Search over Encrypted Data in Cloud Computing

Bing Wang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Application

April 28, 2016
Falls Church, Virginia

Keywords: Cloud Computing, Data Privacy, Searchable Encryption, Secure Pattern Matching, Secure Computation

Search over Encrypted Data in Cloud Computing

Bing Wang

ABSTRACT

Cloud computing which provides computation and storage resources in a pay-per-usage manner has emerged as the most popular computation model nowadays. Under the new paradigm, users are able to request computation resources dynamically in real-time to accommodate their workload requirements. The flexible resource allocation feature endows cloud computing services with the capability to offer affordable and efficient computation services. However, moving data and applications into the cloud exposes a privacy leakage risk of the user data. As the growing awareness of data privacy, more and more users begin to choose proactive protection for their data in the cloud through data encryption. One major problem of data encryption is that it hinders many necessary data utilization functions since most of the functions cannot be directly applied to the encrypted data. The problem could potentially jeopardize the popularity of the cloud computing, therefore, achieving efficient data utilization over encrypted data while preserving user data privacy is an important research problem in cloud computing.

The focus of this dissertation is to design secure and efficient schemes to address essential data utilization functions over encrypted data in cloud computing. To this end, we studied three problems in this research area. The first problem that is studied in this dissertation is fuzzy multi-keyword search over encrypted data. As fuzzy search is one of the most useful and essential data utilization functions in our daily life, we propose a novel design that incorporates Bloom filter and *locality-sensitive hashing* to fulfill the security and function requirements of the problem. Secondly, we propose a secure index which is based on the most popular index structure, i.e., the inverted index. Our innovative design provides privacy protection over the secure index, the user query as well as the search pattern and the search result. Also, users can verify the correctness of the search results to ensure the proper computation is performed by the cloud. Finally, we investigate an important privacy-sensitive data application in cloud computing, i.e., genetic testings over DNA sequences. To provide secure and efficient genetic testings in the cloud, we utilize *predicate encryption* and design a bilinear pairing based secure sequence matching scheme to achieve strong privacy guarantee while fulfilling the functionality requirement efficiently. In all of the three research thrusts, we present thorough theoretical security analysis and extensive simulation studies to evaluate the performance of the proposed schemes. The results demonstrate that the proposed schemes can effectively and efficiently address the challenging problems in practice.

This work was supported by the National Science Foundation under the grands CNS-1217889, CNS-1338102, CNS-1405747 and CNS-1446478.

To my beloved wife, Ran Wei, and my parents Taizong Wang and Suyun Yin.

Acknowledgments

It takes five years to finish this dissertation. Without the generous support and guidance from many people, this dissertation would not be possible. I would like to express my deepest appreciations to them.

First of all, I would like to give my sincere gratitude to my advisor Dr. Wenjing Lou, who guided me into the research area of cloud computing security. I am always inspired by her rigorous research attitude and professional dedication. During the past five years, she has helped me from an amateur in research to finish my Ph.D. dissertation finally. I cannot thank her more for the long-hour discussion to go through my ideas and research works, the meticulous proofreading for my papers, and the inspiration during my research process. Not only she mentored my research, but also, she offered me cares of my life and valuable suggestions for my personal growth. It is my great honor to study and work with such a mentor who has always shown respects for my interests, and helped me to make them possible. I am truly grateful for what she has done.

I am also extremely grateful to Dr. Thomas Hou, who gave me invaluable advice and support on my research and personal life. I am inspired and benefited from his determination, hard-working and detail-oriented research attitude.

I am also thankful to Dr. Ing-Ray Chen, Dr. Danfeng Yao and Dr. David Evans for serving on my dissertation committee. Their insightful comments and suggestions have helped me to make this work better.

I would like to thank Dr. Shucheng Yu for his valuable suggestions and comments on my papers. As a collaborator, his research experience and wide knowledge are always invaluable for me and my research. I am also very grateful to Dr. Song Wei. During his visiting, I have the opportunity to work with him closely to discuss my research. My sincere appreciation also goes to Dr. Ning Cao, Dr. Qiben Yan, two formal labmates, who discussed with me my research and helped me to improve it.

I wish to thank my labmates in the Complex Networks and Security Research (CNSR) Lab at Northern Virginia Center: Changlai Du, Ethan Gaebel, Divya Ramanujachari, Dr. Wenhai Sun, Ning Zhang, Ruide Zhang, Yao Zheng. Working with these talented people is a real joy and a memorable experience. I benefited tremendously from the discussion and interaction with them. I am especially indebted to Dr. Wenhai Sun and Yao Zheng, who put significant efforts in improving and implementing our research ideas.

Most importantly, I am greatly indebted to my father Taizong Wang, and my mother Suyun Yin. They always understand me and backed my every choice with their 100% support both emotionally and financially. They have sacrificed a lot to support me to pursuing my dream in the United States. I feel grateful for their love, unshakable faith in my ability, earnest instructions, and encouragement through my entire life. To make them happy and proud of me is my strongest motivation to getting better.

Finally, I am especially grateful to my wife, Ran Wei. I am truly thankful for what she has been doing for me since we have married seven years ago. She has sacrificed her career opportunity to come with me to the United States and wholeheartedly took care of me through the most difficult time. Her optimistic attitude towards our future and inspiring words to cheer me up are my power to get through tough days. Without her love, support and understanding, I could not finish what I have started.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Emergence of Cloud computing

The concept of cloud computing in computer science can be traced back to Strachy's concept of *time-sharing* [2] in high-performance computers. The idea of time-sharing is to share the CPU time among the users to better utilize the high-performance computer. *Cloud computing* extends the concept of time-sharing from a single computer to a cluster of computers. The shared resources are also extended from CPU time only to a variety of computation resources such as memory and storage thank to hardware virtualization technique. The cloud service providers treat the computation resources as utility and charge the users based on the amount of the computation resource usage. A general system structure of cloud computing is shown in Fig. 1.1. According to the different service types, cloud computing services can be categorized as infrastructure-as-a-service (IaaS), platform-as-a-service (Paas), and software-as-a-service (SaaS). IaaS such as the Amazon Web Services (AWS) gives its user the control of the virtualized infrastructure. PaaS such as the Google App Engine provides a well-constructed platform with a set of common APIs which users implement their services based on. SaaS such as Dropbox provides its users specific services through software. On the other hand, cloud services can be classified as public cloud, private cloud, or hybrid cloud regarding deployment. While public and private clouds are open to public and private groups, respectively, hybrid cloud is a combination of public and private infrastructures.

The new computation paradigm has the following superiorities comparing with the traditional one. First of all, cloud computing provides affordable yet powerful computation platforms. Small businesses and individuals who may not be able to purchase or rent expensive servers can now enjoy the computation service provided by the cloud service providers in a pay-per-usage manner. Secondly, the elasticity and scalability of cloud computing significantly improve the usage efficiency of the computation resource, and thus, enable a more cost-effective computation service. When faced with the higher computation demands than

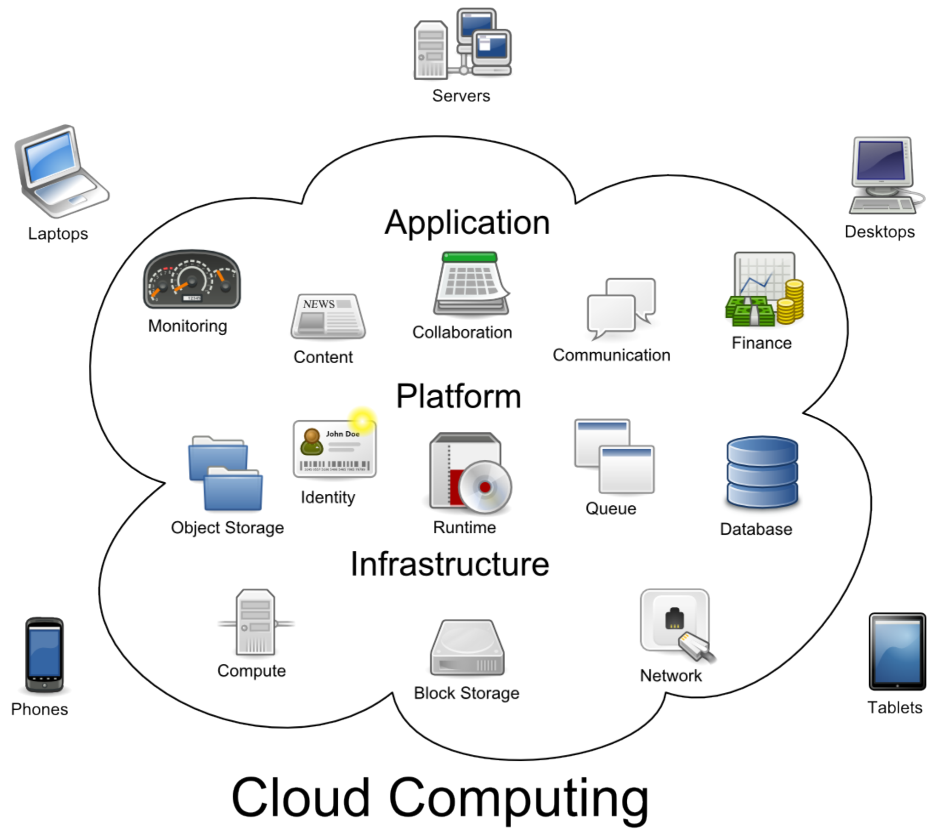


Figure 1.1: Cloud Computing: a general system structure [1]

usual, users can request temporal computation resources in real time for the bursting workload. Finally, cloud computing relieves the taunting IT management burden from users. Managing an IT system can be error-prone, time-consuming as well as costly especially for small business owners. With the unified system management from the cloud, users worry less about the reliability of the underlying computation platform of their services and will focus more on their computation tasks. Combining these advantages, cloud computing has become one of the most popular computation solutions for both companies and individuals. As a result, more and more individuals and companies begin to embrace the new paradigm and move their services along with the data into the cloud.

As the computation moves into the cloud, the data that is used during the computation is moved into the cloud storage as well. Cloud storage is one of the most popular cloud services because storage is an indispensable component of a computing system. There are enterprise-level cloud storage services such as the Amazon Simple Storage Service (Amazon S3) and Microsoft Azure storage, and consumer-level cloud storage services such as Dropbox and Google Drive. The primary purpose of the enterprise-level cloud storage services is to provide a virtual storage for the cloud computing applications, which usually have well-defined application programming interfaces (APIs) to interact with other cloud computing services. Since network bandwidth is a more valuable resource, it is natural to use cloud storage to avoid transmitting data through the Internet. In fact, most cloud computing providers also provide cloud storage which is usually on the same premise with the physical servers. On the other hand, consumer-level cloud storage services focus on hosting user files. Users are allowed to visit their files from anywhere as long as the Internet is available. Advanced functions such as auto-synchronizing among all registered devices are also available. Due to its convenience, cloud-based file hosting services become more and more popular, and people begin to utilize the cloud storage as their primary way to store and share data. However, as users outsource their data into the cloud, the control of the data is also handed over, which raises data privacy concerns especially when the data contains sensitive private information.

1.2 Security and privacy challenges in cloud data applications

In this section, we unfold the data privacy challenges in cloud computing applications. The data used in cloud applications is directly exposed to the cloud service provider and could also be learned by adversaries because of the potential compromise of the cloud. A straightforward solution is to encrypt the data before outsourcing it to the cloud. However, the cloud is not able to perform the original computation tasks over the data that is encrypted using the traditional data encryption schemes such as DES [3] or AES [4]. The security challenge comes from the urgent call of secure data encryption schemes that enable various data utilization functions over the encrypted data. Although this dissertation is not able to address the data privacy in all cloud computing applications, we focus our attention on a fundamental yet important data utilization function, i.e., data search function.

Keyword search is one of the most commonly used data utilization functions in our daily life. It has wide applications in information retrieval, data mining, and machine learning. As the data volume of the Internet is still growing, keyword search has become one of the most fundamental yet important functions. The keyword search problem has been well studied in plaintext scenario for decades. The common approach is to build a keyword index from a collection of documents, and the query is performed against the index. According to the number of the keywords, queries can be categorized as single keyword queries and multi-keyword queries. The later has become the dominant query type in today's Internet search engine. Multi-keyword queries can be further classified into conjunctive and disjunctive queries. While the search result must contain all the query keywords for a conjunctive query, a disjunctive query requires at least one of the query keywords exists in the search result. Other important search functions include search result ranking and approximate keyword search also known as the fuzzy keyword search. Search result ranking is achieved by giving scores to matching documents based on certain measurements. Fuzzy search is to measure the distance between the query keywords and the keywords in the document given a similarity measurement. If the distance is within a threshold, the document is included in the search result. Although keyword search that supports all the aforementioned functions have been well studied in plaintext scenario, searching over encrypted data is a challenging problem. The challenge comes from two aspects, i.e., the security aspect and the functionality aspect. Searchable encryption is a technique that is meant to address the aforementioned problem. A common approach is to build a secure index, and the keyword search is performed over the secure index. To guarantee user data privacy, the secure index should not leak any information regarding the underlying document set. Additionally, queries also need to be encrypted to ensure query privacy. Because the search process is performed by the cloud, intermediate information, such as search pattern which is the collection of the submitted queries and access pattern which is the output of the search algorithm, is available to the cloud. As the cloud provider could perform analysis on this information to gain knowledge about the documents and the queries, this information should also be well protected. In the following sections, we list all the security challenges we are going to address in this dissertation.

1.2.1 Approximate keyword search over encrypted data

Approximate keyword search also known as fuzzy search has become one of the essential functions in today's search engines considering the amount of misspelled keywords people make every day. The search engine should return search results that correspond to a proper keyword when a query keyword is misspelled. The process is done by calculating the distance between the indexed keywords and the query word in real-time in plaintext domain. However, it becomes a technical challenge when the data is encrypted because the distance between the encrypted keyword is not preserved.

Research efforts have been made to address the privacy-preserving fuzzy search problem over encrypted data. The existing solutions avoid the distance computation problem by enumerating all possible misspells as keywords. The expansion of the dictionary will clearly increase the storage overhead and eventually slows down the search computation efficiency. Although fully homomorphic encryption is another solution to the problem, the computation is too expensive to scale for today's data volume. Therefore, we focus ourselves on the design of an efficient searchable encryption scheme that supports fuzzy search in Chapter 2.

1.2.2 Secure inverted index and search result verification

The inverted index is an index data structure that stores the keyword-document mapping information. Different from the forward index which stores lists of words per document, the inverted index consists of document lists for each keyword in the dictionary. Each document list contains the IDs of all the documents that have the keyword. Compared with sequential iteration through each document for each keyword using the forward index, the inverted index limits the search within a subset of the documents. The inverted index is considered as the central component of a typical search engine indexing algorithm. Most of the well-known search engines libraries such as Apache Lucene [5] and Sphinx [6] are based on the inverted index. Therefore, the inverted index is an indispensable index structure to power up today's Internet.

To provide privacy-preserving keyword search over encrypted data, we can either build the secure index from scratch or secure an existing index structure such as the inverted index. Considering the importance and the popularity of the inverted index, converting the inverted index to support privacy-preserving keyword search will be not only important but also convenient for users to move their data into the cloud. Therefore, we focus ourselves on designing an efficient and secure inverted index in Chapter 3.

1.2.3 Secure genetic sequence matching

Genetic sequences contain important personal information that can be used to identify genetic diseases. The human genome consists of the long sequences of the DNA pairs. As recent advances in genetics have revealed more and more genetic mutations that associate with certain diseases, a new health care model known as *personalized medicine* [7] has emerged. Instead of the "one-size-fit-all" approach of the traditional health care, personalized medicine advances by being able to provide an efficient and effective medical treatment utilizing the patient's DNA information. Under the new health care paradigm, doctors perform genetic disease diagnoses and prediction by matching the known disease-related genetic patterns against the patient's DNA sequence. For instance, certain mutations in the genes *BRCA1* and *BRCA2* are related to high risk of breast cancer [8]. By testing the existence of those gene mutations in a patient's DNA, early intervention measures can be taken to prevent breast cancer from developing. Therefore, personalized medicine has been gaining great popularity and is considered as the future health care model.

Recently, the cost of the DNA sequencing decreases from ten million dollars per genome a decade ago to a few thousand dollars [9]. The personal genomics company, 23andMe, offers DNA ancestry service for only $99 [10]. The affordable DNA sequence services will make the DNA data of individuals available for various genetic applications including personalized medicine. Considering the enormous volume of the storage to keep everyone's DNA data, cloud computing is the perfect computing model to implement genetic applications. However, implementing cloud-based genetic application raises data privacy challenge in practice because of the extremely high sensitivity of the genetic data. Existing privacy protection of genetic data mainly relies on the regulations such as Health Insurance Portability and Accountability Act (HIPAA), which is unable to prevent privacy leakage caused by system failure or malicious behaviors. De-identification is another widely adopted technique to protect user privacy in genetic data application. Unfortunately, the technique is not effective as the user's identity could be recovered through analyzing the genetic data along with the public available records [11,12]. Therefore, we study the secure sequence matching problem for genetic applications in cloud computing in Chapter 4.

1.3 Research contributions

This dissertation research uncovers emerging data privacy issues in fundamental cloud computing applications, including privacy-preserving keyword search over encrypted data and secure genetic testing for personalized medicine. We make the following contribution from this research.

1. **Multi-keyword fuzzy search over encrypted data.** We design a novel secure index structure combining Bloom filter and Locality-Sensitive Hashing. Our secure index tolerates the misspelled query keywords that are within the pre-defined distance threshold and correctly match the query with the index. Compared with the existing solutions [13–15] which expand a pre-defined dictionary to include all possible misspellings within a certain edit distance, our scheme converts a keyword into a vector and measures the distance in the vector space. Therefore, our scheme does not need to expand the keyword space and can be easily scaled when the volume of the documents increases. We theoretically analyze the trade-off between the search efficiency and the result accuracy for different choices of the parameters of our scheme. The simulation results demonstrate that our scheme is accurate and efficient. This part of the dissertation is published in part in [16].

2. **Inverted index based dynamic searchable encryption.** We design a cryptographic provable secure inverted index, which supports various functions including generic multi-keyword query, document updates, and search result verification. Compared with the existing works [17–19], our scheme provides stronger privacy guarantee as it protects search pattern, access pattern as well as forward privacy and backward privacy for index update. In addition, we embed verification tokens into the index structure so that the user is able to verify the correctness of the search result. We provide a complete security proof under the random oracle model. Extensive analysis and a series of experimental results are presented to show the efficiency of the proposed scheme. This part of the dissertation is published in part in [20].

3. **Secure pattern matching for genetic testings.** We propose a novel privacy-preserving genetic testing scheme for personalized medicine in cloud computing. To achieve practical and secure computation over the encrypted DNA data, we leverage *predicate encryption* techniques [21, 22] to encrypt the DNA data. Different from previous works in secure DNA computation, we reduce the communication overhead from a linear complexity respect with the length of the inputs to $O(1)$ round, which

greatly improves the scheme usability. Furthermore, our scheme support wild-card based pattern matching. The testing result and the search pattern are both protected to guarantee a strong data privacy. We have proved that our scheme is secure under the well-defined cryptographic assumption. Extensive analytical and experimental results are presented to show the security and efficiency of our proposed scheme.

