Privacy preserving query processing on outsourced data via secure multiparty computation

Do, Hoang Giang

2018

http://hdl.handle.net/10356/73577

https://doi.org/10.32657/10356/73577

NANYANG TECHNOLOGICAL UNIVERSITY

Privacy Preserving Query Processing on Outsourced Data via Secure Multiparty Computation

Do Hoang Giang

School of Computer Science and Engineering

2017

Privacy Preserving Query Processing on Outsourced Data via Secure Multiparty Computation

Do Hoang Giang

School of Computer Science and Engineering
Nanyang Technological University

A thesis submitted to the Nanyang Technological University in fulfilment of the requirement for the degree of

Doctor of Philosophy

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor and teacher, Dr. Ng Wee Keong. It has been an honor to be his Ph.D. student. I am truly privileged to have had the chance to learn from him ever since I was a young and inexperienced undergraduate student. His wise guidance helped me during all of my research. His constructive criticism constantly motivated me to improve, and his careful approach to mentoring me allowed me to grow up. He has shown limitless patience when working with me and has taught me countless invaluable life lessons.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Wang Lipo and Prof. Alwen Tiu, for their encouragement, insightful comments, and hard questions.

In addition, I also want to thank all my friends and collaborators, and the technicians in Nanyang Technological University for their unconditional help. I gratefully acknowledge the scholarship for my Ph.D. from Nanyang Technological University. Last but not least, my deepest love and gratitude are devoted to all of my family members: my parents, my brothers, and especially my wife. Their love will be an endless motivation for me to strive forward and help me overcome the difficult times in my life.

Abstract

Recent advances in technology have given rise to the popularity and success of many data-related services. This new paradigm allows the client to reduce the cost of operations by providing cost-efficient architectures that support the storage and intensive computation of data, and hence increases the throughput of businesses. However, these promising data services incur multiple and challenging design issues, considerably due to the leakage of confidential data. Losing control over the hardware typically means giving the rights of data access to a third party; as a result, the client faces new threats coming from the server-side.

Typical data-management service providers should not be fully trusted, thus storing encrypted data needs to be considered for high-level security assurance. Another potential threat is employees who do not follow the company's privacy policies and may, intentionally or unintentionally, reveal sensitive client information. Even when the provider claims to enforce strict policies pertaining to privacy, there is still a chance that the database systems are vulnerable to malicious external attacks.

This thesis aims at investigating privacy–preserving solutions for various important data query classes in different ubiquitous scenarios. The security issues of existing secure data processing protocols are also discussed. We focus on the provision of a rigorous se-

curity guarantee when processing data and answering queries. Cryptographic techniques from multi-party secure computation are leveraged to enhance security. Specifically, our proposed research objectives are as follows:

- The security problem will be analyzed under the semi-honest secure multi-party computation model. The semi-honest model assumes all the participating parties correctly follow the protocol specifications but actively collect information from the data storage and data processing protocols to discover confidential data.
- Security requirements for various secure data processing models are proposed to ensure strong confidentiality protection. In this thesis, by considering access pattern and query privacy requirements, we aim to address the security limitations of the existing solutions.
- Security requirements for various secure data processing models are proposed to ensure strong confidentiality protection. In this thesis, by considering access pattern and query privacy requirements, we aim to address the security limitations of the existing solutions.
- We investigate various secure query processing algorithms in different ubiquitous scenarios:
  (i) *Secure Conjunctive Matching* - A solution supports conjunctive queries over an encrypted numerical dataset. An extension to support range queries is also proposed.
  (ii) *Boolean Keyword Search* - A scheme allows the client to securely evaluate a boolean expression on a keyword set for an encrypted outsourced corpus of documents.

(iii) *Multi-dimensional Range Query* - a set of protocols support multi-dimensional range queries over a set of points of high dimensional space. The high dimensional space represents the multidimensional datasets of numerical domains.

(iv) *Secure Confidential Information verification.* - a framework for verifying personal or confidential information against a set of criteria. The proposed framework addresses a number of shortcomings of the current state of the process of physical document verification

These protocols are proposed, analyzed, and evaluated under the semi-honest model and with the proposed security requirements.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The security of data processing is a vital problem concerning nearly all aspects of Internet–connected systems. Outsourced data storage and personal information verification are two examples of ubiquitous situations where data confidentiality is one of the most basic requirements.

For outsourced data storage, clients outsource the storage of private data from their own infrastructure to a remote cloud server. This paradigm offers the user an opportunity to drastically reduce the costs of storing and processing data. Outsourcing data storage, as well as data management to the clouds, is useful for many services including law enforcement, finance, healthcare, etc. However, data outsourcing deprives the data owners of direct control of their data, and that brings new crucial security risks. Since there are many possibilities for data leakage to occur at the server side, the user should not fully trust the cloud for data privacy. One possibility of data leakage is corrupt employees who do not follow privacy policies. They may intentionally or unintentionally reveal sensitive information such as personal health records, financial transactions, or personal contacts, etc. Even when the cloud service provider claims to enforce sufficient policies to prevent such privacy violations, there is still a chance that cloud computing systems may be vulnerable to external malicious attacks. Once intrusions take place, sensitive data may be publicly exposed. Finally, due to the privacy regulations of certain countries, cloud data may be required to be shared with certain third parties. Therefore, storing plaintext data in the cloud creates the danger of full exposure of your data.

While outsourced storage settings address the issue of data privacy of the enterprise, personal information verification requires individual confidentiality. Physical document verification is a necessary task for the process of reviewing applications in many services, such as loans, insurances, and mortgages. This process consumes a large amount of time, money, and human resources. Consider a loan or an insurance application as an example. The applicants are usually required to provide numerous documents to certify their relevant personal information, such as a birth certificate, statement of monthly income, marriage certificate, medical records, and so on. All these documents are stored in the provider's database. If the client applies for multiple schemes or subscriptions, multiple copies of his or her personal data are stored in different places. Since data can be leaked from the server, storing personal information in multiple third-party databases is not recommended.

1.2 Research Challenges

Data encryption has been introduced to address the issue of data confidentiality. However, at the same time, it also creates a new challenge for data management.

In the context of outsourced data storage, a natural starting point is to encrypt the data stored on the server that only the client holds the secret key. When data passes to or from the server, it is encrypted/decrypted at the client side. This approach is promising because an attacker at the server can access only encrypted data and thus does not see the data content. However, it also creates a new challenge– the data usability of the remotely encrypted data. Since the data owner should have the ability to query and obtain useful data when needed, the simple encryption method is insufficient in fulfilling this basic requirement. The trivial solution is to retrieve back the whole encrypted dataset, and decrypt and filter it according to the query. However, this requires an excessive amount of bandwidth and processing, especially for large databases, and defeats the purpose of data outsourcing.

For the task of personal information verification, standard data encryption may be applied to personal documents; then the client outsources encrypted personal data to a data storage server for maintenance. When the client wishes to share the information with a certain third party (e.g., loan or insurance providers), he/she first instructs the storage server to send the encrypted documents and then provides the decryption key to the third party for verification. While this approach eliminates the cost of maintaining all physical documents on the client side, it still exposes the privacy of the client to the service provider. At the same time, service providers are still required to provide an excessive amount of human resources to verify all of the documents.

The second challenge to the problem of secure data processing in different contexts is access pattern. When processing the outsourced data, the program control flow and memory-access behavior may reveal sensitive information clients wish to hide. S.Islam *et* *al.* [59] showed that data access pattern leakage could lead to the disclosure of a significant amount of sensitive information. This leakage and its consequences apply to a variety of realistic contexts. For example:

(i) A hospital outsources its patients' medical records to a remote server. Later when a disease is spreading, the hospital may want to check whether certain patients have had the disease synonyms before. If an observer learns which records the hospital is looking for, the observer can infer details about the client's medical records.

(ii) A client outsources their documents to a remote data storage and wishes to provide them to a third party for personal verification. The third party may be a loan provider or an insurance company that provides a financial product to the client. If an observer learns which the data records are accessed by the service provider, the observer is able to know the service that the client wishes to apply for and can even infer the financial condition of the client.

The third challenge when considering the task of secure data processing is query privacy. Query privacy is orthogonal to access-pattern privacy. While access-pattern privacy addresses the confidentiality of the query results, query privacy aims to protect the content of the query itself. More specifically, access-pattern privacy preserves the privacy of the data records when processing, while query privacy considers the privacy of the query initiator's interest. A secure data processing framework should take this aspect into consideration. It also applies to a number of realistic scenarios. Consider the case of a criminal investigation; once the observers are able to determine that several queries are the same, with certain side-knowledge they will be able to infer the main interest of the investigation.

1.3 Contribution

This thesis attempts to protect data confidentiality for data processing tasks. We assume the attacker has the power of observing all the data movement and memory access during query-answering processing. That assumption is much stronger than in the state-of-art systems. Most existing works on data privacy only consider passive attackers who only can view the database at rest. While the proposals of existing works only investigate how to withstand weak outsider attacks, our proposal takes insider threats into consideration.

We leverage the computation model of secure multi-party computation that was introduced by C. Yao [86] and homomorphic encryption techniques to propose solutions to the problem of privacy preserving data processing. We adopt a use-inspired approach to design our solutions. Different protocols have been proposed to address different practical use-cases. For each proposed secure protocol, we analyze its correctness, security, and complexity. We also investigate their practicality based on experimental results. Our meta-strategy for building a secure data processing system is as follows:

(i) We identify and understand a few probable use cases.

(ii) We identify the basic operations (called primitives) that enable a wide class of applications and translate the requirements for these primitives into research problems.

(iii) We design and build a practical system that uses these primitives as building blocks.

More concretely, we have investigated and proposed various secure query processing algorithms in different ubiquitous sceanarios:

- *Secure Conjunctive Matching* - A solution supports conjunctive queries over an encrypted numerical dataset. An extension to support range queries is also proposed.

- *Boolean Keyword Search* - A scheme allows the client to securely evaluate a boolean expression on a keyword set for an encrypted outsourced corpus of documents.
- *Multi-dimensional Range Query* - a set of protocols support multi-dimensional range queries over a set of points of high dimensional space. The high dimensional space represents the multidimensional datasets of numerical domains.
- *Secure Confidential Information verification.* - a framework for verifying personal or confidential information against a set of criteria. The proposed framework addresses a number of shortcomings of the current state of the process of physical document verification.

Furthermore, our work on the problem of secure personal information not only addresses the aforementioned challenges in the problem of secure data processing but also suggests an opportunity for new digital services. The details of the work are described in Chapter 6.

1.4 Thesis Roadmap

This thesis is organized as follows:

Chapter 2 considers the evolution and current state of the field of encrypted data storage and processing, and the related areas of secure multiparty computation.

Chapter 3 presents a simple but secure protocol that supports exact matching on encrypted data. We also discuss how to extend it to support range queries.

In Chapter 4, we present a systematic approach that supports complex Boolean keyword search. The protocol leverages bloom-filter data structure and homomorphic encryption techniques.

Chapter 5 discusses a solution for secure multidimensional range queries. We make use of the bucketization method to support multidimensional range query, while cryptographic techniques are utilized to protect the bucket content.

Chapter 6 describes our proposed systematic solution for the problem of secure personal information verification. Our approach not only provides a cost-efficient and secure solution for the process of document verification but also creates an opportunity for a new service.

Chapter 7 presents the conclusion of this thesis and proposes some potential research directions.

2

Background

2.1 Introduction

This chapter gives an overview of most relevant and influential works related to secure searching and processing data in remote databases and data storages. The chapter is organized as follows. Section 2.2 contains definitions and basic concepts that are widely used in secure data processing research. The existing solutions to the problems of query processing on encrypted data, including exact matching, range queries, and complex queries are discussed in Sections 2.3, 2.4 and 2.5. Section 2.6 presents the two most important techniques for designing secure query processing protocols, homomorphic encryption and Oblivious RAM. Finally, Section 2.7 provides a brief review of the basic concepts of the Secure Multiparty Computation (SMC) model. In this thesis, we leverage the SMC model for both designing and analyzing solutions.

2.2 Definitions

This thesis contains problem statements, definitions, and discussions that rely on basic concepts widely used by researchers of secure data processing. Before we proceed, these basic concepts sh ould be properly defined for both structured and unstructured data.

Definition 2.1 ***Domain***. *The domain of a certain variable v is the set of the possible values of the variable v.*

Definition 2.2 ***Attribute***. *An attribute is the pair of* $\langle Identifier, Domain\rangle$. *When we refer to an attribute of a particular data record, we mean this is a pair of* $\langle Identifier, Value\rangle$ *where the* $Value$ *belongs to the* $Domain$ *of the attribute.*

Definition 2.3 ***Keyword***. *A keyword is an index entry that identifies a specific data record or document. A data record or document may contain multiple keywords.*

Definition 2.4 ***Data Record***. *In this thesis, we use the word data record in the context of structured data. A record is a tuple of pairs* $\langle Identifier; Value\rangle$, *such that there are no two pairs with the same* $Identifier$ *and* $Value$ *belonging to the* $Domain$ *of the attribute of the same* $Identifier$.

Definition 2.5 ***Document***. *In this thesis, we use the word document in the context of unstructured data. A document is electronic matter that contains information. A document is usually identified by a set of keywords.*

Definition 2.6 ***Dataset***. *A dataset is a set of data records or documents. We use this term in a general context for both structured and unstructured data.*

2.3 Secure Single Equality Test

An equality test allows a client to perform an exact match for a single keyword search query. Research on exact keyword searches on encrypted data began a decade ago with two seminal works by Song *et al.* [79] and Boneh *et al.* [13]. Since then, the field of study has gained significant attention from academia.

Based on the nature of search operations, the works on this topic can be divided into two broad categories: *full-domain* and *index-based search*. The former requires sequential scanning through every data item in order to test criteria. The full-domain search takes linear complexity to cover the whole dataset and is flexible since the query can be anything and can be defined on the fly. On the other hand, index-based search, as shown in Figure 2.1, first characterizes every document in the dataset by a list of keywords. The query is later evaluated based on the indices built upon the keyword set. There are two basic ways to construct the indices, namely forward indexing and inverted indexing.

$$\begin{tikzcd}
 & \underline{\textbf{\textit{User}}} & & \underline{\textbf{\textit{Database}}} \\
\boxed{\text{Upload}} & I = \mathsf{BuildIndex}_K(\mathcal{DB}, \mathcal{W}) \arrow[rr, "I||\mathsf{Enc}(M_1,\ldots,M_n)"] & & I||\mathsf{Enc}(M_1,\ldots,M_n) \\
\boxed{\text{Query}} & T = \mathsf{Trapdoor}_K(f) \arrow[rr, shift left, "T"] & & id = \mathsf{Search}(I,T) \arrow[ll, shift left, "\mathsf{Enc}(M_{id})"]
\end{tikzcd}$$

Figure 2.1: General Model of a Secure Index-based Search

Based on the types of encrypted systems, secure keyword search systems can be divided into: symmetric settings and asymmetric settings. This classification is orthogonal to the previous method.

Symmetric Setting. This direction of research leverages symmetric encryptions to allow the data owner, i.e. the secret key holder, to create a searchable ciphertext and trapdoors for encrypted search.

- Song *et al.* [79] presented the first practical scheme for a private search that supports full-domain search. The intuition of the method is to encrypt each word separately and embeds a special hash function into the ciphertext. Later, to search for a certain word, the data owner generates a hash corresponding to the search query and sends it the server. The server can extract the hash embedded in the ciphertext to perform an exact equality test with the query. The idea is depicted in Figure 2.2. The protocol firstly applies a pseudorandom permutation on each word w_i, the result is divided into two parts L_i and R_i. They are latter XORed with a random seed S_i, and a key-hashed value of it to produce a ciphertext C_i. To search for a keyword w, the client has to provide $E(w)$ to the server. The server can compute $E(w) \oplus C_i$ for each word i. If the result can be separated into two parts s and $F_k(s)$, that means the ciphertext match the keyword search with a high probability.

  Brinkman *et al.* [21] utilized this approach to support a secure search in XML data. It was also applied by Popa *et al.* [70] in designing a practical system that supports complex queries.

  While the protocol is IND-CPA secure, it does not cover the new security requirements for secure query processing on outsourced data. The scheme leaks the potential positions of the matched keywords in a document. Hence after several rounds, an observer is able to learn the words inside the document by statistical analysis.

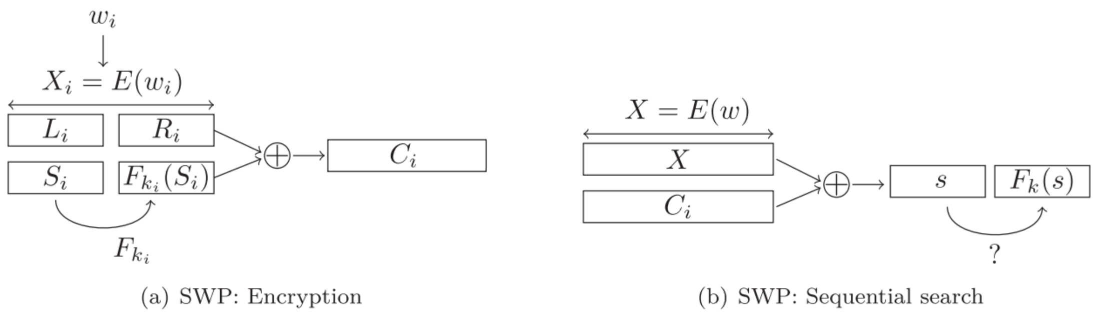


Figure 2.2: Song *et al.* [79] for Private Keyword Search

- Goh *et al.* [45] addressed some of the limitations of the previous method by examining ways to build a secure index to search through encrypted data in a symmetric setting. The idea of their method is to add a secure forward index for each document. More specifically, a Bloom-filter is used as a per-document index. Goh *et al.* encoded each keyword in a document into the Bloom filter by applying a pseudo-random function twice on the keyword. The unique document identifiers ensures that all the Bloom Filters look different, even for two documents with the same keyword set. This approach was later used by Chang and Mitzenmacher [27] to support users with limited storage space and bandwidth.

  While the index structure helped Goh to overcome the shortcomings of Song *et al.* [79] that reveals the positions of the matched keyword, it still poses certain limitations. Firstly, the curious observer can easily learn if the same keyword is queried multiple times just by observing the content of the queries. Secondly, the access pattern leakage is still not completely eliminated. The curious observer still is able to know which subset of the documents satisfies the query. As discussed in the previous chapter, this leakage might end up with the disclosure of a significant amount of sensitive information.

- Curtmola *et al.* [34] proposed two efficient schemes using an inverted index structure. The index structure consists of a linked list L per distinct keyword and a look-up table to access the first node of the linked lists. The pointer and the identifier of the document are randomized and encrypted. To search for a particular keyword, the client and the server perform multiple interactive rounds. At each round, the client obtains one document identifier and the pointer to the next one in the inverted list. The inverted index structure allows efficient lookups for a particular keyword. However, it still is not able to satisfy the requirements of query and access pattern privacy. Moreover, it is difficult to leverage the same approach to support complex queries such as conjunctive keyword searches.

Asymmetric Setting. The asymmetric encryption approach allows every entity to produce a searchable ciphertext without any explicit user authorization, but only the user can issue meaningful search queries and decrypt the data.

- Boneh *et al.* [13] proposed the first searchable encryption scheme using a public key system called the Public-key Encryption with Keyword Search (PEKS). The idea for their scheme is to use identity-based encryption (IBE) in which the keyword acts as the identity. The scheme allows multiple entities to write and encrypt data (with the public key) while only one client is able to search the encrypted data (with the private key). Figure 2.3 describes the algorithms presented by Boneh *et al.* for PEKS. This seminal paper made use of Boneh and Franklin's work on IBE [14] for the construction. Different IBE public-key techniques have been explored so far to achieve this setting, including Anonymous IBE and Hierarchical IBE [1].

  Since the trapdoor for a keyword is never refreshed, it violates the query privacy requirements. Byun *et al.* [23] and Yau *et al.* [87] discussed this issue and proposed an attack against this vulnerability called the offline keyword guessing attack. The attack is straightforward. The attacker firstly creates the trapdoors for the chosen keywords, and then analyzes the results of the queries.

The scheme requires two groups G_1, G_2 of prime order p and a bilinear map $e : G_1 \times G_1 \to G_2$ between them, two hash functions hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : G_2 \to \{0,1\}^{log\ p}$.

- KeyGen: The input security parameter determines the size, p, of the groups G_1 and G_2. The algorithm picks a random $\alpha \in Z^*$ and a generator g of G_1. It outputs $A_{pub} = [g, h = g^\alpha]$ and $A_{priv} = \alpha$.
- $PEKS(A_{pub}, W)$: First compute $t = e(H_1(W), h^r) \in G_2$ for a random $r \in Z$. Output $PEKS(A_{pub}, W) = [g^r, H_2(t)]$.
- $Trapdoor(A_{priv}, W)$: output $T_W = H_1(W)^\alpha \in G_1$.
- $Seach(A_{pub}, C, T_W)$: Let $C = [A, B]$. Test $H_2(e(T_W, A)) \overset{?}{=} B$.

Figure 2.3: The method of Boneh *et al.* [13] for Private Keyword Search

- Baek *et al.* [5] argued that the need for a secure channel for trapdoor transmission as in Boneh *et al.* [13] is not realistic in many application scenarios. They addressed this limitation by introducing a server public/private keypair to the PEKS scheme, where the encryption of the keywords is done under both public keys of the client and a specific server through an aggregation technique [16]. However, the scheme is still vulnerable to offline keyword recovery attacks.

- Baek *et al.* [4] raised the natural question of whether the same public key can be used for the encryption of both keyword and the messages. They gave a concrete construction for this integration by combining the variation of ElGamal cryptosystem [41] and PEKS.

- Bellare *et al.* [8] presented the concept of efficient full-domain searchable deterministic encryption using an asymmetric cryptosystem. However, in this approach, the ciphertext is deterministic so that a curious server can test any guessed plaintext by an encryption operation.

- Fuhr and Paillier [39] introduced a full-domain searchable encryption protocol that is secure under the random oracle model. Hofhinz and Weinreb [56] revisited this concept and propose a solution using the standard model. However, both papers only considered the information leakage from the ciphertext and do not address the privacy of the trapdoors or query privacy.

Categorization. The intended search functionality–full-domain search or index-based search–and the choice between symmetric and asymmetric cryptosystems have strong a impact on each other. Table 2.1 summarizes the classification of secure single equality tests on encrypted data. We emphasize that this summary table only contains the representative schemes mentioned above, and it is not inteneded to be a full list of existing solutions.

Table 2.1: Summary of *Single Equality Test* categorizations

	Symmetric-Setting	Asymmetric-Setting
Full-Domain	[21,46,79]	[8,39,56,85]
Index-based	[27,34,45]	[1,4,5,13]

2.4 Secure Multidimensional Range Query

A multidimensional range query allows inequality evaluations of the value of data records. More formally, if we consider each data record is a point in a high-dimensional space, the multidimensional range query should return all the points within a certain hyper-rectangle corresponding to the query's criteria.

There are essentially three general approaches for tackling multi-dimensional range queries on encrypted data: special data structure for range query evaluation, bucketization-based scheme, and order-preserving encryption-based techniques.

Special data structure for range query evaluation. Boneh *et al.* [18][4] proposed a general public-key approach to support comparison, subset, and range queries on encrypted data by using hidden vector encryption, whose search complexity is $O(mT)$, where m and T are the number of attributes and number of discrete values for each attribute, respectively. Moreover, the ciphertext size is relatively large owing to the usage of composite-order bilinear map groups [17], which makes it infeasible in many applications.

Shi *et al.* [77] proposed a scheme that supports a conjunction of range queries over multiple attributes (i.e., multi-dimensional range queries). The idea is to encrypt each data record as a point in multi-dimensional space. The query processing is equivalent to testing whether a point falls inside a hyper-rectangle for each data record. The authors used a segment tree data structure to represent the ranges for each dimension. Anonymous identity-based encryption is applied at each node of the tree with a different key. For a certain query, the client needs to reveal the keys corresponding to each range on each dimension. Hence, all the attributes on the range will be revealed after successful decryption at the server.

