along with this direction, this study proposed three different sets of Privacy-Preserving Query Processing (PPQP) protocols to facilitate different types of queries, namely, the k-Nearest Neighbor (kNN) query, advanced analytical query, and correlated range query, which preserve both the data confidentiality and the query privacy. In the proposed PPQP protocols, once the data owner has outsourced his/her encrypted data to the cloud, he/she does not participate in any computations. The proposed protocols utilize additive homomorphic cryptosystem and/or garbled circuit technique at different stages of query processing to achieve the best performance. In addition, all computations can be done on the encrypted data without using very expensive fully homomorphic encryptions by adopting a multi-cloud computing paradigm. This work empirically analyzed the efficiency of the proposed protocols through various experiments. These results indicated that the PPQP protocols are efficient from the end-user's perspective. This work also emphasized that the proposed protocols are practical in the cloud environment.

One possible extension to the current work is to explore alternative ways of developing efficient PPQP protocols. The construction of the PPQP protocols is the most efficient and secure two-party implementation known today. However, the empirical results clearly showed that they are not practical. The scalability issue of PPQP protocols can be eliminated or mitigated, especially in a cloud computing environment, where high-performance parallel processing can easily be achieved. Thus, one possible work would be to implement and evaluate the PPQP protocols using a MapReduce techniques in real cloud environment. The set of sub-protocols that would be developed and used as basic primitives when constructing the proposed PPQP protocols need efficient implementation in order to improve the performance of PPQP. One way to achieve that is to try to develop parallel solutions to those basic primitives, which, in turn, will improve the overall performance of the PPQP protocols.

Encryption is not the only way to protect data confidentiality, and a variety of different techniques, such as randomization and secret sharing, exist. One can plan to investigate whether these techniques are more efficient and scalable than the encryption based solutions. In addition, the current work can be extended to other adversary models, such as the malicious model. One can develop PPQP protocols that are secure under the malicious model and evaluate the trade-offs between security and efficiency.

BIBLIOGRAPHY

[1] D. J. Abadi. Data management in the cloud: Limitations and opportunities. *IEEE Data Eng. Bull*, 32(1):3–12, 2009.

[2] C. C. Aggarwal and P. S. Yu. A general survey of privacy-preserving data mining models and algorithms. *Privacy-preserving data mining*, pages 11–52, 2008.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM, 2000.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53:50–58, April 2010.

[5] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2):281–343, Apr. 2010.

[6] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *IEE ICDE*, pages 217–228, 2005.

[7] D. Beaver. Foundations of secure interactive computing. In *Advances in Cryptology - CRYPTO '91*, pages 377–391. Springer-Verlag, 1991.

[8] A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp - a system for secure multi-party computation. In *ACM CCS*, October 2008.

[9] I. F. Blake and V. Kolesnikov. One-round secure comparison of integers. *Journal of Mathematical Cryptology*, 3(1):37–68, May 2009.

[10] M. Blanton and M. Aliasgari. Secure outsourced computation of iris matching. *Journal of Computer Security*, 20(2):259–305, 2012.

[11] M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. In *Computer Security–ESORICS 2011*, pages 190–209. Springer, 2011.

[12] M. Bohanec and B. Zupan. The UCI KDD Archive. University of California, Department of Information and Computer Science, Irvine, CA, 1997. http://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

[13] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, page 78, 2012.

[14] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Proceedings of the* 31^{st} *Annual Conference on Advances in Cryptology*, pages 505–524, 2011.

[15] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing. In *Workshop on Cryptography and Security in Clouds (WCSC 2011)*, 2011.

[16] S. Bugiel, S. Nürnberger, A.-R. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing (extended abstract). In *Workshop on Cryptography and Security in Clouds*, March 2011.

[17] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[18] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[19] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *IEEE FOCS*, pages 136 – 145, oct. 2001.

[20] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina. Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, CCSW '09, pages 85–90. ACM, 2009.

[21] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang. Outsourceable two-party privacy-preserving biometric authentication. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 401–412. ACM, 2014.

[22] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, 2002.

[23] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*, pages 280–299, 2001.

[24] L. Dagum and R. Enon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

[25] I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In *Proceedings of the 12th Australasian conference on Information security and privacy*, pages 416–430. Springer-Verlag, 2007.

[26] I. Damgard, M. Geisler, and M. Kroigard. Homomorphic encryption and secure comparison. *International Journal of Applied Cryptology*, 1(1):22–31, 2008.

[27] S. De Capitani di Vimercati, S. Foresti, and P. Samarati. Managing and accessing data in the cloud: Privacy risks and approaches. In *CRiSIS*, pages 1–9. IEEE, 2012.

[28] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. *Information Security*, pages 471–483, 2002.

[29] Y. Elmehdwi, B. K. Samanthula, and W. Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 664–675. IEEE, 2014.

[30] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies*, pages 235–253. Springer, 2009.

[31] T. Ermakova, B. Fabian, and R. Zarnekow. Security and privacy system requirements for adopting cloud computing in healthcare data sharing scenarios. 2013.