1.4 Organization

This dissertation is organized as follow. Chapter 2 presents our secure index design to support multi-keyword fuzzy query search. We describe our secure inverted index design and the performance evaluation in Chapter 3. Chapter 4 introduces our secure pattern matching algorithm for genetic testing applications in cloud computing. Finally, we summarize our research achievements, conclude the dissertation and point out several future research directions in Chapter 5.

Chapter 2

Privacy-Preserving Multi-Keyword Fuzzy Search over Encrypted Data in the Cloud

Enabling keyword search directly over encrypted data is a desirable technique for effective utilization of the encrypted data which is outsourced to the cloud. Since Song *et al.*'s seminal work on searchable encryption [23], much effort has been made to design effective and efficient mechanisms to enable search over encrypted data [14, 15, 24–36]. Instead of a word-by-word linear scan in the full text search [34], early works [32,33,37] built various types of secure index and corresponding index-based keyword matching algorithms to improve search efficiency. However, all these works only support the search of a single keyword. Subsequent works [24, 27–29, 38] extended the search capability to multiple keyword search, either conjunctive or disjunctive. However, they support only exact keyword matching. Misspelled keywords in the query will result in a wrong matching or no matching at all. Very recently, a few works [14,15,35] extended the search capability to approximate keyword matching (also known as *fuzzy search*). These are all for single keyword search, with a common approach involving expanding the index file by covering possible combinations of keyword misspelling so that a certain degree of the spelling error, measured by edit distance, can be tolerated. Although a wild-card approach is adopted to minimize the expansion of the resulting index file, for a l-letter long keyword to tolerate an error up to an edit distance of d, the index has to be expanded by $O(l^d)$ times. Thus, it is not scalable as the storage complexity increases exponentially with the increase of the error tolerance. To support multi-keyword search, the search algorithm will have to run multiple rounds. Very recently, Boldreva and Chenette proposed a fuzzy-searchable encryption scheme which represents the similar keywords within a distance with a single tag to improve the storage efficiency of [35]. However, their scheme still lacks a generic multi-keyword query support.

To date, multi-keyword fuzzy search over encrypted data remains a challenging problem. We want to point out that the efforts on search over encrypted data involve not only information retrieval techniques such as advanced data structures used to represent the searchable index, and efficient search algorithms that run over the corresponding data structure, but also the proper design of cryptographic protocols to ensure the security and privacy of the overall system. Although multi-keyword search and fuzzy search have been implemented separately, a combination of the two does not lead to a secure and efficient multi-keyword fuzzy search scheme. In this chapter, a brand new idea for achieving multi-keyword (conjunctive keywords) fuzzy search is proposed. Different from existing multi-keyword search schemes, the proposed scheme eliminates the requirement of a predefined keyword dictionary. The fuzziness of the keyword is captured by an innovative data structure and algorithmic design without expanding the keyword index and hence exhibits a high efficiency regarding computation and storage. This is achieved by several novel designs based on locality-sensitive hashing (LSH) [39] and the Bloom filter [40]. We convert each keyword to its bigram vector representation and utilize Euclidean distance to capture keywords similarity. By constructing file indexes using LSH in the Bloom filter, the proposed scheme finds documents with matching keywords efficiently. For ease of presentation, we first present a basic scheme that fulfills the functionality of multi-keyword fuzzy search but has some security vulnerabilities. Based on the basic scheme, we design our enhanced solution by incorporating another layer of security protection. Extensive analysis and simulation show that our scheme is secure.

2.1 Related works

2.1.1 Searchable encryption scheme without fuzzy search support

Single keyword searchable encryption

Song *et al.* [34] studied this problem first under the symmetric key setting for email systems. Their scheme did not contain an index. Thus, the search operation went through the entire file. Goh proposed a secure index using the Bloom filter in [41]. Curtmola *et al.* gave the formal definition of the searchable encryption and proposed an index structure based on the inverted list in [32]. In [30], Wang *et al.* solved the result ranking problem utilizing the keyword frequency and order-preserving encryption. Boneh *et al.* [37] proposed the first searchable encryption scheme using the asymmetric encryption scheme. All of these works only supported the single keyword search over the encrypted data.

Multiple keywords searchable encryption

To enrich the search functionality, the schemes supporting conjunctive keywords search have been proposed [21, 22, 27–29, 38, 42]. Many works which supported the conjunctive keyword search, subset search, range queries were using the asymmetric encryption [27–29]. [21,22,42] used the predicate encryption to achieve the conjunctive keywords search over encrypted data. In [43], a logarithmic-time search scheme was presented to support the range queries. Cao *et al.* [38] proposed a privacy-preserving multi-keyword ranked search scheme using symmetric encryption. Sun *et al.* [44] proposed an efficient privacy-preserving multi-keyword supporting cosine similarity measurement. However, none of the schemes can support fuzzy keyword search.

2.1.2 Searchable encryption scheme support fuzzy search

Li *et al.* proposed a wildcard-based fuzzy search scheme over encrypted data in [35]. Then Liu *et al.* [14] improved the scheme by reducing the index size. In [45], the LSH functions are used to generate file index. But it took two rounds of communication to achieve results ranking and only supported the single keyword search. All the aforementioned schemes only support the single keyword search, the fuzzy match OR the exact match. In [15], Chuah *et al.* improved [35] by introducing a tree structure index and enriched the search functionality by treating the pre-defined phrases, for example, “cloud computing", as a single keyword.

Boldyreva and Chenette generalized Li *et al.*'s scheme [35] to a primitive called efficiently fuzzy-searchable encryption (EFSE) for fuzzy search on encrypted data in [36]. The primitive works on a general closeness functions which translate data to a collection of tags. The tags of data represent the closeness which is the supported fuzziness by design. The authors also suggest using LSH functions as tag-encoding functions. Although this is a concurrent work, our scheme has two major contributions compared to theirs. First of all, our approach which utilizes Bloom filter is different from their tag-based scheme. As a result, we eliminate the per-defined dictionary which is necessary for their scheme. Secondly, our scheme supports conjunctive keyword search. Although the EFSE can be extended to support connective search in multiple attributes such as in a database, it is in fact achieved by combining multiple single attribute search results. Our scheme achieves the conjunctive keyword search through our innovative design and is computationally independent with the number of keywords in the query.

2.2 Problem formulation

The privacy preserving problem of multiple keywords fuzzy search over encrypted data is formulated in this section. We denote a keyword collection of a file as an *index* and an encrypted index as a *secure index.* Similarly, a *query* is a keyword collection and a *trapdoor* is the encryption of a query.

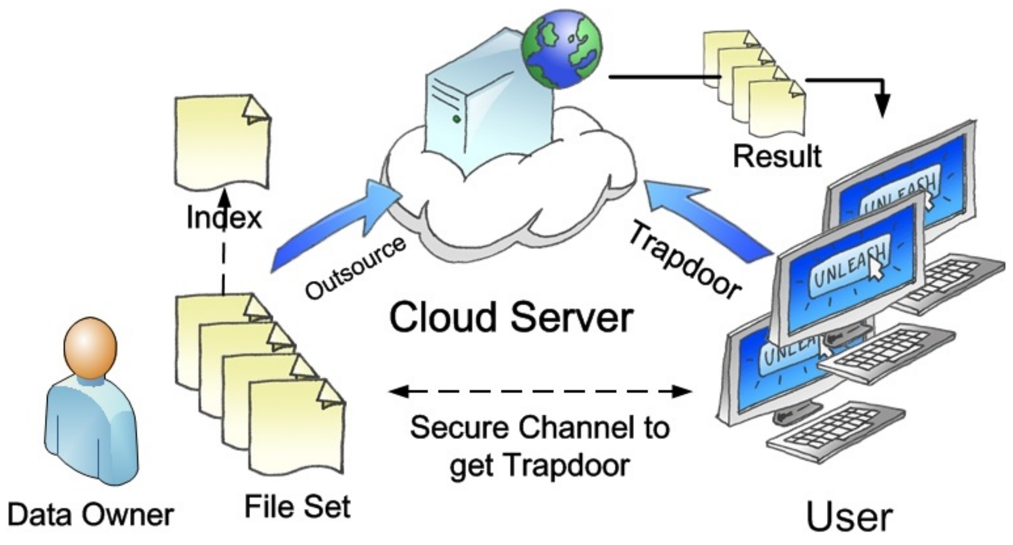


Figure 2.1: System model for search over encrypted data problem in cloud computing

2.2.1 System model

Fig. 2.1 shows the overall system architecture. To outsource a file collection to the cloud, the *data owner* first builds a secure index then uploads the encrypted files together with the secure index to the *cloud server*. To perform a query over the encrypted files, an *authorized user* first obtains the trapdoor, i.e., the encrypted query keyword(s), from the data owner, then submits the trapdoor to the cloud server. Upon receiving the trapdoor, the cloud server executes the search algorithm over the secure index and returns the matched files to the user as the search result.

Users are assumed as trusted entities which have pre-existing mutual trust with the data owner. The assumption is based on the fact that the access to the file collection is restricted to authorized users only in the practical application of searchable encryption. Open access to the file collection for all users including the cloud server will have no incentive for the data owner to adopt searchable encryption to protect the content of the files because the cloud server can learn the file content as search results through querying the file collection. Therefore, it is reasonable and practical to assume trusted users and rule out collusion between the cloud server and the authorized users in our threat model. The trapdoors can be transferred through regular authentication and secure channel establishment protocols based on the prior security context shared between each user and the data owner. The encryption of the individual files and the distribution of the decryption keys to authorized users are separate issues which have been discussed in other publications [46, 47].

2.2.2 Security model

Our scheme aims to protect information privacy of the outsourced data, which includes: 1) content privacy; 2) index privacy; and 3) query privacy. Because protecting content privacy can be achieved by encryption-before-outsourcing schemes [46, 47], we focus on preserving index privacy and query privacy.

- **Index privacy:** The secure index of a file should not 1) leak the indexed keywords of the file; 2) be distinguishable from other secure indexes of different files.
- **Query privacy:** The trapdoor of a query should not 1) leak the query keywords; 2) be linked to trapdoors of previous queries including the identical ones.

We exclude the security requirement of the access pattern in our discussion although we are aware the privacy leakage caused by the access pattern [48]. Research works such as [49, 50] have been proposed to address the privacy leakage issue of the access pattern. However, the gain of the privacy protection will cost either computation or communication. Our scheme can be modified easily to adopt these techniques to protect the access pattern with additional cost. Therefore, our scheme will focus on protecting index privacy and query privacy. We will briefly discuss the modification of the scheme to hide the access pattern as well.

We adopt the “honest-but-curious" model for the cloud server as in [14, 15, 35]. It assumes that the cloud server would honestly follow the designated protocols and procedures to fulfill its service provider's role, while it may analyze the information stored and generated on the server to learn additional information about the file collection. Regarding level of privacy protection, we consider three attack models which are also used in other related works [30, 31, 33, 51].

- **Ciphertext-only attack (COA) model.** In this model, the cloud server can only access the encrypted files, the secure indexes, and the submitted trapdoors. Although the encrypted files and the secure indexes are always available to the cloud server, the cloud server can only access to a trapdoor when a user submits it.
- **Known-keyword attack (KKA) model.** In this model, the cloud server has a limited number of the samples of the keyword-trapdoor pairs. Because the trapdoor can be used to retrieve files that contain the keyword, the cloud server will learn the existence of the keyword in these files. Some researchers refer this weaker model as known-background attack model [38, 44].
- **Chosen-keyword attack (CKA) model.** In this model, an adaptive adversary has the access to the trapdoor generation algorithm to generate trapdoors for any selected keywords. The objective of the adversary is to fully recover the secure indexes.

The COA model is the least powerful attack model, and it is suitable when the file set has an evenly distributed keyword set. If a document set focuses on a specific topic such as a conference proceeding, the background information of the file set is predictable by the cloud server. In practice, the KKA model is the most realistic model. The adversary has the most attacking power in the CKA model because the adversary controls the access to the trapdoor generation algorithm. Therefore, it is the hardest attack model to defend against.

2.2.3 Design objective

Our design bears the following security and performance goals.

- **Multi-keyword fuzzy search:** Our primary goal is to support multi-keyword fuzzy search. For example, “network security" related files should be found for a misspelled query “netword security".
- **Privacy guarantee:** Our scheme should protect the index privacy and the query privacy. It is worth mentioning that the search result, i.e., the access pattern, is known to the cloud server in most scenarios. However, in the CKA model, the adversary should not be given the search result. The adversary will be able to recover the index by enumerating all the possible keyword since he has the access to generate the proper trapdoors corresponding to the keywords.
- **Space efficiency and result accuracy:** We design our scheme using the Bloom filter which is famous for its computation and space efficiency. However, the Bloom filter is a probabilistic data structure. Therefore, we should ensure that the accuracy of the search result should be within an acceptable range.
- **No dictionary required:** The need of a pre-defined dictionary is a limiting factor that makes dynamic data operations, such as dataset/index update, very difficult. In our design, we would like to eliminate this requirement in contrast to many previous solutions [30–32, 35, 38, 44].

2.2.4 Preliminaries

Our scheme utilizes the Bloom filter and the locality-sensitive hashing (LSH). The brief introductions of them are given below.

Bloom filter

A Bloom filter is a bit array of m bits, all of which are set to 0 initially. A Bloom filter uses l independent hash functions from $\mathcal{H} = \{h_i | h_i : S \to [1, m], 1 \le i \le l\}$. To insert an element

$a \in \mathcal{S}$ into the Bloom filter, set the bits at the $h_i(a)$-th positions in the array to 1 for all hash functions. To test whether an element b is in $\mathcal{S}$, feed it to each of the l hash functions to get l array positions. If any of the l bits is 0, then $b \notin \mathcal{S}$; otherwise, either b belongs to $\mathcal{S}$ or b yields a false positive. The false positive rate of a m-bit Bloom filter is approximately $(1 - e^{-\frac{ln}{m}})^l$. The optimal false positive rate is $(1/2)^l$ when $k = \frac{m}{n} \cdot \ln 2$ [40].

Locality-Sensitive Hashing (LSH)

Given a distance metric d, e.g. Euclidean distance, a LSH function hashes close items to the same hash value with higher probability than the items that are far apart. A hash function family $\mathcal{H}$ is (r_1, r_2, p_1, p_2)-sensitive if any two points s, t and $h \in \mathcal{H}$ satisfy:

$$\text{if } d(s,t) \leq r_1 : Pr[h(s) = h(t)] \geq p_1 \tag{2.1}$$

$$\text{if } d(s,t) \geq r_2 : Pr[h(s) = h(t)] \leq p_2 \tag{2.2}$$

where $d(s,t)$ is the distance between the point s and the point t. We use the p-stable LSH family [52] in our scheme. A p-stable LSH function has the form $h_{a,b}(\mathbf{v}) = \lfloor \frac{\mathbf{a} \cdot \mathbf{v} + b}{w} \rfloor$ where $\mathbf{a}, \mathbf{v}$ are vectors and b, w are real numbers.

Definition 1. *A distribution $\mathcal{D}$ over $\mathbb{R}$ is called a p-stable distribution, if $\exists p \geq 0$ such that for n real numbers $v_1, v_2, \cdots, v_n$ and i.i.d. variables $X_1, X_2, \cdots, X_n$ with same distribution, the summation $\sum_i v_i X_i$ also follows the distribution $\mathcal{D}$ with the variable $(\sum_i |v_i|^p)^{1/p} X$, where X is a random variable with distribution $\mathcal{D}$.*

- when $p = 1$, it is **Cauchy distribution**, defined by the density function $f_p(x) = \frac{1}{\pi} \frac{1}{1+x^2}$, is 1-stable.
- when $p = 2$, it is **Gaussian distribution**, defined by the density function $f_p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$, is 2-stable.

The p-stable LSH function is given by $h_{a,b}(v) = \lfloor \frac{\mathbf{a} \cdot \mathbf{v} + b}{w} \rfloor$, where $\mathbf{a}$ is a d-dimensional vector, $b \in [0, w]$ is a real number and w is a fixed constant for one family.

The hash function $h_{\mathbf{a},b}(\mathbf{v}) : \mathbb{R}^d \rightarrow \mathbb{Z}$ maps a d-dimensional vector $\mathbf{v}$ onto the set of integers. By choosing different $\mathbf{a}, b$, different hash functions in the family can be generated. Given two vectors $\mathbf{v}_1, \mathbf{v}_2$, let $c = \parallel \mathbf{v}_1 - \mathbf{v}_2 \parallel_p$, it is easy to see

$$p(c) = Pr_{\mathbf{a},b}[h_{\mathbf{a},b}(\mathbf{v}_1) = h_{\mathbf{a},b}(\mathbf{v}_2)] = \int_0^w \frac{1}{c} f_p(\frac{t}{c})(1 - \frac{t}{w}) dt, \tag{2.3}$$

where f_p is the probability density function. Denote $\epsilon = r_2/r_1$, then p-stable LSH is showed $(r, \epsilon r, p_1, p_2)$-sensitive where $p_1 = p(r), p_2 = p(\epsilon r)$ in [52].

2.3 Multi-keyword fuzzy search scheme

2.3.1 Main idea

To design a secure and well-functioning search scheme over encrypted data, one has to make three important design choices that are closely inter-related and largely determine the performance of the resulting search scheme. They are 1) data structure used to build secure indexes and trapdoors; 2) effective search algorithm that can quantify the level of match between keywords in the query and keywords in the index with high efficiency; and 3) security and privacy mechanisms that can be integrated in the above two design choices thus the index privacy and search privacy can be protected.

In this subsection, we outline the key ideas behind our design for 1) and 2). We will present the key idea of the data structure and search algorithm in the plaintext format for ease of understanding. More detailed scheme design with integrated security and privacy mechanisms will be described in section 2.3.2.

Our scheme builds index on a per file basis, namely, $\mathcal{I}_\mathcal{D}$ for file $\mathcal{D}$. The index $\mathcal{I}_\mathcal{D}$ is a m-bit Bloom filter which contains all the keywords in $\mathcal{D}$. To support fuzzy and multiple keywords search, we first convert each keyword into a *bigram vector*. The Euclidean distance between bigram vectors is used to measure the closeness between keywords. Then we use LSH functions to replace standard hash functions to insert the keywords in the Bloom filter $\mathcal{I}_\mathcal{D}$. By choosing proper LSH function family, the keywords, of which the Euclidean distance of the bigram vectors is within a threshold, will match in the Bloom filter. The main steps are illustrated in Fig. 2.2 and explained as follows.