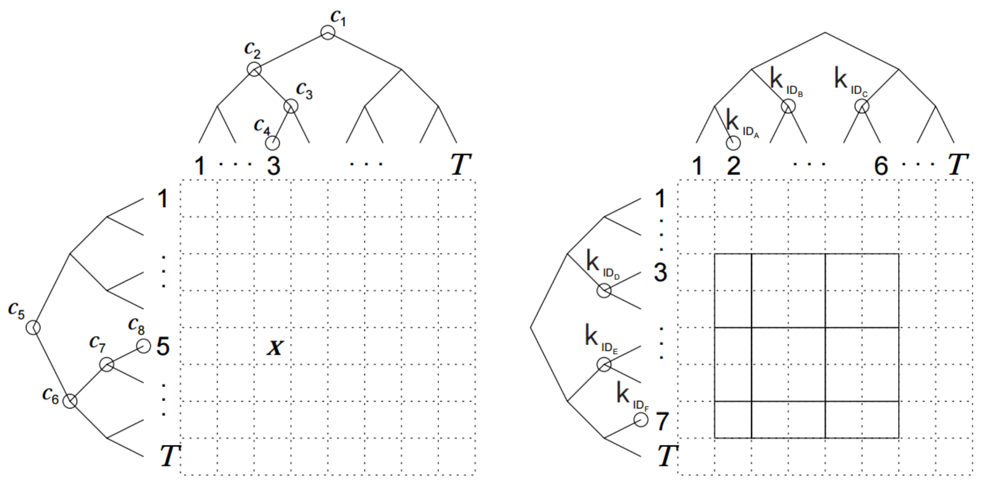


Figure 2.4: Tree-based Representation by Shi *et al.* [77] for Multidemsional Search

Subsequently, different tree-based approaches have been proposed to tackle the problem of secure multi-dimensional range queries. Lu [64] proposed a range search scheme on encrypted data by leveraging predicate encryption and B^+ tree. He extended the original scheme to support multi-dimensional range queries by replacing B+ trees with kd-trees in the implementation. Wang *et al.* [82] presented a scheme for evaluating multi-dimensional range queries by using asymmetric scalar-product preserving encryption and R-trees. However, these two methods lead to single-dimensional privacy leakage.

Bucketization-based scheme. Bucketization-based data representation for query processing in an untrusted environment was originally leveraged by Hacigumus *et al.* [54]. Their bucketization simply involves a data partitioning step based on equi-depth or equi-width partitioning to support single-dimensional range queries. The queries are mapped to a set of buckets that contain any value satisfying the range of the query. The original queries are translated into bucket-level queries, which request the encrypted buckets containing the desired values. Several studies have attempted to reduce bucket costs (i.e., false positives) while preserving the anonymity of the data set [3, 58]. However, only until the work of Hore [57], the bucketization-based method was extended to support secure multidimensional range queries. Their method tries to cluster d-dimensional space into various hyper-rectangles, and bucket indexing is performed on the clustered hyper-rectangles.

Order-preserving encryption based techniques. An order-preserving encryption scheme is a deterministic cryptosystem whose encryption algorithm produces ciphertexts that preserve the numerical ordering of the plaintexts. Roughly speaking, for any two ciphertexts c_1 and c_2 corresponding to plaintexts p_1 and p_2, respectively, if $p_1 \leq p_2$ then it is guaranteed that $c_1 \leq c_2$. Order-preserving encryption was first proposed by Agrawal *et al.* [2] to support range queries on encrypted data. When the dataset is encrypted by an order-preserving encryption scheme, the result of a single-dimensional range query is simply determined by the encryptions of the two end-points. The concept of order-preserving encryption was efficiently revised with formal security analysis by Boldyreva *et al.* [11, 12] and Mavroforakis *et al.* [65]. Although these techniques are highly efficient, they provide a low level of privacy. As such, traditional efficient indexes can be built directly on encrypted data and queried in the same manner as plaintexts. Nevertheless, order-preserving encryption is deterministic and hence suffers from inherent distribution leakage. In addition, it inevitably leaks the ordering of the data.

2.5 Secure Complex Query Processing

It is worth noting that the abovementioned techniques are only suitable for processing specific queries (e.g., exact matching, range query) and are not directly applicable for evaluating complex queries such as combinations of multiple sub-queries. The first attempt toward the solution for secure complex query processing was conjunctive search. A conjunctive keyword search system allows a client to find documents containing several keywords with a single query. The naive approach is to perform multiple individual keyword searches with the server. The server returns the intersection of all results. This approach leaks the individual results for each keyword search, that is, which documents contain each particular keyword, and may allow the observer to run statistical analysis.

Along the symmetric settings, Golle *et al.* [50] gave the first construction of conjunctive keyword searches. Their idea is to assume that there are special keyword fields associated with each document. The keyword only appears in these certain fields. The construction leaks the number of encrypted keywords. To address the different shortcomings of Golle *et al.* [50], various protocols have been proposed. Ballard *et al.* [7] proposed a construction for secure conjunctive keywords searches, which are secured using the standard model. The Byun *et al.* [22] protocol reduces the communication and storage cost, while Ryu and Takagi [75] enhanced the trapdoor size. Recently, Cash *et al.* [26] proposed the first sublinear construction supporting conjunctive queries. Their construction leverages the idea of inverted index approach of Curtmola *et al.* [34].

With asymmetric settings, Park *et al.* [68] studied the problem of public key encryption with a conjunctive keyword search. The idea of the solution is to use bilinear maps to aggregate the tags for the encrypted index and the conjunctive query and compared the aggregated results. The protocols leak the position of the keyword in the query, hence, two queries are distinguishable.

Boneh and Waters [18] proposed a new primitive, called hidden vector encryption, to support a number types of queries including conjunctive, subset, and range queries. However, their technique is costly and complex to implement [77]. Katz *et al.* [60] extended the list with disjunctions, polynomial equations, and inner products, but the complexity of the approach is not improved.

Popa *et al.* [71] proposed CryptDB, a system that supports a wide range of queries. Figure 2.5 illustrates the architecture of CryptDB. Their idea is to encrypt each data record with an onion of encryption schemes with the outermost layer providing maximum security, whereas the innermost layer provides more functionality but with weak security. In order to support multiple types of queries, the client needs to provide the cloud the corresponding secret keys to decrypts the outer layers so that the necessary operation can be performed with the inner layers. CryptDB has some major drawbacks. Firstly, the

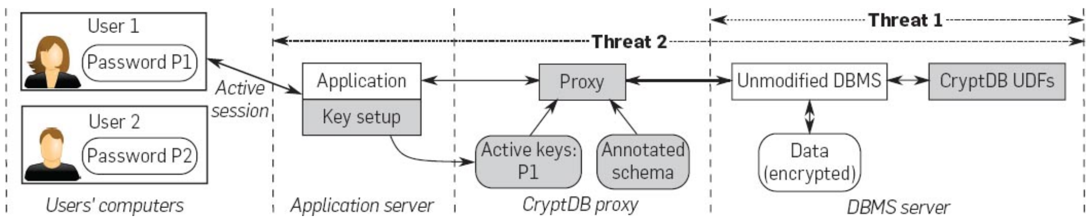


Figure 2.5: CryptDB architecture

system reveals different types of information at different layers, once a layer is decrypted there is no way for the system to recover to the maximum security level. Morever, multiple onions may have to generated for each data item which makes the approach very expensive.

Recently, Samanthula *et al.* [76] presented a systematic approach to support complex query evaluation over secure encrypted data. The authors leverage the secure multiparty computation to design the solutions. Our solutions presented in this thesis share the similar approach with their works. However, Samanthula *et al.* [76]'s solution requires the client to share the private key with a third party, which is undesirable in many applications.

2.6 Computing on Encrypted Data

In this section, we briefly review two general cryptographic techniques that we utilize to design and implement our solutions for secure data processing. The first one is homomorphic encryption. Homomorphic encryption techniques hold the promise of attaining truly secure ways to gather, share, and process data. The second is oblivious RAM, which is a cryptographic primitive that obfuscates a client's access pattern (address, data, read/write) in an untrusted memory source.

2.6.1 Homomorphic Encryption

Homomorphic encryption (HE) is a form of encryption that allows arithmetic operations to be carried out on ciphertexts and generates an encrypted result that, when decrypted, matches the result of the operations performed on the plaintexts. Based on the specific types of computation they support in the ciphertext space, the homomorphic encryption schemes are generally categorized into two classes *additive homomorphic encryption* and *multiplicative homomorphic encryption*.

Additive Homomorphic Encryption. For any two plaintexts m_1 and m_2, consider two operators $\times$ and $+$ on the ciphertext and plaintext domain, respectively, an additive HE scheme *Enc* satisfies

$$Enc(m_1, r_1) \times Enc(m_2, r_2) = Enc(m_1 + m_2, r_{12}),$$

where $Enc(m, r)$ is the encrypted value of plaintext m with another random input r. This random input r is required to ensure the semantic secure[1] of the cryptosystem.

Based on the additive homomorphism, we have the pseudo *homomorphic multiplication* property of the cryptosystem as follows:

$$Enc(k \times m_1, r_1) = Enc(m_1, r_2)^k.$$

Various additive HE cryptosystems have been proposed, such as the Pallier cryptosystem [67], GoldwasserMicali Encryption scheme [49], Damgard and Jurik cryptosystems [35], and the Okamoto-Uchiyama [66] cryptosystem. For simplicity, this thesis makes use of the Pallier cryptosystem as the additive HE encryption method. We will present details of this protocol in the subsequent subsection.

Multiplicative Homomorphic Encryption. A cryptosystem is said to have the multiplicative homomorphic property if for any two plaintexts m_1 and m_2, considering the two operators $\times$ and $+$ on the ciphertext and plaintext domain, respectively,the following property holds:

$$Enc(m_1, r_1) \times Enc(m_2, r_2) = Enc(m_1 \times m_2, r_{12}),$$

where $Enc(m, r)$ is the encrypted value of plaintext m with a other random input r. RSA [74] and ElGamal [40] are two examples of cryptosystems that have this property.

[1]Informally, a semantically secure encryption scheme is a probabilistic, polynomial-time algorithm such that given the ciphertext, an adversary cannot deduce any additional information about the plaintexts.

Fully Homomorphic Encryption. The two aforementioned classes of homomorphic encryption are homomorphic with respect to a single operation. The natural question was whether there exists an encryption scheme that is fully homomorphic, namely, homomorphic with respect to both addition and multiplication. In 2009, the first fully HE was discovered by Gentry [42]. Principally, a fully HE allows for arbitrary computations on encrypted data. Computing on encrypted data means that if a user has a function f and want to obtain $f(m_1, ..., m_n)$ for some inputs $\{m_1, ..., m_n\}$, it is instead possible to compute the encryptions of these inputs, $\{c_1, ..., c_n\}$, obtaining a result which decrypts to $f(m_1, ..., m_n)$. However, despite significant advances [19, 29, 43], fully homomorphic encryption remains largely impractical [55].

Addressing the performance limitation of fully homomorphic encryption, somewhat homomorphic schemes have been proposed. A somewhat homomorphic scheme is an HE scheme that is capable of evaluating a limited number of homomorphic operations. This thesis utilizes the Boneh-Goh-Nissim Encryption Scheme [17] for the purpose of implementing a somewhat HE scheme.

2.6.1.1 The Paillier Cryptosystem

The Paillier cryptosystem, named after and invented by Pascal Paillier [67], is a probabilistic public-key encryption scheme. The cryptosystem relies on the computational hardness assumption of a novel mathematical problem called composite residuosity. The decision version of this problem class assumes that no polynomial-time algorithm can distinguish the *n-th* residues modulo N^2 with a non-negligible probability.

The Paillier cryptosystem is composed of three algorithms: key generation, encryption and decryption.

Key Generation. The Paillier key generation algorithm inputs the security parameter ℓ and outputs a public and private key: (pk, sk).

(i) Generate two large prime p and q of $\ell/2$ bits.

(ii) Compute $n = pq$ and $\lambda = lcm(p-1, q-1)$ where lcm is least common multiple function.

(iii) Select random integer g where $g \in Z^*_{n^2}$.

(iv) Compute the following modular multiplicative inverse

$$\mu \;=\; (L(g^\lambda \;mod\; n^2))^{-1} \;mod\; n$$

, where

$$L(u) \;=\; \frac{u-1}{n}$$

(v) Output key-pair $pk = (n, g)$, and $sk = (\lambda, \mu)$.

Encryption. The Paillier encryption algorithm inputs a message m and a public key $pk = (n, g)$ and outputs a ciphertext $c \in Z^*_{n^2}$.

- Choose a random $r \in Z^*_{n^2}$
- Output ciphertext $c \;=\; g^m \cdot r^n \;mod\; n^2$

Decryption. The Paillier decryption algorithm inputs a ciphertext c and a keypair (pk, sk) and outputs a decrypted message m.

$$m \;=\; L(c^\lambda \;mod\; n^2) \cdot \mu \;mod\; n$$

.

Homomorphism. Assume that we have two messages m_1 and m_2, which are encrypted using a Paillier public key pk. The Paillier cryptosystem satisfies the additive homomorphic encryption property:

$$Enc_{pk}(m_1, r_1) \times Enc_{pk}(m_2, r_2) = (g^{m_1} \cdot r_1^n)(g^{m_2} \cdot r_2^n) = g^{m_1+m_2}(r_1 r_2)^n = Enc_{pk}(m_1+m_2, r_1 r_2)$$

We implemented Pailler using Java with BigInteger library. The performance of the implemementation is presented in Section 3.5 of Chapter 3.5. It is also used as the building blocks for developing the solutions for almost solutions presented in this thesis.

2.6.1.2 The ElGamal Crypttosystem

The ElGamal cryptosystem is a public-key encryption scheme invented by Taher Elgamal [40] based on the Differ-Hellman key exchange. The cryptosystem is defined over a cyclic group G. Its security depends upon the difficulty of computing discrete logarithms in group G. The ElGamal cryptosystem is composed of three algorithms as follows:

Key Generation. The ElGamal key generation algorithm outputs public and private key: (pk, sk).

(i) Choose a multiplicative cyclic group G of order p with generator g, where p is big prime number.

(ii) Choose a random $x \in \{0, \ldots, p-1\}$.

(iii) Compute $y = g^x \ \ modp$.

(iv) Output key pair $pk = (G, p, g, y)$, and $sk = x$.

Encryption. The ElGamal encryption algorithm inputs a message m and a public key $pk = (G, p, g, y)$ and outputs a ciphertext c.

(i) Choose a random $r \in \{0, \ldots, p-1\}$

(ii) Output ciphertext $c = (a, b)$. where

$$a = g^r \ mod \ \ p.$$

$$a = m \cdot y^r \ mod \ p.$$

Decryption. The ElGamal decryption algorithm inputs a ciphertext $c = (a, b)$ and a keypair (pk, sk) and outputs a decrypted message m.

$$m \;=\; b \cdot a^{-x} \;\; mod \;\; p.$$

Homomorphism. Assume that we have two messages m_1 and m_2, which are encrypted by an ElGamal cryptosystem. This ElGamal cryptosystem satisfies the multiplicative homomorphic encryption property:

$$Enc_{pk}(m_1, r_1) \times Enc_{pk}(m_2, r_2) = (g^{r_1}, m_1 \cdot y^{r_1})(g^{r_2}, m_2 \cdot y^{r_2 s})$$

$$= (g^{r_1+r_2}, (m_1 \cdot m_2) \cdot y^{r_1+r_2} = Enc_{pk}(m_1 \cdot m_2, r_1 + r_2)$$

2.6.1.3 BonehGohNissim Encryption Scheme

The Boneh-Goh-Nissim [17] cryptosystem, or BGN for brevity is a somewhat homomorphic public key encryption scheme. It allows both additions and multiplications on the ciphertext space. The multiplication is constructed based on a bilinear map over the elliptic curve groups.

Let G_1, G_2 be additive groups and G_T be a multiplicative group. All groups are of prime order p. Let g_1, g_2 be generators of G_1, G_2 respectively. A bilinear map $e : G_1 \times G_2 \to G_T$ has the following properties:

- Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in Z$, then $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degenerate: $e(g_1, g_2) \neq 1$.

For practical purposes, e has to be efficiently computable. In cases when $G_1 = G_2 = G$, the bilinear map is called symmetric. Furthermore, if G is cyclic, the map e will be commutative.

The BGN cryptosystem is described as the follows:

Key Generation. BGN key generation outputs a keypair as the following steps:

(i) Generate a tuple (q_1, q_2, G, G_1, e), where q_1 and q_2 are two large primes, G is a cyclic group of order q_1q_2.

(ii) Let e be a bilinear map $e : G \times G \rightarrow G_1$.

(iii) Set $N = q_1q_2$.

(iv) Pick two generators $g, u \in G$, compute $h = g^{uq_2}$. Hence, h is the generator of G with order q_1

(v) Output $pk = \{N, G, G_1, e, g, h\}$, and $sk = q_1$.

Encryption. BGN key generation inputs a message $m \in 1, \cdots, T,\ T < q_2$ and outputs a ciphertext $c \in G$.

- Choose a random $r \in \{0, \ldots, N\}$
- Output ciphertext $c = g^m h^r \in G$.

Decryption. BGN decryption inputs a ciphertext $c \in G$ with the private key sk and outputs a message m. We have:

$$c^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m.$$

To recover the message m, we compute the discrete logarithm of c_1^q to the base g^{q_1} . Since $0 < m < T$, this takes expected time $O(\sqrt{N})$ using Pollard's method [69].

Homomorphism. Consider c_1 and c_2 as the ciphertexts of two messages $m_1, m_2 \in \{0, 1, \cdots, T\}$ respectively. We have

$$c_1 = g^{m_1} h^{r_1} \in G,\ \ c_2 = g^{m_2} h^{r_2} \in G$$

then,

$$C = c_1 \times c_2 = g^{m_1} h^{r_1} g^{m_2} h^{r_2} = g^{m_1+m_2} h^{r_1+r_2} \in G$$

is an encryption of $m_1 + m_2$.

In addition, we also can compute the product of two encrypted messages using the bilinear map $e : G \times G \to G$. More specifically, if $C = e(c_1, c_2)$ then,

$$\begin{aligned} C &= e(g^{m_1}h^{r_1}, g^{m_2}h^{r_2}) \\ &= e(g^{m_1+\alpha q_2 r_1}, g^{m_2+\alpha q_2 r_2}) \\ &= e(g,g)^{m_1 m_2} h^{m_1 r_2 + m_2 r_1 + \alpha q_2 r_1 r_2}. \end{aligned}$$

and is a uniformly distributed encryption of $m_1 m_2$ in G_1. Moreover, C still has additive homomorphism in G_1.

2.6.2 Oblivious RAM

Oblivious RAM (ORAM) is a cryptographic primitive that conceals memory access patterns. The idea of ORAM is to continuously reshuffle the memory and translate the address of each memory access to a set of randomized memory locations. It can be proved that these randomized memory locations are independent of, and thus leak no information about, the logical addresses that are actually requested.

Square root ORAM The study of ORAM was initiated with the "square-root" solution by Goldreich [46]. The author addressed the problem of software protection, i.e., the prevention of unauthorized copying. The idea for the solution is to design a remote secure processor that allows hiding a program's control flow determined by the access pattern to the main memory. A trivial solution is to scan all the memory locations of each access, which is infeasible for a complex program.

The Goldreich solution partitions the server memory into two regions: a main region of $O(N)$ blocks and a shelter of size $(O\sqrt{N})$ blocks. Initially, the main memory is filled with $O(N)$ real data blocks and $(O\sqrt{N})$ dummy blocks. All blocks are encrypted by a semantically secure encryption scheme and randomly shuffled. To perform an access, the client first scans the shelter region. If the block is found there, the client performs a random access to the unread dummy block in the main memory. Otherwise, the client computes the location by a pre-defined hash function of the requested block and accesses it. After that, the client re-encrypts the accessed block in the main region and appends it to the shelter.

After $O(\sqrt{N})$ accesses, the main region runs out of dummy blocks and needs to be reshuffled. We denote B as the size of the block since the scheme requires $O(BNlogN)$ (due to the oblivious sorting approach) data cells to be transferred for every $(O\sqrt{N})$, and thus the solution requires a $O(B\sqrt{N}logN)$ amortized bandwidth.

Hierarchical ORAM Goldreich and Ostrosvky [48] presented the famous 'hierarchical' solution to achieve polylogarithmic memory bandwidth overhead. The key idea of the solution is to, instead of using one main and shelter memory region, organize the server memory into a hierarchy of buffers whose sizes grow at a geometric rate as in Figure 2.6. Each permuted buffer at a particular level acts as the main region in the square root approach and thus is parameterized by a hash function.

To access a block, each level in the data structure is accessed as if it were in the main region. Each buffer slot acts like a bucket of size $O(logN)$ blocks. The hash function now maps blocks into random buckets. Each block accessed is appended to the top level of the data structure. When a certain level fills, it is merged with the lower level.

The bucket of size $O(logN)$ is downloaded atomically by the client for each data access. There are $O(logN)$ levels in the hierarchy, and together with the merge operations, the solution requires a $O(log^3N)$ amortized bandwidth.

Goldreich and Ostrosvky's hierarchical construction inspired many subsequent works. Williams and Sion [83] reduced the bandwidth overhead to $O(log^2N)$ using oblivious sorting, with $O(\sqrt{N})$ of client storage. Goodrich *et al.* [51, 52] used cuckoo hashing to further reduce the bandwidth overhead to O(log N) with $O(N^\epsilon)$ $(\epsilon > 0)$ client storage.

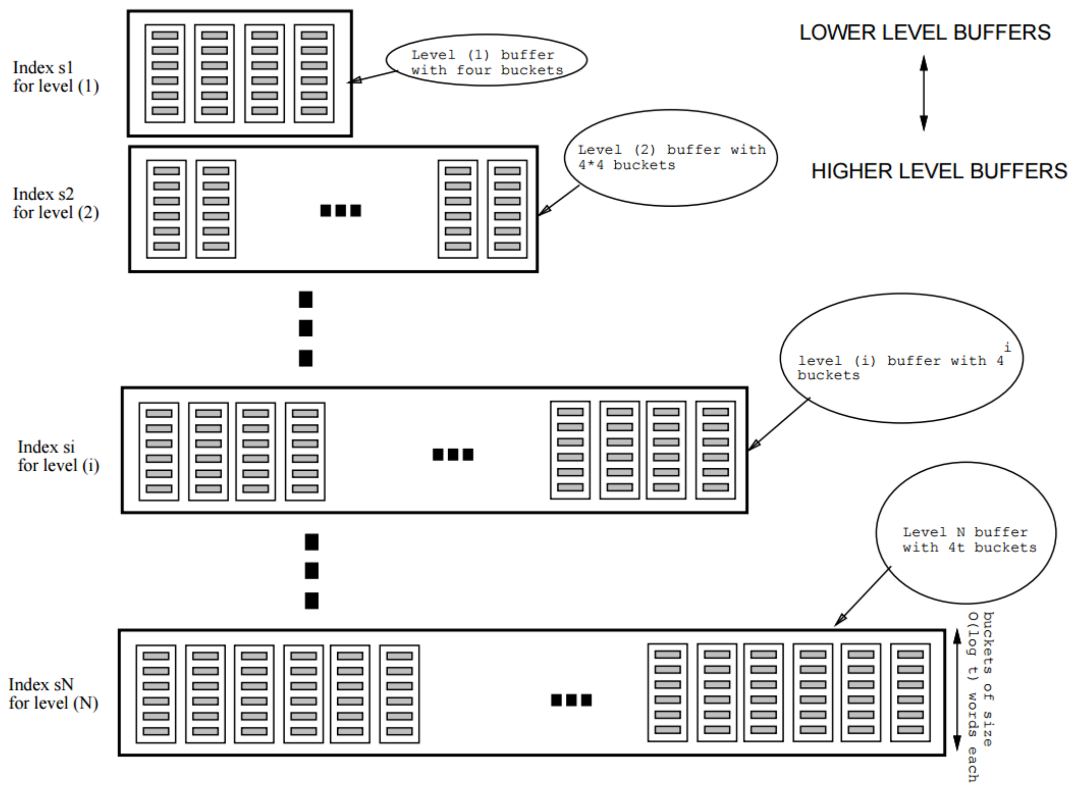

Index s1 for level (1)
Level (1) buffer with four buckets
LOWER LEVEL BUFFERS
HIGHER LEVEL BUFFERS
Index s2 for level (2)
Level (2) buffer with 4*4 buckets
level (i) buffer with 4^i buckets
Index si for level (i)
Level N buffer with 4t buckets
Index sN for level (N)
buckets of size O(log t) words each

Figure 2.6: The hierarchical ORAM of Goldreich and Ostrosvky [48]

Tree-based ORAM A recent breakthrough is the development of tree-based ORAM by Stefanov *et al.* [80]. In this construction, each data block is mapped on to a random path of a binary tree. This concept is different from the idea of hierarchical ORAM where each block stored in a level has complete freedom on where they will be reshuffled into the next level. The tree path containing a particular block associates with a certain leaf node and that address of the leaf node is stored in a position map at the client side.