[32] D. Evans, Y. Huang, J. Katz, and L. Malka. Efficient privacy-preserving biometric identification. In *Proceedings of the 17th conference Network and Distributed System Security Symposium, NDSS*, 2011.

[33] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29(4):343–364, 2004.

[34] S. Fienberg and J. McIntyre. Data swapping: Variations on a theme by dalenius and reiss. In *Privacy in statistical databases*, pages 519–519. Springer, 2004.

[35] J. Garay, B. Schoenmakers, and J. Villegas. Practical and secure solutions for integer comparison. In *Proceedings of the 10th international conference on Practice and theory in public-key cryptography*, pages 330–342. Springer-Verlag, 2007.

[36] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, 2009.

[37] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, pages 129–148. Springer-Verlag, 2011.

[38] C. Gentry, S. Halevi, and N. Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology - EUROCRYPT'2012*, 2012.

[39] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD*, pages 121–132. ACM, 2008.

[40] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols, pages 599–746. Cambridge, University Press, Cambridge, England, 2004.

[41] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter Encryption Schemes, pages 373–470. Cambridge University Press, Cambridge, England, 2004.

[42] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[43] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38:690–728, 1991.

[44] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18:186–208, February 1989.

[45] T. Graepel, K. Lauter, and M. Naehrig. Ml confidential: Machine learning on encrypted data. In *Information Security and Cryptology–ICISC 2012*, pages 1–21. Springer, 2013.

[46] H. Hacıgümüş, B. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *Database systems for Advanced Applications*, pages 633–650. Springer, 2004.

[47] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez. An analysis of security issues for cloud computing. *Journal of Internet Services and Applications*, 4(1):1–13, 2013.

[48] W. Henecka, S. K ögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Tasty: tool for automating secure two-party computations. In *ACM CCS*, pages 451–462. ACM, 2010.

[49] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multidimensional range queries over outsourced data. *The VLDB Journal*, 21(3):333–358, 2012.

[50] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 720–731. VLDB Endowment, 2004.

[51] H. Hu, J. Xu, C. Ren, and B. Choi. Processing private queries over untrusted data cloud through privacy homomorphism. In *IEEE ICDE*, pages 601–612, 2011.

[52] Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*, 2012.

[53] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of the 20th USENIX conference on Security (SEC '11)*, pages 35–35, 2011.

[54] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Proceedings of the* 19^{th} *Annual Network & Distributed System Security Symposium (NDSS)*, February 2012.

[55] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-based fingerprint matching. *Image Processing, IEEE Transactions on*, 9(5):846–859, 2000.

[56] A. Janosi, W. Steinbrunn, M. Pfisterer, and R. Detrano. Heart disease data set. The UCI KDD Archive, 1988. http://archive.ics.uci.edu/ml/datasets/Heart+Disease.

[57] M. Kantarcioglu and C. Clifton. Privately computing a distributed k-nn classifier. In *PKDD*, pages 279–290, 2004.

[58] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall, CRC Press, 2007.

[59] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology–EUROCRYPT 2008*, pages 146–162. Springer, 2008.

[60] K. Kumar and Y.-H. Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51 –56, april 2010.

[61] M. Li, S. Yu, W. Lou, and Y. T. Hou. Toward privacy-assured cloud data services with flexible search functionalities. In *32nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 466–470. IEEE, 2012.

[62] Y. Lindell. General composition and universal composability in secure multiparty computation. *Journal of Cryptology*, 22(3):395–428, 2009.

[63] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology (CRYPTO)*, pages 36–54. Springer, 2000.

[64] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):5, 2009.

[65] W. Melicher, S. Zahur, and D. Evans. An intermediate language for garbled circuits. In *IEEE Symposium on Security and Privacy Poster Abstract*. Citeseer, 2012.

[66] P. Mell and T. Grance. The nist definition of cloud computing (draft). *NIST special publication*, 800:145, 2011.

[67] D. Micciancio. A first glimpse of cryptography's holy grail. *Communications of the ACM*, 53(3):96–96, 2010.

[68] E. Mykletun and G. Tsudik. Aggregation queries in the database-as-a-service model. In *Data and Applications Security XX*, pages 89–103. Springer, 2006.

[69] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, pages 245–254, Atlanta, Georgia, United States, 1999. ACM Press.

[70] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[71] A. E. Nergiz, M. E. Nergiz, T. Pedersen, and C. Clifton. Practical and secure integer comparison and interval check. In *Proceedings of the IEEE Second International Conference on Social Computing*, pages 791–799, 2010.

[72] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *IEEE Symposium on Security and Privacy (SP '13)*, pages 334–348. IEEE Computer Society, 2013.

[73] S. R. Oliveira and O. R. Zaiane. Privacy preserving clustering by data transformation. In *Proc. of the 18th Brazilian Symposium on Databases*, pages 304–318, 2003.

[74] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. Scifi-a system for secure face identification. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 239–254. IEEE, 2010.