Bigram vector representation of keyword

One key step to building index is *keyword transformation* because the LSH function does not apply to string. We use the following method to transform a keyword to its vector representation. A keyword is first transformed to a *bigram set*. A bigram set of a keyword contains all the contiguous two letters appeared in the keyword. For example, the bigram set of the keyword "network" is {ne,et,tw,wo,or,rk}. We use a 26×26-bit long vector to represent a bigram set. The order of the bigrams in the vector is fixed for all keywords. Each element in the vector represents one of the 26^2 possible bigrams. The element is set to 1 if the corresponding bigram exists in the bigram set of a given keyword. This bigram vector based keyword representation is not sensitive to the position of the misspelling, nor is it sensitive to which letter it was misspelled to. Therefore, the distance between the vectors is based on the number of the different bigrams. The misspelled keywords "n$w$twork", "n$v$twork", or "netwo$y$k" will all be mapped to a vector with two-element difference from the bigram vector of the keyword "network". By this representation, a keyword can be misspelled in many different ways but still be represented in a vector that is very close to the correct one.

We measure closeness (distance) using Euclidean distance, the well-known distance metric for vector-type data items. The bigram vector representation is robust, inclusive, and the key to enabling the use of LSH functions.

Bloom filter representation of index/query

Bloom filter has been used to build per document index for single keyword exact search scenario before [33, 53]. However, these schemes use regular hash functions in Bloom filter so that they cannot support fuzzy keyword search. In our scheme, we adopt a special class of hash functions - locality sensitive hash - to build the index. Because LSH functions hash inputs between which the similarity is within a certain threshold into the same output with high probability, our index supports fuzzy keyword search. Fig. 2.2 explains the process using an example that a misspelled keyword "netword" in the user query is hashed into the same bucket as the correctly spelled keyword "network" under the LSH functions. The first step is to transform the keywords "network", "security" and "privacy" into vectors. Then two LSH functions h_1, h_2 are used from the same hash family to generate the index. When a query is submitted, i.e., "netword" and "security", the same LSH functions are used to generate the trapdoor. Because the word "network" has the same hash value of the misspelled work "netword" using LSH function $h_{\mathbf{a},b}$, the misspelled query matches exactly with the index containing the keywords "network" and "security" (encryption is not shown in the example, and we use '$\cdots$' to represent all the 0s). As a result, a match will be found during the search process. The use of the LSH functions in building the per-file Bloom filter based index is the key to implementing fuzzy search. Therefore, indexes and queries now are represented as vectors instead of words. However, to ease the presentation, we still use the terms *index, query.*

Inner product based matching algorithm

The last step of our scheme is to find all the matches and sort them based on the matching score. As shown in Fig. 2.2, the secure index of each file is a Bloom filter that contains all the keywords in the file. We generate the query vector in the same way as generating the index vector.

First of all, we represent each of the query keywords as a bigram vector. Then we insert the bigram vectors to a Bloom filter which has the same length as the index and uses the same LSH functions. Since the query vector and the index Bloom filter have the same length, we can computer the inner product of them. Note that we use the same LSH hash functions in both the index Bloom filter and the query vector. If every position of 1s in the query vector also matches an 1 in the index Bloom filter, the query matches with the index. As a result, the inner product result is the largest value that it can be.

Lemma 1. *The inner product between the index of a file $\mathcal{D}$ and the query vector of a query*

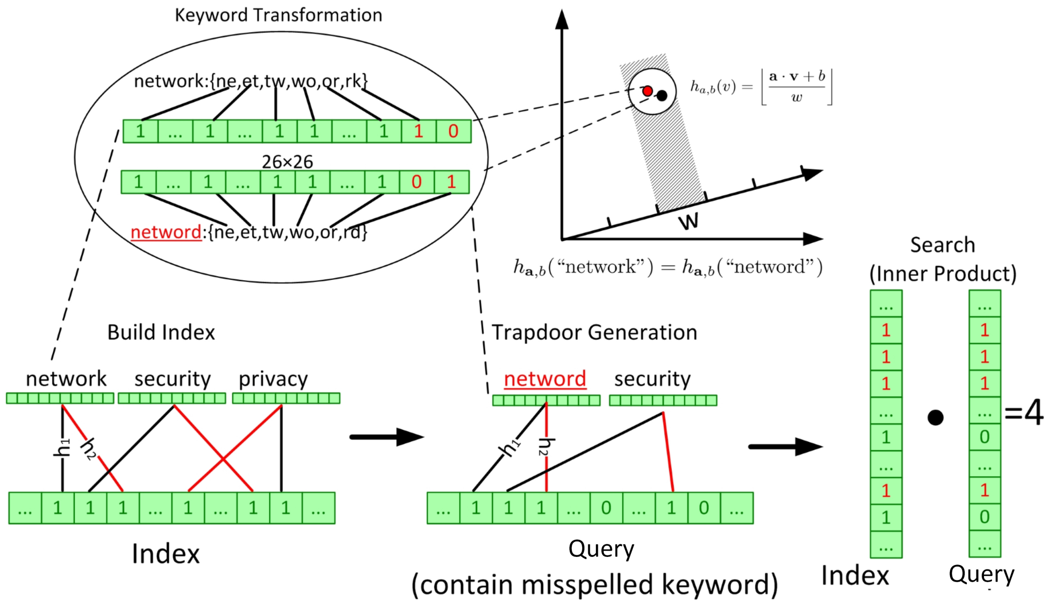

Keyword Transformation
network:{ne,et,tw,wo,or,rk}
26×26
netword:{ne,et,tw,wo,or,rd}
$h_{a,b}(v) = \left\lfloor \frac{\mathbf{a}\cdot\mathbf{v}+b}{w} \right\rfloor$
w
$h_{\mathbf{a},b}(\text{“network”}) = h_{\mathbf{a},b}(\text{“netword”})$
Build Index
network
security
privacy
h1
h2
Index
Trapdoor Generation
netword
security
Query
(contain misspelled keyword)
Search
(Inner Product)
=4
Index
Query

Figure 2.2: Simplified scheme illustration: fuzzy query support.

$\mathcal{Q}$ yields the largest value which is the number of 1s in the query vector if and only if the file $\mathcal{D}$ contains all the keywords in the query $\mathcal{Q}$. [1]

Proof. Recall that we use the same LSH functions to generate the query vector which has the same length as the index Bloom filter. If the file $\mathcal{D}$ contains all the keywords in the query $\mathcal{Q}$, every keyword in the query will result in a match when querying the index Bloom filter, which means the every corresponding position of the 1s in the query vector is 1 as well in the index Bloom filter. As a result, the inner product yields the maximum value which is the number of the 1s in the query vector.

If the inner product value between an index Bloom filter and a query vector is the number of the 1s in the query vector, then the every corresponding position of the 1s in the query vector is 1 as well in the index Bloom filter, which implies that all the keywords in the query are contained in the file. □

Based on this property, the search can be done through a simple inner product of the index vector and the query vector.

[1]For now, we do not differentiate the false positive from the true positive. We will discuss the result accuracy later.

2.3.2 Scheme construction

We present the detailed description of the proposed scheme in this section, with integrated security and privacy mechanisms. Our scheme is based on symmetric cryptography and consists of six polynomial-time algorithms:

- **KeyGen(m):** Given a security parameter m, output the secret key $SK(M_1, M_2, S)$, where $M_1, M_2 \in \mathbb{R}^{m\times m}$ are invertible matrices and $S \in \{0,1\}^m$ is a vector.
- **IndexEnc($SK, \mathcal{I}$:)** Generate two vectors $\{\mathcal{I}', \mathcal{I}''\}$ where $|\mathcal{I}'| = |\mathcal{I}''| = |\mathcal{I}|$. For each element $i_j \in \mathcal{I}, 1 \le j \le |\mathcal{I}|$, set $i'_j = i''_j = i_j$ where $i'_j \in \mathcal{I}', i''_j \in \mathcal{I}''$ if $s_j = 1, s_j \in S$; otherwise $i'_j = \frac{1}{2}i_j + r, i''_j = \frac{1}{2}i_j - r$ where r is a random number. Output $Enc_{SK}(\mathcal{I}) = \{M_1^T \cdot \mathcal{I}', M_2^T \cdot \mathcal{I}''\}$ as the secure index.
- **QueryEnc($SK, \mathcal{Q}$:)** Generate two vectors $\{\mathcal{Q}', \mathcal{Q}''\}$ where $|\mathcal{Q}'| = |\mathcal{Q}''| = |\mathcal{Q}|$. For each element $q_j \in \mathcal{Q}, 0 \le j \le |\mathcal{Q}|$, set $q'_j = q''_j = q_j$ where $q'_j \in \mathcal{Q}', q''_j \in \mathcal{Q}''$ if $s_j = 0, s_j \in S$; otherwise $q'_j = \frac{1}{2}q_j + r', q''_j = \frac{1}{2}q_j - r'$ where r' is another random number. Output $Enc_{SK}(\mathcal{Q}) = \{M_1^{-1} \cdot \mathcal{Q}', M_2^{-1} \cdot \mathcal{Q}''\}$ as the trapdoor.
- **BuildIndex($\mathcal{D}, SK, l$):** Choose l independent LSH functions from the p-stable LSH family $\mathcal{H} = \{h : \{0,1\}^{26\times 26} \to \{0,1\}^m\}$. Construct a m-bit Bloom filter $\mathcal{I}_\mathcal{D}$ as the index for each file $\mathcal{D}$.
  1. Extract the keywords set $\mathcal{W}_\mathcal{D} = \{w_1, w_2, \cdots\}, w_i \in \{0,1\}^{26\times 26}$ from $\mathcal{D}$.
  2. For each keyword w_i, insert it into the index $\mathcal{I}_\mathcal{D}$ using $h_j \in \mathcal{H}, 1 \le j \le l$.
  3. Encrypt the index $\mathcal{I}_\mathcal{D}$ using IndexEnc($SK, \mathcal{I}_\mathcal{D}$) and output $Enc_{SK}(\mathcal{I}_\mathcal{D})$.
- **Trapdoor($\mathcal{Q}, SK$):** Generate a m-bit long Bloom filter for the query $\mathcal{Q}$. For each search keyword q_i, insert q_i using the same l LSH functions $h_j \in \mathcal{H}, 1 \le j \le l$ into the Bloom filter. Encrypt $\mathcal{Q}$ using QueryEnc($SK, \mathcal{Q}$), and output the $Enc_{SK}(\mathcal{Q})$.
- **Search($Enc_{SK}(\mathcal{Q}), Enc_{SK}(\mathcal{I}_\mathcal{D})$):** Output

$$M_1^T\mathcal{I}' \cdot M_1^{-1}\mathcal{Q}' + M_2^T\mathcal{I}'' \cdot M_2^{-1}\mathcal{Q}''$$

  as the search result for the query $\mathcal{Q}$ and the document $\mathcal{D}$, which can be shown as equivalent to compute

$$\mathcal{I}'^T \cdot \mathcal{Q}' + \mathcal{I}''^T \cdot \mathcal{Q}'' = \mathcal{I}^T \cdot \mathcal{Q}$$

2.3.3 Soundness of the scheme

Now we prove that the inner product of the encrypted index and the trapdoor is equivalent to the inner product of the original index and the query. *IndexEnc*($SK, \mathcal{I}$), *QueryEnc*($SK, \mathcal{Q}$)

are index and query encryption function respectively, which are tailored for our design from the secure kNN scheme [51].

Proof. Recall the **Search** operation, the cloud server computes the following:

$$\begin{aligned} & M_1^T\mathcal{I}' \cdot M_1^{-1}\mathcal{Q}' + M_2^T\mathcal{I}'' \cdot M_2^{-1}\mathcal{Q}'' && (2.4)\\ = \;& \mathcal{I}'^T(M_1 \cdot M_1^{-1})\mathcal{Q}' + \mathcal{I}''^T(M_2 \cdot M_2^{-1})\mathcal{Q}'' && (2.5)\\ = \;& \mathcal{I}'^T \cdot \mathcal{Q}' + \mathcal{I}''^T \cdot \mathcal{Q}'' && (2.6)\\ = \;& \textstyle\sum_{i'_j \in \mathcal{I}', q'_j \in \mathcal{Q}'} i'_j q'_j + \sum_{i''_j \in \mathcal{I}'', q''_j \in \mathcal{Q}''} i''_j q''_j && (2.7) \end{aligned}$$

Recall the split process in **IndexEnc** and **QueryEnc**, if $s_j = 1, s_j \in \mathcal{S}$

$$i'_j = i''_j = i_j, q'_j = \frac{1}{2}q_j + r, q''_j = \frac{1}{2}q_j - r;$$

otherwise,

$$q'_j = q''_j = q_j, i'_j = \frac{1}{2}i_j + r, i''_j = \frac{1}{2}i_j - r.$$

Therefore, the Eq. (2.7) can be rewritten as

$$\begin{aligned} & \sum_{s_j=1, j\in[1,m]} i_j(q'_j + q''_j) + \sum_{s_k=0, k\in[1,m]} q_k(i'_k + i''_k) \\ = & \sum_{j=1}^{m} i_j q_j = \mathcal{I}^T\mathcal{Q} \end{aligned}$$

□

The search result of our basic scheme reflects whether a query $\mathcal{Q}$ matches with a file $\mathcal{D}$.

1. Our basic scheme returns the correct results for the exact keyword search. If the query keywords $\mathcal{Q} \subset \mathcal{W}_{\mathcal{D}}$, the cloud server should include file $\mathcal{D}$ in the result set. Recall that we use the same l hash functions $h_j \in \mathcal{H}, 1 \leq j \leq l$ when building the index and the query. The positions which are set to 1 in the query $\mathcal{Q}$ are also set to 1 the index $\mathcal{I}_{\mathcal{D}}$, which implicates that the inner product reaches the maximum value that the query can produce. Therefore, the file $\mathcal{D}$ is included in the result set.

2. Our basic scheme returns the correct results with high probability for the fuzzy keyword search. Suppose the keyword $w \in \mathcal{Q}$ is slightly different from the keyword $w' \in \mathcal{W}_{\mathcal{D}}$, i.e., $d(w, w') \leq r_1$ where r_1 is the distance threshold defined in the LSH function. If $h_j(w) = h_j(w'), h_j \in \mathcal{H}, 1 \leq j \leq l$ for all the l LSH functions, the *Search* returns the maximum value as the exact keyword search.

If $d(w, w') \leq r_1$ but $h_j(w) \neq h_j(w')$, then we call it a *miss*, which lowers the inner product. The probability of k *misses* happen is $\binom{l}{k}(1-p_1)^k p_1^{l-k}$, where p_1 is the probability defined in the LSH function. If $d(w, w') > r_1$, the probability that the LSH functions hash them together is very low. Thus, our basic scheme returns the relatively high inner product with high probability.

Discussion: The inner product of the secure index $Enc_{SK}(\mathcal{I}_{\mathcal{D}})$ and the trapdoor $Enc_{SK}(\mathcal{Q})$ is the exact number of the matching bits in the Bloom filter, which shows whether the query keywords existed in the document. For security consideration, we follow several rules when choosing the secret key SK in *KeyGen* and random numbers during the encryption. First, the security parameter m must be greater than 1024 to prevent the cloud server from launching brute force attacks. Second, the number of 0s should be equal to the number of 1s in the split vector S to maximize the randomness introduced by S. Last, the random number r used during the split process should be picked uniformly from $\mathbb{R}$.

Dataset update: A particular advantage of our scheme over the previous multi-keyword search schemes [38, 44] is that our scheme can support dataset updates efficiently, due to the facts that our scheme does not require a pre-defined global dictionary and each document is individually indexed. Therefore, dataset updates, such as *file adding, file deleting* and *file modifying*, can be done efficiently, involving only the indexes of the files to be modified, without affecting any other files.

Hiding the access pattern: As we mentioned earlier, the access pattern, i.e., the inner product result, can be hidden from the cloud. One intuitive method is to encrypt the elements of the secure index and the trapdoor with a multiplicative homomorphic encryption such as the ElGamal encryption algorithm [54]. Then the inner product results must be first returned to the user for decryption, and the user can retrieve the documents through oblivious transfer protocols.

2.4 Security and privacy analysis

Since a block cipher such as AES [4] is used to protect the file content, there are only two ways for the cloud to learn the privacy of the indexes and the queries. The first one is through the secure index and the trapdoors. The other way is through the search process and the access pattern which is the search result, i.e., the inner product results, in our case. The access pattern in our scheme would leak the following information. First, it may leak the number of the keywords in a query. Because we assume the design of our scheme is publicly known except the secret key, the cloud could refer the number of the keywords in a query from the inner product results. Second, the cloud can link trapdoors of the same query by the search results. This is intuitive since the same query will always lead to the same search results. Finally, the search results also reveal the document set that shares the same query keywords. To conceal the leakage caused by the access pattern is another active research topic. It is worth mentioning that the existing access pattern protection schemes such as oblivious transfer protocols can be applied to our scheme directly. Therefore, we exclude the discuss of the privacy leakage caused by the access pattern in this section and focus on the privacy leakage caused by the secure index and the trapdoors. We consider the followings as the privacy.

1. The keywords in the index and the query,
2. index-linkability through the secure indexes, i.e., whether different indexes can be linked through shared keywords,
3. query-linkability through the trapdoors, i.e., whether a query can be linked with a previous one.

We adopt the security games proposed in [33] to define the security of the proposed scheme. The first game is used to define the index security. Intuitively, the security game tries to capture the notation that the adversary cannot deduce the content information from the secure index. When the adversary is given two keyword sets, i.e., V_0, V_1, the adversary's challenge is to determine which keyword set is encoded into the secure index. If the problem of distinguishing the index for V_0, V_1 is hard, then it is also hard to determine at least one keyword that is not in both V_0, V_1. Therefore, the secure index reveals no useful information to the adversary. We require the number of the keywords in V_0, V_1 to be equal, which is possible by adding dummy keywords.

Setup

The challenger $\mathcal{C}$ creates a set of keywords $\mathcal{W}$. Then $\mathcal{C}$ chooses a subset $\mathcal{W}^\star \subset \mathcal{W}$ and chooses a number of subsets from $\mathcal{W}^\star$ as the file collection. The collection of the subsets is denoted as $\mathcal{D}$. The challenger $\mathcal{C}$ first runs **KeyGen** to generate the secret key, then creates the index for each subset $d_i \in \mathcal{D}$ using **BuildIndex**. Finally, $\mathcal{C}$ publishes the keyword set $\mathcal{W}$, $\mathcal{W}^\star$, and the secure indexes $\mathcal{I}_D$ to $\mathcal{A}$.

Queries

$\mathcal{A}$ is allowed to request the trapdoor T_Q for a query $Q \subset \mathcal{W}$ from $\mathcal{C}$ at most n times.

Challenge

At some point, $\mathcal{A}$ chooses a non-empty keywords set $V_0 \subset \mathcal{W}^\star$ and another non-empty keywords set $V_1 \subset \mathcal{W}$ such that $|V_0 - V_1| \neq 0, |V_1 - V_0| \neq 0$, and $|V_0| = |V_1|$. The restriction is that $\mathcal{A}$ has not requested the trapdoor for queries which contain any keyword in $V_0 \triangle V_1 = (V_0 - V_1) \cup (V_1 - V_0)$.