To access a block, the client first looks up the position map and reads all the buckets on that path. For the binary tree construction, the path contains $O(logN)$ data buckets. After the access operation, the data block will be re-assigned a new path or a new leaf node. It is re-encrypted and put at the root of the tree. Hence, some eviction procedures are needed to percolate the blocks towards the leaves of the tree and prevent the root from overflowing.

Each data access requires $O(log^2 N)$ bandwidth overhead because it requires fetching of the $O(logN)$ nodes where each node is a bucket of $O(logN)$ data blocks. Moreover, the original solution requires $O(N)$ client side storage for the position map. Shi *et al.* [78] proposed ORAM recursion to reduce the client storage to $O(1)$. The key idea is to store the position map at the server side as a second ORAM. This may be performed recursively until the client storage is $O(1)$. Clearly, the access operation now includes looking up all the position map ORAMs in addition to the main data ORAM, which increases the bandwidth overhead to $O(log^3 N)$.

Stefanov *et al.* [81] propose Path-ORAM, an extremely simple and efficient ORAM protocol. It only contains 16 lines of pseudocode, shown in Figure 2.7 below. Path ORAM for the first time achieved an $O(logN)$ memory bandwidth overhead under small client storage.

```
Access(op, a, data*):
1:  $x \leftarrow \mathsf{position}[\mathsf{a}]$
2:  $\mathsf{position}[\mathsf{a}] \leftarrow \mathsf{UniformRandom}(0 \ldots 2^L - 1)$
3:  for $\ell \in \{0, 1, \ldots, L\}$ do
4:      $S \leftarrow S \cup \mathsf{ReadBucket}(\mathcal{P}(x, \ell))$
5:  end for
6:  $\mathsf{data} \leftarrow$ Read block $\mathsf{a}$ from $S$
7:  if $\mathsf{op} = \mathsf{write}$ then
8:      $S \leftarrow (S - \{(\mathsf{a}, \mathsf{data})\}) \cup \{(\mathsf{a}, \mathsf{data}^*)\}$
9:  end if
10: for $\ell \in \{L, L-1, \ldots, 0\}$ do
11:     $S' \leftarrow \{(\mathsf{a}', \mathsf{data}') \in S : \mathcal{P}(x, \ell) = \mathcal{P}(\mathsf{position}[\mathsf{a}'], \ell)\}$
12:     $S' \leftarrow$ Select $\min(|S'|, Z)$ blocks from $S'$.
13:     $S \leftarrow S - S'$
14:     $\mathsf{WriteBucket}(\mathcal{P}(x, \ell), S')$
15: end for
16: return $\mathsf{data}$
```

Figure 2.7: Path-ORAM description [81]

The notation for the Path-ORAM description is defined in Figure 2.8 below.

N	Total # blocks outsourced to server
L	Height of binary tree
B	Block size (in bits)
Z	Capacity of each bucket (in blocks)
$\mathcal{P}(x)$	path from leaf node x to the root
$\mathcal{P}(x, \ell)$	the bucket at level ℓ along the path $\mathcal{P}(x)$
S	client's local stash
position	client's local position map
$x :=$ position[a]	block a is currently associated with leaf node x, i.e., block a resides somewhere along $\mathcal{P}(x)$ or in the stash.

Figure 2.8: Path-ORAM notation [81]

This thesis makes use of Path-ORAM for the implementation of ORAM due to its simplicity and efficiency.

2.7 Secure Multiparty Computation

In this thesis, we leverage the techniques as well as the security model from secure multiparty computation to provide solutions to the problem of secure query processing. This section reviews the concepts of secure multiparty computation (SMC). In the mid-1980s, C. Yao [86] introduced the idea of securely computing any two-party functionality in the presence of dishonest adversaries. Since then, various privacy-preserving protocols have been proposed to address different classes of computation problems in the context of private data. This section does not intend to describe the details of the mathematical model of SMC, but aims to provide an intuitive interpretation of the basic concepts. Interested readers can find more details in many books and papers such as "Foundations of Cryptography" by Goldreich [47] and "Composition of Secure Multi-Party Protocols" by Lindell [63].

2.7.1 What is Secure Multiparty Computation

We consider the scenario of n parties who want to compute the output of a function f, a scenario such as a comparison evaluation. Each party i owns a private input x_i. After a procedure, they obtain the result $f(x_1, \ldots, x_n)$. The result may belong to one certain party or be shared between the participating parties as $(y_1, \ldots, y_n)$ depending on the purpose of the protocol. This procedure is called multiparty computation (MPC). Let us assume that they compute the output of function f in a secure way such that the ith party knows y_i, but gets nothing more than that. In this case, the procedure is called secure multiparty computation (SMC).

To illustrate the intuition of SMC, we consider a classic example of Yao's millionaire's problem [86]. Two millionaires want to know who is richer without disclosing their fortunes. Yao proposed a secure comparison method such that each millionaire knows the result (who is richer) but nothing else. The protocol acts as a black box such that there is a trusted party who is able to receive two numbers from the two parties and reveal the comparison result and nothing else. While the presence of a trusted party is unrealistic in many applications, SMC has become a very prominent solution for these scenarios and has attracted a significant attention from both the industry and academia.

2.7.2 Semi-Honest Adversary

There are several notions of security with various degrees of strength. In this work, we focus on the special case of private computation, which assumes that the adversary is passive (also called semi-honest or honest-but-curious) and cannot modify the behavior of corrupted parties. In the semi-honest model, each party strictly follows and executes the specified protocol and provides the correct input data when executing the protocol. However, the attacker is subsequently free to compute additional information based on his or her private input, output and messages received during the execution of the secure protocol. Loosely speaking, the semi-honest model is only concerned with the information learned by the adversary, and not with the effect of misbehaviors may have on the protocol's correctness.

The stronger notion of the adversary is a fully-malicious one who can modify the corrupted parties' behavior arbitrarily. However, in this thesis, we do not consider this notion of the adversary. Although the assumptions of the semi-honest adversarial model are weaker than those of the malicious model, we insist that this assumption is realistic under the problem settings. Firstly, participating parties do not want to deviate from the protocol specifications since it is also against their interest. Moreover, it is often non-trivial for one party to maliciously deviate from a particular protocol that may be hidden in a complex process.

The semi-honest party model has been widely accepted and applied in many secure data processing protocols such as [9, 33] as generally, each party does not wish to collaborate with any malicious parties for risks of compromising data privacy.

2.7.3 Security Requirements

In an SMC protocol, the result of the computed function is inevitable. For example, in Yao's millionaire problem, if a millionaire knows he is richer, he can then conclude that the other's fortune is smaller than his. Any additional information leakage beside this is undesirable. Hence, the security of an SMC protocol should be properly defined.

A naive approach to define the security of an SMC protocol is to exhaustively enumerate all the security requirements for each participating party and the independence of inputs. However, since there is no universal standard for these requirements, it is hard to know whether the list is complete. Hence, defining the security of an SMC protocol is not easy. In this thesis, we will take a different approach.

We make use of the simulation-based approach as suggested by Goldreich [47]. Intuitively, a protocol is secure if whatever can be computed by a party participating in the protocol can be computed based on its input and output only. More concretely, for each participating party, its view in the protocol execution should be simulatable by a simulator with only access to its input and output.

2.7.4 The Composition Theorem

Composition theorems state that if a protocol π is secure and it makes direct call to an ideal functionality g during execution, and σ is a secure protocol that computes g, then π is secure if it replaces g by σ in its definition. It essentially means we are able to construct a secure protocol by sequentially composing secure functions. A detailed presentation on the theorems and their proof can be found in Canetti [25].

3 Private Query Processing via Proxy-Assisted Computation

3.1 Introduction

In this chapter, we present a simple but secure scheme that supports conjunctive query evaluation over the numerical domain. More concretely, we consider the scenario where the client outsources a multidimensional dataset of numerical values, and later he/she is able to query to obtain the useful subset of the dataset. The scheme allows exact matching and range query in the numerical domain.

3.1.1 Motivating Scenario

In this chapter, we consider the process of alert correlation in network security as the motivating scenario. In a network system, suspicious events are recorded by a variety of sensors (e.g., IDS, system logs, antiviruses, etc.). When an intrusion takes place, security analysts are required to examine past related events. These events and the current attacks typically share the same characteristics such as source and destination ports, source subnet, and destination IP addresses. This chapter presents a solution that allows organizations to outsource their network encrypted event logs, and later are able to issue queries to retrieve partial but useful records from the database.

We leverage an additive homomorphic encryption (i.e., Paillier cryptosystem) to present a solution that supports exact matching in a numerical domain such as querying a destination IP or a communication port. To support range queries, we propose a prefix–encoding extension to transform them to exact matching queries. Our solution allows the dynamically updating of the outsourced dataset.

3.1.2 Organization

The rest of this chapter is organized as follows. The main content of the chapter starts with the formulation of the problem in Section 3.2. The section also discusses the requirements for the problem. Section 3.3 presents the solution for the abovementioned problem. We begin with the description of the exact matching query and then describe the extension to support range query. We analyze the security as well as the complexity of the proposed approach in Section 3.4. Section 3.5 presents experimental evaluations of the proposed solution. The final section concludes the chapter. Throughout this chapter, we make use of the alert correlation process to illustrate our proposed approach.

3.2 Problem Formulation

3.2.1 Problem Statement

We consider the scenario where a client poses a multidimensional dataset $D = \{d_1, \cdots, d_n\}$ of n-records. Each data record consists of m numerical attributes $\{t^1, t^2, \cdots, t^m\}$. We denote t_i^j as the value of jth attributed of data record d_i.

The client outsources his/her database (i.e., a set of data records) to the cloud server in order to save the local storage costs. In our example of network security, the client outsources the network event logs that are generated by different sensors (e.g. IDS, system logs, antiviruses, etc.). The dataset may contain sensitive information of business operations. Hence, a client-side encryption is required to ensure data confidentiality.

Let D' denote the encrypted database stored in the server. The client wants to securely retrieve a subset of the data from D' in the cloud using his/her private query Q. In this chapter, we assume the query is represented in a conjunctive normal form as follows:

$$Q = s_1 \wedge s_2 \cdots \wedge s_k \rightarrow \{0, 1\}$$

The input to query Q is an encrypted data record d'_i. Here, s_i is a simple equality predicate whether attribute j_i of the data record has value $x_i : t_{j_i} \overset{?}{=} x_i$. The output of the query evaluation on the record d'_i is 1 if it satisfies all the equality predicates at the same time and 0 otherwise.

The goal of the solution presented in this chapter is to facilitate clients efficient retrieval of data records from D' (stored at the server) that satisfy Q in a privacy-preserving manner. More formally, we define a privacy preserving query processing (PPQP) protocol as follows:

$$PPQP(D', Q) \rightarrow S,$$

where S is a set of indices that denotes the output set of records that satisfy Q. That is, $\forall t' \in S, Q(t') = 1$ always holds.

3.2.2 Privacy Requirements

When designing the solution for a secure conjunctive query processing system, we consider the following security requirements:

(i) Query Confidentiality. No information about the dataset should be revealed to a passive observer at the server.

(ii) Query Privacy. Given two queries, one query is created by a client, and the other one is randomly simulated, an observer should not be able to distinguish (with probability non-negligibly exceeding 1/2) them.

(iii) Server Unlinkability. A client should not be able to detect any changes in the server's database, except for the information implied (by the client) from the results of the queries.

(iv) Access Pattern. The identification of the matching or requested records should not be revealed to the server.

Cristofaro *et al.* [32] pointed out that query privacy and server unlinkability requirements are necessary in many practical scenarios. In general, these requirements prevent one party (client or server) from noticing whether the other party's inputs have been changed. Hence, the risks of privacy leaks is minimized. Without the third requirement, the client always can determine if the server has updated its database in the duration between two queries. In the alert correlation example, the necessary records may be shared or queried by different parties or different departments. The unlinkability requirement prevents unnecessary information to be exposed. Besides that, unless query privacy is guaranteed, the server can learn that the client sends two of the same queries in two interactions; in that case, if one query's content is exposed, the other would be immediately leaked.

3.3 Private Conjunctive Query Processing

3.3.1 Solution Overview

We propose a system architecture consisting of three general parties.

- *The client* is the data owner who outsources the database D to the server.
- *The server* provides data storage and computing services. The server may be an untrusted entity, and it may try to collect information about clients from the dataset.
- *The proxy* is a computing server. The proxy will communicate with the server, receive some immediate results to evaluate, filter the encrypted results and return them to the client.

The workflow of our proposed approach consists of three main stages:

- Setup. In this stage, the client generates two public keypairs so that they can be used for encryption, decryption, and communication during the later stages.
- Encryption. The client encrypts the dataset at the client-side before outsourcing the data to the server. The semantic security property of the cryptosystem provides client data confidentiality.

- Query Preparation. To retrieve the necessary data from the server-side, the client constructs a query, encrypts it and sends the encrypted query content to the server.
- Query Evaluation. Receiving the encrypted query content, the server and the proxy initiate an interactive process so that at the end of the stage, the client will receive the index set of matching data records.

3.3.2 Basic Secure Conjunctive Query Processing

This section describes the construction of our basic secure conjunctive query processing scheme:

(i) *Setup.* The client generates two Paillier keypairs (pk_1, sk_1) and (pk_2, sk_2). The first keypair is utilized to encrypt data and query content at the second and third stage. The latter keypair is used for the query evaluation stage. We denote $Enc(pk_i, m)$ as the encryption of a message m under the public key pk_i. The client shares sk_1 with the proxy while keeping sk_2 in secret.

(ii) *Encryption.* For each data record $d_i \in D$, the client encrypts the attribute values of the data record one by one using the public key pk_1, and outsources the whole encrypted dataset to the server. The server stores one encrypted data record in the following form:

$$\langle id, Enc(pk_1, t_{id}^1), Enc(pk_1, t_{id}^2), \cdots, Enc(pk_1, t_{id}^m)\rangle$$

,where id is the index of the data record in the server. The server index allows an authorized client to obliviously retrieve the data record from the server database (i.e. Oblivious RAM techniques [81]).

(iii) *Query Preparation.* The client wishes to query for the data record satisfying the following requirements:

$$Q \leftarrow t^{j_1} = x_1 \wedge t^{j_2} = x_2 \wedge \cdots \wedge t^{j_k} = x_k$$

The query Q comprises k expressions ($k \leq m$). Each expression is a simple equality predicate on one attribute j_i: $t^{j_i} \overset{?}{=} x_i$.

We can represent a query Q by two components the attribute identifier, and the query values:

$$Q = \{[j_1, j_2, \cdots, j_k],\ [x_1, x_2, \cdots, x_k]\}$$

. The second component is encrypted under the public key pk_1 and sent to the server:

$$Enc(Q) = \{[j_1, j_2, \cdots, j_k],\ [Enc(pk_1, x_1), Enc(pk_1, x_2), \cdots, Enc(pk_1, x_k)]\}$$

.

(iv) *Query Processing.* Protocol 1 presents the query processing protocol specification.

After the execution of the algorithm, the client receives the indexes of all satisfied records. He/she can subsequently perform private information retrieval to obtain the necessary information.

Correctness. The soundness of the protocol follows the additive homomorphic property of the additive homomorphic cryptosystem:

$$\begin{aligned} A^{j_i} &= Enc_{pk}(t^{i_j}) \times {c_1}^{-1}\ (modN^2) \\ &= Enc_{pk}(t^{j_i} - x_i)\ (mod\ N^2) \end{aligned}$$

Hence,

$$C = \prod_{i=1}^{k} B^{j_i} = \prod_{i=1}^{k} (A^{j_i})^{r_i} = Enc(\sum_{i=1}^{k} r_i(t^{j_i} - x_i))(mod\ N^2)$$

Algorithm 1: Basic Query Processing Algorithm

```
**Input:** The attribute index set $\{[j_1, j_2, \cdots, j_k]$ and the value set
    $\{c_1 = Enc(pk_1, x_1), c_2 = Enc(pk_2, x_2), \cdots, c_k = Enc(pk_1, x_k)\}$
**Output:** Indexes of satisfied records
1  for each record, Server does do
2      for i = 1 to k do do
3          Compute $A_{j_i} = Enc_(pk_1, t^{j_i}) \times {c_1}^{-1}\ (modN^2)$;
4          Compute $B^{i_j} = (A^{j_i})^{r_i}\ (modN^2)$ where $r_i$ is a random positive number ;
5      end
6      Compute $C = \prod_{i=1}^{k} B^{j_i}$;
7      Send $C$ and encrypted index $Enc(pk_2, id)$ to Proxy;
8  end
9  for each received record, Proxy do
10     Send $En(pk_2, id)$ to Client if $Dec(C) = 0$;
11 end
12 Client decrypts $Enc(pk_2, id)$ and receives the result;
```

C is a linear combination of k random positive numbers. Hence, once a record passes the test when $C = 0$, it must satisfy the conditions: $t^{j_1} = x_1$ and ... $t_{j_k} = x_k$, with negligible false positives.

Security. Data records stored in the server's database are encrypted by the Paillier cryptosystem, which is semantically secure. Hence, the server cannot distinguish any particular alert from other ones. Besides that, the server cannot gain any information from the query as well as it cannot distinguish any two queries from each other since all the information, that it obtains, is an encrypted form of the query. Therefore, client privacy and client unlinkability requirements are satisfied. The only inevitable information leakage is the number of records that the server receives from each client.

On the other hand, for the client, without colluding with the proxy, all data he/she receives during the protocol are the indexes of records that match the query's conditions. Hence, he/she does not get any information of the records from the server except for the records that satisfy the query. Similarly, given two queries, the client is not able to detect any changes in the server's database, except for the information implied (by the client) from the results of the queries.

For each record, the proxy receives the value of $Dec(C) = \sum_{i=1}^{k} r_j(t^{j_i} - x_i)$. If there is $\exists j : t^{j_i} \neq x_i$, that value is a random number. Hence, the only knowledge the proxy obtains is the number of matched records. In order to resolve this issue, the server can encrypt a random number of "fake" matched immediate results (encryption of value zero), and send them to the proxy.

3.3.3 Secure Range Query Processing

In a particular context, the protocol can be modified to support certain range queries. One possible scenario is the alert correlation process described in Section 3.1.1. Upon the arrival of a new alert data, a client often wants to know past related intrusion alerts. Rather than examining the exact source and destination IP addresses, security analysts are often more interested in the subnet addresses.

An IP4 address can be represented by a 32-bit integer. A subnet address query such as "10.10.1.0/24" (a class C network) can be represented by a range query as follows:

$$168427776 \leq x \leq 168428031$$

In order to support such a type of range query, we perform a prefix encryption as the follows:

(i) Transform the IP4 address to binary representation. For example, 10.10.1.24 is recorded as 1010000010100000000100011000.

(ii) Represent different subnet classes of interest in binary format. For example, 10.10.1.24/32 (exact address), 10.10.1.24/24 is represented as 1010000010100000000100011000, and 1010000010100000000100000000, respectively.

(iii) Encrypts the IP4 address multiple times, each encryption for a subnet class of interest.

(iv) The query process for particular subnet addresses can be performed using the similar method described in Section 3.3.2.

The intuitive concept behind the prefix encryption in this scenario is that the boundaries of the range and the actual data point have the same prefix in binary representation.

This concept can also be extended to support the more general problem of an integral range query such as a 32-bit integer range query. In order to support such queries, each data point should be encrypted multiple times (i.e., 32 times) for each prefix in the binary representation of this data point. The query for an arbitrary range $[a, b]$ is now translated into a union of prefix range queries.

We illustrate this approach by a simple example of the query: $50 \leq x \leq 100$. We only consider an 8-bit integer representation for simplicity.

(i) Represent the boundaries in binary format: $50 = 00110010,\ 100 = 01100100$.

(ii) Create the queries for the binary prefix of these boundaries. There will six queries as follows: $00110010, 00110100, 00111000, 01000000, 01010000, 01100001$.

Here, we note that the union of six queries would include all the numbers between 50 and 100. However, we also emphasize that while this approach is not impossible, it is impractical in almost all other scenarios. Even for such simple query, we are required to process six separate subqueries, and create a huge overhead for data storage (i.e., 32 times larger).

Chapter 5 discusses a much more efficient method to address the problem of range as well as multi-dimensional range queries.

3.4 Discussion

Since for each query, the data server performs an indistinguishable computation process for each data record, there is no information it can gain from the query processing process. Hence, the access pattern and the query privacy requirements are satisfied.

This chapter presents a protocol to support conjunctive exact matching queries and a certain type of range query. This query model is useful for the process for alert correlation. A more complex query that is required in the alert correlation process can also be supported by this model. More specifically, sequential information can be included as the query criteria. This knowledge is encoded as a binary matrix, and homomorphic encryption is utilized to deliver the corresponding records. Interested readers can refer to our paper [36] for more details.

While the scheme we present in this chapter provides rigorous security in terms of access pattern, and query privacy, etc., there is one known shortcoming that affects the practicality of the solution, that is, the private key sharing with the proxy. The client is normally reluctant to share the key with a third party, even when they claim that there will be no collusion with the data server.

3.5 Experimental Results

This section examines the performance of our proposed solution to the problem of secure conjunctive equality tests on encrypted remote data. We firstly discuss the implementation of the Paillier cryptosystem–an additive homomorphic encryption scheme. In the subsequent chapters, when we refer to the performance of the additive homomorphic encryption, we refer to the practical results we obtain in this section. Later the performance of the cryptosystem for the proposed solution is presented.

Paillier Implementation. We implemented a Paillier cryptosystem using Java BigInteger class. All the experiments were conducted using a Windows 10.0 machine with a 3-GHz processor and 16 GB of RAM. To examine the performance of the Paillier implementation, we performed 200,000 encryption/decryption operations with a varying key length of $\{512, 640, 768, 896, 1024, 2048\}$. These experiments were also used to test the correctness of the Paillier implementation. The input of the experiment is the SHA-256 value of the random text. The execution time for Paillier encryption/decryption is calculated by taking the average time from 200,000 operations and the results are graphed in Figure 3.1.

The results show that the encryption operation is slightly more expensive than the decryption. The experimental results also guide us on the performance of the Paillier cryptosystem in practice. With a medium-sized data of 1,000 to 10,000 records, it may take up to minutes just to encrypt and decrypt the whole dataset. We also note that the execution time of the addition operation in ciphertext space is much shorter than the execution time of the encryption/decryption operation. To perform one addition operation in 1024-bit length Paillier takes 0.019 ms.

Secure Conjunctive Query Processing. To investigate the efficiency of the proposed solution, we test a simulated network log dataset of 10,000 data records. The dataset contains multiple attributes: username, IP4 address, country, gender, datetime,

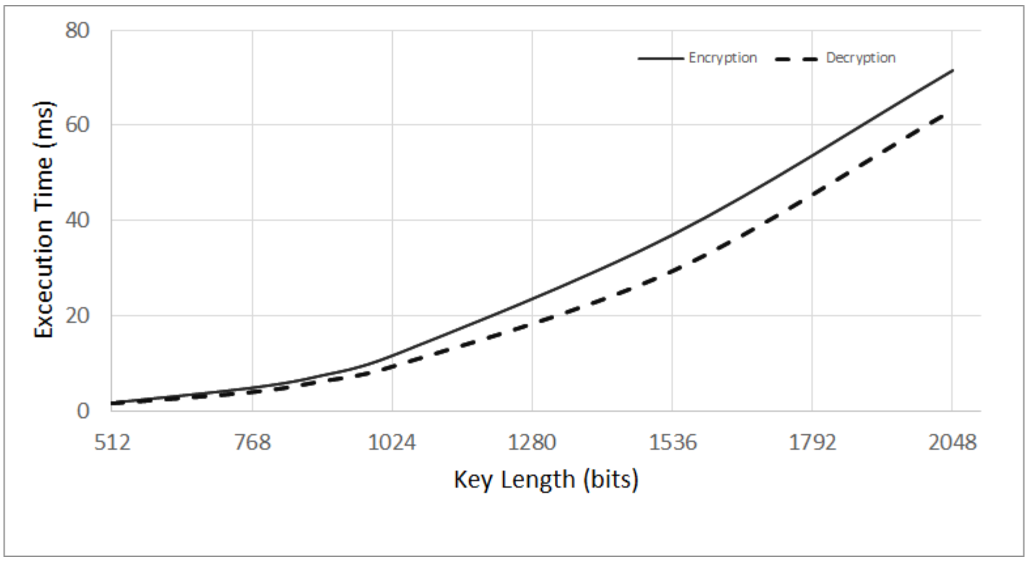


Figure 3.1: Execution times of Paillier cryptosystem's operations

and time interval. We conduct the experiment with different Paillier key lengths of $\{512, 1024, 2048\}$.