[75] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Eurocrypt*, pages 223–238. Springer-Verlag, 1999.

[76] S. Pearson and A. Benameur. Privacy, security and trust issues arising from cloud computing. In *IEEE CloudCom*, pages 693–702, 2010.

[77] S. Pearson, Y. Shen, and M. Mowbray. A privacy manager for cloud computing. *Cloud Computing*, pages 90–106, 2009.

[78] M. Prabhakaran and M. Rosulek. Rerandomizable rcca encryption. In *Advances in Cryptology-CRYPTO 2007*, pages 517–534. Springer, 2007.

[79] Y. Qi and M. J. Atallah. Efficient privacy-preserving k-nearest neighbor search. In *ICDCS*, pages 311–319. IEEE, 2008.

[80] S. Ravu, P. Neelakandan, M. Gorai, R. Mukkamala, and P. Baruah. A computationally efficient and scalable approach for privacy preserving knn classification. In *IEEE International Conference on High Performance Computing (HiPC)*, 2012.

[81] J. J. Rodrigues, I. de la Torre, G. FernÃąndez, and M. LÃşpez-Coronado. Analysis of the security and privacy requirements of cloud-based electronic health records systems. *Journal of medical Internet research*, 15(5), 2013.

[82] M. D. Ryan. Cloud computing security: The scientific challenge, and a survey of solutions. *Journal of Systems and Software*, 2013.

[83] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *Information, Security and Cryptology–ICISC 2009*, pages 229–244. Springer, 2010.

[84] A. Sahai. Computing on encrypted data. *Information Systems Security*, pages 148–153, 2008.

[85] B. K. Samanthula, Y. Elmehdwi, and W. Jiang. k-nearest neighbor classification over semantically secure encrypted relational data. *Knowledge and Data Engineering, IEEE Transactions on*, 27(5):1261–1273, 2015.

[86] B. K. Samanthula and W. Jiang. An efficient and probabilistic secure bit-decomposition. In *8th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 541–546, 2013.

[87] B. Schoenmakers and P. Tuyls. Efficient binary conversion for paillier encrypted values. In *EUROCRYPT*, pages 522–537. Springer-Verlag, 2006.

[88] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612 – 613, November 1979.

[89] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. *Machine Learning in Cyber Trust*, pages 247–276, 2009.

[90] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *Theory of Cryptography*, pages 457–473. Springer, 2009.

[91] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy (SP'07)*, pages 350–364. IEEE, 2007.

[92] J. Vaidya and C. Clifton. Privacy-preserving top-k queries. In *ICDE*, pages 545–546. IEEE, 2005.

[93] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of the* 29^{th} *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2010)*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[94] B. Wang, S. S. Chow, M. Li, and H. Li. Storing shared data on the cloud via security-mediator. In *International Conference on Distributed Computing Systems-ICDCS 2013*, 2013.

[95] J. Wang, H. Ma, Q. Tang, J. Li, H. Zhu, S. Ma, and X. Chen. Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. *Computer Science and Information Systems*, 10(2):667–684, 2013.

[96] J. Wang, H. Ma, Q. Tang, J. Li, H. Zhu, S. Ma, and X. Chen. Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. *Computer Science and Information Systems*, 10(2):667–684, 2013.

[97] P. Williams, R. Sion, and B. Carbunar. Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In *ACM CCS*, pages 139–148, 2008.

[98] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis. Secure knn computation on encrypted databases. In *ACM SIGMOD*, pages 139–152, 2009.

[99] X. Xiao, F. Li, and B. Yao. Secure nearest neighbor revisited. In *IEEE ICDE*, pages 733–744, 2013.

[100] L. Xiong, S. Chitti, and L. Liu. K nearest neighbor classification across multiple private databases. In *CIKM*, 2006.

[101] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[102] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Symposium on Foundations of Computer Science*, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.

[103] H.-J. Yu, H.-S. Lai, K.-H. Chen, H.-C. Chou, J.-M. Wu, S. Dorjgochoo, A. Mendjargal, E. Altangerel, Y.-W. Tien, C.-W. Hsueh, et al. A sharable cloud-based pancreaticoduodenectomy collaborative database for physicians: Emphasis on security and clinical rule supporting. *Computer methods and programs in biomedicine*, 2013.

[104] P. Zhang, Y. Tong, S. Tang, and D. Yang. Privacy preserving naive bayes classification. *Advanced Data Mining and Applications*, pages 730–730, 2005.

[105] Y. Zhu, R. Xu, and T. Takagi. Secure k-nn computation on encrypted cloud data without sharing key with query users. In *Cloud Computing*, pages 55–60. ACM, 2013.

VITA

Yousef M. Elmehdwi received the Bachelor degree in computer science from Benghazi University (formerly known as the University of Garyounis), Libya, in 1993 and the Master degree in information technology from Mannheim University of Applied Science, Germany, in 2005. He received the Ph.D. degree in computer science from Missouri University of Science and Technology, Rolla, Missouri, in December 2015.