$\mathcal{C}$ chooses $b \xleftarrow{R} \{0,1\}$ and generates the secure index for V_b

The challenge for $\mathcal{A}$ is to decide b. After the challenge is issued, $\mathcal{A}$ can continue request trapdoor if the query number limit n is not reached.

Response

$\mathcal{A}$ outputs a bit b' as his guess of b. The advantage of $\mathcal{A}$ in winning the game is defined as

$$\text{Adv}_{\mathcal{A}} = |\Pr[b = b'] - 1/2|.$$

The second game is used to define the trapdoor security. Because the index construction is very similar to the trapdoor construction in the proposed scheme, we adopt the similar security game to define the trapdoor security. We need to ensure that the trapdoor does not reveal useful information about the query keywords.

Setup

The challenger $\mathcal{C}$ creates a set of keywords $\mathcal{W}$ and chooses a number of subsets of $\mathcal{W}$ as the file collection. The collection of the subsets is denoted as $\mathcal{D}$. The challenger $\mathcal{C}$ first runs **KeyGen** to generate the secret key, then creates the index for each subset $d_i \in \mathcal{D}$ using **BuildIndex**. Finally, $\mathcal{C}$ publishes the keyword set $\mathcal{W}$ and the secure indexes $\mathcal{I}_D$ to $\mathcal{A}$.

Queries

$\mathcal{A}$ is allowed to request the trapdoor T_Q for a query $Q \subset \mathcal{W}$ from $\mathcal{C}$ at most n times.

Challenge

At some point, $\mathcal{A}$ chooses two queries Q_0, Q_1 on which it wishes to be challenged and sends them to $\mathcal{C}$. The restriction is that $\mathcal{A}$ has not requested the trapdoor for neither Q_0, Q_1.

$\mathcal{C}$ chooses $b \overset{R}{\leftarrow} \{0, 1\}$ and generates the trapdoor T_{Q_b}
The challenge for $\mathcal{A}$ is to decide b. After the challenge is issued, $\mathcal{A}$ can continue request trapdoor if the query number limit n is not reached.

Response

$\mathcal{A}$ outputs a bit b' as his guess of b. The advantage of $\mathcal{A}$ in winning the game is defined as

$$\text{Adv}_{\mathcal{A}} = |\Pr[b = b'] - 1/2|.$$

We set a limit on how many times the adversary can access to the trapdoor algorithm. If we set $n = 0$, the security game simulates the COA model. Similarly, if we set $n = N$ where N is a fixed integer, the security game simulates the KKA model, and if we set $n = MAX$ where MAX is the maximum number of possible queries, the security game simulates the CKA model. Therefore, we can use the above games to define the security of our scheme under all three attack models. Based on the security game, we present the security definition as follows.

Definition 2 ($\epsilon-$indistinguishability)**.** *A searchable encryption scheme is $\epsilon-$indistinguishable secure if no probabilistic polynomial-time (PPT) adversary can win the above games with an advantage greater than ϵ*

The security definition is based on the semantic security in cryptanalysis proposed by Goldwasser *et al.* [55]. Based on the above security definition, we have the following claim.

Theorem 1. *The secure index and the trapdoor of the proposed scheme satisfy the Def. 2 under the ciphertext-only attack model.*

Proof. Under the COA model, the trapdoor generation algorithm access limit n is set to 0 which means the adversary has no query-trapdoor pair available. Assume that the adversary can win the games using a polynomial-time algorithm $\mathcal{A}$ with $\text{Adv}_{\mathcal{A}}$. We show that we can build an algorithm $\mathcal{B}$ that use $\mathcal{A}$ to distinguish the ciphertext of the secure kNN encryption from a random function $f : \{0,1\}^{26\times 26} \to \{\mathcal{R}^m, \mathcal{R}^m\}$ with $\text{Adv}_{\mathcal{A}}$.

$\mathcal{B}$ has access to a random oracle $\mathcal{O}_f$ where $f : \{0,1\}^{26\times 26} \to \{\mathcal{R}^m, \mathcal{R}^m\}$ is the unknown function. The objective of $\mathcal{B}$ is to determine whether f is a random function or the secure kNN encryption function. $\mathcal{B}$ substitutes the **IndexEnc** and the **QueryEnc** in our protocol with the queries to the oracle and plays the security game using the algorithm $\mathcal{A}$. $\mathcal{B}$ indicates f is the random function if $\mathcal{A}$ outputs 0; otherwise, $\mathcal{B}$ guesses f is the secure kNN encryption algorithm. If f is a random function, then $\Pr[b' = 0] = 1/2$. If f is the secure kNN encryption algorithm, $\mathcal{A}$ has the advantage bigger than ϵ to win the game, i.e., $|\Pr[b' = 1] - 1/2| > \epsilon$. Therefore, $\mathcal{B}$ can determine f in $\mathcal{O}_f$ using $\mathcal{A}$.

However, according to [51], the secure kNN encryption algorithm uses two full rank matrices and random split vector to obfuscate the ciphertext. As shown in the original paper, the ciphertext is indistinguishable from the random vectors without the plaintext-ciphertext pairs. Therefore, our original assumption is incorrect, which means there exists no such an algorithm $\mathcal{A}$. □

While our secure index satisfies the security definition under the *ciphertext-only attack model*, the cloud server may obtain additional information besides the secure indexes and the trapdoors such as the keyword frequency and distribution among the dataset [44]. As a result, the cloud server should be considered as an adversary under the *known-keyword attack (KKA) model* in practice. A more powerful attack model is the *chosen-keyword attack (CKA) model*, where the adversary has the access to the trapdoor generation algorithm. The adversary could upgrade itself from a known-keyword attacker to a chosen-keyword attack through a so-called *signature linking attack* according to [51]. Because the CKA model is stronger than and inclusive of the KKA model, we will analyze the security of the scheme under the CKA model only.

Our basic scheme does not satisfy the security definition under the CKA model. Recall **IndexEnc** and **QueryEnc**. The encryption ensures the following property when we perform the inner product between an index $\mathcal{I}$ and a query vector $\mathcal{Q}$.

$$\mathcal{I}^T \cdot \mathcal{Q} = \mathcal{I}' \cdot \mathcal{Q}' + \mathcal{I}'' \cdot \mathcal{Q}''$$

Because the adversary can access to **Trapdoor**, given an index, there are only m unknown variables which are the elements of the index. Therefore, m different queries are sufficient to build an equation system to solve the variables. After revealing the index, the adversary can map the Bloom filter back to keywords since we assume the LSH functions used in our scheme are known to the public.

2.4.1 Enhanced multi-keywords fuzzy search scheme

Because the adversary potentially can recover the encrypted indexes through linear analysis under *chosen-keyword attack model* and further infers the keywords in the index, we introduce an extra security layer, i.e., a pseudo-random function f, to secure the linkage between the keywords and the Bloom filter.

Our enhanced scheme contains the following four processes:

- **KeyGen**(m, s)**:** Given a parameter m, generate the secret key $SK(M_1, M_2, S)$, where $M_1, M_2 \in \mathbb{R}^{m \times m}$ are invertible matrices while $S \in \{0,1\}^m$ is a vector. Given another parameter s, generate the hash key pool $HK = \{k_i | k_i \overset{R}{\leftarrow} \{0,1\}^s\}$.
- **BuildIndex**$(\mathcal{D}, SK, l)$: Choose l independent LSH functions from the p-stable LSH family $\mathcal{H}$ and one pseudo-random function $f : \{0,1\}^* \times \{0,1\}^s \rightarrow \{0,1\}^*$. For each file $\mathcal{D}$,
  1. Extract the keywords set $\mathcal{W} = \{w_1, w_2, \cdots\}$ from $\mathcal{D}$.
  2. Generate a m-bit Bloom filter $\mathcal{I}_\mathcal{D}$. Insert $\mathcal{W}$ into $\mathcal{I}_\mathcal{D}$ using the hash functions $\{g_i | g_i = f_{k_i} \circ h_i, h_i \in \mathcal{H}, 1 \leq i \leq l\}$.
  3. Encrypt the $\mathcal{I}_\mathcal{D}$ with SK and return $Enc_{SK}(\mathcal{I}_\mathcal{D})$ as the index.
- **Trapdoor**$(\mathcal{Q}, SK)$**:** Generate a m-bit long Bloom filter. Insert the $\mathcal{Q}$ using the same hash functions g_i, i.e., $g_i = f_{k_i} \circ h_i, h_i \in \mathcal{H}, 1 \leq i \leq l$ into the Bloom filter. Encrypt the $\mathcal{Q}$ with SK and return the $Enc_{SK}(\mathcal{Q})$ as the trapdoor.
- **Search**$(Enc_{SK}(\mathcal{Q}), Enc_{SK}(\mathcal{I}_\mathcal{D}))$**:** Output the inner product $< Enc_{SK}(\mathcal{Q}), Enc_{SK}(\mathcal{I}_\mathcal{D}) >$ as the search result for the query $\mathcal{Q}$ and the document $\mathcal{D}$.

Note that the extra security layer in the enhanced scheme does not affect the search result because the pseudo-random functions are one-to-one mapping. In practice, we can use a keyed cryptographic hash function, e.g. HMAC-SHA1 as our pseudo-random function.

2.4.2 Security analysis of enhanced scheme

Before analyzing the security of our enhanced scheme, we first define *ϵ-pseudo-random function.*

Definition 3. *A function $G : \{0,1\}^\star \to \{0,1\}^n$ is a ϵ-pseudo-random function if*

1. *G is efficiently commutable by a deterministic algorithm;*
2. *For all polynomial-time probabilistic algorithm $\mathcal{A}$,*

$$|Pr[Rand^{\mathcal{A}}_{G(s)} = 1] - Pr[Rand^{\mathcal{A}}_{r} = 1]| < \epsilon$$

   where $s \xleftarrow{R} \{0,1\}^\star, r \xleftarrow{R} \{0,1\}^n$.

Theorem 2. *The secure index in our enhanced scheme satisfies the security definition if the pseudo-random function f used in the scheme is a ϵ-pseudo-random function.*

It is worth mentioning that under the CKA model, the objective of the adversary is to recover the keywords in the secure index.

Proof. Assume there exists an algorithm $\mathcal{A}$ that wins the security game with advantage greater than ϵ. Now we build an algorithm $\mathcal{B}$ which has access to an oracle $\mathcal{O}_f$ where $f : \{0,1\}^\star \to \{0,1\}^{\log m}$. We replace the evaluation of f in our enhanced scheme with $\mathcal{B}$. We show that $\mathcal{B}$ is able to determine whether f is a ϵ-pseudo-random function or a random function. Our detailed proof is presented as follows.

Setup

The algorithm $\mathcal{B}$ picks a set S and sends it to the adversary $\mathcal{A}$. $\mathcal{A}$ chooses a number of subsets from S as the file collection denoted as $S^\star$ which is then returned to $\mathcal{B}$.

After receiving $S^\star$, $\mathcal{B}$ runs **KeyGen** and generates the index of each subset using **BuildIndex**. Finally, the secure index is published to $\mathcal{A}$.

Queries

$\mathcal{A}$ is allowed to query $\mathcal{B}$ on a query x and receive the trapdoor T_x.

Challenge

After making some **Trapdoor** queries, $\mathcal{A}$ picks a non-empty subset $V_0 \in S^\star$, and generates another non-empty subset $V_1 \in S$ such that $|V_0 - V_1| \neq 0, |V_1 - V_0| \neq 0$ and $|V_0| = |V_1|$. Most importantly, $\mathcal{A}$ must not have queried for the trapdoor of any word in $V_0 \triangle V_1 = (V_0 - V_1) \cup (V_1 - V_0)$.

$\mathcal{A}$ then sends V_0, V_1 to $\mathcal{B}$. $\mathcal{B}$ chooses $b \overset{R}{\leftarrow} \{0,1\}$ and generates the secure index for V_b The challenge for $\mathcal{A}$ is to decide b. After the challenge is issued, $\mathcal{A}$ is not allowed to query $\forall x \in V_0 \triangle V_1$.

Response
$\mathcal{A}$ outputs a bit b' as his guess of b.

If $\mathcal{A}$ outputs 0, then $\mathcal{B}$ guesses f used in the random oracle is a random function, denoted as $\mathcal{B}_f = 0$. Otherwise, $\mathcal{B}$ guesses f is a pseudo-random function, denoted as $\mathcal{B}_f = 1$. Similar with our previous proof, $\mathcal{A}$ has advantage greater than ϵ to win the security game, which means

$$|Pr(\mathcal{B}_f = 1) - Pr(\mathcal{B}_f = 0)| > \epsilon. \tag{2.8}$$

Now we prove that if f is a random function, $\mathcal{A}$ guesses b at best with $\frac{1}{2}$ probability. Because the only differences between V_0, V_1 are $V_0 \triangle V_1$, without loss generality, we assume there are only two keywords $x, y \in V_0 \triangle V_1$. $\mathcal{A}$ is not able to link x, y from the trapdoors it has already queried because x, y have not been queried and are independent with others. Therefore, $\mathcal{A}$ at best distinguishes x, y from the outputs of a random function with the probability 1/2. Therefore, Eq. (2.9) can be written as

$$|Pr(\mathcal{B}_f = 1) - 1/2| > \epsilon, \tag{2.9}$$

which means $\mathcal{B}$ can distinguish a ϵ-pseudo-random function from a random function. But this is contradict with the definition of the ϵ-pseudo-random function. □

Since the adversary can generate the trapdoors for any keywords the trapdoor linkability cannot be protected under the CKA model. Due to the same reason, the index linkability is leaked as well. However, it is worth mentioning that the leakage is unavoidable under the CKA model.

2.5 Performance analysis

In this section, we analyze the performance of our scheme in terms of the search result accuracy, i.e., false positive rate and false negative rate.

2.5.1 False positive rate

A *false positive* is that a query keyword q matches with an index $\mathcal{I}$ but $q \notin \mathcal{I}$. It is well known that the Bloom filter will generate false positive when testing the existence of an element. The general false positive rate of an m-bit Bloom filter using l hash functions is $(1 - e^{-\frac{ln}{m}})^l$, where n is the number of the items inserted into the Bloom filter. However, the above result is based on the assumption that all l hash functions used in the Bloom filter hash elements to one of the m array positions with a uniform random distribution. In our scheme, because we use LSH functions to replace standard hash functions, the assumption does not hold. Therefore, we cannot use the above result directly.

To analyze the false positive rate, we first need to study the distribution of the hash results. Recall the form of the p-stable LSH function which is

$$h_{a,b}(\mathbf{v}) = \lfloor \frac{\mathbf{a} \cdot \mathbf{v} + b}{w} \rfloor.$$

Because w is fixed for every hash function and b will not change the shape of the probability density function of the distribution, we focus on the probability distribution of the random variable $Y = \mathbf{a} \cdot \mathbf{v}$ where $\mathbf{v}$ is the bigram vector and $\mathbf{a}$ is the parameter of the LSH function. For each element $a_i, 1 \leq i \leq n = 26^2$, it is drawn independently from a p-staple distribution $\mathcal{D}$ which is Gaussian distribution ($p = 2$) in our case. Without loss generality, we assume $\mathcal{D} \sim \mathcal{N}(0, 1)$.

One property of choosing the 2-stable distribution $\mathcal{D}$ is that when $v_1, v_2, \cdots, v_n$ are real numbers and $a_1, a_2, \cdots, a_n$ are i.i.d. random variables drawn from $\mathcal{D}$, the inner product of $\mathbf{v}, \mathbf{a}$, i.e., $\sum_i v_i a_i$, forms a new random variable

$$\left(\sum_i |v_i|^2 \right)^{1/2} X \tag{2.10}$$

where X follows distribution $\mathcal{D}$ as well. In our case, the bigram vector $\mathbf{v}$ is a $0 - 1$ vector. Therefore, equation 2.10 can be written as

$$\left(\sum_i v_i \right)^{1/2} X \tag{2.11}$$

Because we want to examine the hash result distribution over all possible elements, i.e., bigram vectors, we need to determine the distribution of $\sum_i v_i$. Instead of uniformly distributed, the bigrams of English words approximately follow Zipf's distribution which is also known as discrete Pareto distribution [56]. To simplify our analysis, we assume the appearance of each bigram is independent[2]; otherwise the problem becomes impossible to analyze due to the complicated correlation amongst all the bigrams. Based on the independence assumption, we can model each element v_i in the bigram vector as a Bernoulli trial with the success probability p_i. Then the summation $\sum_{i=1}^{n} v_i$ forms a Poisson binomial distribution of which the mean and the variance is

$$\mu = \sum_{i=1}^{n} p_i, \ \sigma^2 = \sum_{i=1}^{n} (1 - p_i) p_i,$$

[2]The appearance of each bigram is not independent because it is decided by the word. Furthermore, the vector space in practice is much less than the theoretical one because the average length of an English word is five letters.

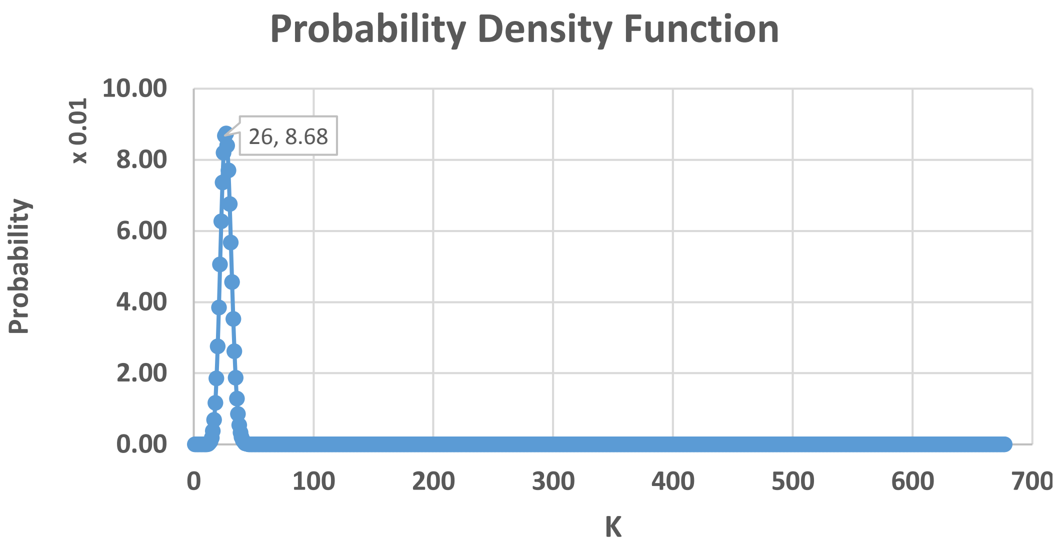


Figure 2.3: Probability density function of $\sum_{i=1}^{n} v_i$.

respectively. We use the data from [57] to calculate the density function based on the recursive method which is shown in equation 2.12. We draw the probability density function in figure 2.3.

$$Pr(K = k) = \begin{cases} \prod_{i=1}^{n}(1 - p_i) & k = 0 \\ \frac{1}{k}\sum_{i=1}^{k}(-1)^{i-1}Pr(K = k - i) \cdot T(i) & k > 0 \end{cases} \tag{2.12}$$

where $T(i) = \sum_{j=1}^{n}\left(\frac{p_j}{1-p_j}\right)^i$. Note that because taking root square is a one-to-one mapping, it will not change the shape of the distribution curve. Therefore, with the distribution of $\|\mathbf{v}\|$, we can now analyze the distribution of the hash result of the 2-stable LSH function.