The data storage overheads when encrypting the dataset using different key lengths are 39.1, 78.1, and 156.2 times, respectively. The storage overhead is roundly doubled when the key length doubles. The execution time of the protocol is reported in seconds in Table 3.1.

For the standard 1024-bit length, the solution only takes around three minutes in order to return results of the whole dataset of 10,000 records. This is clearly acceptable in most applications. Moreover, we also note that the execution times scale at a magnitude of seven times when the key length doubles.

3.6 Summary

This chapter presents a simple but secure solution for the problem of conjunctive equality matching. The extension that supports range queries in certain scenarios is also discussed.

Table 3.1: Execution time of private query processing via proxy-assisted computation

No. of Records	512-bit	1024-bits	2048-bit
500	1.659	9.06	68.322
1000	3.172	18.026	149.56
2000	6.194	36.425	318.84
3000	8.815	56.483	438.941
4000	11.425	74.107	541.164
5000	15.308	97.252	701.143
6000	18.22	113.999	908.048
7000	20.872	133.758	1052.807
8000	24.253	151.372	1119.793
9000	27.126	166.588	1346.356
10000	29.849	190.15	1380.214

The solution reveals no information to the data server as well as the proxy. The security of the protocol is analyzed against the secure multi-party computation model. However, there is one certain limitation of the protocol. The protocol requires the client to share his/her private key with a third party (i.e., the proxy), which is unrealistic for many applications. In addition, the solution only supports conjunctive queries. In the subsequent chapters, we discuss different approaches that support more general queries.

4 Private Boolean Keyword Search on Encrypted Data

4.1 Introduction

In the previous chapter, we presented a simple but secure solution that supports conjunctive exact matching queries. This chapter aims to solve a more general problem: private Boolean keyword searches. The solution presented in this chapter allows the evaluation of any combination of Boolean functions over a set of keywords. The problem we address in this chapter is the same as the classical Boolean information retrieval problem.

We consider a scenario where an encrypted dataset contains a number of encrypted documents. Each document is associated with a certain set of keywords. An authorized user is able to perform search queries that are a combination of logic predicates of the keyword set. The output of the query processing is the index set of satisfied documents and nothing else to the data consumer. During the query processing, not only the outsourced data is kept private from the data storage provider, but also the user's input query remains confidential.

There are two general approaches to address the problem of secure query processing over encrypted data without downloading the entire dataset. The first approach deploys tamper-proof trusted hardware (which is either trusted or certified by the clients) on the cloud-side. The hardware provides a secure environment that allows the cloud to perform secure operations over the data in critical query processing stages. Along with this direction, Bajaj and Sion [6] leveraged the existence of trusted hardware to design TrustedDB, an outsourced database prototype that allows a client to execute SQL queries with privacy and under regulatory compliance constraints. However, the secure hardware is costly and may not be suitable for a general cloud computing paradigm, which typically makes use of cheap commoditized machines.

The second approach makes use of cryptographic protocols to perform operations over the encrypted data. Chapter 2 provides an extensive review of existing works on searching over encrypted data. We emphasize that existing works either only support single (or only conjunctive) keyword search or fail to provide rigorous privacy protection due to access pattern leakage. In this chapter, we present a secure protocol that not only preserves query privacy but also keeps the access pattern confidential.

4.1.1 Motivating Scenario

We consider an example of outsourcing personal email. Alice wishes to outsource her archived emails to free up her mobile device storage. However, these emails contain important information that she is required to use in the future. Each email belongs to some certain categories or includes several keywords such as “work”, “education”, “leave”, etc. Later, she wishes to search and retrieve the useful emails using combinations of these keywords.

4.1.2 Organization

We organize the chapter described by the following. In the next section, we review the existing works on secure query processing over encrypted data. Section 4.2 describes our problem statement, data model, and query model as well as the requirements for the designed framework. Section 4.3 reviews the necessary building blocks which are Bloom Filters and Oblivious Transfer. The proposed solution for complex Boolean query processing on encrypted data is presented in Section 4.4. The section systematically discusses four stages of the solution. Section 4.5 presents our experimental evaluations of the proposed sub-protocols. The final section discusses future works and concludes the chapter.

4.2 Problem Formulation

4.2.1 Problem Statement

In our problem settings, we consider three different parties: the data owner, the data server and the data consumer.

- *The Data Owner* outsources a set of documents. In order to ensure the data security, these documents are encrypted at the client side.
- *The Data Server* has the resposiblity to store the data owner's dataset and provide the search service to the data consumer.
- *Data Consumer* is authorized entity that has permission to search on the encrypted dataset and obtain partial but useful data content.

Let D be the data owner's dataset with n documents. Each document is associated with a certain set of keywords that enable it to be searched or retrieved efficiently. We assume that the data owner wishes to encrypt D using his/her key and outsources the encrypted data to a cloud so that later an authorized data consumer is able to search on the encrypted data. The input query is represented by a logical combination (i.e., negation, conjunction, and disjunction) of the keyword predicates. At the same time, several security issues should be addressed, such as data confidentiality, the privacy of query content, etc.

Formally, let $W = \{0,1\}^*$ be a universe of words and $D \subseteq W$ be the corpus. Let $Kw = \{w_1, \cdots, w_n\}$ denote a set of searchable keywords. The keyword set Kw is pre-defined (hence it is called a common reference keyword set). While Kw can be any searchable property set of the corpus D, for simplicity, we assume the problem as a general Boolean keyword search. That means $Kw \subseteq D$ and the predicate $d(w_i) = 1$ if and only if the document d contains the keyword w_i. Otherwise, $d(w_i) = 0$.

A Boolean query q_K contains a keyword predicate and a set of logical expressions $\wedge, \vee, \neg$. Let $d_C = \{d^{(1)}, \ldots, d^{(n)}\}$ be n documents stored in a server C. With a set of keywords $K \subseteq Kw$, we define a query $q_K : d \to \{0,1\}$ that takes a document d as input and outputs 1, if and only if d matches the criteria.

4.2.2 Security Requirements

In this chapter, we consider the following security requirements for the problem of private Boolean keyword searches on encrypted data:

- Data Confidentiality. The data are encrypted by a provably secure cryptosystem. Besides, during the query processing, the data server should not gain any new knowledge of the stored data.

- Query Privacy. At any point of time, the data consumer's query should never be revealed to the data server and the data owner.

- Access Pattern Privacy. Data access patterns of the data consumer should not be disclosed to the data server and the data owner. Data access patterns are the information about the documents that satisfy the query (even the attackers do not know the query content).

- End-user Privacy. At the end of the query processing protocol, only the satisfied results should be revealed to the data consumer and nothing else.

4.2.3 Comparision with Related Work

Section 2.5 briefly review representative approach to the problem of the secure complex query over the encrypted dataset. Table 4.1 shows the comparisons of the presented solution in this chapter with the existing ones on different security aspects.

Table 4.1: Comparision with Related Work

	Data Confi-dentiality	Query Privacy	Access Pattern	CNF & DNF	No Key Sharing
Golle *et al.* [50]	✓	✓	✗	✗	✓
Boneh *et al.* [18]	✓	✓	✗	✗	✓
Popa *et al.* [71]	✗	✗	✗	✓	✓
Do *et al.* [36]	✓	✓	✓	✗	✗
Samanthula *et al.* [76]	✓	✓	✓	✓	✗
Proposed method	✓	✓	✓	✓	✓

4.3 Building Blocks

4.3.1 Bloom Filter

A Bloom filter [10] provides a way to probabilistically represent set membership of elements using a small amount of space, even when the universe set is large. It represents a set $S = \{s_1, \dots, s_n\}$ of n elements by a space-efficient m-element array $B = \{B[1], B[2], \dots, B[m]\}$. A random set of hash functions $h_1, \dots, h_t$, where each function $h_i : \{0, 1\}^* \to [0, \dots, m]$ is chosen to associate with the Bloom filter B.

The filter algorithm is constructed as follows. The bit array B is initially set to 0. For each element $s_i \in S$, the bits corresponding to the positions $h_1(s_i), h_2(s_i), ..., h_t(s_i)$ are set. The same bit in the array may be set several times without any restriction. Figure 4.1 depicts how an element is inserted into a Bloom filter.

After the Bloom filter is constructed, membership queries can be easily answered. To determine whether an element x belongs to the set S, we check all the bits corresponding to the positions $h_1(s), h_2(s), ..., h_t(s)$. If at least one bit is 0, then $x \notin S$ is certain.

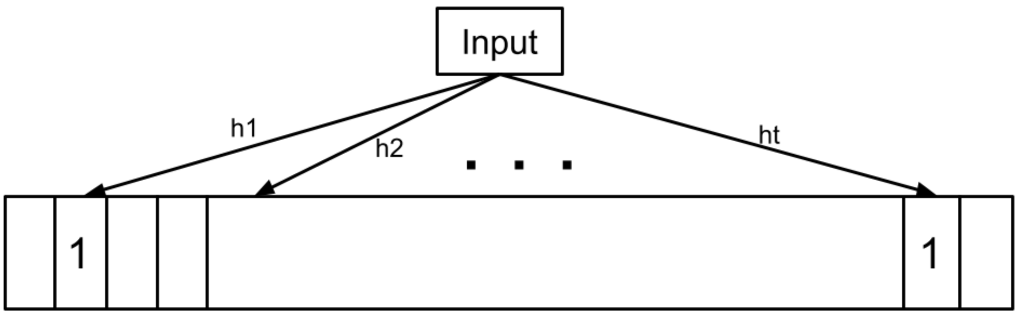


Figure 4.1: Bloom Filter Insertion

Otherwise, we conclude that $x \in S$. A false positive may occur when an element $x \notin S$ is recognized as an element of the set. However, mathematical analysis shows that the probability of the algorithm returning 1 for $x \notin S$ is approximately $(1 - e^{\frac{-tn}{m}})^t$, which is small enough for practical use.

4.3.2 Oblivious Transfer and OT Extension

Oblivious transfer (OT) is a major building block for designing a number of secure computation protocols. The protocol consists of two parties: the receiver and the sender. The basic 1-out-of-2 OT_1^2 allows the receiver to choose either one from two input of the senders without learning anything regarding the other. OT_1^2 was introduced by Even *et al.* [37] as a generalization of Rabin's "oblivious transfer" [73]. Brassard *et al.* [20] further extended OT_1^2 to 1-out-of-n OT_1^n where the receiver is able to obtain 1 message from n messages possessed by the sender. Since then, many efficient protocols for OT with different security assumptions have been proposed over the years. The k-out-of-n OT_n^k scheme is the final form of OT schemes and the one we make use of in our solution. In it, from n encrypted messages sent by the sender, the chooser can obtain k of them that he/she had chosen without the sender's knowledge about which part of the messages can be obtained by the chooser. While it is clear that OT_n^k can be constructed by applying k repetitions of OT_1^n, there have been more efficient protocols. Wu *et al.* [84] introduced two-lock cryptosystems to improve the efficiency of the OT_n^k protocol from $O(kn)$ to $O(k+n)$. Recently, Guo *et al.* [53] proposed a cryptographic concept called subset membership encryption and applied it to construct a two round OT_n^k protocol against semi-honest adversaries. The algorithm only requires the communication cost of $O(n)$ for the sender and $O(k)$ for the receiver.

4.4 Proposed Framework

4.4.1 Solution Overview

As mentioned in Section 4.2, the data owner outsources a corpus D of encrypted documents so that later the data consumer is able to perform complex Boolean keyword queries. A query is represented by logical expressions $\wedge, \vee, \neg$ of Boolean keyword predicates q_k. The results of the query are the indices of satisfied documents.

We explicitly assume that each document d_i is associated with a subset $S^{(i)} \subseteq K$ of keywords. For simplicity, we assume that each document has the same number of associated keywords. For each document, we represent these associated keywords by a Bloom-filter of size m. Let $\{h_1, h_2, \ldots, h_t\}$ be independently keyed hash functions. A keyed hash function h_i inputs a secret key from key space K and a keyword k and outputs an integer in the range of $[1, \ldots, m]$. The preprocessing stage when the data owner prepares the data and uploads them to the data server is described as follows:

(i) The data owner generates a secret pseudorandom function F mapping an integer to a random key in the key space K.

(ii) For each document $d^{(i)}$, the data owner does the following:

- Generating a key for hash functions: $k^{(i)} \leftarrow F(i)$.

- Creating a Bloom-filter $B^{(i)}$ associated the documents.
- Inserting the document keyword set $S^{(i)}$ into Bloom-filter $B^{(i)}$ with the hash functions: $h_1, h_2, ..., h_t$ using the key $k^{(i)}$. Concretely, for each keyword $w_j^{(i)}$ of the document $d^{(i)}$, we set the bit $h_1(k^{(i)}, w_j^{(i)}), \ldots, h_t(k^{(i)}, w_j^{(i)})$ in Bloom-filter $B^{(i)}$.
- Encrypting the content of the document with a standard cryptosystem (i.e. AES).

(iii) The data owner sends the encrypted dataset as well as the Bloom-fiters for all the encrypted documents to the data server.

(iv) The data owner shares the secret function F with the authorized data consumer.

Let $D^{'}$ denote the outsourced data of the data owner. $D^{'}$ consists of multiple records and each of them has the following form:

$$\Big\langle \text{Document index, Encrypted content, Bloom Filter data} \Big\rangle$$

Two arbitrary documents have two different Bloom filters that are generated by two different sets of hash functions (i.e., different keys for keyed hash functions). Moreover, the keys are generated by a pseudorandom function F. The input of F is the index of the document. Hence, having only the view of the Bloom filter data, the data server cannot make any conclusion about the associated keyword sets.

Now consider an authorized data consumer (who would typically be authorized by the data owner) who wants to securely retrieve data from $D^{'}$ in the data server. The satisfied documents are defined by his/her private Boolean query. The query phase consists of three stages as follows:

- Secure Single Keyword Evaluation – In this stage, the data consumer evaluates a single keyword query for each encrypted document. The output of this stage is the encryption of either 1 or 0, depending on whether the document contains the keyword.

- Secure Complex Boolean Query Evaluation – Based on the results of the previous stage, the data consumer collaborates with the data server to compute the result of the complex logical combination of Boolean predicates. Again, the output of this stage is the encryption of either 1 or 0 depending on whether the document satisfies the input query.

- Retrieval of Output Data – At this stage, the data consumer collaborates with the data server to securely retrieve the indices of satisfied documents, and obtains the final documents by private information retrieval or oblivious RAM with known indices.

4.4.2 Secure Single Keyword Evaluation

We consider the scenario of evaluating a single Boolean predicate $q_k(d)$ for each document d in the encrypted data corpus D'. More concretely, we propose an algorithm to answer the query whether a particular encrypted document d contains a given keyword k.

We denote Alice, S, and Bob to be the data owner, the cloud, and the data consumer respectively. Let (pk, sk) be the Paillier key pair of S, and $Enc(\cdot)$ be the encryption under public key pk. Protocol 2 describes the algorithm that inputs an index i, and a keyword w and outputs an encrypted bit b. Bit $b = 1$ if the document contains the keyword and 0 otherwise. The result is held by Bob but remains encrypted under S's public key pk.

In line 5 of protocol 2, Bob and S collaboratively compute the multiplication operation on the encrypted data. In this protocol (i.e., SecMul), Bob holds two private encrypted

Algorithm 2: Secure Single Keyword Evaluation

```
Input: Integer $i$ denotes the index of document $d^{(i)}$, keyword $w$, $pk$ is the Paillier
       public key of S
Output: Encrypted bit $Enc(b)$, where $b = q(w, d^{(i)})$ - whether $d^{(i)}$ contain $w$
1 S encrypts each bit in the Bloom filter $B^{(i)}$ by its Paillier public key;
2 Bob generates key $k = F(i)$;
3 S and Bob perform $(t, m)$ oblivious transfer to send encrypted
   $\{Enc(B^{(i)}[h_j(k, w)])\}$ to Bob, $j = \overline{1, t}$;
4 Bob computes $r = Enc(1)$ ;
5 For each corresponding bit $h_j(k, w)$, Bob computes
   $r = SecMul(r, Enc(B^{(i)}[h_j(k, w)]))$ ;
6 Bob outputs $r$.
```

inputs $(Enc(x), Enc(y))$ and S keeps the Paillier secret key sk, where x and y are unknown to both two parties. The output of $SecMul(Enc(x), Enc(y))$ is $Enc(x \times y)$ and revealed only to Bob. The protocol $SecMul$ is presented in protocol 3. Regarding the definition of $SecMul$ (i.e., protocol 3), Bob iteratively computes the product of $\prod Enc(B^{(i)}[h_j(k, w)]$ in the encrypted form for $j = 1, \ldots, t$. The product equals 1 if any only if all the bits of $\{Enc(B^{(i)}[h_j(k, w)]\}$ are set and 0 otherwise. Hence, the correctness of the protocol follows that observation.

We note that this chapter is using the iterative approach to compute the product of multiple encrypted bits for simplicity. The improved version of computing the product encrypted bits will be discussed and presented in Chapter 6.

As described in steps 1 and 2 of protocol 2, S is required to encrypt the Bloom filter sets each time for a single keyword evaluation. However, since a complex Boolean query generally contains multiple keyword evaluations, one time Bloom filters encryption and communication for a query are sufficient. The protocol requires n encrypted bit transfers for the Bloom filter and $2 \times t$ encrypted integer communication for t Secure Multiplication rounds.

Algorithm 3: Secure Multiplication

```
  Input: Bob holds $(Enc(x), Enc(y))$, and S holds the private key $sk$
  Output: Bob holds $Enc(x \times y)$
1 Bob generates two random number $r, s$;
2 Bob computes $Enc(x + r), Enc(y + s)$ sends them to S;
3 S decrypts and obtains $x + r,\ y + s$;
4 S computes $(x + r)(y + s)$ and sends $Enc((x + r)(y + s))$ to Bob;
5 Bob computes
    $Enc(x \times y) = Enc((x + r)(y + s)) - Enc(x \times s) - Enc(y \times r) - Enc(r \times s)$.
```

The computations at lines 2-5 of protocol 3 are simply performed by the homomorphic property of Paillier encryption. During the protocol, Bob only works on the encrypted data, while the server receives two random numbers. Hence, no information regarding x and y is gained by *Bob* and *S*. The correctness of the protocol is trivial as $(x+r)(y+s) = x \times y + x \times s + y \times r + r \times s$. The protocol requires two encrypted integer transfers for communication cost. Bob has to perform five multiplicative operations and five exponential operations in the ciphertext space.

4.4.3 Secure Complex Boolean Query Evaluation

At this stage, Bob holds the encrypted result of the evaluation for each document with each keyword appearing in the private query. This sub-section discusses three basic primitives that operate on the encrypted inputs of the stage. With these primitives, Bob has the capability to compute the encryption of the desired bit result for each document's query evaluation. The output of this stage is a single encrypted bit for each document. This single bit indicates whether the document has satisfied the query.

Inputs of the three primitives are either one single encrypted bit (NOT operation) or two encrypted bits (AND and OR operations). The descriptions of these are presented as follows:

(i) $\neg$ (NOT) - It is straightforward to derive the formula for the bit negation operation: $Enc(\neg x) = Enc(1) - Enc(x)$. Clearly, the operation leaks no information regarding the encrypted bit x to both the cloud S and *Bob*. It requires one exponential operation and one multiplicative operation. Clearly, the protocol leaks no information to Bob and S, since S receives no more data while Bob only works on his inputs which are encrypted data.

(ii) $\wedge$ (AND) - Because $x \wedge y = x \times y$ for any two bits x, y, the primitive is exactly the same as the description of Secure Multiplication (protocol 3). The protocol requires five multiplicative operations and five exponential operations in the ciphertext space. The security of the protocol follows the analysis of Secure Multiplication (i.e., protocol 3).

(iii) $\vee$ (OR) - Since $x \vee y = x + y - x \times y$, we can derive the definition of the OR primitive from protocol 4. The protocol requires seven multiplicative operations and six exponential operations in the ciphertext space. During the protocol, the data that *Bob* and S receive is exactly the same as the data received during protocol 3, hence, *Bob* and S gain nothing after the protocol's execution.

4.4.4 Secure Retrieval of Output Data

Following the output of the Secure complex Boolean query evaluation stage, Bob has the evaluation result (in encrypted form) for the combination of all predicates in the input

```
Algorithm 4: Secure OR Operation
    Input: Bob holds $(Enc(x), Enc(y))$, and S holds the private key $sk$
    Output: Bob holds $Enc(x \vee y)$
1   Bob and S collaboratively compute $Enc(x \times y)$ using protocol 3 ;
2   Bob computes $r = Enc(x) + Enc(y) - Enc(x \times y)$;
3   Bob outputs r;
```

of each data record. The goal of this stage is to utilize these results to reveal the raw evaluation result to Bob. The results are the indices of satisfied records. It is still worthy to point out that after this final stage, Bob can obtain only the result and nothing else, at the same time S gains nothing regarding Bob's query.

Let us denote (pkB, skB) as Bob's Paillier key pair, and $Enc(pkB, \cdot)$ as the encryption using Bob's public key. The process of the Secure retrieval of output data is presented in protocol 5.

At line 1.4, the cloud S receives $Enc(pkB, r_i)$ (encrypted by Bob's public key) and a random number $b_i \times id_i + r_i$ for each document. Clearly, it learns nothing regarding the evaluation results of Bob's input query. On the other hand, while Bob receives two random numbers $b_i \times id_i + r_i + s_i$ and $s_i + r_i$ for document id_i, the only information he obtains is $b_i \times id_i$ and nothing else.

4.5 Experimental Results

We implemented and calculated the CPU time required to run the sub-protocols that we proposed in Section 4.4. Our experiments were conducted on a Windows 10.0 machines with a 3.0GHz processor and 16GB RAM. We used the Paillier cryptosystem as the underlying additive homomorphic encryption scheme and implemented the proposed sub-protocols in Java. In order to make all the same subprotocols tp play a similar role,

```
Algorithm 5: Secure Retrieval of Output Data
  Input: Bob holds $Enc(b_i)$- the encrypted evaluation result of each document,
  Output: Bob holds $S_r$ the set of satisfied indices
1 For each index $id_i$
    (1.1) S sends $Enc(id_i)$ to Bob.
    (1.2) Bob and S compute $Enc(b_i \times id_i) = SecMul(Enc(b_i), Enc(id_i))$
    (1.3) Bob generates a random integer $r_i$, and computes $Enc(b_i \times id_i + r_i)$
    (1.4) Bob sends $Enc(b_i \times id_i + r_i)$ and $Enc(pkB, r_i)$ to S.
    (1.5) S decrypts to get $b_i \times id_i + r_i$
    (1.6) S generates a random integer $s_i$ and encrypts $Enc(pkB, b_i \times id_i + r_i + s_i)$
    (1.7) S computes $Enc(pkB, s_i + r_i)$;
2 S sends pairs of $\{Enc(pkB, b_i \times id_i + r_i + s_i), Enc(pkB, s_i + r_i)\}$ to Bob in a
    random order;
3 Bob decrypts and computes $p_i = b_i \times id_i$;
4 If $p_i \neq 0$, Bob adds $p_i$ to $S_r$;
5 Bob outputs $S_r$;
```

we implemented a simplified version of Secure Retrieval of Output Data protocol (i.e., protocol 5). In this simplified version, we considered only one document. As the result of the assumption, the output of the sub-protocol is either 0 or the index of the single document. Table 4.2 shows the processing time of four sub-protocols with different Paillier encryption key sizes.

While the specification of the secure OR protocol requires four more multiplicative operations compared with the secure AND protocol in the ciphertext space, the result shows that there is not much difference between the running times of secure AND and secure OR. On the other hand, we note that the running time of secure Negation is sig-

Table 4.2: Execution time of sub-protocols in *private Boolean keyword search* system

Key Size	Secure Negation	Secure AND	Secure OR	Secure Retrieval
512	4 ms	20 ms	22ms	38 ms
1024	17 ms	73 ms	86 ms	150 ms
2048	81 ms	517 ms	558 ms	1090 ms

nificantly larger than the difference between the previous two protocols. That means the encryption/decryption operations are more computationally expensive than performing arithmetic calculations on the ciphertext space. The computational cost of our proposed method is much lower than Samanthula *et al.* [76]'s results. Their approach required approximately 350ms on each encrypted AND and OR encrypted operation.