The hash result is a joint-distribution of two independent random variables, i.e., $\|\mathbf{v}\|, X$. The density function will be the product of both individual density functions. We draw the density function graph in figure 2.4. Now we can use the hash result density function which we denote as $f(v, a)$ to analyze the false positive rate. For each element v_i in the Bloom filter, it is not hashed into the jth array positions by l LSH functions is

$$\prod_{k=1}^{l}(1 - f_{v_i}^{-1}(j, a_k), \tag{2.13}$$

where f^{-1} is the inverse of the density function f. Then after inserting n elements, the

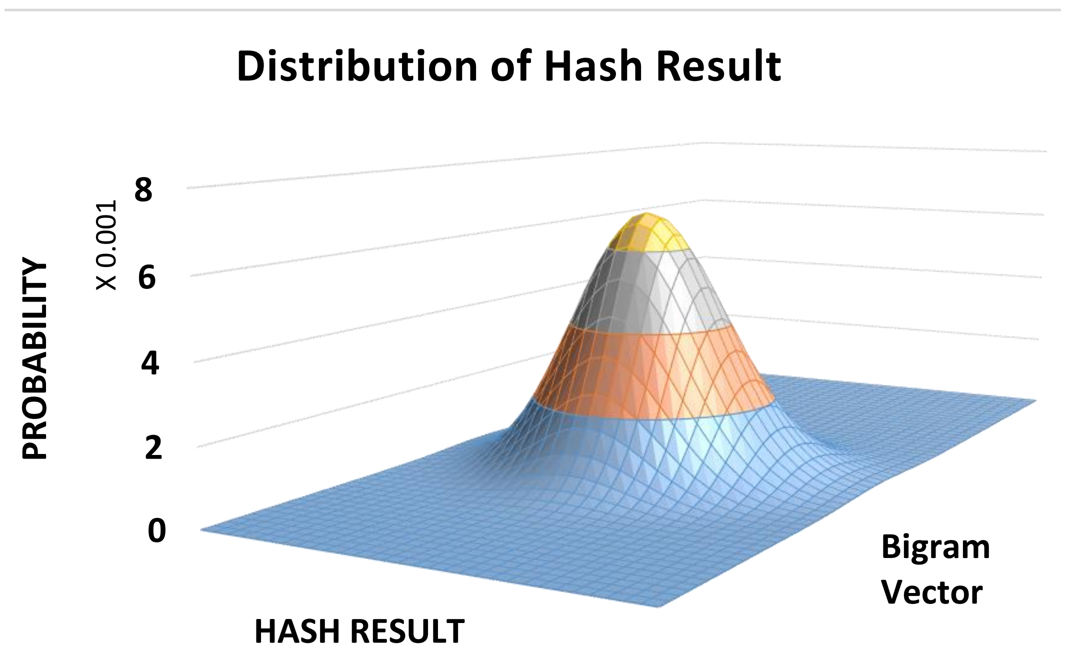


Figure 2.4: The distribution of the hash result, where the x-axis is the hash result, the y-axis is the probability and the depth represents the different $\|v\|$.

probability that the particular position is still 0 will be

$$\prod_{i=1}^{n}\big(\prod_{k=1}^{l}(1-f_{v_i}^{-1}(j,a_k))\big). \tag{2.14}$$

Now it is clear that the false positive rate is

$$\left(1-\prod_{i=1}^{n}\big(\prod_{k=1}^{l}(1-f_{v_i}^{-1}(j,a_k))\big)\right)^{l} \tag{2.15}$$

False positive rate discussion

During the analysis, we made one assumption that the appearance of each bigram was independent. As we stated previously, we made the assumption mainly due to the complicated probability distribution of the bigram and the hash result. However, because the statistic of the frequency of the English bigrams in [57] has already implied the relationship among bigrams, our analysis which is based on their result can be considered as a close approximation of the false positive rate. In practice, we can tune the parameter w which determines the number of buckets of a hash table to achieve lower false positive rate.

2.5.2 False negative rate

A *false negative* is that a query keyword q does not match with an index $\mathcal{I}$ but $\exists p \in \mathcal{I}$. Unlike false positive, the Bloom filter will not generate false negative. Therefore, for exact keyword matching, our scheme will not generate false negative because the same keyword will always be hashed to the same positions by the LSH functions. However, for the fuzzy keyword matching, LSH functions may hash close keywords, i.e., $d(p, q) < r_1$, to different locations, where r_1 is the parameters defined in the (r_1, r_2, p_1, p_2)-sensitive hash functions.

A false negative occurs when there exists at least one hash function that hashes two close keywords separately. As defined in the LSH, the probability of $h_{a,b}(p) = h_{a,b}(q)$ is no less than p_1. Therefore, the false negative rate is no bigger than $1 - p_1^l$ in our case.

False negative discussions

Our design returns the documents with the highest inner product score, which works fine when no fuzzy keyword presents in the query. But when there are fuzzy keywords in the query, some of the matched files may be missed from the search result because the inner products may be lowered due to one or more *misses* caused by LSH functions. This increases the false negative rate. One alternative approach to reducing the false negative rate is to add documents with the relatively high score into the result to reduce the false negative rate. However, the alternative will increase the false positive rate. We notice the trade-off between the false positive rate and the false negative rate of our scheme. It is commonly known that lowing the false positive rate will increase the false negative rate and vice versa. Therefore, the data owner can tune the parameters, i.e. m, l, to specifically fit his accuracy requirements. Additionally, [38, 44] show that returning obfuscated files instead of the exact files into the search result can improve the security.

2.6 Experimental results

2.6.1 Experiment settings

We use the recent 10 years' IEEE INFOCOM publication as our experiment dataset which contains more than 3600 files. We extract 5734 unique keywords in total. The average number of the keywords in a paper is 147 while the minimum and the maximum are 112 and 175, respectively.

We use 2-stable LSH function with $r_1 = \sqrt{3}, r_2 = 2$ to build the index. The LSH functions can tolerant less than 2 bigram difference. Based on the result of [52], the probability of hashing two close bigram vector together(separately) is $p_1 = P(\sqrt{3}) = 0.558864(p_2 =$

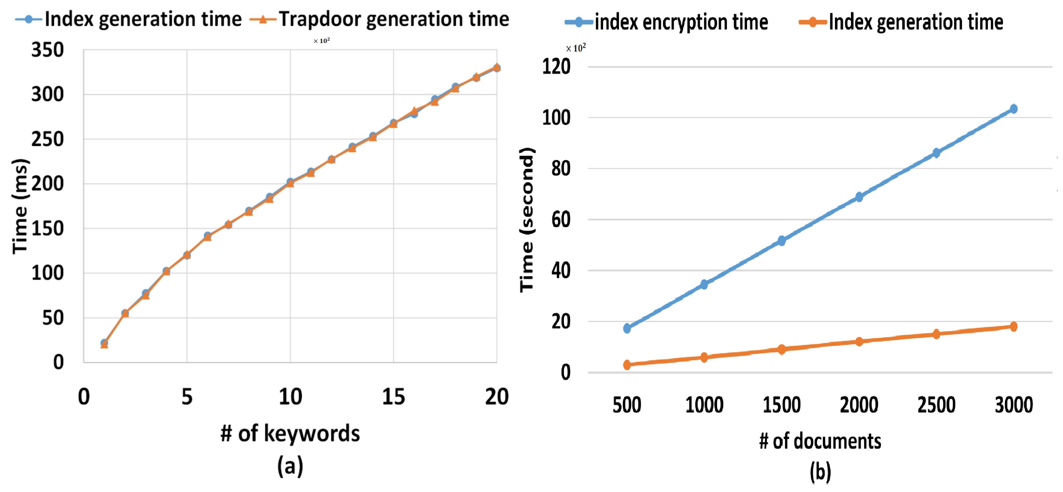


Figure 2.5: Index and trapdoor generation time.

$P(3/4) = 0.285932$) using

$$P(x) = -\frac{\left(1 - e^{-\frac{x^2}{2}}\right)\sqrt{\frac{2}{\pi}}}{x} + \text{Erf}\left[\frac{x}{\sqrt{2}}\right],$$

where Erf is the error function. For the Bloom filter, we choose the number of the hash function as $l = 10$, and set the length of each Bloom filter as $m = 8000$.

We implement our schemes on a desktop PC equipped with Intel Core i3 processor at 3.3 GHz and 4 Gb RAM, which has the same computation power with an Amazon EC2 M1 Medium instance. To generate a fuzzy keyword in a query, we randomly choose one letter from a keyword and replace it with another letter. We allow at most two fuzzy keywords in a query.

2.6.2 Computation efficiency

Index and trapdoor generation

The index generation process is a one-time computation which contains two major steps: the Bloom filter generation and the encryption. During the Bloom filter generation, the computation mainly comes from the hash function calculation. Figure 2.5.(a) shows the Bloom filter generation time for the index and the trapdoor Bloom filter on the number of keywords in the document. The generation time increases linearly on the number of the inserted keywords. The trapdoor generation time is very close to the index generation time due to the identical procedure. The encryption time which involves the matrix multiplications is showed in figure 2.5.(b). The time cost of encryption increases linearly respect to the number of the files in the dataset.

Search over encrypted index

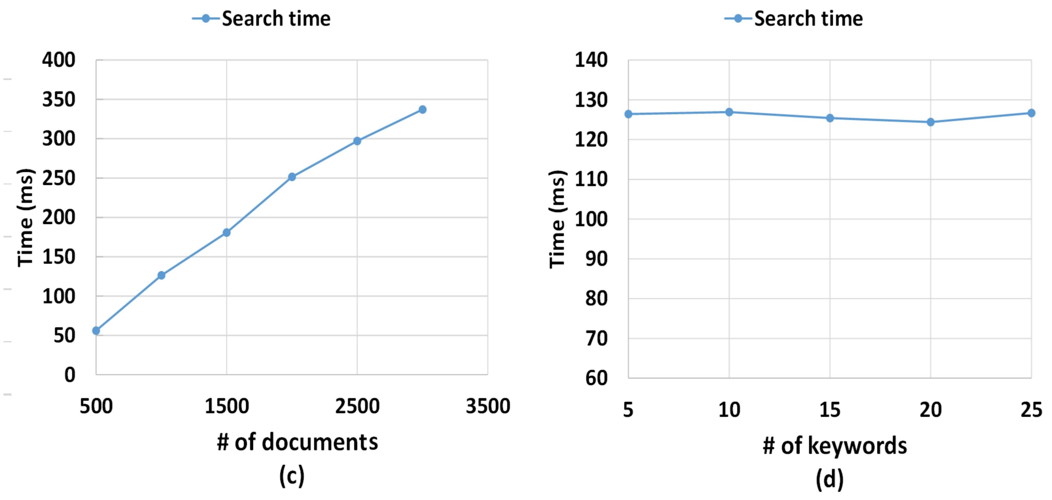


Figure 2.6: Search time for different parameter settings.

The search operation executed at the cloud server side consists of computing the inner product calculation for all the files in the dataset. Figure 2.6.(c) shows the search time grows linearly with the size of the file set while the number of keywords in the query has little impact as showed in figure 2.6.(d). This is intuitive because the search process needs to go over all the files in the dataset before the cloud server can get the final result. The inner product computation is only related to the length of the index, so the computation time changes little in figure 2.6.(d).

2.6.3 Result accuracy

We adopt the definitions of the widely used performance metrics, precision and recall to measure the search result accuracy. Denote tp as true positive, fp as false positive and fn as false negative, then the precision equals to $\frac{tp}{tp+fp}$ while the recall is $\frac{tp}{tp+fn}$. To generate the fuzzy queries, we randomly pick two keywords and modify them into the fuzzy keywords.

Figure 2.7.(a) shows the performance metrics of our scheme according to l. Note that there is no recall for the exact matching because the false negative doesn't exist. One observation is that precision is very low when k is small, i.e. 5% at $l = 1$. Because that multiple LSH functions are used together to enlarge the gap between p_1 and p_2. So when the l is small,

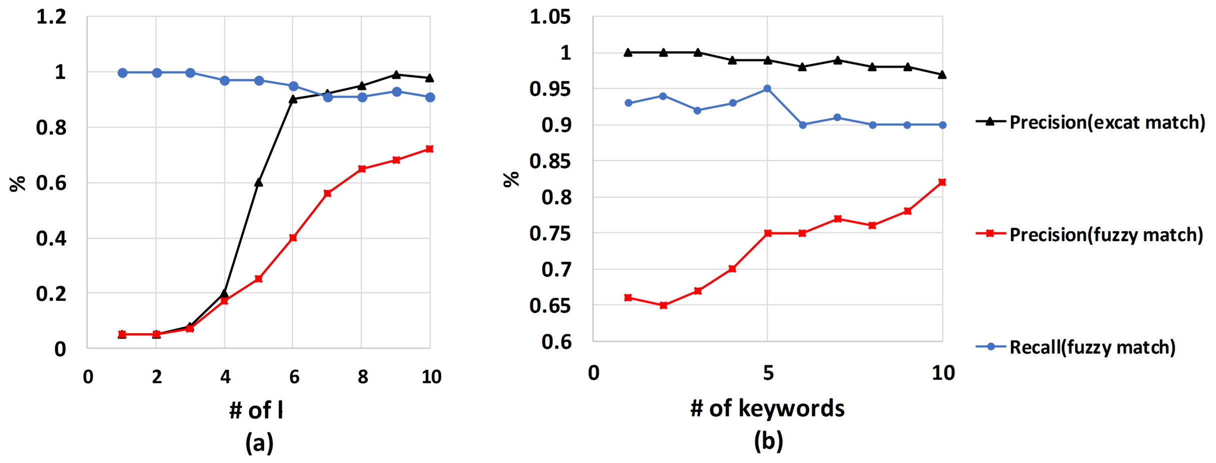


Figure 2.7: Search result accuracy for fuzzy keyword queries.

the gap is not big enough to distinguish the different keywords, and most of the files in the dataset have been returned, which leads to high fp and low fn. The jump at $l = 5$ is due to the gap between p_1 and p_2 increases exponentially respect to l. After a certain l, i.e., $l = 8$, the precision remains at a high level, which is above 90% for the exact search and above 80% for the fuzzy search. Another observation is that the recall drops when increasing the l. This is because that increasing the l will cause more false negatives. In general, the false positive and the false negative cannot be improved at the same time.

Another important parameter is the number of the keywords in the query. Figure 2.7.(b) shows the precision of the exact match decreasing slightly, from 100% to 96% while the number of the keywords in the query increases from 1 to 10. This is reasonable because the false positive generated by each keyword accumulates. But the precision for the fuzzy search doesn't show the same pattern. It is slightly increased from 70% to 81% when the number of the keywords in the query increases from 1 to 10. The reason is that the false positive caused by the LSH functions contributes much more than the false positive introduced by the Bloom filter. As the portion of the fuzzy keywords decreases, the impact of the false positive caused by the fuzzy keyword is reduced since the fuzzy keywords contribute less in the search result.

2.7 Summary

In this chapter, we tackled the challenging multi-keyword fuzzy search problem over the encrypted data. We proposed and integrated several innovative designs to solve the multiple keywords search and the fuzzy search problems simultaneously with high efficiency. Our approach of leveraging LSH functions in the Bloom filter to construct the file index is novel and provides an efficient solution to the secure fuzzy search of multiple keywords. Also, the Euclidean distance is adopted to capture the similarity between the keywords, and the secure inner product computation is used to calculate the similarity score so as to enable result ranking. We proposed a basic scheme as well as an improved scheme to meet different security requirements. Thorough theoretical analysis and simulation were carried out to demonstrate the suitability of our proposed scheme for the practical usage.

Chapter 3

Multi-keyword Dynamic Searchable Encryption based on Inverted-index

As cloud computing is continuing gaining popularity, more and more data has been moved into the cloud. To provide privacy protection while enabling data search functionality, searchable encryption which builds a secure index is a promising technique to achieve the security and functional objectives. In the searchable encryption literature, some of the works [13, 16, 24–26, 41, 58–64] design their own secure index structures. On the other hand, it is also possible to secure an existing index structure. Inverted index is one of the most popular index structures used in plaintext search. It has been widely adopted in today's search engines and databases. Comparing with constructing a secure index from scratch, building a secure index based on the inverted index provides a more convenient transition for a dataset that already has an inverted index to move into cloud with privacy protection as well as keyword search enabled. Several works [17–19] build their secure indexes based on the inverted index. However, the following limitations confine these solutions to be impractical in cloud computing. First of all, the keyword privacy is compromised once a keyword is searched. As a result, the index must be rebuilt for the keyword once it has been searched. Obviously, such a solution is counterproductive due to the high overhead suffered. Secondly, the conjunctive multi-keyword search is not supported in these schemes. As the most popular query type, conjunctive multi-keyword query support is essential in practical searchable encryption schemes. Thirdly, the existing schemes do not support index updates when changes are made to the underlying document set, i.e., document addition and deletion. At last, as the cloud may cheat the users to avoid expensive computation, the users should be able to verify the correctness of the search result.

In this chapter, we propose a dynamic searchable encryption scheme based on the inverted index. Our scheme features a probabilistic verification process which allows users to verify the correctness of the returned result. Compared with the existing scheme, our scheme advances in both privacy-wise and functionality-wise. Privacy-wise, our scheme protects the search pattern as well as the access pattern through a series of novel designs based on a private set intersection protocol. Functionality-wise, our scheme supports multi-keyword queries and allows the data owner to update the secure index when the underlying document set has been changed. The update protocol supports document addition and deletion while preserving the index forward and backward privacy.

3.1 Related works

The searchable encryption problem is first studied by Song *et al.* in [23]. The proposed scheme supports single keyword search without an index which means the server must scan the whole document to perform the keyword search. Follow-ups on searchable encryption usually build a secure searchable index such that certain trapdoors generated via secret keys could match with the index to get the search result while the content of the index is hidden from the cloud. Some of the works [13,16,24–26,41,58–64] design their own index structures while others [17–19] build their secure searchable indexes upon the inverted index.

Among the self-designed secure indexes, Goh *et al.* [41] first proposed a Bloom filter based index which supports single keyword search. Chang *et al.* [58] used a vector index of which the length is the same as the cardinality of the dictionary for each document. Other works [13,16,24,26,60–64] focus on enrich the search functionality including result ranking, multi-keyword search and fuzzy search. In public key settings, Boneh *et al.* [25] proposed the first public key based searchable encryption scheme. Bellare *et al.* [59] introduced an as-strong-as-possible privacy definition for the public-key based searchable encryption and constructed a solution that satisfies their definition. One major disadvantage of using self-designed indexes is that their index structures are not compatible with each other. As a result, it is impossible to provide a service that includes all the useful functions. Additionally, for users who have already built their inverted indexes for plaintext search, they need to re-generate their encrypted searchable index which could be expensive if the data volume is huge.

Curtmola *et al.* [17] proposed the first inverted index based encrypted searchable index. The document list for each keyword is encrypted and obfuscated into an array. However, according to the design, the position and the content of the inverted list will be disclosed to the cloud server once the keyword is searched. As a result, one keyword can only be searched once before regenerating the index for the keyword. Based on [17], Kamara *et al.* [18] put forward the concept of dynamic searchable encryption and constructed an encrypted inverted index that supports dynamic operations such as document updates. Naveed *et al.* [19] designed a primitive named blind storage upon which they implemented the encrypted searchable index scheme of [17]. As these works are based on [17], they share the same limitation. Moreover, these schemes do not support conjunctive multi-keyword search which is the most common query type nowadays. Stefanov *et al.* [65] proposed a practical dynamic searchable encryption scheme that preserves the forward privacy. Compared with their scheme, ours provides a better privacy guarantee because our scheme protects the search

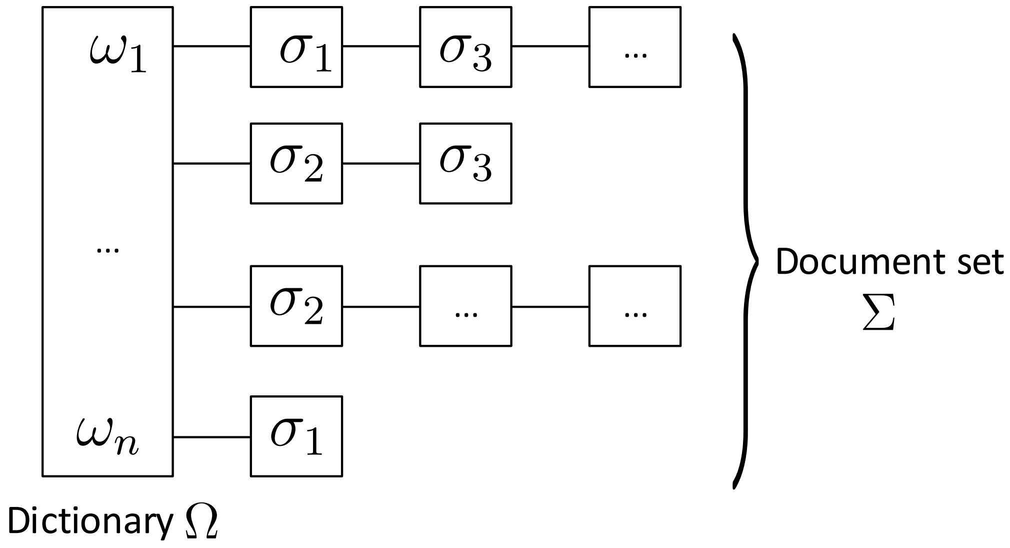


Figure 3.1: Inverted index illustration.

pattern and the access pattern. Additionally, our scheme provides both the forward and the backward privacy for the index updates.

Our scheme falls into the public-key searchable encryption category which includes [25, 59, 60, 64, 66–68]. While those works all adopt a pairing-based cryptosystem to construct the indexes, our scheme only requires multiplication and exponentiation, which are a magnitude less expensive in computation comparing to pairing operations. Additionally, most of these schemes did not provide an implementation evaluation of the algorithm performance (we only found [60, 64] performed the experimental studies). In this work, we not only provide a complexity analysis of our algorithm but also perform a simulation study to evaluate the performance. The simulation results show that our scheme is suitable for practical usage and is more efficient than existing public-key based solutions.

3.2 Preliminary

3.2.1 Inverted index

Inverted index is one of the most popular data structures used in document retrieval systems [69]. As shown in Fig. 3.1, an inverted index contains multiple inverted lists. One inverted list I_{ω_i} corresponds to one keyword ω_i which is contained in all the documents of I_{ω_i}. Additional information can be included in the inverted list such as the numerical statistics of the keyword (to support result ranking) and the positions of the keyword within a document (to support phrase search). The biggest advantage of using the inverted index comes from the search efficiency especially for a large volume of documents. The search operation is performed on a much smaller document set which consists of the document IDs in the inverted lists of the query keywords.

3.2.2 Private set intersection

We build our secure index based on Private set intersection (PSI) technique. PSI is a cryptography primitive that allows two or more parties to calculate the intersection of their sets privately. The output of the PSI reveals no additional information other than the intersection itself. In [70], Freedman *et al.* proposed an efficient PSI protocol, denoted as the FNP protocol, based on an additive homomorphic encryption scheme. They use the Paillier homomorphic cryptosystem [71] in their paper but any additive homomorphic encryption that supports the following operations will work.

$$E(m_1 + m_1) = E(m_1) +_h E(m_2) \tag{3.1}$$

$$E(m_1 \times m_2) = E(m_1) \times_h m_2 \tag{3.2}$$

where $+_h, \times_h$ are homomorphic addition and multiplication, respectively.

A brief description of the Paillier algorithm is shown in Fig.3.2. The Paillier cryptosystem features 1) additive homomorphic, i.e. given the ciphertexts $E(a_1), E(a_2)$, the ciphertext of $a_1 + a_2$ can be calculated as $E(a_1 + a_2) = E(a_1)E(a_2)$, and 2) one-time multiplicative homomorphic, i.e. given the ciphertext $E(a_1)$, the ciphertext of $a_1 \times a_2$ is $E(a_1)^{a_2}$.

Key generation: $pk = (n, g)$, where $n = pq, \gcd(pq, (p-1)(q-1)) = 1, g \in \mathbb{Z}^*_{n^2}$
$sk = (\lambda, \mu)$, where $\lambda = \operatorname{lcm}(p-1, q-1), \mu = \left(\frac{g^\lambda \mod n^2 - 1}{n}\right)^{-1} \mod n$
Encryption: To encrypt a message m to its ciphertext c
$c = g^m \cdot r^n \mod n^2, r \in \mathbb{Z}_n$
Decryption: $m = \frac{c^\lambda \mod n^2 - 1}{n} \cdot \mu \mod n$

Figure 3.2: The scheme of the Paillier homomorphic cryptosystem

The FNP protocol works as follows. 1) Alice represents her set $\mathcal{A}$ as a polynomial $f(x) = \prod_{a_i \in \mathcal{A}}(x - a_i)$. Clearly, the set of the roots of $f(x) = 0$ is $\mathcal{A}$. 2) Alice encrypts the coefficients of the polynomial using the additive homomorphic encryption and sends the encrypted polynomial $f'(x) = Enc(f(x))$ to Bob. 3) Bob calculate $\mathcal{R} : \{r_j = f'(b_j) +_h b_j\}$ with his data $b_j \in \mathcal{B}$. Then Bob sends $\mathcal{R}$ back to Alice. 4) Alice decrypts $\mathcal{R}$ as $\mathcal{R}'$, and the intersection $\mathcal{A} \cap \mathcal{B}$ is the intersection $\mathcal{A} \cap \mathcal{R}'$.

3.2.3 Fast Fourier transform

We utilize Fourier transform to design our secure index update protocol. A polynomial can be determined either by its coefficients or a set of point-value pairs. The later is known as the *point-value representation* of a polynomial. The discrete Fourier transform (DFT) converts the coefficient representation of a polynomial to its point-value representation. The inverse DFT does the opposite. The fast Fourier transform algorithm [72] is an efficient algorithm to compute both the DFT and the inverse DFT with complexity $O(n \log n)$.

Given a polynomial f with degree $d-1$

$$f(x) = a_{d-1}x^{d-1} + \cdots + a_1 x + a_0,$$

we denote the coefficient form of f is the collection of the coefficients, i.e., $\{a_{d-1}, \cdots, a_0\}$ and the point-value form of f as $\{f(1), f(w), \cdots, f(w^{d-1})\}$ where $w \in \mathbb{R}$. We show the recursive version of the FFT algorithm in Fig. 3.3. Given a polynomial f with degree $d-1$

$$f(x) = a_{d-1}x^{d-1} + \cdots + a_1 x + a_0,$$

we denote the coefficient form of f is the collection of the coefficients, i.e., $\{a_{d-1}, \cdots, a_0\}$ and the point-value form of f as $\{f(1), f(w), \cdots, f(w^{d-1})\}$ where $w \in \mathbb{R}$.

To simply the presentation, we assume $d = 2^k$, but the FFT algorithm works on any d. The FFT algorithm is based on the following observation. We denote f_e as

$$f_e(x) = a_0 + a_2 x + \cdots + a_{d-1}x^{d/2},$$

and f_o as

$$f_o(x) = a_1 + a_3 x + \cdots + a_d x^{d/2}.$$

It is easy to see that

$$f(x) = f_e(x^2) + x f_o(x^2).$$

Therefore, the original problem has been divided into two problems each of which is half the size of the original one.

Mohassel proposed an efficient algorithm for encrypted FFT in [73]. The scheme calculates the FFT and the inverse FFT algorithms over a polynomial of which the coefficients are encrypted with an additively homomorphic encryption algorithm such as the Paillier encryption scheme. Because the Paillier encryption scheme can perform addition among ciphertext and multiplication between a ciphertext and a plaintext. The above FFT algorithm can be easily modified to the encrypted FFT in Fig. 3.4.

As mentioned before, we can use the FFT to compute the inverse FFT. Therefore, the inverse encrypted FFT is to compute

$$1/n \times_h \mathrm{EncFFT}_{d,u^{-1}}(Enc(f)),$$

which can be used to calculate the coefficient representation of a polynomial from its point-value representation.

```
FFT$_{d,u}(f)$ :
Input: $d = 2^k \in \mathbb{N}, \{a_{d-1}, \cdots, a_0\}$, and $u \in R$ which is the $n$th root of unity
Output: $\{f(1), \cdots, f(u^{d-1})\}$

    1. if $d = 1$ then return $a_0$.
    2. Compute $r_e = \{a_j + a_{j+d/2}\}_{j=0}^{d/2}$, and $r_o = \{(a_j - a_{j+d/2})u^j\}_{j=0}^{d/2}$.
    3. Compute FFT$_{d/2,u^2}(r_e)$ and FFT$_{d/2,u^2}(r_o)$
```

Figure 3.3: The recursive version of the FFT algorithm

```
EncFFT$_{d,u}(Enc(f))$ :
Input: $d = 2^k \in \mathbb{N}, \{Enc(a_{d-1}), \cdots, Enc(a_0)\}$, and $u \in R$ which is the $n$th root of unity
Output: $\{Enc(f(1)), \cdots, Enc(f(u^{d-1}))\}$

    1. if $d = 1$ then return $Enc(a_0)$.
    2. Compute $r_e = \{Enc(a_j) +_h Enc(a_{j+d/2})\}_{j=0}^{d/2}$, and
       $r_o = \{Enc(a_j) -_h Enc(a_{j+d/2}) \times_h u^j\}_{j=0}^{d/2}$.
    3. Compute EncFFT$_{d/2,u^2}(r_e)$ and EncFFT$_{d/2,u^2}(r_o)$
```

Figure 3.4: The recursive version of the encrypted FFT algorithm

3.3 System model

The system model that we consider in this work is same as the system in Chapter 2. There are three entities in the system, a cloud server, a data owner and multiple users. The data owner generates the secure index and outsources it along with the encrypted data into the cloud. An authorized user submits a query request to the server in the form of a trapdoor which he gets from the data owner through a secure channel. After receiving the trapdoor, the cloud server matches the secure index with the trapdoor. Finally, the cloud server returns the matching documents as the search result. It is worth mentioning that the data owner can outsource the trapdoor generation process to a trusted entity to avoid staying online all the time because the trapdoor generation process does not require the knowledge of the dataset.

We model the cloud server as an *honest-but-curious* adversary. We assume that the users need to be authorized before accessing the document set through keyword search. The assumption is based on the fact that the access to the document collection is restricted to authorized users in practical application of searchable encryption. Open access to the document collection for all users including the cloud server will have no incentive for the data owner to adopt searchable encryption to protect the content of the documents because the cloud server can learn the document content as search results through querying the file collection. Therefore, the users are fully trusted by the data owner. The access control between the data owner and the users is out of the scope of this paper as it can be achieved using existing protocols such as [64, 74].

3.3.1 Threat model

A searchable encryption scheme should protect data owner's *data privacy.* The data privacy violation could come from three aspects, i.e., the document set, the secure index and the query trapdoors. In most scenarios, the data owner will encrypt his document contents using a block cipher such as AES [4]. Therefore, it is safe to claim that the privacy of the document set itself is well protected. So we focus on the privacy of the other two aspects, i.e., the *secure index privacy* and the *trapdoor privacy.*

Index privacy: The index privacy is twofold. Firstly, the cloud server should not learn the content of the index since the content of the index directly reflects the content of the documents. Secondly, the cloud server should deduce no information about the document through analyzing the encrypted index. Such information includes 1) whether a document contains the certain keyword(s), and 2) whether different documents contain a common keyword.

Trapdoor privacy: A trapdoor is generated for each query request to allow the cloud server to search over the secure index. Intuitively, the trapdoor contains the query information but in an encrypted form. Given a trapdoor, the cloud server should learn nothing about the user's query from it. We consider the protection of the following information for trapdoor privacy: the content of the query, the number of the keywords in the query, and the fact that whether the same query has been searched before.

Access pattern refers to the accessed documents, i.e. the search results. As pointed out by [75], the adversary could further deduce the private information of the index and the trapdoor from the access pattern. To avoid such leakage, the search results of queries must be indistinguishable from each other.

Forward privacy & backward privacy are properties of a dynamic searchable encryption protocol. If the cloud server cannot learn whether the newly added document has the previously queried keywords through the index update process, the update protocol preserves the forward privacy. Backward privacy means the cloud server cannot perform queries over the deleted documents.

3.3.2 Definition and notation

We will use the following notations through the rest of the paper.

- $\Sigma = (\sigma_1, \sigma_2, \cdots, \sigma_n)$ is a finite set of document collection, where σ_i is the ID of the ith document.
- $\Omega = (\omega_1, \omega_2, \cdots, \omega_m)$ is a finite set of the keywords collected from Σ. We refer Ω as the dictionary.
- $\mathcal{I} = (I_{\omega_1}, I_{\omega_2}, \cdots, I_{\omega_m})$ is an inverted index for the document set Σ and the dictionary Ω. Each I_{ω_i} is a list which contains $\Sigma_i = (\sigma_{i1}, \sigma_{i2}, \cdots, \sigma_{ip}) \subset \Sigma$ where $\omega_i \in \sigma_{ij}, 1 \leq j \leq p$.
- $\tilde{\mathcal{I}}$ is the encrypted searchable index based on $\mathcal{I}$.
- $\mathcal{Q} \subset \Omega$ is a query request which is a subset of the dictionary.
- $\mathcal{T}_\mathcal{Q}$ is the trapdoor for the query $\mathcal{Q}$.
- $[n]$ means an integer set from 1 to n.
- $|\mathcal{S}|$ refers to the cardinality of $\mathcal{S}$ which can be a set, a list or a vector.

Before delving into the detail of our scheme, we first present the searchable encryption definition given in [17,25].

Definition 4 (Searchable Encryption Scheme)**.** *A searchable encryption scheme consists of the following probabilistic polynomial time algorithms.*

- ***Setup***(k) *takes a security parameter k as input. It outputs a master key MK.*
- ***IndexGen(***MK,$\mathcal{I}$***)*** *takes as input the master key MK and an index $\mathcal{I}$ for the document set Σ. It outputs the encrypted searchable index $\tilde{\mathcal{I}}$.*
- ***TrapdoorGen***$(MK, \mathcal{Q})$ *takes as input the master key MK and a query $\mathcal{Q}$. It outputs the trapdoor $\mathcal{T}_\mathcal{Q}$ for the query.*
- ***Query(***$\tilde{\mathcal{I}}, \mathcal{T}_\mathcal{Q}$*)* *takes as input the encrypted index $\tilde{\mathcal{I}}$ and the trapdoor $\mathcal{T}_\mathcal{Q}$. It outputs $\mathcal{R} \subset \Sigma$ as the search result.*

A verifiable searchable encryption has an additional algorithm besides the above four algorithms to allow the user to verify the correctness of the search result.

- ***Verify(***$\mathcal{R}, \tau$*)* *takes the search result $\mathcal{R}$ and the verification token τ as inputs. It outputs* 1 *if $\mathcal{R}$ is correct; otherwise, it outputs $\perp$.*

3.4 Inverted index based public-key searchable encryption scheme

We present our scheme in this section. We assume the data owner already has an inverted index available for the document collection. Thus, we skip the process of building an inverted index from a document set.

3.4.1 Overview

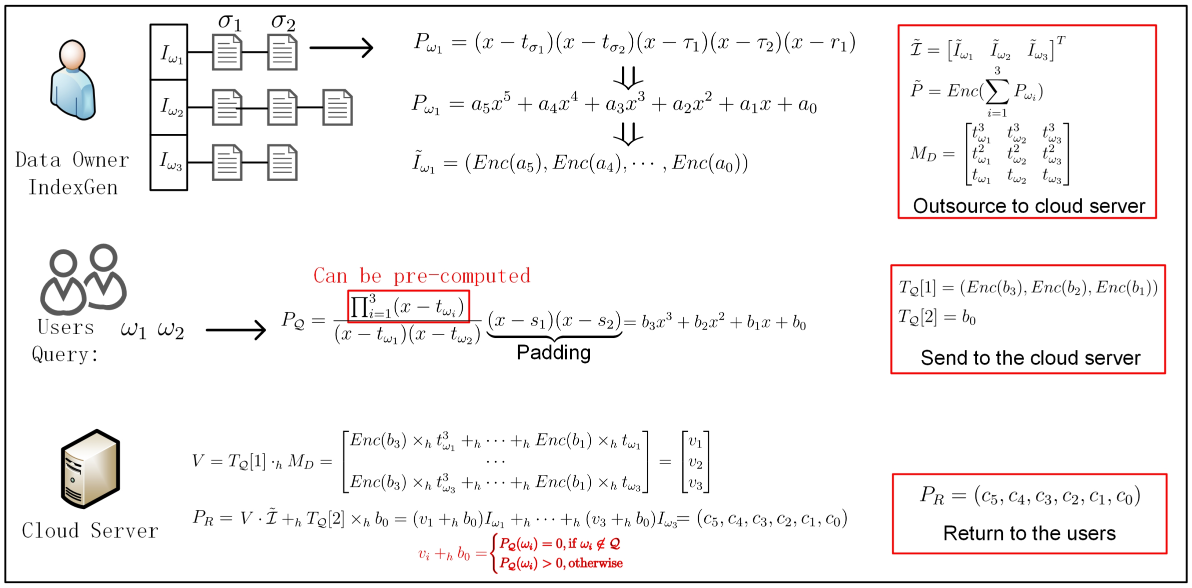
As discussed in section 3.2, an inverted index consists of two parts: 1) a keyword dictionary and 2) a document list for each keyword. When performing a search over plaintext, the server matches the query keyword(s) to the dictionary to locate the target document list(s) first. Then the server gets the document candidates by integrating the document list(s) together. Finally, the document candidates are returned to the user as the search result.