If we fix the Paillier encryption key size to 1024 bits, we note that the running time of the secure retrieval of output data is 150 milliseconds. However, we observed that the computation cost of the sub-protocols increases by almost a factor of seven when the Paillier key size is doubled.

Figure 4.2 shows the computation cost for the frameworks for a synthetic dataset respects to the number of predicates in the query. The dataset contains 2000 documents, each a document contains a random set of keywords (ranging from 50-250 keywords). The size of the whole keyword set is 1000.

The execution time of the proposed framework goes linearly with the size of the dataset. To process the dataset with average of 5 predicates in a query, it requires around 20 minutes for completion. It is important to note that the computations performed on each document are independent of other documents. Hence, by utilizing the parallel processing paradigm to parallelize the processing, we are able to improve the performance drastically. We leave these implementation details for future work.

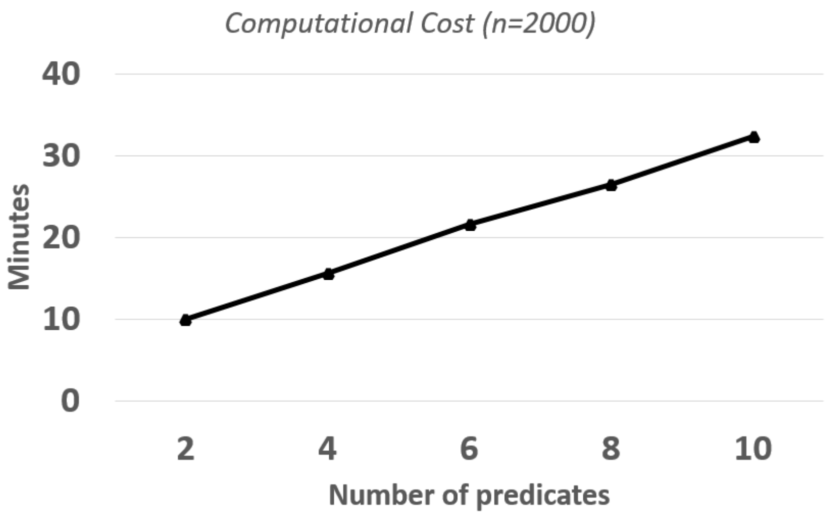


Figure 4.2: Computational cost of proposed framework (corpus size = 2000)

4.6 Summary

This chapter presents a framework to securely evaluate Boolean queries over encrypted data in the cloud. We applied the Bloom filter and additive homomorphic encryption to construct a secure single keyword evaluation. We also presented an efficient mechanism to systematically combine the evaluation results of individual predicates to compute the corresponding query evaluation result. Our protocol not only protects data confidentiality and the privacy of user input queries but also hides the access patterns of the queries. The experimental results show that our protocol is practical for a small and medium size datasets. Since the evaluation for each record can be done independently, the protocol is able to be performed in parallel paradigms such as Map-Reduce or GPU.

5 Secure Multidimensional Range Query on Outsourced Database

5.1 Introduction

In this chapter, we study the other important class of complex search operations: multi-dimensional range query. Chapter 3 discussed a possible scheme to support range query. However, in general settings, it creates a signification overhead for both of complexity and storage. The complexity of the solution, presented in this chapter, only is sublinear to the size of the database. Moreover, instead of assuming the presence of a proxy, this chapter utilizes a multi-server approach. The client is not required to share the secret key with the proxy, and thus, the solution inherently eliminates the shortcomings of the previous approach.

Multi-dimensional range queries are required for a wide range of practical applications, including network traffic log analysis, long-term health monitoring, and banking audits. There are essentially three broad categories of solutions: special data structure for range query evaluation, bucketization-based schemes, and order-preserving encryption-based techniques. However, these techniques fail to provide rigorous privacy protection due to their statistical patterns, access patterns, or query leakage. Moreover, nearly all existing solutions support only static data (i.e., data that has been uploaded to the server only once) and previously known queries. To address the abovementioned issues, this chapter proposes a three-party architecture and investigates different protocols that support multi-dimensional range queries over encrypted remote data while rigorously guaranteeing privacy. We examine both static and dynamic cases in which data records can be appended to the existing data set. In addition, solutions for both fixed and unknown sets of queried attributes are studied.

5.1.1 Motivating Scenario

We consider this context similar to the scenario we described in Chapter 3. Network audit logs of an enterprise are outsourced to a third party for future usage and sharing among different security groups of the company. Storing raw network audit logs is hampered by the presence of security and privacy sensitive information. The contents of the log should be encrypted by a certain method so that necessary query operations can be carried out. Suppose a particular host with a range of IP addresses is determined to have been compromised at time t1 and later involved in scanning other hosts for vulnerabilities at a certain range of ports $[p1, p2]$. The audit logs should be examined for :

$$(ip_1 \leq ip \leq ip_2) \wedge (t_1 \leq t \leq t_2) \wedge (p_1 \leq p \leq p_2)$$

Other applications of secure multidimensional range query include medical records, and financial audit logs, etc.

5.1.2 Organization

The remainder of this chapter is organized as follows. Section 5.2 formulates the problem statement. Section 5.3 describes our system model as well as the requirements for the designed protocols. Section 5.5 discusses the solution for full-domain queries where the set of queried attributes is fixed and unchanged for all queries. This type of query is similar to most existing techniques for multi-dimensional range queries. Section 5.4 presents solutions for a generalized version where the attributes of interest can dynamically vary for different queries. The naive approach to this problem is to apply multiple single range queries. However, this approach leads to single-dimensional privacy leakage. In the proposed protocol, we leverage the concept of secure set intersection to resolve this issue. Section 5.6 discusses practical considerations for system implementation. Finally, Section 5.7 concludes the chapter.

5.2 Problem Formulation

We consider a scenario involving a data owner Alice. Alice possesses a multi-dimensional dataset D of n records. Each data record has m numerical attributes. Let a_i denote the identifier of the ith attribute and v_j^i denote the value of the jth attribute of the ith record.

Alice wishes to outsource her dataset (i.e., the set of data records) to a cloud server to save on local storage costs. A typical example is the data of body measurement recorded by sensors or mobile devices. Because mobile devices are normally limited in terms of storage capability, outsourcing such information to cloud services offers easy and low-cost solutions for long-term continuous health monitoring. On the other hand, such data includes confidential personal information; hence, they should be stored in an encrypted form. At the same time, to facilitate health monitoring and treatment, the data owner needs to use the outsourced data efficiently.

To query the outsourced dataset, Alice performs a multi-dimensional range query. A multi-dimensional range query is a process of retrieving the data records that satisfy query values from the relative attribute domains:

$$\text{Select * From } D \text{ where } a_{x_i} \in [s_{x_i}, t_{x_i}], \text{ where } x_i \in \{1, \ldots, n\}$$

We follow the convention to denote $[a, b]$ representing an interval of integers from a to b, inclusively. The set of attribute identifiers $S_q = \{x_i\}$ is called the *set of queried attributes*. A multi-dimensional range query requests data records such as each attribute a_{x_i} in the *set of queried attributes*, and takes the value in the interval $[s_{(x_i)}, t_{(x_i)}]$. Roughly speaking, each data record can be considered as a point in m-dimensional space, and the multi-dimensional range query is defined by a hyper-rectangle. All the points inside this hyper-rectangle are considered to be the results of the query.

When the *set of queried attributes* covers all m dimensions of the database D: $S_q = \{x_i\} = \{1, \cdots, m\}$, we say that the query is a full-domain query. Section 5.5 discusses viable solutions where the query's dimensions change dynamically from one query to another, with no fixed set of the concerned attribute domain for the query. Although studies [76] have been conducted to address such a multi-dimensional query, they require the data owner to share his/her private key with a semi-trusted third party, which is not practical owing to privacy threats. In Sections 5.5 and 5.4, we will discuss the solutions for the settings of static and dynamic data storage. In the static database setting, the data owner uploads the entire data only once and is later able to query for the necessary records. On the other hand, in the dynamic database setting, new data records can be appended to the previously uploaded data, and expired data can be transferred a data storage archive and removed from the query results. We assume that the three parties loosely synchronize the time of archiving the data.

5.3 System Overview

5.3.1 System Model

We adopt the three-party architecture described by Boneh *et al.* [15] and Cristofaro *et al.* [32]. The system consists of three parties: the data server, the client, and the index server as depicted in Figure 5.1

(i) *Client*: The client possesses a numerical multi-dimensional dataset and wishes to outsource his/her dataset in an encrypted form to the data server. At some point, he/she should have the capability to securely construct a multi-dimensional range query to obtain the requested data.

(ii) *Data server*: The data server provides storage services to the client. The cloud server is trusted to provide reliable services but it may also be curious about the content of the data records stored in its database or the content of the queries submitted by the data owner.

(iii) *Index server*: The index server stores meta-data to support answering of the query. As with the data server, it is trusted to store the meta-data as well as correctly

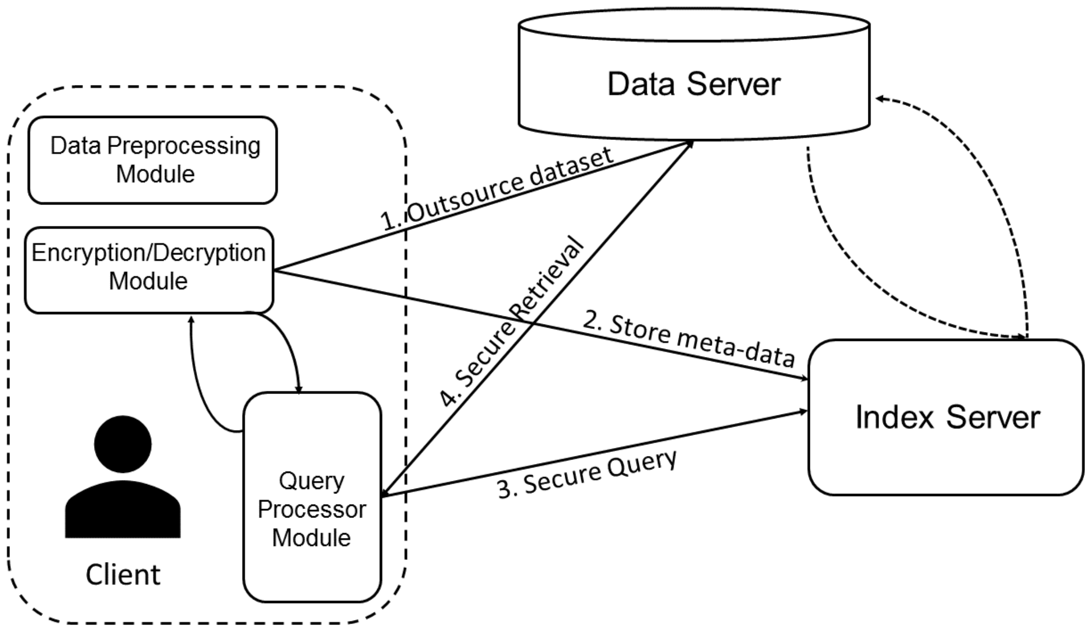


Figure 5.1: Three Party Model for Secure Multidimensional Range Query

process the query initiated by the data owner. In addition, it may be curious about the client's data as well as the query content.

Workflow. Based on the notation in Figure 5.1, the workflow of our system can be described as follows. The data is first preprocessed by the data preparation module on the client side. The dataset and the meta-data (i.e., the output of the preprocess phase) are encrypted by the encryption/decryption module and sent to the data server and the index server. The entire dataset should be encrypted by a symmetric encryption scheme. It involves low overhead for data storage and bandwidth, as well as efficient encryption/decryption operations. On the other hand, the meta-data should be encrypted by a specific public key cryptosystem and embedded in a special data structure to enable secure query processing.

To construct a multi-dimensional range query, the client constructs the query content using the query processor module, which has an interface with the encryption/decryption module. The encrypted query content is sent to the index server. The index server will output the corresponding indices of the satisfied data records so that the client is able to efficiently retrieve them from the data server. Furthermore, the query processor module may be invoked to filter the results returned from the data server.

5.3.2 Security Requirements

In this work, we attempt to design a secure multi-dimensional range query while preserving the privacy of the database content and query values. Ideally, a secure protocol should only reveal what is leaked by system parameters known to all parties and by the intended functional output. More specifically, we consider the following information leakage:

(i) Data Privacy. A passive attacker who gets a snapshot of the encrypted database and encrypted index data should not be able to obtain any information about the user's private data. This implies that these two types of data should be encrypted by a probabilistic cryptosystem.

(ii) Query Privacy. The cloud and index servers should not be able to determine whether two queries are the same.

(iii) Access Pattern. Access to satisfied data records should not be revealed to the cloud server. Islam *et al.* [59] showed that data access pattern leakage could lead to the disclosure of a significant amount of sensitive information.

(iv) 1-d Dimension Privacy. Informally, single-dimensional privacy means that given a search token of a multi-dimensional range query, a computationally bounded adversary is not able to independently obtain the exact search results for any single-dimensional query.

While allowing the user to download the entire database and performing the query locally will achieve all the abovementioned privacy requirements, this solution is infeasible. The proposed solution should be efficient in terms of time complexity, communication, and round complexity.

5.4 Full-Domain Multidimensional Range Query

First, we describe the construction of full-domain multi-dimensional range queries. The dataset contains multiple data records that are points $\{x_j\}_{j=1}^{n}$ in a m-dimensional lattices. A full-domain query is defined by a hyper-rectangle in the m-dimensional space:

$$B = \{[s_1, t_1], [s_2, t_2], \dots, [s_m, t_m]\},$$

where $[s_i, t_i]$ represents a single dimensional range. A data record $\{v_j\}_{j=1}^{m}$ satisfies a query represented by the hyper-rectangle B if and only if $v_j \in [s_j, t_j]$ for $\forall j : 1 \leq j \leq m$.

As mentioned above, each record of the dataset is encrypted individually by a symmetric encryption scheme such as AES. This allows for the efficient retrieval of specific records from the data server. Hence, this work focuses on designing the data structure and algorithms to find the indices of data records that satisfy the query criteria. Specifically, we present a method for constructing the meta-data, the data structure to store the metadata in the index server, and the algorithms to process the query.

Our solution for the full-domain multi-dimensional range query problem is inspired by the multi-dimensional bucketization approach. First, we partition the data space into M disjoint buckets. The number of buckets M is smaller than the number of data points n. Each data record belongs to exactly one bucket. The query processing translates into the problem of retrieving the bucket content. Hore *et al.* [57] claimed that nondeterministic encryption of the bucket labels does not raise the level of security, because simply encrypting bucket labels cannot protect the query privacy or access pattern from an adversary. In this chapter, we leverage the ORAM technique to hide the access pattern and thus provide a stronger security guarantee.

5.4.1 Solution Overview

The dataset is first pre-processed by the data preprocessing module. It is partitioned into non-overlapping buckets, and the bucket label is set as the tag for each data record in the bucket. The data owner encrypts each original data record (i.e., a d-dimensional data point) and uploads it to the data server. Furthermore, an encrypted inverted index table is stored at the index server in the following form:

$$\left\langle Bucket\ Label, \quad \text{encrypted-set-of-record-indices} \right\rangle$$

The idea of the solution is to leverage ORAM data access for each intersecting bucket without leaking any information about the data or query content. The satisfied records are determined by the buckets intersecting the query. Many existing studies have indicated that ORAM is impractical due to its high computational cost. However, this claim is not necessarily true, especially because nearly all existing studies focus only on designing various methods to retrieve the indices of the satisfied records while leaving the actual data retrieval process to a black box. The black box is assumed to be a standard ORAM protocol for data records with known indices. It contradicts the abovementioned concerns with regard to practical applications. Recent results of Path-ORAM have shown that it is a practical tool for data access. When the record indices are known to the client, a constant number of communication rounds of ORAM data access should be required to retrieve the data.

The two main challenges facing this approach are how to partition the space into buckets and how to embed the record indices into the ORAM data structure stored at the index server.

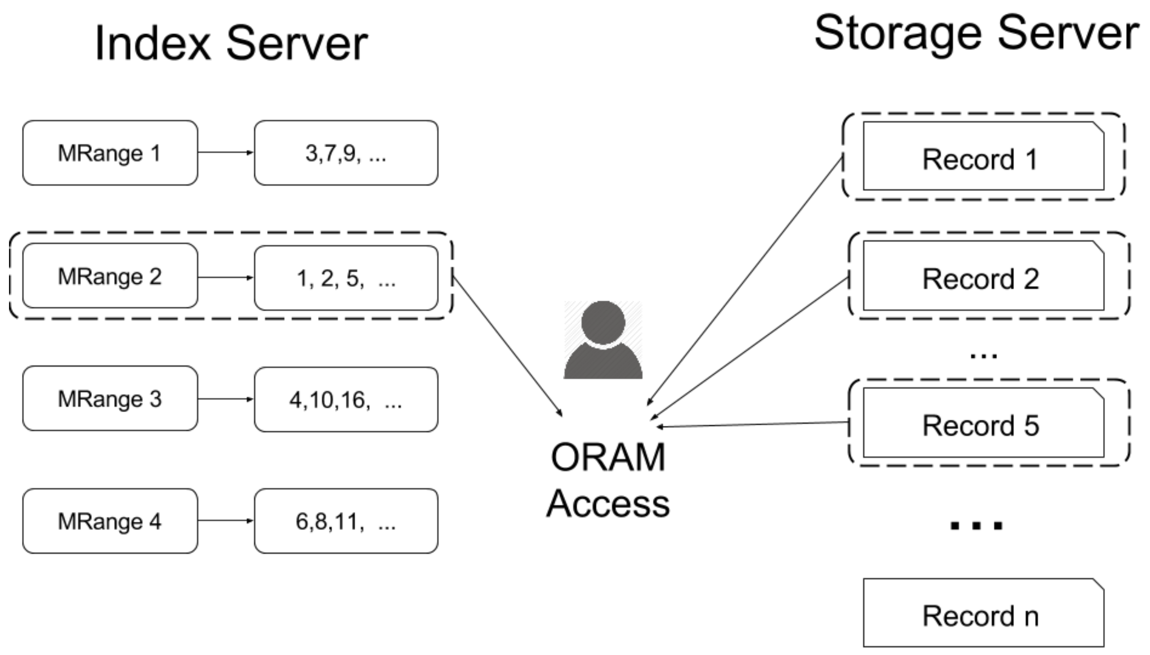


Figure 5.2: Secure multidimensional Range Query from ORAM

Bucketization Algorithms. We retrieve a bucket if and only if it overlaps the query rectangle. We partition the space into M non-overlapping hyper-rectangles. Each rectangle contains approximately the same number of data points. The parameter M is determined by the tradeoff between the cost of false positives and the cost of communication between the client and the index server. When the number of buckets increases, each bucket contains fewer points, and the false positive rate decreases. However, the number of buckets intersecting with the query increases, which means that the number of ORAM accesses increases. On the other hand, when the number of buckets decreases to one, fewer ORAM accesses are required for each query. Furthermore, when the false positive rate increases, additional bandwidth overhead is incurred to retrieve the actual data records from the server. With a certain parameter M, we use the Mondrian multi-dimensional partition algorithm presented by LeFerve *et al.* [62]. The algorithm is described in protocol 6. In this protocol, we assume that $M = 2^k$ for simplicity.

Algorithm 6: $Partition(D, M)$

```
Input: Dataset of multidimnensional points D, number of bucket $M = 2^k$
Output: M buckets
1 $attr \rightarrow 1, \cdots, m$ ;
2 $splitVal \rightarrow find_{median}(D, attr)$ ;
3 $left\_half \rightarrow \{r \in D : r.attr \leq splitVal\}$ ;
4 $right\_half \rightarrow \{r \in D : r.attr > splitVal\}$;
5 return $Partition(left_half, M/2) \bigcup Partition(right_half, M/2)$;
```

At each call, the algorithm partitions the space according to the value of attribute *attr*, which is chosen at step 1. The data space is partitioned into two parts. Each part has approximately the same number of elements. The complexity of the preprocessing phase is $O(logn)$.

ORAM storage. Each data label is associated with a list of record indices. Our solution stores these indices in the ORAM data structures so that we can have secure access to the contents of a particular bucket. A naive solution is to use a sufficiently large ORAM data block to store the entire set of encrypted record indices. The user can retrieve the entire set of indices for a block label at once. However, because the size of the index set of each bucket label may vary, it is wasteful to use a large-sized data block to store a small piece of information. Moreover, it is computationally expensive to read from and write to a large-sized data block. Our approach is to represent the index sets in a compact form so that the data from an ORAM block can be determined beforehand with reasonable size.

5.4.2 Static dataset

First, we consider the simplest scenario where the dataset is intact after being uploaded to the server. Archiving data is an example of this scenario.

The dataset is preprocessed and encrypted once before being uploaded to the data server. In the preprocessing stage, the bucketization algorithm (Algorithm 6) is applied to partition the space into buckets. Each record (i.e., a data point) belongs to exactly one bucket. Data is rearranged so that the identifiers of data records in the same bucket label form a consecutive numerical counter. Hence, the necessary information to reconstruct an index set is the starting counter and ending counter. The meta-data stored in the index server is ORAM data structures such that each block has an encrypted bucket label as the key for the encryption of the starting counter and ending counter as the values. Hence, to query for a data bucket, the data owner performs an ORAM data access with the index server; the access key is the bucket label. Then, the client obtains the range of data identifiers. The last step is to perform one more round of data access for each data identifier with the data server.

Analysis. Because the data and their indices are encrypted on the client side by a standard secure cryptosystem, the data server and index server obtain no information from the view of the stored data. Furthermore, the data owner accesses the index data at the index server and the data content at the data server by means of ORAM. The two servers are not able to obtain additional information on the query content or the access pattern. Because the data is obtained by the bucket label, one-dimensional privacy is satisfied for the query.

The bandwidth overhead of the solution is $O(logn)$ owing to the overhead of the Path ORAM. The client is also required to maintain a local storage of $O(logn))$. Finally, the complexity of each data access incurred at the server is $O(logn)$.

We also note that in the case of the static dataset, private information retrieval (PIR) can be used as an alternative to ORAM. In the PIR approach, for reading data, the computational complexity for the data owner is linearly related to the size of the retrieved data; in terms of the server size, the computational cost is linearly related to the size of the entire data set.

5.4.3 Dynamic dataset.

We consider the case where the update operation is allowed. More specifically, new data records may be appended to the existing dataset. The log file is an example of this scenario.

Because new data records can be in any data bucket, we cannot use the previous approach. In the abovementioned approach, the indices of data records with the same bucket label are required to form consecutive integers. Because the new records are not necessarily in the bucket containing the previous records, this requirement does not hold. We adopt another approach to index the records. We make use of the collision-resistance hash function. The index of a new record is determined by the hash of the previous record index that is in the same bucket. More specifically, we consider a collision-resistant hash function $h : \{0,1\}^n \mapsto \{0,1\}^m$ and define a sequence generated by h as follows:

$$a_0 = \text{label-id},\ a_i = h(a_{i-1}).$$

To obtain the index set of the bucket label for the ith bucket, we need to know the values of $a_0^{(i)}$ and $a_k^{(i)}$ where $a_k^{(i)}$ is the last element in the sequence generated for data records belonging to the ith bucket. In this case, when the data is appended, the data owner is required to modify the corresponding ORAM data block to update the last element of the current index set. To obtain the index sets of buckets intersecting the query, the data owner retrieves the corresponding block to get $Enc(a_0^{(i)})$ and $Enc(a_n^{(i)})$.

He/she decrypts the encrypted content and iteratively generates the sequence starting from the first element until the last element using the formula $a_i = h(a_{i-1})$. A collision-resistant hash function is required for this construction so that different data records are not mapped to the same index.

To insert a new record into the data set, the data owner first determines the bucket label of the new data record. Next, he/she performs a read and an update operation with the index server by means of ORAM. The new content of the ORAM block now contains the encrypted next sequential number of the last record index in this bucket.

Analysis. We use a collision-resistant hash function to index the records because each data record belongs to exactly one data bucket. Moreover, it is difficult to find a collision; each data record has a unique identifier determined by the hash functions.

The security of dynamic case construction is analyzed in the same way as that of the static case. In both cases, the user accesses the indices and the real data records using ORAM. Hence, the only information leakage to the cloud server is the result size, which is the lower bound of the number of satisfied records. The reason is that the user is required to make at least the minimum number of satisfied records ORAM data accesses to the cloud server data.

The complexity and bandwidth overhead of the solution remains polylogarithmic due to the use of Path ORAM. The update operation as an ORAM write operation also requires $O(logn)$ memory bandwidth and computational complexity.

5.5 Dynamic Multidimensional Range query

The aforementioned approach can be generalized to address the scenario where the *set of queried attributes* is fixed. Roughly speaking, we should be able to efficiently represent the query space by a small number of fixed hyper-rectangles. In this section, we examine a more general case where the *set of queried attributes* varies for different queries. In other words, the number of dimensions in each query can vary from one to m – the number of attributes in each data record. Clearly, we can treat the query as a full-domain query by considering each missing dimension as the full range. However, this approach normally leads to an excessive number of communication rounds with the index server. Instead of using the multi-dimensional bucketization technique, we apply the single-dimensional bucketization approach as a solution to this generalized problem.

A general multi-dimensional range query is defined by an unordered set:

$$S = \{[s_{i_1}, t_{i_1}], [s_{i_2}, t_{i_2}], \ldots, [s_{i_k}, t_{i_k}]\},$$

$1 \leq i_j \leq m$ and $[s_i, t_i]$ represents a single dimensional range. The size of the *set of queried attributes* is k, $1 \leq k \leq m$.

5.5.1 Solution Overview

Our solution for the generalized problem employs private set intersection techniques. First, we partition each single-dimensional domain into buckets. A data record of m attributes falls into m buckets. For each single-dimensional range in the query, the requested data records must belong to the buckets intersecting with the range. A naive solution is to perform multiple queries on each queried dimension. The results are the intersection of the results. This approach has been used by Zhang *et al.* [88] to design multi-dimensional range queries in sensor environments. However, the naive approach fails to provide single-dimensional privacy because the server gains the results for each queried dimension. In this chapter, instead of performing the intersection operation on the client side, we conduct it on the server side in a manner that preserves privacy.

The query is firstly decomposed into each corresponding dimension. For each queried dimension, the intersecting buckets are enumerated. The query now can be translated into one or multiple conjunctive queries. The variables in the conjunctive query are intersection buckets in each dimension. Finally, a secure set intersection algorithm should be applied to answer the conjunctive query.

To illustrate our idea, we consider the following example. Alice has a dataset where each data record is a three-dimensional point (x, y, z). The attributes receive integer values in the domain $[1, 10]$. Alice decomposes the data space with eight buckets: $[0, 5]_x, [6, 10]_x, [0, 5]_y, [6, 10]_y, [0, 5]_z, [6, 10]_z$. Each record is in exactly 2 buckets. Alice wishes to perform a simple multi-dimensional range query: $3 \leq x \leq 6, 2 \leq y \leq 4$. The query can be transformed into two conjunctive queries: $[0, 5]_x \wedge [0, 5]_y$ and $[6, 10]_x \wedge [0, 5]_y$. Query $[0, 5]_x \wedge [0, 5]_y$ (similar for the other) is translated as retrieving all the data records belonging to both $[0, 5]_x$ and $[0, 5]_y$. The answer to it is the intersection between the two buckets.

Bucketization Algorithms. We apply a simple equi-depth bucketization approach to partition each dimension for the data space. The number of buckets M in each dimension is determined by the tradeoff between the number of conjunctive queries and the false-positive rate. If we partition the dimension into many small buckets, we are able to reduce the false positive rate of the results. On the other hand, if M in each dimension is small, we are required to perform fewer conjunctive queries but additional work is required for the post-processing.

We now describe the method for securely answering each transformed conjunctive query in two different settings.

5.5.2 Static data case

For the case of a static dataset, the entire dataset is processed and uploaded only once to the servers. We leverage the Kissner-Song private set intersection [61] to perform conjunctive queries.

The idea behind the Kissner-Song protocol is to fix a large field F and represent a set $S \subset F$ by a polynomial A_S that has zeros in all the elements of S i.e., $A_S(x) = \prod_{s \in S}(x-s)$. To compute the intersection of many sets S_i , we construct a polynomial B whose zeros are the intersection of these sets. Clearly, if a point $s \in F$ is contained in all the sets S_i, then $A_{S_i}(s) = 0 \ \forall i$, and therefore, if we compute B as a linear combination of the A_{S_i} s, then $B(s) = 0$ also. On the other hand, if $A_{S_i}(s) \neq 0$ for some i and B is a random linear combination of the A_{S_i}s, then there is a high probability that $B(s) \neq 0$. Roughly speaking, instead of storing the record indices for each bucket, we store the coefficients of the polynomial that represents the index set. However, the coefficients should be encrypted so that the index server is not able to trace the indices.

In this chapter, we use the BGN encryption technique [17] to encrypt the polynomial coefficients. The BGN cryptosystem, proposed by Boneh, Goh, and Nissim, allows both addition and multiplication with ciphertexts of constant size. However, there is a catch: while the addition can be performed multiple times, only one instance of multiplication is permitted. Nevertheless, this protocol is considered to be much more practical than fully homomorphic encryption schemes. The homomorphism allows us to compute a linear combination of the polynomial in encrypted form so that set intersection operations can be securely performed on the index server. Chapter 2 provides a quick overview of the BGN cryptosystem.

Consider a bucket consisting of ℓ data records with indices $i_1, , i_\ell$. It can be represented by a polynomial in the form $A(x) = (x - i_1) \cdots (x - i_\ell) = \sum c_i \times x^i$. The coefficients c_i of the polynomial are encrypted by a BGN cryptosystem and stored at the index server. The index server stores $m \times M$ encrypted polynomials, where M is the number of buckets for each dimension and m is the number of attributes in each data record.

To issue a multi-dimensional range query (i.e. $Q = \{[s_{i_1}, t_{i_1}], \ldots, [s_{i_k}, t_{i_k}]\}$), the client first decomposes the query into k dimensions: $i_1, i_2, \ldots, i_k$. The query is translated into multiple conjunctive queries; each one is associated with k buckets corresponding to k dimensions. To obtain the answer for each conjunctive query, the client does the following:

(i) We consider a bit $b_{i_j} = 1$ if the jth bucket of ith dimension intersects with the query and is zero otherwise.

(ii) The client generates appropriate tags: $\sigma_{i_k} = Enc(b_{i_j})$ and sends σ_{i_k} to the index server.

(iii) The index server generates random non-zero numbers p_{i_k} and computes $B(x) = \sum_{i,j} \sigma_{i_k} \cdot p_{i_j} \cdot A_{i_j}$ (in encrypted form), and sends the corresponding result to the user.

(iv) The client decrypts and factors the polynomial $B(x)$ and finds its roots, which are the indices of the records that the user is interested in.

(v) The client performs ORAM data access to obtain the necessary data records from the cloud server.

At step 3, the outer sum and $p_{i_j} \cdot A_{i_j}$ are calculated using the additive homomorphic property of BGN encryption, while the product with the encrypted bit $\sigma_{i_k} = Enc(b_{i_j})$ is performed using the one time multiplication property of the cryptosystem.

If $A_{i_1}, \ldots, A_{i_n}$ are the polynomial representation of the index set for the query q, B is a random linear combination of them. Clearly, when record id satisfies the query, it must be the common root of $A_{i_1}, \ldots, A_{i_n}$. Hence, id is also included in the root of $B(x)$, which leads to the correctness of the protocol. Since $B(x)$ may have superfluous roots, we use a large space F so that there is a probability that these roots are identified as invalid. Moreover, the user can also perform post-processing to filter out the false positives of the processes.

Analysis. To analyze the security of the proposed protocol, we need to examine the data view of the two servers. Because the server stores securely encrypted data and the data is accessed only by the ORAM, it can gain no knowledge of the user sensitive data, query content, or access pattern. Moreover, the index server receives only BGN-encrypted bits; it is not able to obtain any information about the query. Hence, the proposed protocol leaks no information of the data content, query content, or query result to the server or the index server, except for the upper bound of the number of matching records (due to the number of ORAM accesses).

The proposed solution requires the client to send $O(m \times M)$ encrypted bits to the index server and receive approximately $O(n/M)$ bits from the index server. At the same time, the index server is also required to perform up to $O(n)$ multiplication and exponentiation operations in the ciphertext space. The computation and communication costs for the server are the same as those of the previous methods.

The presented solution has a linear complexity in terms of both communication and computation. Moreover, it requires a somewhat homomorphic encryption for computation. A minor modification can be applied to achieve sub-linear complexity, and only an additive homomorphic encryption is required, with a security trade-off. The idea can be expressed as follows:

- Apply an additive homomorphic cryptosystem (e.g., Paillier) to encrypt the polynomials that represent the buckets' content.
- To answer a conjunctive query on bucket $\{b_{i_j}\}_{j=1,\cdots,k}$, the client sends the indices of this bucket to the index server.
- The index server computes the random linear combination of the corresponding polynomials by additive homomorphism.

There is one trade-off for the security of the modified solutions. Since the index server is able to observe the requested buckets for each conjunctive query, the query privacy requirement is violated.

5.5.3 Dynamic data case.

In the case of a dynamic dataset, the new data records may be dynamically appended to the existing dataset. The aforementioned approach, which represents the index sets by polynomials, requires all encrypted coefficients for each corresponding bucket of the new record to be recomputed. We propose another approach that requires interaction between the server and the index server. We note that the abovementioned methods for both full-domain queries and dynamic multi-dimensional range queries do not require any communication between the two servers. The proposed protocol is inspired by a conjunctive keyword searchable encryption protocol proposed by Cash *et al.* [26]. We use it with a major modification to adapt it to our privacy requirements.

We start the protocol description by reviewing a few concepts related to bilinear maps. We will use the following notation:

(i) G_1 and G_2 are two (multiplicative) cyclic groups of prime order p.

(ii) g_1 is a generator of G_1 and g_2 is a generator of G_2.

A bilinear map is a map $e : G_1 \times G_2 \rightarrow G_T$ with the two following properties:

(i) Bilinear: for all $u \in G1, v \in G2$ and $a, b \in Z$, then $e(u^a, v^b) = e(u, v)^{ab}$.

(ii) Non-degenerate: $e(g_1, g_2) \neq 1$.

The proposed solution consists of two algorithms: INSERT and SEARCH for appending new records and performing multi-dimensional range queries on the encrypted data set respectively.

Let F denote a keyed pseudo-random function $f : Z_p \times K \to Z_p$, and select keys K_S, K_X, K_I, K_Z for F.

<u>*INSERT(Record id:* $x = \{x_1, \cdots, x_m\}$ *)*</u>

- For each ith dimension ($1 \leq i \leq m$), the client does:
  - Let jth bucket of ith dimension: bucket l_i^j contains attribute value x_i.
  - Compute $xind \leftarrow F_{K_I}(id),\ z \leftarrow F_{K_Z}(l_i^j)$ and $y \leftarrow g_1^{xind/z}$.
  - Set $eid \leftarrow Enc(id)$, append (eid, y) to a list for bucket l_i^j at the index server.
  - Set $xtag \leftarrow e(g_1, g_2)^{F_{K_x}(l_i^j)\cdot xind}$ and append to a set S at the data server .
- Encrypt the data record and upload the cloud server.

<u>*SEARCH(*$Q = \{(s_{i_1}, t_{i_1}), \ldots, (s_{i_k}, t_{i_k})\}$*)*</u>

- The client decomposes the query into k dimensions: $i_1, i_2, \ldots, i_k$. The query is transformed into conjunctive queries of buckets. For each conjunctive query q, we denote the buckets of interest for these k dimensions as $L_1, \ldots, L_k$.
- For $i = 2, \ldots, k$:
  - The client computes $xtoken_i \leftarrow g_2^{F_{K_z}(L_1)\cdot F_{K_x}(L_i)}$
  - The client sends $L1, \{xtoken_2, \cdots, xtoken_k\}$ to the index server.
  - For each item (eid, y) in L_1 list in random order:
    - The index server sends eid to the cloud server.
    - The index server computes the cardinality $size$ of the intersection between set S of cloud server and $\{e(y, xtoken_2), \ldots, e(y, xtoken_k)\}$

$*$ If $size = k - 1$ cloud server sends eid to the client.

- The client decrypts and obtains the indices.
- The client performs ORAM data access to obtain the necessary data records from the cloud server.

The correctness of the search protocol relies on the following fact:

$$\begin{aligned} e(y, xtoken_i) &= e(g_1^{xind/z}, g_2^{F_{K_z}(L_1) \cdot F_{K_x}(L_i)}) \\ &= e(g_1^{F_{K_I}(id)/F_{K_Z}(L_1)}, g_2^{F_{K_Z}(L_1) \cdot F_{K_X}(L_i)}) \\ &= e(g_1, g_2)^{F_{K_I}(id) \cdot F_{K_X}(L_i)} \end{aligned}$$

Hence, if the set $\{e(y, xtoken_2), \ldots, e(y, xtoken_k)\}$ is a subset of set S, the data record belongs to exactly all the requested buckets.

Our construction of the index list for the bucket and for set S is similar to the construction of TSet and XSet proposed by Cash *et al.* [26]. In their protocol, TSet and XSet are stored in the same place in the cloud server. The correctness and the privacy of the two sets follow the proofs given by the authors [26]. However, because the set intersection is performed locally by the cloud server, that construction leads to unnecessary information leakage (i.e., access pattern leakage). In the proposed construction, the storage of the bucket list and set S is separated into two parts, and the cardinality of the set intersection is obtained by a secure two-party computation protocol. We use the protocol proposed by Cristofaro *et al.* [31]. Figure 5.1 shows the workflow of the protocol. The protocol is secure under the assumptions of the semi-honest model. The complexity is linearly related to the sizes of the two sets.

Analysis. . The insertion algorithm only appends the semantically secure encryptions to the server and index server. It leaks no information to the two servers; for details of the proof, readers may refer to [26].

During the search phase, the server only receives the encryption of the record index *eid* and knows whether the encrypted index belongs to the results. By observing the value of *eid*, the server is able to determine whether there are records that satisfy multiple queries. This information leakage can be eliminated by applying one more encryption layer on *eid*.

On the other hand, the newly proposed protocol does not provide query privacy for the index server. The index server is able to determine whether two queries are the same by observing the *xtoken* sets received from the client. However, this is the only information leakage of the client's private data to the index server. Because the cardinality of private set intersection is only revealed to the cloud server, the index server does not obtain any information about the satisfied records.

The computational complexity and communication cost of the two servers are linearly related to the number of data records. The communication cost and computational complexity for the client are $O(max(log|D|, k))$, where $|D|$ is the number of data records and k is the size of the *set of queried attributes*.

5.6 Experimental Results

We conducted a number of experiments to verify the practicality of the proposed solution. The experiments were performed on a synthetic dataset. We created the synthetic dataset by sampling 20,000 data points, each having four integral attributes from the domain [0,999].

Our implementation of Path ORAM required up to two minutes to construct the ORAM data structures of the dataset of 20,000 data records. It also only took only 0.02 s per access, including the time for decryption. This execution time is considered a criterion for the trade-off between the accuracy and the number of buckets, as discussed in Section V. For full-domain multi-dimensional range queries, we randomly generated 10,000 queries where each dimension is randomly selected from a uniform distribution from the same domain with the dataset. We took the average results of these queries and reported them.

Consider the number of buckets M. The accuracy of the intersection bucket approach was $0.1, 0.15, 0.3, 0.5$ for $M = 128,\ 256,\ 512,\ 1028$, respectively. On the other hand, the number of buckets intersecting with the query was 22, 33, 53, 107 for each configuration of M. Thus, when the parameter M increases, we are able to reduce the false-positive rate; however, we need to perform additional ORAM queries with the index server. When we fixed the number of bucket $M = 1028$, the solution requires 50 seconds for the search time. Here we note that is is much faster than Wang *et al.* [82]'s method. The previous method took around 250 second for 2D multidimensional range queries and doubles for 4D dimensions.

To test dynamic multi-dimensional range queries, we considered 5000 random queries. The set of interested attributes contains 13 attributes. The two end points of each query are uniformly generated from the attribute domain.

Figure 5.3 shows the relation between the accuracy of the proposed approach and the number of buckets in each dimension (M). We conducted experiments with M in the range of 520. When we used only five buckets per dimension, the reported false positive rate was high, but we were only required to perform five conjunctive queries on average for each query. On the other hand, when we increased M to 20, the accuracy increased to 70%. However, we also needed to answer 60 conjunctive queries (i.e., depicted Figure 5.4) to obtain the complete set of requested records.

We also report that the execution time of our implementation of BGN encryption to evaluate the combination of two sets of size 128 is 2 seconds.

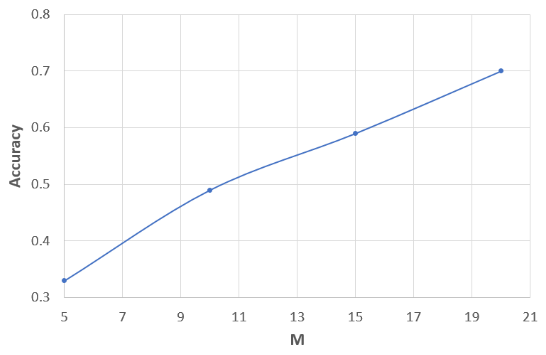


Figure 5.3: Accuracy vs number of buckets (M)

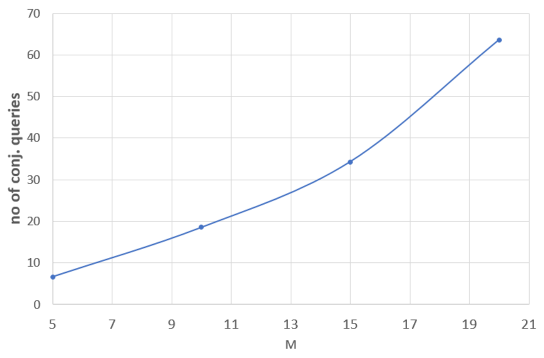


Figure 5.4: Number of conjunctive queries vs number of buckets (M)

5.7 Summary

This chapter investigates different protocols to securely evaluate multi-dimensional range queries over encrypted data in cloud platforms. Our main idea is to leverage the bucketization algorithm to label the numerical range. The server stores the encrypted data, while the index server stores the meta-data output by the bucketization algorithm. When the multi-dimensional range queries are fixed beforehand, we perform two rounds of ORAM data access to answer the queries. The data records are labeled before encryption and sent to the server. On the other hand, to support more general queries, we presented novel solutions that allow multi-dimensional range queries to be answered, where the query constraints are not required to be fixed. The idea is to bucketize each attribute of a multi-dimensional data record and perform set intersection to answer conjunctive queries for multiple data labels. While the latter approach is able to support more general requirements, it is costly in terms of both computational complexity and bandwidth communication. In the most general case, where the dataset can be dynamically appended and the set of queried attributes is not fixed, our approach leaks a small amount of information to the index server.

Client, on input $C = \{c1, \ldots, c_v\}$		Server, on input $S = \{s1, \ldots, s_w\}$
$R_c \leftarrow Z_q, R'_c \leftarrow Z_q$		$(\widehat{s_1}, \ldots, \widehat{s_w}) \leftarrow \psi(S)$
$X = G^{R_c}$		with ψ random permutation
$\forall i\ 1 \leq i \leq v :$		$\forall j\ 1 \leq j \leq j : hs_j = H(\widehat{s_j})$
$hc_i = H(c_i);$		
$a_i = (hc_i)^{R'_c}$	$\xrightarrow{X, \{a_1, \ldots, a_v\}}$	$R_s \leftarrow Z_q, R'_s \leftarrow Z_q$
		$Y = g^{R_s}$
		$\forall i\ 1 \leq i \leq v : a'_i = (a_i)^{R'_s}$
		$(a'_{l_1}, \ldots, a'_{l_v}) = \psi(a'_1, \ldots, a'_v)$
		$\forall j\ 1 \leq j \leq w : bs_j = X^{R_s} \cdot (hs_j)^{R'_s}$
		$\forall j\ 1 \leq j \leq w : ts_j = H'(bs_j)$
$\forall i\ 1 \leq i \leq v :$	$\xleftarrow[\{ts_1, \ldots, ts_w\}]{Y, \{a'_{l_1}, \ldots, a'_{l_v}\}}$	
$bc_i = (Y^{R_c})(a'_{l_i})^{1/R'_c}$		
$\forall i\ 1 \leq i \leq v :$		
$tc_i = H'(bc_i)$		
	Output $= \lvert\{ts_1, \ldots, ts_w\} \cap \{tc_1, \ldots, tc_v\}\rvert$	

Table 5.1: PSI-CA Protocol [31]

6 Secure Personal Information Verification

6.1 Introduction

In this chapter, we investigate the problem of secure personal information verification. The scenario we consider is different with the data outsourcing contexts that we discussed in the previous three chapters. However, the two problems are on the same theme of secure query processing. They share the same challenges and requires a similar approach to design the solutions.

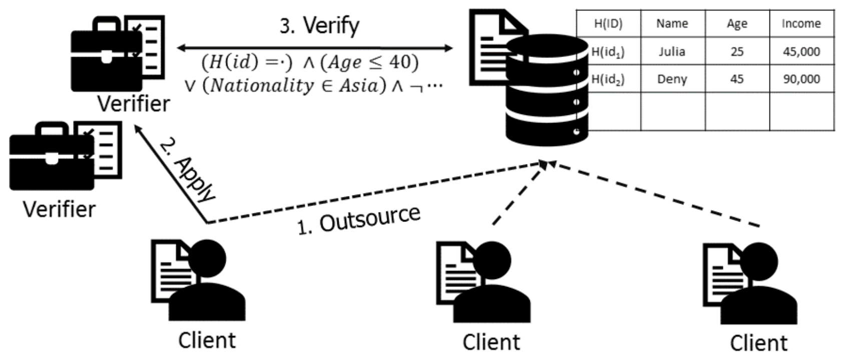
Recent advances in technology have led to the introduction of many digital and automated services, such as e-shopping, e-learning, and e-banking. These digitalised services not only reduce the cost of operation, but also increases throughput for businesses. However, several tasks in these services continue to involve a considerable amount of human effort. Physical document verification is a necessary task in the process of reviewing applications for many services, such as loans, insurances, and mortgages. This process consumes a large amount of time, money, and human resources. Consider the example of a loan or an insurance application. The applicants are usually required to provide numerous documents to certify their relevant personal information, such as birth certificate, statement of monthly income, marriage certificate, medical records, and so on. At the same time, the loan/insurance provider requires a considerable amount of human resource to verify and store these documents. This process can take several weeks to complete, and serves to limit business throughput.

Moreover, the process of physical document verification incurs a critical privacy risk for applicants. They provide to a third party (i.e. the service provider) many sensitive documents, such as birth certificates, IDs, health records, and so on. All these documents are stored in the provider's database. If the client applies for multiple schemes or subscriptions, multiple copies of his/her personal data are stored in different places. Since data can be leaked from the server, storing personal information in multiple third-party databases is not recommended. One source of such a leak is employees who do not follow the company's privacy policies, and may, intentionally or unintentionally, reveal sensitive client information. Even when the provider claims to enforce strict policies pertaining to privacy, there is still a chance that the database systems are vulnerable to malicious external attacks.

This chapter presents a systematic approach to address the abovementioned shortcomings of the current state of the process of physical document verification. We assume that there is a trusted data source that stores the certified personal information of clients. We also assume that a list of requirements (maybe involving the divulgence of private information) needs to be fulfilled by the applicant to qualify for a given scheme or subscription. We present a series of protocols that allow the verifier and the data keeper to communicate with each other and securely verify the applicant's information according to the requirements proposed by the verifier. The proposed approach creates space for new services for information data storage and verification.

6.1.1 Motivating Scenario

To illustrate, consider the following example: A client first outsources his/her personal information to a data keeper called Alice. Alice can be a governmental agency. The client and Alice are responsible for ensuring the correctness of the information. The client can pay a small fee to Alice for keeping track of his/her data. The verifier Bob provides a subscription scheme. In order to subscribe to the scheme, the client needs to satisfy a number of statements for personal information, including age, income, nationality, health condition, etc. He/She wants to prove that he/she qualifies for the scheme, but does not want to reveal exact information. At the same time, he/she also wishes to hide the fact that he/she is applying the certain scheme through others (such as Alice). He/She should anonymously authenticate Bob to communicate with Alice. Bob interacts with Alice by our proposed approach. Finally, Bob should be able to decide whether the client qualifies for the given scheme.