When searching over encrypted inverted indexes, we have to solve three challenging tasks. We first need a privacy preserving method to determine the match between the query keyword(s) and the dictionary. To conceal the search pattern, we need to select the related inverted lists without letting the cloud server know which ones are retrieved. Finally, the search result must not directly reveal the access pattern, i.e., the document IDs of the search result. We address these challenges through a series of novel designs based on the FNP PSI protocol. To make the presentation clear, we use a simple example in Fig. 3.5 to illustrate our scheme. 1) The data owner transforms each inverted list to a polynomial as described in the algorithm. The polynomial is represented using its coefficients. To secure the index, the data owner encrypts the coefficients of each polynomial. The data owner also generates a dictionary matrix M_D. 2) When generating a trapdoor for a query, the data owner first calculates P_Ω/P_Q, then adding randomness to hide the number of the keywords in the query. The trapdoor is a 2-tuple. 3) When the cloud server receives the trapdoor, he calculates V as shown in the figure. Because $P_R(\omega_i) = 0$ if $\omega_i \notin \mathcal{Q}$, the result polynomial $P_R(x)$ only contains the inverted lists corresponding to the query keywords. We now present the details of our scheme as follows.

3.4.2 Scheme details

Each document list in the inverted index is a set of document IDs. We use a polynomial to represent a document list in our scheme.

- **Setup**(k) takes a security parameter k as input. It outputs a key pair E_k, D_k for the additive homomorphic encryption algorithm and two pseudorandom permutations (PRP) f_1, f_2 as the master key MK.



Data Owner
IndexGen
$P_{\omega_1} = (x-t_{\sigma_1})(x-t_{\sigma_2})(x-\tau_1)(x-\tau_2)(x-r_1)$
$P_{\omega_1} = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$
$\tilde{I}_{\omega_1} = (Enc(a_5), Enc(a_4), \cdots, Enc(a_0))$
Outsource to cloud server
Users
Query:
Can be pre-computed
Padding
Send to the cloud server
Cloud Server
$v_i +_h b_0 = P_Q(\omega_i)=0$, if $\omega_i \notin Q$; $P_Q(\omega_i) > 0$, otherwise
Return to the users

Figure 3.5: Scheme process illustration.

- **IndexGen**$(MK, \mathcal{I})$ takes an inverted index $\mathcal{I}$ and the master key MK. It outputs a secure index $(\tilde{\mathcal{I}}, \tilde{P})$, where $\tilde{\mathcal{I}}$ is a vector with each element representing a document list polynomial and $\tilde{P}$ is the summation of all the document list polynomials. Given an inverted index, the data owner encrypts it as follows.

  1. The data own encodes each keyword $\omega_i \in \Omega$ with f_1 as $t_{\omega_i} = f_1(\omega_i)$. Each document list I_{ω_i} for the keyword ω_i is represented as a polynomial with two parts.

     (a) The first part represents the document ID set. A tag $t_{\sigma_j} = f_1(\sigma_j)$ is generated for each document ID. We use the polynomial to represent the set. Therefore, the first part of the polynomial is

     $$\prod_{\sigma_j \in I_{\omega_i}} (x - t_{\sigma_j}).$$

     (b) The second part contains the verification tokens. For each verification token $\tau_j, 1 \le j \le N$ where N is a fixed number, generate a random number $f_2(\omega_i, \tau_j) = e_j$. The second part of the polynomial is

     $$\prod_{1 \le j \le N} (x - \tau_j)^{e_j}.$$

     (c) Combining the two parts together, the polynomial is

     $$P_{\omega_i} = \prod_{\sigma_j \in I_{\omega_i}} (x - t_{\sigma_j}) \prod_{1 \le j \le N} (x - \tau_j)^{e_j}$$

2. The data owner calculates a polynomial vector as follows,

$$\mathcal{I} = (P_{\omega_1}, P_{\omega_2}, \cdots, P_{\omega_m})^T$$

   (a) The data owner calculate an addition polynomial $P = \sum_{i=1}^{m} P_{\omega_i}$. We denote the encrypted polynomial as $\tilde{P} = Enc_{E_k}(P)$.

3. The data owner encrypts the coefficients of each polynomial P_{ω_i} using the public key, and sets the encrypted index as $\tilde{\mathcal{I}} = Enc_{E_k}(\mathcal{I})$.

4. The data owner constructs a dictionary matrix

$$M_D = \begin{pmatrix} t_{\omega_1}^m & t_{\omega_2}^m & \cdots & t_{\omega_m}^m \\ t_{\omega_1}^{m-1} & t_{\omega_2}^{m-1} & \cdots & t_{\omega_m}^{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ t_{\omega_1} & t_{\omega_2} & \cdots & t_{\omega_m} \end{pmatrix}.$$

5. Finally, the data owner outsources the matrix M_D, the index polynomial vector $\tilde{\mathcal{I}}$ and $\tilde{P}$ to the cloud.

- **TrapdoorGen**$(MK, \mathcal{Q})$: Before generating the trapdoors, the data owner first generates a polynomial for the entire dictionary P_D as

$$P_D = \prod_{\omega_i \in \Omega} (x - t_{\omega_i})$$

  Note that this is a one-time cost.

  When the data owner receives a query request $\mathcal{Q}$ from a user, he constructs a polynomial $P_{\mathcal{Q}} = P_D / \prod_{\omega_i \in \mathcal{Q}} (x - t_{\omega_i})$. Then, the data owner generates $P'_{\mathcal{Q}}$ by padding random terms to $P_{\mathcal{Q}}$.

$$P'_{\mathcal{Q}} = P_{\mathcal{Q}} \prod_{q+1}^{m} (x - r_j), q = |\mathcal{Q}|, r_j \notin f(\Omega)$$

  We represent $P'_{\mathcal{Q}}$ using its coefficients (a_m, a_{m-1}, a_1, a_0). Finally, the data owner returns the trapdoor which is a 2-tuple to the user as

$$\mathcal{T}_{\mathcal{Q}}[1] = (Enc(a_m), \cdots, Enc(a_1)), \mathcal{T}_{\mathcal{Q}}[2] = a_0$$

  The data owner also returns f_2,$\{\tau_i\}_{i=1}^{N}$ as the verification token.

- **Query**$(\tilde{\mathcal{I}}, \mathcal{T}_{\mathcal{Q}})$: After receiving the trapdoor, the cloud server first calculates

$$V = T_{\mathcal{Q}}[1] \cdot_h M_D = (v_1, v_2, \cdots, v_m).$$

and

$$\tilde{P}_R = V \cdot_h \tilde{\mathcal{I}}^T +_h T_{\mathcal{Q}}[2] \times_h \tilde{P} \tag{3.3}$$

where $\cdot_h$ is to perform matrix multiplication using homomorphic addition and multiplication.

and returns $\tilde{P}_R$ back to the user.

- **Verify**$(\tilde{P}_R, f_2)$: After receiving the $\tilde{P}_R$, the user first calculates $P_R = Dec_{D_k}(\tilde{P}_R)$.
  1. The user factors P_R and finds the roots of $P_R = 0$. The root set exclude the verification tokens is the document IDs of the search result.
  2. For each verification token $\tau_i, 1 \leq i \leq N$, the user computes $e_i = \min\ f_2(\omega_j, \tau_i),\ \omega_j \in \mathcal{Q}$. At last, the algorithm output 1 if P_R can be divided by $\prod_{1 \leq i \leq N}(x - \tau_i)^{e_i}$; otherwise, $\perp$.

Discussion. The number of the verification tokens is pre-defined in our scheme, i.e., N. The number N determines the probability of the cloud server being able to bypass the verification with an incorrect search result successfully. Obviously, the more verification tokens, the harder the cloud can defeat the verification process. The exponent e_i for the verification token τ_i also affects the verification probability. On the other side, the verification terms in the index increase the degree of the polynomial which leads to additional storage and computation overhead. Therefore, there is a trade-off between the scheme efficiency and the scheme security when choosing the parameters. We delay the discussion of the trade-off in section 3.6.

3.5 Secure index update protocol

3.5.1 Basic idea

Because our scheme represents a document list I_ω in the inverted index as a polynomial, $P_\omega = \prod_{\sigma_i \in I_\omega}(x - t_{\sigma_i})$ (we ignore the verification tokens for simplicity), adding a document $\sigma^\star$ to I_ω is to multiply P_ω with $(x - t_{\sigma^\star})$. In practice, we can calculate the multiplication of two polynomials in a three-step method. First, we use Discrete Fourier Transform (DFT) to convert the two polynomials to their point-value representations. Then we calculate the point-wise multiplications between the two point-value representations. At last, we use inverse DFT to transfer the point-value representation to its coefficient representation which is the product of the two polynomials. Therefore, we can utilize the secure FFT algorithm to perform the document addition.

Similarly, to delete a document $\sigma^\star$ from I_ω is equivalent to divide P_ω by $(x - t_{\sigma^\star})$. The polynomial division can be achieved through polynomial multiplication using the Newton iteration. Therefore, we can also use the secure FFT algorithm to perform the index update for document deletion.

3.5.2 Document set update

We consider two operations of document update, i.e., the document addition and the document deletion. The document modification can be achieved through a two-step operation by deleting the original document first and then adding the modified document as new to the document set. We focus ourselves on the discussion of the update on an existing document lists because a new document list can be generated following the secure index generation process.

Setup

Definition 5 (Root of unity)**.** *Let R be a ring, $n \in \mathbb{N}_{\geq 1}$, and $u \in R$.*

1. *u is an nth root of unity if $u^n = 1$.*
2. *u is a primitive nth root of unity if it is an nth root of unity, n is a unit in R, and $u^{n/t} - 1$ is not a zero divisor for any prime divisor t of n.*

The following lemma says when primitive roots of unity exist in a finite field $\mathbb{F}_q$ with q elements.

Lemma 2. *For a prime power q and $n \in \mathbb{N}$, a finite field $\mathbb{F}_q$ contains a primitive nth root of unity if and only if n divides $q - 1$.*

The lemma has been proved in [76]. We refer readers to Chapter 8.2 in [76] for detail.

To perform the FFT algorithm, we require that the domain of the additive homomorphic encryption algorithm contains a primitive nth root of unity. However, as shown in Chapter 8.3 in [76], the FFT can be performed over any commutative ring. In practice, most additive homomorphic encryption algorithms satisfy the requirement. To simply our description, we present our protocol based on an additive encryption algorithm of which the domain of the plaintext contains a primitive nth root of unity. We denote the degree of a polynomial f as deg_f through the remaining content.

Adding a document

When adding a document σ, we first extract all the keywords in the document, denoted as Ω_σ. Then for each keyword $\omega \in \Omega_\sigma$, we add σ into the corresponding document list, i.e.,

I_ω. Recall that each of the document lists of our secure index is represented as an encrypted polynomial. To simplify our presentation, we denote the polynomial of the document list I_ω as $f(x)$ and represent the newly added document using the polynomial $g(x) = x - t_\sigma$, where t_σ is the document tag for σ. The document addition protocol works as follows.

- The data owner performs the following computation.
  1. Calculate
     $$\{f(1), f(u), \cdots, f(u^n)\},$$
     and
     $$\{g(1), g(u), \cdots, g(u^n)\},$$
     where u is the primitive nth root of unity, and $n > deg_f + deg_g$.
  2. Compute a new keyword tag t'_ω for the document list I_ω.
  3. The data owner calculates
     $$\Pi_{f \cdot g} = \{f(1)g(1), f(u)g(u), \cdots, f(u^n)g(u^n)\}.$$
  4. Send $\Pi_{f \cdot g}$, u, and t'_ω to the cloud.
- The cloud performs the following computation.
  1. The cloud runs the encrypted FFT algorithm over $(\Pi_{f \cdot g}, u)$ to compute the new encrypted polynomial $f'(x)$.
  2. The cloud updates $\tilde{P}$ as $\tilde{P} -_h f(x) +_h f'(x)$ and replaces $f(x)$ with $f'(x)$ in $\tilde{\mathcal{I}}$.
  3. The cloud replaces t_ω with t'_ω in M_D.

It is worth mentioning that the algorithm returns a n-length vector to represent the result polynomial. Although $n > deg_f + deg_g$, the coefficients of x^i for all i that $n \geq i > deg_f + deg_g$ will be 0. It is known that the FFT algorithm can be used to calculate the inverse FFT, which we briefly show in the appendix. Therefore, the cloud is able to calculate the updated document list.

Deleting a document

Deleting a document σ is to perform a polynomial division. We utilize the Newton's iteration trick to convert polynomial division to polynomial multiplication. The same technique has also been used in [73, 77].

Given a polynomial $f(x)$, we define its *reverse* $rev_d f(x) = x^d f(1/x)$ where d is the degree of f. It is easy to see that

$$rev_d f(x) = a_0 x^d + \cdots + a_{d-1} x + a_d.$$

We focus on the case where the dividend polynomial $g(x) = b_k x^k + \cdots + b_0$ of degree $k < d$ is a *monic* polynomial, i.e., $b_k = 1$. Then we have

$$f(x) = q(x) \cdot g(x) + r(x)$$

and

$$rev_d f(x) = rev_{d-k} q(x) rev_k g(x) + x^{d-k+1} rev_{k-1} r(x).$$

We can re-write the above equations as

$$rev_d f(x) \equiv rev_{d-k} q(x) rev_k g(x) \mod x^{d-k+1},$$

and

$$rev_{d-k} q(x) \equiv rev_d f(x) rev_k^{-1} g(x) \mod x^{d-k+1},$$

where $rev_k^{-1} g$ is the inverse of $rev_k g$. Because g is monic, $rev_k^{-1} g$ exists. Now we have convert a polynomial division to a polynomial multiplication. Note that we are only interested in the quotient. The document deletion protocol is as follows.

- The data owner performs the following computation.
  1. First compute
  $$\{rev_d f(1), rev_d f(u), \cdots, rev_d f(u^n)\},$$
  and
  $$\{rev_k^{-1} g(1), rev_k^{-1} g(u), \cdots, rev_k^{-1} g(u^n)\},$$
  where u is the primitive nth root of the unity.
  2. Then calculate $\Pi_{rev_d f \cdot rev_k g}$ as
  $$\{rev_d f(1) \cdot rev_k^{-1} g(1), \cdots, rev_d f(u^n) \cdot rev_k^{-1} g(u^n)\}.$$
  3. Compute a new keyword tag t'_ω for the document list I_ω.
  4. Finally send $\Pi_{rev_d f \cdot rev_k^{-1} g}$, u, and t'_ω to the cloud.
- The cloud performs the following computation.
  1. The cloud runs the encrypted FFT algorithm over $(\Pi_{rev_d f \cdot rev_k^{-1} g}, u)$ to compute the new encrypted polynomial $f'(x)$.
  2. The cloud updates $\tilde{P}$ as $\tilde{P} -_h f(x) +_h f'(x)$ and replaces $f(x)$ with $f'(x)$ in $\tilde{\mathcal{I}}$.
  3. The cloud replaces t_ω with t'_ω in M_D.

3.6 Security and performance analysis

In this section, we first prove the correctness and the security of our scheme. Then we analyze the computation performance of the proposed scheme.

3.6.1 Correctness

In our scheme, we represent each document list as a polynomial. When the context is clear, we may use document list to refer the polynomial.

Theorem 3 (Completeness)**.** *Our scheme returns all the documents that contain the query keyword(s).*

Proof. We represent the query trapdoor as a polynomial $P'_{\mathcal{Q}}(x)$ so that the roots of $P'_{\mathcal{Q}}(x) = 0$ are the tags of the keywords except those in the query, i.e. $\{x | x = t_{\omega_i}, \omega_i \notin \mathcal{Q}\}$.

Then, the result polynomial P_R which is

$$\begin{aligned} & V \cdot_h \tilde{\mathcal{I}}^T +_h T_{\mathcal{Q}}[2] \times_h \tilde{P} && (3.4) \\ = & \sum v_i P_{\omega_i} + a_0 \sum P_{\omega_i} && (3.5) \\ = & \sum (v_i + a_0) P_{\omega_i} && (3.6) \\ = & \sum P'_{\mathcal{Q}}(\omega_i) P_{\omega_i} && (3.7) \\ & && (3.8) \end{aligned}$$

Clearly, P_R consists of the query-related inverted lists only. The roots of equation 3.4 actually is $\cap_{\omega_j \in \mathcal{Q}} \Sigma_{\omega_j}$, which is the set of the documents that contain all the query keywords. Therefore, the user will fetch the correct documents from the cloud. □

Verification process. Because the verification tokens are included in every document list polynomial, $V_{\text{Token}} = \prod_{1 \le i \le N} (x - \tau_i)^{e_i}$ must be a common factor for all the polynomials $P_{\omega_i}, \omega_i \in \mathcal{Q}$ where $e_i = \min\ f_2(\omega_j, \tau_i)$. Clearly, V_{Token} can divide the search result polynomial P_R if P_{ω_i} is included in P_R. On the other hand, if P_R is divisible by V_{Token}, we have the following claim.

Theorem 4. *Given a negligible number ϵ, there exists a number m such that the probability of an adversary to forge a verification token V_{Token} that can divide P_R is bounded by ϵ.*

Proof. Each verification token $\tau_i, 1 \le i \le m$ is selected uniformly from a finite field with size M. Because f_2 is a pseudo-random function, the distribution of the exponent e_i for each keyword-token pair (ω, τ_j) is also uniform. We denote N as the size of the range of f_2. Therefore, the probability of the cloud to forge a V_{Token} for a query with q keywords to bypass the verification process is

$$\text{Pr}_{\text{forge}} = \frac{(M-m)!}{M!} \cdot \frac{1}{N^m}.$$

Because Pr_{forge} decreases exponentially as m increases, we can find a m such that $\text{Pr}_{\text{forge}} \le \epsilon$.

On the other hand, it is possible that two search results P_{R_1}, P_{R_2} share a same V_{Token}. Since P_{R_1}, P_{R_2} correspond to queries Q_1, Q_2, respectively, we analyze the probability based on the query. There are two cases.

- $Q_1 \cap Q_2 = \emptyset$. Each $(x - \tau_j)^{e_j}, 1 \leq j \leq m$, is shared by P_{R_1}, P_{R_2}. The probability of two search results has the same e_j for τ_j is $\frac{q_1 q_2}{N^2}$, where $q_1 = |Q_1|, q_2 = |Q_2|$. Therefore, the probability of two search result share a same V_{Token} is

$$\text{Pr}_{Q_1 \cap Q_2 = \emptyset} = \left(\frac{q_1 q_2}{N^2}\right)^m$$

- $Q_1 \cap Q_2 \neq \emptyset$. We denote $Q_1 \cap Q_2 = \triangle Q$. Thus we can represent $Q_1 = \triangle Q + \tilde{Q}_1$ and $Q_2 = \triangle Q + \tilde{Q}_2$. There are m verification tokens in V_{Token}. If Q_1, Q_2 generate a same V_{Token}, we assume m_1 verification tokens come from $\triangle Q$ and the remaining m_2 come from $\tilde{Q}_1, \tilde{Q}_2$.