3. Verify
$(H(id) = \cdot) \wedge (Age \leq 40)$
$\vee\ (Nationality \in Asia) \wedge \neg \cdots$
Verifier
Verifier
2. Apply
1. Outsource
Client
Client
Client

H(ID)	Name	Age	Income
$H(id_1)$	Julia	25	45,000
$H(id_2)$	Deny	45	90,000

Figure 6.1: Secure personal information verification system model

6.1.2 Organization

The remainder of the chapter is organised as follows: In the next section, we review related work in the literature. Section 6.2 contains our problem formulation as well as our security model and assumptions. The proposed solution to the problem of secure personal information verification is described in Section 6.3, which systematically discusses four stages of the solution. Section 6.4 presents experimental evaluations of the proposed sub-protocols. The final section discusses future work and our conclusions.

6.2 Problem Formulation

6.2.1 Problem Statement

Definitions. Our proposed system involves three general parties—the client, the verifier, and the data keeper—as illustrated in Figure 6.1.

- The client. The client wishes to privately prove that his/her personal data satisfy the predicates predefined by the verifier.
- The verifier. The verifier (we call the verifier Bob) provides a series of predicates that need to be satisfied by the client.

- The data keeper. The data keeper (whom we call Alice) stores the personal data of the client and provides a security guarantee for the data storage.

The database stored by the data keeper consists of n records. Each record describes a client by m attributes. Table 6.1 presents a simple example of data content maintained by the data keeper.

Table 6.1: Sample personal data records

ID	Name	Age	Sex	Income	Nationality	Marital Status
1	Julia	29	F	30,000	Singaporean	Married
2	Deny	32	M	35,000	Singaporean	Single
3	Christina	38	F	80,000	Myanmar	Single
4	Alwen	41	M	120,000	Indonesian	Married
5	Dino	37	M	90,000	Malaysian	Divorced

A *personal information verification scheme* is a Boolean function on a data record. The Boolean function is informally described by single and complex predicates. We assume that the single predicates are equality, inequality, membership, and non-membership.

- An equality predicate examines whether a variable x is equal to a certain value a: $x \overset{?}{=} a$.
- An inequality predicate inputs a variable x and a certain value a, and outputs 1 when $x \overset{?}{<} a$ (and 0 otherwise).
- The membership and non-membership predicates check whether variable x belongs (or does not belong) to a set A of elements: $x \overset{?}{\in} A,\ A = \{e_1, e_2, \cdots, e_n\}$.

A complex statement contains multiple single predicates and a set of logical expressions $\wedge, \vee, \neg$. A series of predicates (provided by the verifier) can be expressed as a complex predicate by combining them using the $\wedge$ operator. The personal data of clients is accepted by the verifier only if they satisfy the final complex predicate.

6.2.2 Security Assumptions

In this paper, the privacy/security of the proposed protocols is measured by the amount of information disclosed during execution. We adopt the security definitions and proof techniques from the literature on secure multi-party computation to analyze. The secure multi-party computation problem involves multiple parties collaboratively performing various types of computation without compromising the privacy of data. In the mid-1980s, C. Yao [86] introduced the idea of securely computing any two-party functionality in the presence of dishonest adversaries. Since then, various privacy-preserving protocols have been proposed to address different class of computation problems in the context of private data.

There are two common adversarial models under secure multi-party computation: semi-honest and malicious. In the malicious model, the adversary has the ability to arbitrarily deviate from the protocol specifications. On the other hand, in the semi-honest model, an attacker (i.e. one of the participating parties) is expected to follow the prescribed steps of the protocol. However, the attacker is subsequently free to compute additional information based on his or her private input, output and messages received during the execution of the secure protocol. Although the assumptions of the semi-honest adversarial model are weaker than those of the malicious model, we insist that this assumption is realistic under the problem settings. We assume that the data keepers are trusted (as they are governmental agencies) to ensure the confidentiality of sensitive client data. It is difficult to imagine them colluding with other companies to damage their own reputation. Moreover, it is often non-trivial for one party to maliciously deviate from a particular protocol which may be hidden in a complex process.

In short, we assume that the verifier and the data keeper are semi-honest. They will correctly follow the protocol specifications. However, at the same time, they are also curious about the applicants' information. In general, secure personal information as described in Section 6.3 should meet the following privacy requirements:

- Client-to-verifier privacy. The verifier should not be able to gain any details concerning the client's personal data stored in the data keeper's database, except for those he can learn from the result (i.e. the client qualifies or not).

- Client-to-data-keeper privacy. At any point during protocol execution, the identity of the applicant should not be revealed to the data keeper.

- End user's privacy. The verifier should not be able to obtain any information relating to other clients stored in the data keeper's database.

- Verifier-to-data-keeper privacy. The details of the predicates should not be leaked to the data keeper. This requirement is particularly applicable to private services where the selection criteria may be private to the provider.

6.3 Secure Information Verification

6.3.1 Overview

Our proposed approach to the secure information verification problem consists of four stages:

(i) *Setup* - During this phase, the client goes through an anonymous authentication process so that the verifier is authenticated to communicate with the data keepers for the verification stage. In addition, the data keeper and the verifier generate an encryption key pair and exchange the public key of the homomorphic cryptosystem. These public keypairs are utilized for secure communication and computation at the later stages.

(ii) *Single Predicate* - In this stage, the data consumer evaluates a predicate for each entity in the dataset of the data keeper. The output of this stage is the encryption of either 1 or 0, depending on whether the entity satisfies the predicate.

(iii) *Secure Complex Predicate Evaluation* - Based on the results of the previous stage, the verifier collaborates with the data keeper to compute the result of the complex logical combination of Boolean predicates. Again, the output of this stage is the encryption of either 1 or 0 depending on whether the entity satisfies the predicate.

(iv) *Aggregation of Output Data* - At this stage, the final result is aggregated, decrypted and shown to the verifier. Since the data keeper computes the decryption, we propose a secure protocol to generate the outcome so that the data keeper cannot obtain any information concerning the final result.

6.3.2 Setup

In the setup phase, the client is first required to complete an anonymous authentication with the data keeper Alice, who then allows the verifier Bob to initiate the secure information verification process for the records of Alice's database. When the client agrees to his/her personal information being stored in Alice's database, she issues to the client credentials to be used for authentication. Each time a client subsequently requests access to Alice's database, he/she uses the credentials for verification with Alice, who begins communication with Bob for the information verification process.

Traditional password-based authentication systems expose the identity of the client to the data keeper Alice. Hence, they violate the client-to-data-keeper privacy requirement. To satisfy this, it is desirable to have an authentication scheme that promises unlinkability, i.e. the server should not be able to link user requests such that access to the same user cannot be recognized as such.

As the anonymous authentication process is not our main contribution here, we only briefly review possible approaches to satisfy this requirement. The most feasible solution is anonymous credentials introduced by Chaum [28]. This allows a user to prove that he/she has obtained a credential issued by an organization without revealing anything regarding his/her identity other than the credential. Camenisch [24] proposed a protocol that allows an organization to issue a credential by obtaining a signature on a committed value. The client can then prove with zero knowledge that he/she has a signature under the organization's public key on the given value.

When applied to our problem setting, the client first generates a non-interactive zero-knowledge proof (i.e. applying the Fiat–Shamir transform) of his/her credentials with Alice. The client transfers the proof to Bob, who submits the proof to Alice. Finally, Alice authenticates Bob to communicate and verify the client's information.

Following the authentication process, Alice and Bob generate two Paillier key pairs using the $KeyGen$ algorithms and agree on two public key pairs for communication during the verification execution. We denote $Enc_A(\cdot)$ and $Enc_B(\cdot)$ as the Paillier encryption under Alice's public key and that under Bob's public key, respectively.

6.3.3 Single Individual Predicate Evaluation

In the single predicate evaluation stage, for each data record and each attribute that needs to be verified, the verifier Bob and the data keeper Alice together perform one of the following protocols: equality predicate evaluation, inequality predicate evaluation and (non-) membership predicate evaluation. The output of each protocol is an encrypted bit maintained by Bob. The resulting bit is encrypted under the data keeper's public key so that Bob cannot obtain any information relating to the other entities in the database. We now describe the three protocols to securely evaluate the results of these predicates.

6.3.3.1 Equality Predicate

A secure equality predicate evaluation tests whether two private inputs x and y are equal: $x \stackrel{?}{=} y$. We use the protocol presented by C. Gentry *et al.* [44] to develop the protocol for secure equality predicate evaluation. Gentry's equal-to-zero protocol [44] allows the comparison between a private value and zero. To be able to apply the equal-to-zero protocol, we execute a transformation (as presented in Protocol 7) on the two private inputs.

The computation in Steps 1-2 transforms the problem into a secure equal-to-zero protocol. In this protocol, Alice holds an encrypted message with value a. The message is encrypted under Bob's key; hence, neither Alice nor Bob has information concerning the value a. The remaining part of the protocol involves comparing a with 0. In the last step, Bob is required to compute an AND operator in the ciphertext space. In a binary setting, the AND operator is an exact multiplication scheme. In Chapter 2, we introduced the secure multiplication scheme. That protocol works on general integer inputs. Here, we present a variant of that protocol that only works with binary inputs. However, with this modification, we are able to improve the performance of the Secure Equality Evaluation protocol.

The key idea of the modification is instead of masking the input by adding a random number, we only need to mask a random bit in the case of binary inputs. The variant of the Secure Multiplication protocol is presented in Protocol 8.

Algorithm 7: Secure Equality Evaluation

```
Input: Alice holds integer $x$, Bob holds integer $y$
Output: Bob holds an encrypted bit $Enc_A(b)$ such that $b = 1$ if $x = y$, and 0
        otherwise.
1 Bob computes and sends $Enc_B(y)$ to Alice;
2 Alice computes $Enc_B(a) = Enc_B(x - y)$ using the homomorphism;
3 Alice chooses a random $r$ and uses the homomorphism to compute
   $c \to Enc_B(a + r)$;
4 Denote the bit representation of r by $r_n \cdots r_2 r_1$. Alice encrypts the bits $r_i$ under
   her key to obtain $c_i = Enc_A(r_i)$;
5 Alice sends to Bob both $c$ and all $c_i$;
6 Bob decrypts c to obtain $a' = a + r$. Let denote $a'_n \cdots a'_2 a'_1$ be the binary
   representation of $a'$;
7 For each $i$, Bob computes $c'_i = Enc(1 - r_i)$ if $a'_i = 0$; otherwise, keep $c_i$ unchanged;
8 Bob computes $rc = \bigwedge_i c'_i$, and outputs $Enc_A(b) = 1 \ominus rc$;
```

Bob is able to evaluate $Enc_A(x \oplus r)$ since $Enc_A(x \oplus r) = Enc_A(1 - x)$ when $r = 1$ and $Enc_A(x \oplus r) = Enc_A(x)$ when $r = 0$. $Enc(y \oplus s)$ at line 2 is similarly computed. During the protocol, Bob only works on encrypted data while the server receives two random numbers. Hence, no information regarding x and y is obtained by *Bob* and S. The correctness of the protocol is trivial. Step 4 presents a data packing technique of encrypted data. The modification reduces the bandwidth memory, in which Alice sends to Bob, from two to only one ciphertext. It also further reduces the number of decryption operations of Bob from two to one when compared to the original protocol described in Chapter 4.

As mentioned above, the modification allows us to improve the efficiency of the Secure

Algorithm 8: Secure AND

```
**Input:** Bob holds $(Enc_A(x), Enc(y))$- the encryption of two bits $x$ and $y$, and
        Alice holds private key $sk_A$
**Output:** Bob holds $Enc_A(x \wedge y)$
1 Bob generates two random bit $r, s$;
2 Bob computes $Enc_A(z_0) = Enc_A(x \oplus r), Enc_A(z_1) = Enc_A(y \oplus s)$;
3 Bob packs the encrypted data: $Enc_A(z) = Enc_A(z_0 + 2 * z_1)$ and sends them to
   Alice;
4 Alice decrypts, unpacks and obtains $x \oplus r,\ y \oplus s$;
5 Alice computes $(x \oplus r)(y \oplus s)$ and sends $Enc((x \oplus r)(y \oplus s))$ to Bob;
6 Bob computes
   $Enc(x \wedge y) = Enc((x \oplus r)(y \oplus s)) - Enc(x \wedge s) - Enc(y \wedge r) + Enc(r \wedge s)$;
```

Equality Evaluation. The last step of the protocol requires the computation of AND operations on n encrypted bits $\{c_1, \cdots, c_n\}$. The straightforward solution is performing Protocol 8 n times on the aggregated results. This approach requires n rounds of communication, n encryption operations, and $2n-2$ decryption operations. For the text domain, the size of the inputs may be as high as several hundred bits (after hashing), it incurs a high cost for communication and computation. Here, we introduce an improvement to reduce the number of communication round and encryption/decryption operations to $O(logn)$. The improvement is based on data packing technique. The protocol describes details of our improvements. For simplicity, we assume that the size of input in bits: $n = 2^k$.

The correctness of Protocol 9 can be proven in similar way as Protocol 8. Each round reduces the size of the input by a factor of two, hence there is only $logn$ communication rounds. At each round, Bob is only required to decrypt one ciphertext. Hence, the

Algorithm 9: Secure AND on multiple binary inputs: $f(c, n)$

```
    Input: Bob holds $n$ encrypted bits: $\{Enc_A(x_1), \cdots, Enc(x_n\}$
    Output: Bob holds $Enc_A(\bigwedge_i x_i)$
1  if $n$=1 then
2      Bob returns $c_1$;
3  for $i = 1, \cdots, n$ do
4      Bob computes $Enc_A(x'_i) = Enc_A(x_i \oplus r_i)$, where $r_i$ is a random bits;
5      Bob computes $Enc_A(x'_i) = Enc_A(2^i \times x_i)$;
6  end
7  Bob computes $Enc_A(x') = Enc_A(\sum 2^i \times x_i)$ and sends to Alice ;
8  Alice decrypt $x'$ and unpack $x'$ to bits $\{x'_1, \cdots, x'_n\}$;
9  for $i = 1, \cdots, n/2$ do
10     With $(2i) - th$ and $(2i - 1) - th$ bits, Alice computes $z'_i = x'_{2i-1} x'_{2i} \cdot$;
11     Alice encrypt $Enc_A(z'_i)$ and sends to Bob;
12     Bob computes
         $Enc_A(z_i) = Enc(z'_i) - Enc(x'_{2i-1} \wedge r_{2i-1}) - Enc(x'_{2i} \wedge r_{2i}) + Enc(r_{2i-1} \wedge r_{2i})$
13 end
14 Bob recursively performs $f(z, n/2)$;
```

number of decryption operations is reduced to *logn*. On the other hand, the bandwidth and the encryption operations reduce by a constant factor (i.e. two times), and the asymptotic complexity remains the same.

Analysis. We now analyze the correctness and security of the equality predicate evaluation protocol (Protocol 7). Due to the transformation in Steps 1-2, we only need to examine the remaining parts, where the two parties together compare the encrypted value a with 0.

We note that in Step 7, Alice reserves bit c_i when $a'_i = 0$. This means that we perform the $XNOR$ operation on bit a'_i and ciphertext c_i for all $i = \overline{1, n}$. Remember that $c_i = Enc_A(r_i)$, so at this step, we actually compute $c'_i = Enc_A(f_i) = Enc_A(r_i\ XNOR\ a'_i)$. Moreover, since $a' = a + r$, all $r_i \oplus a'_i$s are zero if and only if $a = 0$. Therefore, all $f_i = 1$ if $a = 0$; the last step concludes the protocol, where Bob computes the AND of all bits f_i and outputs the inverted result.

The security of the two parties follows the semantic security properties of the employed encryption scheme—the Paillier cryptosystem. Alice only obtains the encryption version of y. On the other hand, Bob receives a randomized value $a' = a + r$ and an array of encrypted bits $\{Enc(r_n), \ldots, Enc(r_1)\}$. Hence, no more information is leaked to either party.

6.3.3.2 Inequality Predicate

The inequality predicate considers two parties that pose two private integral values x and y and wish to evaluate the predicate $x \overset{?}{<} y$. This problem is known as secure comparison. The first solution to it was proposed by A. Yao [86] in the 1980s, and pioneered research on secure multi-party computation. Since then, extensive research has been conducted to address the problem of secure comparison. Di Crescenzo [30] proposed a secure comparison protocol with $O(n^2 logN)$ complexity, where n is the length of the compared numbers in bits, and N is the group size of the plaintext of the employed encryption scheme. Fischlin [38] and Blake [9] reduced the complexity of the solutions to $O(nlogN)$.

We propose a variant of Blake's protocol [9] as a building block to compare two private inputs. In our problem setting, at this stage, it is expected that no information relating to the results of the evaluation are known to the verifier or the data keeper. Therefore, we cannot directly apply the protocol proposed by Blake [9], as it leaks the comparison results to one of the parties. In order to prevent such information leakage, we propose a mechanism, as an extension of the original scheme [9], shown in Protocol 10.

Blake's protocol allows us to obliviously transfer one over two secrets depending on the result of the secure comparison. The protocol considers the scenario where there are two parties holding two private inputs x and y. The second party holds two secrets (s_0, s_1) (in addition to private input y). Blake's protocol allows the two parties obliviously transfer s_0 when $x > y$ and s_1 in the other case. Our modification adds one more step which is the secure equality evaluation protocol to determine the secret that has been sent. At the end of the protocol, Bob obtains an encrypted bit that indicates the result of the inequality comparison. To present the protocol, we follow Blake [9] and denote with D_S a set of integers agreed to by the two parties before executing the protocol.

In Step $3.b$, Bob is required to compute the XOR of two encrypted bits x_i, and y_i. Since $f_i = x_i \oplus y_i = x_i + y_i - 2x_iy_i$, we can evaluate the result with the help of the secure multiplication protocol (Protocol 8). To compute the encryption of vector γ, δ, μ, Bob only needs to apply the homomorphic property of the Paillier cryptosystem.

Analysis. We first show that the protocol correctly computes the desired functionality. The flag vector $f = \{f_i = x_i \oplus y_i\}$ is a binary vector, where the i-th bit indicates whether $x_i \neq y_i$. Therefore, vector γ as constructed in Step $2c$ is a vector with the following structure: it starts with one or more 0s followed by a 1, and then a sequence of non-1s.

Let k be the first position where x_i and y_i differ, which implies that $\gamma_k = 1$ and d_k determine the result of predicate $x < y$. δ randomizes the value of γ but keeps $\delta_k = d_k$. We note that with a large probability, γ_i is statistically close to being uniformly random in Z_N. Finally, the transformation in Step 2e is a permutation of Z_n where set $-1 \to s_0$ and $1 \to s_1$. The final step, where a random permutation $\pi(\mu)$ is sent back to Alice, hides information concerning index k.

Since there is a negligible minority of elements of D_S in a group of size N, with an overwhelming probability, there is exactly one element in vector μ belonging to set D_S.

```
Algorithm 10: Secure Inequality Evaluation
  Input: Alice holds integer $x$ and Bob holds integer $y$; common integer set $D_s$
  Output: Bob holds an encrypted bit $Enc_A(b)$ such that $b = 1$ if $x < y$, and 0
          otherwise
1 Bob draws two random number $s_0 \neq s_1$ from the set $D_s$;
2 Denote the bit representation of $x$ by $x_n \cdots x_2 x_1$; Alice encrypts each bit $x_i$ under
    her key and sends $(Enc_A(x_1), \ldots, Enc(x_n))$ to Bob;
3 For each $i = 1, \ldots, n$, Bob does the following:
    $(a)$ Compute $Enc_A(d_i) = Enc_A(x_i - y_i)$
    $(b)$ Compute the encryption of the XOR between $i - th$ bits
    $Enc_A(f_i) = Enc_A(x_i \oplus y_i)$ using homomorphism.
    $(c)$ Compute an encryption of vector $\gamma$, where $\gamma_0 = 0$ and $\gamma_i = 2\gamma_{i-1} + f_i$
    $(d)$ Compute an encryption of vector $\delta$, where $\delta_i = d_i + r_i(\gamma_i - 1)$, where $r_i$ is a
    random number in $Z_n$
    $(e)$ Compute a random encryption of vector $\mu$, where $\mu_i = \frac{s_1 - s_0}{2}\gamma_i + \frac{s_1 + s_0}{2}$
    and send a random permutation $\pi(Enc_A(\mu))$ to Alice.;
4 Alice obtains $\pi(Enc_A(\mu))$, decrypts it and determines that $\mu$ contains a single
    value $v \in D_S$;
5 Alice and Bob perform secure equality evaluation on inputs $v, s_0$;
6 Bob outputs the encrypted result $Enc(b)$;
```

In Step 3, Alice can output either s_0 or s_1 depending on whether $x < y$. The last two steps conclude our construction of the protocol, where $v = s_0$ only if $x < y$ as desired.

We now prove the security of the protocol. Due to the universal security of the secure equality evaluation protocol, we only need to consider the first part (i.e. Steps $1 - 3$). Alices privacy trivially holds because of the semantic security properties of the employed encryption scheme—the Paillier cryptosystem. Bob only receives from Alice a list of encryption messages, and obtains no more information about Alice's private input.

Bob's privacy against the semi-honest party Alice is proven by constructing a simulator $Sim_A(x, v)$, where x is the private input of Alice and v the value obtained in the part of the protocol that is examined. $Sim_A(x, v)$ needs to generate a distribution statistically close to the view of Alice in real execution. The simulator generates a random vector μ': for $i = 1, \ldots, n$, and a random element $\mu'_i \in Z_n$ is chosen. It then replaces the randomly chosen element of μ' with s (i.e. $\mu'_i \leftarrow s$), and outputs $\{x, Enc(\mu')\}$. As discussed above, due to the randomization in Step 2.d, vector μ is statistically close to being uniformly random in Z_N (except one element in D_S).

6.3.3.3 (Non-)Membership Predicate

A membership predicate allows the verifier to examine whether an attribute of the client falls into certain categories. A simple example is the case where the verifier wishes to know if an applicant works in the education industry (e.g. as a teacher, student, librarian, or school counselor). A non-membership predicate should be the complement of the membership query, and tests whether a particular value is excluded from a set.

The membership predicate evaluation protocol is presented in Protocol 11. The non-membership predicate can be easily derived from Protocol 11 by applying the *NOT* operator discussed in Section 6.3.4.

In the protocol, Alice is required to evaluate the encrypted polynomial $P(x)$ at point $x = a$ (line 3). She can do so due to the homomorphism of the cryptosystem. She first computes a^i in plaintext, and then computes $Enc_B(c_i \times a^i) = Enc_B(c_i)^{a^i}$ by the homomorphic multiplication property of the Paillier cryptosystem. Finally, she calculates $P(a) = \sum c_i \times a^i$ in encrypted form by applying the homomorphic addition property.

Algorithm 11: Secure Membership Evaluation

```
Input: Alice holds an integer $a$, Bob holds a set if integer $S = \{x_1, x_2, \cdots, x_n\}$
Output: Bob holds an encrypted bit $Enc_A(b)$ such that $b = 1$ if $a \in S$ and 0
        otherwise
1 Bob computes the characteristic polynomial of the set:
   $P(x) = c_n x^n + \cdots + c_1 x + c_0 = (x - x_1)(x - x_2)\cdots(x - x_n)$;
2 Bob encrypts the coefficients of a polynomial $\{c_0, c_1, \ldots, c_n\}$ under his own key,
   obtaining $\{Enc_B(c_0), Enc_B(c_1), \ldots, Enc_B(c_n)\}$, sends the encrypted set to Alice;
3 Alice securely evaluates the encrypted value of the polynomial at $x = a$, denote
   $v \to P(a)$;
4 Alice generates random $r$, and computes $Enc_B(v + r)$ and sends it to Bob;
5 Bob decrypts to obtain $v + r$;
6 Alice and Bob perform Secure Equality Evaluation to with inputs $r$,$v + r$;
7 Bob outputs the encrypted result Enc(b);
```

Analysis. We first analyze the correctness of the protocol. If $a \in S$, there exists one x_i such that $x = x_i$. This implies $v = P(a) = 0$. This observation leads to the final step of the protocol, where Bob and Alice collaboratively evaluate the equality predicate with inputs r and $v + r$. Hence, the encrypted bit $Enc(b)$ is an indicator of whether value a belongs to set S.

The security of the protocol can be proven with two simulators that generate the views of the two parties, Alice and Bob. For Alice, a simulator that generates and sends n random encrypted values is a valid simulator. Due to semantic security, she cannot distinguish the simulator from a real-world scenario. Similarly for Bob, a random number $v + r$ can be easily simulated. Finally, the security of Protocol 11 concludes the proof of security of the secure membership evaluation protocol.

6.3.4 Complex Predicate Evaluation

At this stage, Bob holds the encrypted result of the evaluation for each data record, with each attribute in a complex predicate that needs to be verified. There are three basic primitives that operate on the encrypted inputs at this stage. With these primitives, Bob has the capability to compute the results of the encryption of the desired bit to evaluate each data record. The output of this stage is an encrypted bit for each data record. This bit indicates whether the given record satisfies the complex statement.

The inputs of the three primitives are either one encrypted bit (NOT operation) or two encrypted bits (AND and OR operations).

(i) $\neg$ (NOT) - The negation operation is computed by the formula: $Enc(\neg x) = Enc(1) - Enc(x)$.

(ii) $\wedge$ (AND) - Protocol 8 describes the method to compute AND operation on two encrypted binary inputs.

(iii) $\vee$ (OR) - OR operation evaluation protocol is derived from the formula: $x \vee y = x + y - x \times y$.

They are described as similar to the those described in Chapter 4. However, with the data packing techniques we introduce in this chapter, the bandwidth memory as well as the number of decryption operations can be further reduced by a constant factor. This modification significantly improves the performance of the sub-protocols.

6.3.5 Aggregation of Output Data

For the input of this stage, Bob holds an encrypted bit, for each entity in Alice's database, that determines whether the data record satisfies the complex statement. In order to ensure that there is exactly one qualified data record in the case that the application is successful, we introduce one special attribute to the final complex predicate. The attribute is the secret identification of the client in the database.

We assume that when the client registers his/her data with Alice the data keeper, Alice generates a secret random number r_c to identify the client. The number is stored in the database as an attribute of the client. We introduce additional steps to address the requirement:

(i) The client encrypts the random secret under Bob's key, obtaining $Enc_B(r_c)$.

(ii) The client anonymously sends the encryption of secret value to Alice.

(iii) Alice and Bob perform secure equality evaluation (starting from Step 2) and get the result $Enc_A(b)$.

(iv) Bob applies the AND operation on $Enc_A(b)$ and the current result of the evaluation process.

With the additional step, Bob now holds an array of encrypted bits with all 0s and at most one bit 1. Bob uses a homomorphism to compute the encrypted sum of these bits; the result is the encryption of either 1 or 0. He can send it to Alice for decryption and obtain the final result to determine whether the applicant qualifies. However, this may compromise Bob's privacy, especially when he wants to hide his business progress. In order to maintain his privacy, we introduce one step for the randomization of the decryption process as follows:

(i) Bob computes the encrypted sum using a homomorphism to obtain $Enc_A(s)$.

(ii) Bob generates a random number r, and computes $c = Enc_A(s + r)$ and sends it to Alice.

(iii) Alice decrypts c to obtain $s + r$ and sends it back to Bob. Note that Alice only receives a random number so that she learns nothing about the result of the application.

(iv) Bob computes the result $s = s + r - r$.

Finally, Bob is able to decide the result of the verification process using bit s.

6.3.6 Discussion

We first consider the security of the entire system, since all intermediate results, that are revealed to Alice and Bob, are either random or semantically secure encryption of numbers. Furthermore, the outputs of all sub-protocols (only seen by Bob) are always encrypted under Alice's key. Under the assumptions of the semi-honest model, we claim that the sequential composition of these sub-protocols leaks no details of the client or the predicates proposed by the verifier.

The second issue we consider is the practical implementation of the system. Since the same procedure is applied for all data entries, the verification results for each data record can be computed in parallel. That means we are able to construct multiple verification threads, each one corresponding to one data entry. By the batch verification approach, we can improve the running time of the whole process by a factor of n/m, where n is the number of data records and m is the number of threads.

While the same procedure is applied for each data record, the data keeper is not able to know who the applicant is. In practice, there are some cases where the data keeper (e.g. a governmental agency) is allowed to know the identity of the applicant, and this rigorous security feature is then not required. The proposed solution can be modified, and inherently improves performance. Specifically, the client can perform a simple authentication rather than an anonymous solution to allow the verifier to communicate with the data keeper. The verification process only needs to be performed on the one data record only identified by the client. Hence, the cost of the proposed solution is reduced by a factor of n where n is the number of data records in the data keeper's database.

In our proposed solution, an applicant qualifies only if he/she satisfies all criteria specified by a single predicate or several complex predicates. Hence, we can define a complex predicate to cover all criteria using the AND operation. We also can extend our protocol to adapt to threshold criteria, where the applicant qualifies only if he/she satisfies more than k criteria. The idea is to compute the sum of each of the predicate's evaluation (in encrypted form) and apply a slightly modified version of Protocol 10 to compare the encrypted value with threshold k.

6.4 Implementation

We implemented our proposed method, and calculated the CPU time required to run our sub-protocols from Section 6.3. Our experiments were conducted on a Windows 10.0 machine with a 3-GHz processor and 16 GB of RAM. We used the Paillier cryptosystem as the underlying additive homomorphic encryption scheme and implemented the proposed sub-protocols in Java.

We first examined the operation of the secure equality evaluation and the secure inequality evaluation protocols. Two factors affect the performance of these protocols: the Paillier key size and the domain size of the input. Table 6.2 shows the processing times of Algorithms 7 and 10 with different settings of bit size and key size. We performed the experiment with bit lengths of 32, 64 and 160. The latter was the size of the output of the SHA-1 hash function we used for the secret identification described in Section 6.3.5. The result showed that these protocols require twice the time for double-bit size of inputs; the time needed increased by a factor of nearly seven when the Paillier key size was doubled.

1024 bit Key size

Size	Algo.7	Algo.10
32	796	1769
64	1472	3542
160	3277	8623

2028 bit Key size

Size	Algo.7	Algo.10
32	4477	12047
64	9983	24755
160	22569	57393

Table 6.2: Run times of secure equality and inequality evaluation protocols (ms)

The third building block of the single-predicate evaluation was the (non-) membership predicate. The run time of the building block depends on three factors: the Paillier key size, the number of elements in the set, and the bit size of the inputs; bit size only affects the final step of Protocol 10, which is the secure equality evaluation protocol. Figure 6.2 below shows the relationship between the run times of the two remaining factors and the performance of the building block.

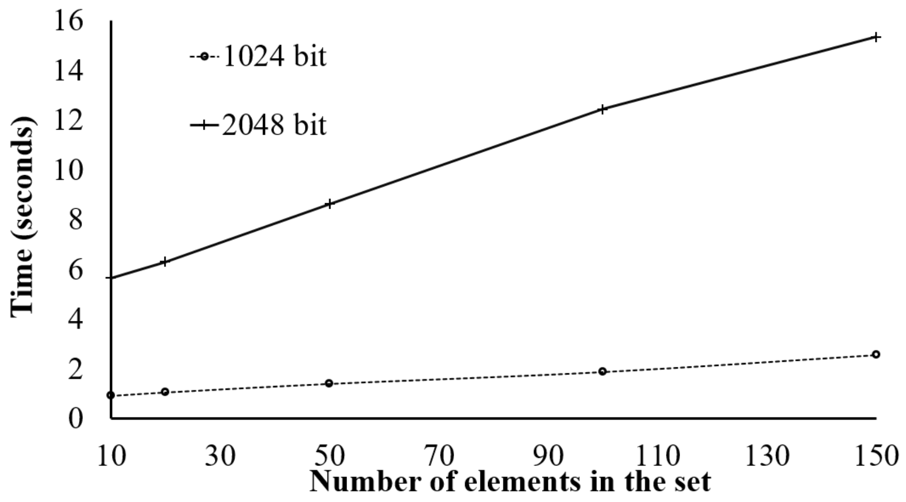


Figure 6.2: Running time of Secure Membership Evaluation

We had made a similar observation earlier: the cost of the secure membership evaluation protocol when the key size was 1024 bits was roughly six to seven times more efficient than with a length of 2048 bits for the Paillier key. The computational cost of the protocol also increased linearly with the size of the set.

In order to verify the feasibility of the whole proposed system, we conducted an experiment on a simulated dataset. We considered a complex statement verification comprising of 10 single predicates linking together by two Boolean operations AND, OR. The running time for verifying a single data record was 25 seconds, and it took approximately one hour to verify one thousand data records in the parallel mode of 10 threads running simultaneously.

6.5 Summary

This chapter presents a framework for the privacy-preserving verification of personal information. We used the secure multi-party computation model and homomorphic encryption to develop a systematic solution to the problem in four stages. We showed that the proposed scheme can protect the client's privacy from both the verifier and the data keeper, and at the same time, protects the privacy of the verifier. Different ways to further enhance the performance of the proposed method and a scheme extension for threshold verification were discussed. The experimental results highlighted the efficiency and feasibility of our proposed scheme under different security settings.

7 Conclusion

In this chapter, we recapitulate the previous chapters and highlight key points that are the major contributions from this thesis. We then discuss several promising directions of research in secure data processing in the future.

7.1 Conclusion

Data privacy has emerged as an important area of research in the recent years, due to the increasing amount of information available that contains sensitive data from people and companies. Securing data at rest does not adequately address the modern requirements for data confidentiality. The demand for supporting various digital services requires efficient methods to process and extract useful subsets from encrypted data while preserving data confidentiality. The new challenges need new approaches rather than simple data encryption. In this thesis, we have proposed various techniques to efficiently process the encrypted data in a secure manner.

In Chapter 3, we presented a simple but secure solution that supports conjunctive matching queries. The searching scheme is based on the additive homomorphism property of cryptosystems such as the Paillier cryptosystem. We presented the three-party architecture and applied the secure multi-party model to analyze the security of the solution. The solution can also be extended to support range and multi-dimensional range queries. We also investigated the overhead incurred by the solution.

Chapter 4 presented a more general problem of searching encrypted data. In this chapter, we considered the classical problem of Boolean information retrieval. The solution allows the evaluation of any combination of Boolean functions over a set of keywords. The solution employs Bloom-filter data structure, oblivious transfer techniques, and homomorphic encryption. The solution consists of several phases. At each stage, secure protocols were proposed to solve a subproblem.

Chapter 5 considered the problem of multi-dimensional range queries. It addressed the limitation of the previous solution presented in Chapter 3. In this chapter, we examined various scenarios. Different approaches were proposed to address the requirements for these different contexts. The basic idea behind the solution is to combine the bucketization and Oblivious RAM techniques to secure access to the data.

Finally, in Chapter 6, wwe considered the problem of personal information verification. The root of the problem is similar to the problems that have been discussed in the three previous chapters. We took a similar approach to address the security requirements. The solution eliminates the limitations of the current personal verification process. Moreover, it also creates an opportunity for a new kind of digital service.

7.2 Future Work

As an ongoing effort, we are exploring several extensions to the work in this dissertation.

Stronger Security Notions. In the thesis, we assume the participating parties follow the semi-honest model. We assume they faithfully act according to the protocol specification. While we emphasize that this model is more practical than the concept of a strong security notion with a fully malicious adversary. The fully malicious adversary model makes no assumptions of the behaviors of the adversaries. They may diverge from the normal execution of a protocol, as long as the deviation cannot be detected. It is possible for them to craft the input or wrongly compute certain corresponding functions. While this is unlikely to happen to the well-established service providers, the fully malicious adversary model still provides a stronger security guarantee and covers more real-world scenarios.

In Chapter 3 and 5 we assumed that the participating parties (i.e., proxy and the servers, index server and data server) do not collude. This may be not sufficient for several applications. In a future work, we may explore possible approaches to eliminate this requirement.

Multi-user Support Our works support single user model. Loosely speaking, the client is only able to interact with his/her own dataset. In order to support multi-user scenario, Chapter 3 and 4 required the data owner to share the secret key with the authorized data consumers. Popa and Zeldovich [72] proposed an interesting scheme that enables single keyword search in the outsourced database for multi-user settings. In future works, we will explore possible approaches that support secure complex query processing in multi-user contexts without key sharing.

Multi-source Support. The idea of supporting multi-source datasets should be an interesting extension. In the context of data outsourcing, different types of data may be required to be outsourced to different service providers. The research question is how to efficiently support the queries whose answers are fused information from these various encrypted databases. In the context of personal information verification, the client may wish to have different organizations handle different personal documents; for example, a bank to take care of financial documents or a hospital to store healthcare records. We wish to explore secure multi-party computation solutions to allow the private verification process to be performed in these distributed settings.

Graph dataset. Graph structure provides a way to represent the complex data structure and relationships between data entities. The abundance of information in the real world today can be modeled as graph-structured data, e.g. communication networks, biological networks, or social networks. Therefore, graph analysis has become an interesting and attractive research topic and will be for a long time. In future research, we will examine efficient methods for storing graph-based data as well as securely answering graph queries such as connectivity or shortest-path queries.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *J. Cryptology*, 21(3):350–391, 2008.

[2] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 563–574, 2004.

[3] Moheeb Alwarsh and Ray Kresman. On querying encrypted databases. In *10th International Conference on Security and Management, Las Vegas, USA, June 18-21, 2011.*, 2011.

[4] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*, pages 217–232, 2006.

[5] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In *Computational Science and Its Applications - ICCSA 2008, International Conference, Perugia, Italy, June 30 - July 3, 2008, Proceedings, Part I*, pages 1249–1259, 2008.

[6] Sumeet Bajaj and Radu Sion. Trusteddb: a trusted hardware based database with privacy and data confidentiality. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 205–216, 2011.

[7] Lucas Ballard, Seny Kamara, and Fabian Monrose. Achieving efficient conjunctive keyword searches over encrypted data. In *Information and Communications Security, 7th International Conference, ICICS 2005, Beijing, China, December 10-13, 2005, Proceedings*, pages 414–426, 2005.

[8] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 535–552, 2007.

[9] Ian F. Blake and Vladimir Kolesnikov. One-round secure comparison of integers. *J. Mathematical Cryptology*, 3(1), 2009.

[10] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[11] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 224–241, 2009.

[12] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 578–595, 2011.

[13] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 506–522, 2004.

[14] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[15] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J. Wu. Private database queries using somewhat homomorphic encryption. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pages 102–118, 2013.

[16] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 416–432, 2003.

[17] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 325–341, 2005.

[18] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 535–554, 2007.

[19] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 505–524, 2011.

[20] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, 1986.

[21] Richard Brinkman, Ling Feng, Jeroen Doumen, Pieter H. Hartel, and Willem Jonker. Efficient tree search in encrypted data. *Information Systems Security*, 13(3):14–21, 2004.

[22] Jin Wook Byun, Dong Hoon Lee, and Jongin Lim. Efficient conjunctive keyword search on encrypted data storage system. In *Public Key Infrastructure, Third European PKI Workshop: Theory and Practice, EuroPKI 2006, Turin, Italy, June 19-20, 2006, Proceedings*, pages 184–196, 2006.

[23] Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Secure Data Management, Third VLDB Workshop, SDM 2006, Seoul, Korea, September 10-11, 2006, Proceedings*, pages 75–83, 2006.

[24] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, pages 268–289, 2002.

[25] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145, 2001.

[26] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 353–373, 2013.

[27] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 442–455, 2005.

[28] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[29] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 487–504, 2011.

[30] Giovanni Di Crescenzo. Private selective payment protocols. In *Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings*, pages 72–89, 2000.

[31] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In *Cryptology and Network Security,*

11th International Conference, CANS 2012, Darmstadt, Germany, December 12-14, 2012. Proceedings, pages 218–231, 2012.

[32] Emiliano De Cristofaro, Yanbin Lu, and Gene Tsudik. Efficient techniques for privacy-preserving sharing of sensitive information. In *Trust and Trustworthy Computing - 4th International Conference, TRUST 2011, Pittsburgh, PA, USA, June 22-24, 2011. Proceedings*, pages 239–253, 2011.

[33] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers*, pages 143–159, 2010.

[34] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 79–88, 2006.

[35] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, Cheju Island, Korea*, 2001.

[36] Hoang Giang Do and Wee Keong Ng. Privacy-preserving approach for sharing and processing intrusion alert data. In *Tenth IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2015, Singapore, April 7-9, 2015*, pages 1–6, 2015.

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

[38] Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, pages 457–472, 2001.

[39] Thomas Fuhr and Pascal Paillier. Decryptable searchable encryption. In *Provable Security, First International Conference, ProvSec 2007, Wollongong, Australia, November 1-2, 2007, Proceedings*, pages 228–236, 2007.

[40] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 10–18, 1984.

[41] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[42] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[43] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 107–109, 2011.

[44] Craig Gentry, Shai Halevi, Charanjit S. Jutla, and Mariana Raykova. Private database access with he-over-oram architecture. In *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, pages 172–191, 2015.

[45] Eu-Jin Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.

[46] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 182–194, 1987.

[47] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[48] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[49] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377, 1982.

[50] Philippe Golle, Jessica Staddon, and Brent R. Waters. Secure conjunctive keyword search over encrypted data. In *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings*, pages 31–45, 2004.

[51] Michael T. Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 576–587, 2011.

[52] Michael T. Goodrich, Michael Mitzenmacher, Olga Ohrimenko, and Roberto Tamassia. Privacy-preserving group data access via stateless oblivious RAM simulation. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 157–167, 2012.

[53] Fuchun Guo, Yi Mu, and Willy Susilo. Subset membership encryption and its applications to oblivious transfer. *IEEE Trans. Information Forensics and Security*, 9(7), 2014.

[54] Hakan Hacigümüs, Balakrishna R. Iyer, Chen Li, and Sharad Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*, pages 216–227, 2002.

[55] Shai Halevi and Victor Shoup. Bootstrapping for helib. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 641–670, 2015.

[56] Dennis Hofheinz and Enav Weinreb. Searchable encryption with decryption in the standard model. *IACR Cryptology ePrint Archive*, 2008:423, 2008.

[57] Bijit Hore, Sharad Mehrotra, Mustafa Canim, and Murat Kantarcioglu. Secure multidimensional range queries over outsourced data. *VLDB J.*, 21(3):333–358, 2012.

[58] Bijit Hore, Sharad Mehrotra, and Gene Tsudik. A privacy-preserving index for range queries. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 720–731, 2004.

[59] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[60] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *J. Cryptology*, 26(2):191–224, 2013.

[61] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 241–257, 2005.

[62] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 25, 2006.

[63] Yehuda Lindell. *Composition of Secure Multi-Party Protocols, A Comprehensive Study*, volume 2815 of *Lecture Notes in Computer Science*. Springer, 2003.

[64] Yanbin Lu. Privacy-preserving logarithmic-time search on encrypted data in cloud. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[65] Charalampos Mavroforakis, Nathan Chenette, Adam O'Neill, George Kollios, and Ran Canetti. Modular order-preserving encryption, revisited. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 763–777, 2015.

[66] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 308–318, 1998.

[67] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238, 1999.

[68] Dong Jin Park, Kihyun Kim, and Pil Joong Lee. Public key encryption with conjunctive field keyword search. In *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*, pages 73–86, 2004.

[69] J. M. Pollard. Theorems on factorization and primality testing. *Mathematical Proceedings of the Cambridge Philosophical Society*, 76(3):521528, 1974.

[70] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011, SOSP 2011, Cascais, Portugal, October 23-26, 2011*, pages 85–100, 2011.

[71] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: processing queries on an encrypted database. *Commun. ACM*, 55(9):103–111, 2012.

[72] Raluca A. Popa and Nickolai Zeldovich. Multi-key searchable encryption. *IACR Cryptology ePrint Archive*, 2013:508, 2013.

[73] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005, 2005.

[74] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[75] Eun-Kyung Ryu and Tsuyoshi Takagi. Efficient conjunctive keyword-searchable encryption. In *21st International Conference on Advanced Information Networking and Applications (AINA 2007), Workshops Proceedings, Volume 1, May 21-23, 2007, Niagara Falls, Canada*, pages 409–414, 2007.

[76] Bharath Kumar Samanthula, Wei Jiang, and Elisa Bertino. Privacy-preserving complex query evaluation over semantically secure encrypted data. In *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I*, pages 400–418, 2014.

[77] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 350–364, 2007.

[78] Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with o((logn)3) worst-case cost. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 197–214, 2011.

[79] Dawn Xiaodong Song, David A. Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*, pages 44–55, 2000.

[80] Emil Stefanov, Elaine Shi, and Dawn Xiaodong Song. Towards practical oblivious RAM. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[81] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 299–310, 2013.

[82] Boyang Wang, Yantian Hou, Ming Li, Haitao Wang, and Hui Li. Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index. In *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 111–122, 2014.

[83] Peter Williams and Radu Sion. Usable PIR. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*, 2008.

[84] Qianhong Wu, Jianhong Zhang, and Yumin Wang. Practical t-out-n oblivious transfer and its applications. In *Information and Communications Security, 5th International Conference, ICICS 2003, Huhehaote, China, Proceedings*, 2003.

[85] Guomin Yang, Chik How Tan, Qiong Huang, and Duncan S. Wong. Probabilistic public key encryption with equality test. In *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, pages 119–131, 2010.

[86] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164, 1982.

[87] Wei-Chuen Yau, Swee-Huay Heng, and Bok-Min Goi. Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In *Autonomic*

and Trusted Computing, 5th International Conference, ATC 2008, Oslo, Norway, June 23-25, 2008, Proceedings, pages 100–105, 2008.

[88] Rui Zhang, Jing Shi, and Yanchao Zhang. Secure multidimensional range queries in sensor networks. In *Proceedings of the 10th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2009, New Orleans, LA, USA, May 18-21, 2009*, pages 197–206, 2009.

Author's Publication

[1] **Hoang Giang, Do**, Wee Keong Ng, "Private Personal Information Verification" *Journal of Information Security*, Vol.8, No.3, 2017.

[2] **Hoang Giang, Do**, Wee Keong Ng, "Multi-dimensional Range Query on Outsourced Database with Strong Privacy Guarantee" *International Journal of Computer Network and Information Security*, Vol.9, No.10, 2017.

[3] **Hoang Giang, Do**, Wee Keong Ng, "Blockchain-Based System for Secure Data Storage with Private Keyword Search" *2017 IEEE World Congress on Services, SERVICES 2017*, Honolulu, USA, June 25-30, 2017.

[4] **Hoang Giang, Do**, Wee Keong Ng, "Private Boolean Query Processing on Encrypted Data" *Information and Communications Security - 18th International Conference, ICICS 2016*, Singapore, November 29 - December 2, 2016.

[5] **Hoang Giang, Do**, Wee Keong Ng, "Multidimensional range query on outsourced database with strong privacy guarantee" *14th Annual Conference on Privacy, Security and Trust, PST 2016*, Auckland, New Zealand, December 12-14, 2016.

[6] **Hoang Giang, Do**, Wee Keong Ng, "Privacy-Preserving Triangle Counting in Distributed Graphs" *30th IEEE International Conference on Advanced Information Networking and Applications, AINA 2016*, Crans-Montana, Switzerland, March 23-25, 2016.

[7] **Hoang Giang, Do**, Wee Keong Ng, "Privacy-preserving approach for sharing and processing intrusion alert data" *Tenth IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2015*, Singapore, April 7-9, 2015.

[8] **Hoang Giang, Do**, Wee Keong Ng, "Secure Reachability Query on Private Shared Graphs" *Ninth IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2014*, Singapore, April 21-24, 2014.

[9] Kanishka Ariyapala, **Hoang Giang, Do**, Ngoc Anh Huynh, Wee Keong Ng, "A Host and Network Based Intrusion Detection for Android Smartphones" *30th International Conference on Advanced Information Networking and Applications Workshops, AINA 2016 Workshops*, Crans-Montana, Switzerland, March 23-25, 2016.

[10] Ngoc Anh Huynh, **Hoang Giang, Do**, Wee Keong Ng, "On periodic behavior of malware: experiments, opportunities and challenges" *11th International Conference on Malicious and Unwanted Software, MALWARE 2016*, Fajardo, PR, USA, October 18-21, 2016.

[11] Fang Liu, Wee Keong Ng, Wei Zhang, **Hoang Giang, Do**, Shuguo Han, "Encrypted Set Intersection Protocol for Outsourced Datasets" *2014 IEEE International Conference on Cloud Engineering*, Boston, MA, USA, March 11-14, 2014.