  1. If there is one e_i come from $\triangle Q$, there exists one keyword in $\triangle Q$ that generates the smallest exponent for τ_i among all the query keywords, i.e., $Q_1 \cup Q_2$. The probability of m_1 verification come from $\triangle Q$ is

$$\text{Pr}_{\triangle Q} = \left(\frac{1}{N}\sum_{x=1}^{N}\left(\frac{N-x}{N}\right)^{(\tilde{q}_1 + \tilde{q}_2)}\right)^{m_1}$$

  2. For the remaining $m - m_1$ verification tokens, we can re-use the equation in the first case.

$$\text{Pr}_{\tilde{Q}_1 \cap \tilde{Q}_2 = \emptyset} = \left(\frac{\tilde{q}_1 \tilde{q}_2}{N^2}\right)^{m - m_1}$$

  Therefore, the probability of the second case is

$$\text{Pr}_{Q_1 \cap Q_2 \neq 0} = \sum_{m_1 = 1}^{m} \text{Pr}_{\triangle Q} \cdot \text{Pr}_{\tilde{Q}_1 \cap \tilde{Q}_2 = \emptyset}$$

  Although there is no close form for this case, it is obvious that $\text{Pr}_{Q_1 \cap Q_2 \neq 0}$ also decreases as m increases.

□

3.6.2 Security analysis

Before we prove the security of our basic scheme, we first review the definition of *semantic security.*

Definition 6 (Semantic security)**.** *A cryptosystem is semantically secure if given the ciphertext of a message Msg, any probabilistic polynomial-time algorithm (PPTA) cannot deduce any partial information about Msg computationally with a high non-negligibly probability.*

Similar to the semantic security requirement, we need to ensure that the secure index does not reveal any information about the keywords. The security of the scheme can be defined through the following game which is played between a challenge and an adaptive adversary who can obtain trapdoors for any query of his choice. We follow the security definition of the searchable encryption in [25]. The following game is used to define the security.

Setup

The challenger $\mathcal{C}$ creates a set of keywords Ω. Then $\mathcal{C}$ chooses a number of subsets from Ω as the file collection. The collection of the subsets is denoted as Σ. The challenger $\mathcal{C}$ first runs **Setup** to generate the key, then creates the secure inverted index for Σ using **IndexGen**. Finally, $\mathcal{C}$ publishes the keyword set Ω and the secure indexes $\tilde{\mathcal{I}}$ to the adversary $\mathcal{A}$.

Queries

$\mathcal{A}$ is allowed to request the trapdoor T_Q for a query $Q \subset \Omega$ from $\mathcal{C}$. $\mathcal{A}$ can perform **Query** using T_Q over the secure index to get the $\tilde{P}_R$.

Challenge

At some point, $\mathcal{A}$ chooses two non-empty queries $V_0, V_1 \subset \Omega$ such that $|V_0 - V_1| \neq 0$ and $|V_1 - V_0| \neq 0$. Then send them to $\mathcal{C}$.

After receiving V_0, V_1, $\mathcal{C}$ chooses $b \overset{R}{\leftarrow} \{0, 1\}$ and generates the trapdoor T_b for V_b. $\mathcal{A}$ is allowed to use T_b to get the search result $\tilde{P}_R$ for T_b.
The challenge for $\mathcal{A}$ is to decide b. After the challenge is issued, $\mathcal{A}$ can continue request trapdoor.

Response

$\mathcal{A}$ outputs a bit b' as his guess of b. The advantage of $\mathcal{A}$ in winning the game is defined as

$$\mathrm{Adv}_{\mathcal{A}} = |\Pr[b = b'] - 1/2|.$$

Definition 7 (Semantic security of searchable encryption)**.** *A searchable encryption scheme is semantic secure if no probabilistic polynomial-time adversary can win the above game with non-negligible advantage.*

The underlying security significance of Definition 7 is that any probabilistic polynomial-time adversary (PPTA) cannot determine any partial information of the documents with the encrypted index and the trapdoors. We assume the probabilistic additive homomorphic encryption scheme is a semantic secure cryptosystem. Many additive homomorphic encryption algorithms satisfy the above assumption, for example, the Paillier homomorphic encryption [71] and the ElGamal [54]. Now we are ready to prove the security of our scheme.

Theorem 5 (Security). *Our inverted index based public-key searchable encryption scheme is semantically secure if the underlying additive homomorphic encryption algorithm is a semantic secure cryptosystem.*

Proof. The security of our scheme relies on the semantic security of the additive homomorphic encryption algorithm. Assume that the a polynomial-time algorithm $\mathcal{A}$ can win the game with a non-negligible advantage. We can construct an algorithm $\mathcal{B}$ with $\mathcal{A}$ to break the semantic security of the encryption algorithm under the random oracle model. The algorithm $\mathcal{B}$ has the access to a random oracle $\mathcal{O}_f$ where f is either a random function or the additive homomorphic encryption algorithm. We replace the encryption in our scheme with the evaluation of $\mathcal{B}$. Then we construct the following game.

Setup

The algorithm $\mathcal{B}$ creates a set of keywords Ω. Then $\mathcal{B}$ chooses a number of subsets from Ω as the file collection. The collection of the subsets is denoted as Σ. The challenger $\mathcal{B}$ first runs **Setup** to generate the key, then creates the secure inverted index for Σ using **IndexGen**. Finally, $\mathcal{B}$ publishes the keyword set Ω and the secure indexes $\tilde{\mathcal{I}}$ to the algorithm $\mathcal{A}$.

Queries

$\mathcal{A}$ is allowed to request the trapdoor T_Q for a query $Q \subset \Omega$ from $\mathcal{B}$. $\mathcal{A}$ can perform **Query** using T_Q over the secure index to get the $\tilde{P}_R$.

Challenge

At some point, $\mathcal{A}$ chooses a non-empty query $V_0 \subset \Omega^\star$ and another non-empty query $V_1 \subset \Omega$ such that $|V_0 - V_1| \neq 0$ and $|V_1 - V_0| \neq 0$.

$\mathcal{B}$ chooses $b \overset{R}{\leftarrow} \{0, 1\}$ and generates the trapdoor for V_b
The challenge for $\mathcal{A}$ is to decide b. After the challenge is issued, $\mathcal{A}$ can continue request trapdoor.

Response

$\mathcal{A}$ outputs a bit b' as his guess of b.

If $\mathcal{A}$ outputs 0, then $\mathcal{B}$ guesses f used in $\mathcal{O}_f$ is a random function, denoted as $\mathcal{B}_f = 0$. Otherwise, $\mathcal{B}$ guesses f is the encryption algorithm, denoted as $\mathcal{B}_f = 1$.

Clearly, if f is a random function, the probability $Pr[\mathcal{B}_f = 0] = 1/2$. If f is the encryption algorithm, $\mathcal{B}$ has the same probability as $\mathcal{A}$ to output 1. Therefore, $\mathcal{B}$ has the same advantage to distinguish a semantic secure encryption algorithm from a random oracle as $\mathcal{A}$ to win the security game. However, according to the definition of the semantic security cryptosystem, $\mathcal{B}$ does not exist. Therefore, there exists no algorithm $\mathcal{A}$ to win the security game with non-negligible probability. $\square$

Discussion. Linkability of the trapdoors: In $T_{\mathcal{Q}}[1]$, the coefficients are encrypted. In $T_{\mathcal{Q}}[2]$, a_0 is not encrypted. Because of the random terms in the trapdoor, a_0 will be different each time. As a result, a plaintext query will be transferred to different trapdoors every time due to the introduced randomness.

Forward & backward privacy

We denote an update as $up = (op, \sigma, \omega)$ where $op \in$\{add,del\} is the operation, σ is the newly added document ID, and $\omega \in \Omega$ is used to identify the targeted document list of the operation.

Because the cloud server performs the same operation for both document addition and deletion, i.e., computing the inverse EncFFT, it is impossible to learn the op for the cloud server. Our update protocol also hides the document ID tag, i.e., t_σ, from the cloud server. However, since our update protocol performs on each document list, it leaks the keyword tag set of which is contained in the updated document.

Forward privacy. As we update both the document list and the keyword tag, the previously searched trapdoors cannot apply to the updated secure index. Therefore, our update protocol preserves the forward privacy.

Backward privacy. The later query result, i.e., the result polynomial, will not contain the deleted document term because the delete process remove the term from the secure index. Therefore, our update protocol preserves the backward privacy.

3.6.3 Complexity analysis

In the *Setup* phrase, the data owner needs to generate the key pair for the additive homomorphic encryption algorithm. It takes two exponentiations plus the computation of finding the prime numbers.

In the *IndexGen* process, there are m polynomials need to be encrypted. Each of them has the degree of L. Since the polynomial is represented using its coefficient, we need $m \times L$ encryption operations in total.

In the *TrapdoorGen* process, the data owner needs to perform: 1) a polynomial division which can be efficiently done through the synthetic division method and 2) m exponentiation to encrypt $T_{\mathcal{Q}}[1]$.

Table 3.1: Complexity comparison among the existing schemes

	Setup	Index	Trapdoor	Search
Baek et al. [66]	M	nm(E+M+P+2e)	P+M	nm(M+e)
Rhee et al. [67]	2E	nm(2E+P+e)	2E+2P	nm(2E+P+e)
Zhao et al. [68]	M	nm(4M+P+2e)	3M+4P+e	nm(2M+P+4e)
Li et al. [60]	9mM	n(m^2+3m)M	(m^2+3m)M	(m+3)e
Sun et al. [64]	(3n+1)E+e	m(n+2)E	(2n+1)E	ne+nM+E
Ours	2E	mLE	mE	($m^2 + L$)E+LM

m is the size of the dictionary, n is the size of the document set, L is a constant decided by the dataset.

In the *Query* process, the cloud server needs to perform m^2 exponentiation to calculate $\mathcal{V}$, $m \times L$ exponentiations and L multiplications to calculate $\tilde{P}_R$.

Comparison with the existing public key based schemes: We compare the computational complexity of our scheme with the existing public-key based schemes [60, 64, 66–68] in Table 3.1. Let E denote an exponentiation operation, M denote a multiplication and e denote a pairing operation and P denote a map-to-point hash function which hashes any input to the bilinear paring group G_1 (such operations are not efficient).

In most cases, the number of the documents are greater than the number of the keywords, i.e. $n \gg m$. Pairing operations are far more expensive than multiplication and exponentiation. The comparison table shows that the computation overhead of our scheme is light-weight. It is worth mentioning that although our scheme requires more computation when generating a trapdoor than other schemes, we support multi-keyword conjunctive search while none of them has this functionality.

3.6.4 Document update computation

The update computation consists of the owner part computation and the server part computation. The owner needs to perform n multiplications to compute the product of the two polynomials using their point-value representation. The owner also needs to encrypt the product of the polynomials before sending it to the server. The encryption takes $O(n)$ exponentiation. On the server side, the computation is carried out over the ciphertext. The complexity of EncFFT is $O(n \log n)$ regarding exponentiation operation.

3.6.5 Simulation

We evaluate the performance of our proposed scheme through a series of simulations. We choose the Paillier homomorphic encryption algorithm to implement our scheme. We implement the prototype of our scheme using JAVA on a Windows 8.1 PC with Intel Core i3 3.3 GHz and 4 Gigabyte memory.

Dataset: we use two data sets to perform the evaluation. The first data set is a simulated data set and we use part of the Enron spam email data set as the second one. The Enron email data contains a large number of documents. The statistic of the Enron data set is shown in Table 3.3 (the keyword set excludes the stop words and has been processed using Porter's stemming algorithm).

Inverted index encryption

There are four factors that affect the computation time of the index generation process, i.e., the size of the dictionary, the length of each document list, the number of the verification tokens, and the range of the exponent for each verification token. Thus, we evaluate the performance of the proposed scheme under four different parameter settings. Denote $|\Omega|$ as the size of the dictionary, N as the size of the verification tokens, $range(f_2)$ as the range of the verification exponent, and L as the length of the document list. The parameter setting for each simulation is as follows.

- *Size of Dictionary.* Fix $N = 50, range(f_2) = 100, 500 \leq L \leq 1000$. Increase $|\Omega|$ from 1000 to 5500.
- *Size of Verification Token.* Fix $|\Omega| = 5000, range(f_2) = 100, 500 \leq L \leq 1000$. Increase N from 10 to 100.
- *Range of Verification Exponent.* Fix $|\Omega| = 5000, N = 50, 500 \leq L \leq 1000$. Increase $range(f_2)$ from 100 to 190.
- *Size of Document List.* Fix $|\Omega| = 5000, N = 50, range(f_2) = 100$. Increase the upper bound of L from 1000 to 1900.

We show the computation time with different parameters in Fig. 3.6. The computation of the index generation process mainly comes from the encryption of the polynomial coefficients. As shown in the figure, the major factor that affects the index generation computation time is the size of the dictionary because most of the coefficients are generated by the document lists. Increase the size of the dictionary is the same as increase the number of the document lists. The number of the verification tokens and the range of the verification exponent also have a great impact on the index generation computation as these factors increase the number of the coefficient exponentially. On the other hand, increasing the length of the document list

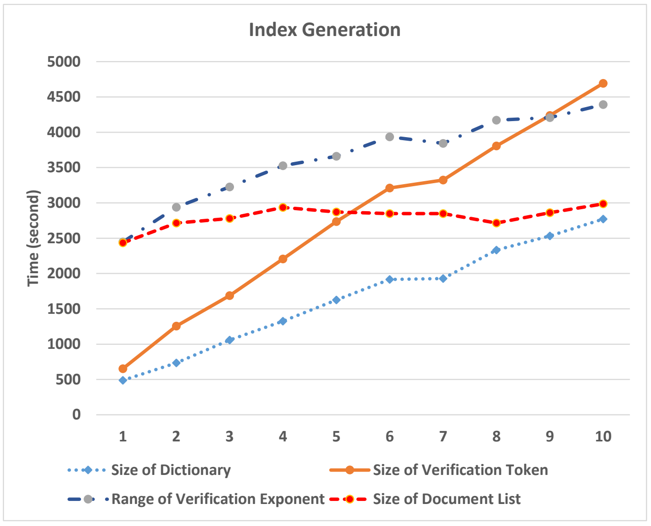


Figure 3.6: Index generation time for different parameter settings

only has a small impact because the increase takes only a very small portion compared with the total number of the coefficients.

Trapdoor generation

The trapdoor generation process involves encryption of the query polynomial only. Therefore, the degree of the query polynomial is the only factor that affects the computation time. As we construct the query polynomial in the way that the degree of it is solely determined by the size of the dictionary, we show the trapdoor generation cost respect with the dictionary size in Fig.3.7.

Search computation

The search time is shown in Fig.3.8. The computation is affected mainly by the size of the dictionary because of the matrix multiplication. On the other hand, the length of each document list also impacts the search time as the longer the document list is, the more homomorphic multiplications are needed. The search computation can be further reduced if multiple threads are used by the cloud server which also has more computation power than our proof-of-concept environment.

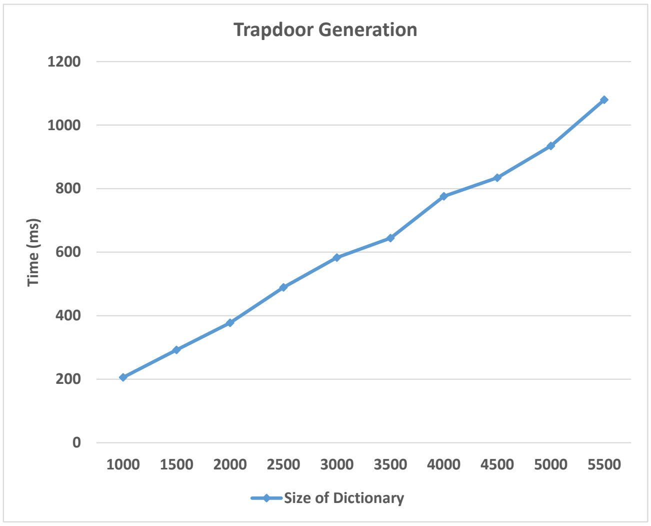


Figure 3.7: Trapdoor generation time respect to the size of the dictionary

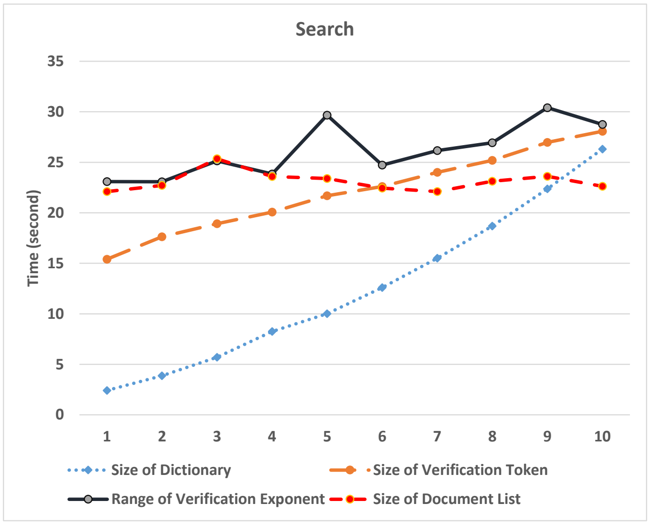


Figure 3.8: Search time for different parameter settings

Update computation

In this simulation, we update the entire secure index to present the worst case computation cost. The update process involves both the data owner and the cloud server. We show the computation cost at the user side and the server side in Fig. 3.9 and Fig. 3.10, respectively. At the user side, the computation involves a fixed number of multiplications of two polynomials in the value-point form. Since the number of multiplication is fixed, i.e., n (we set n=4096 in our simulation, which is large enough to handle all document list update), the

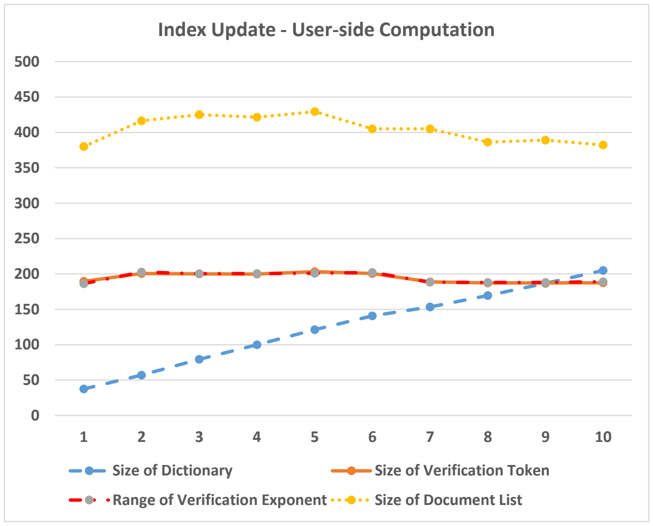


Figure 3.9: Update computation at user side

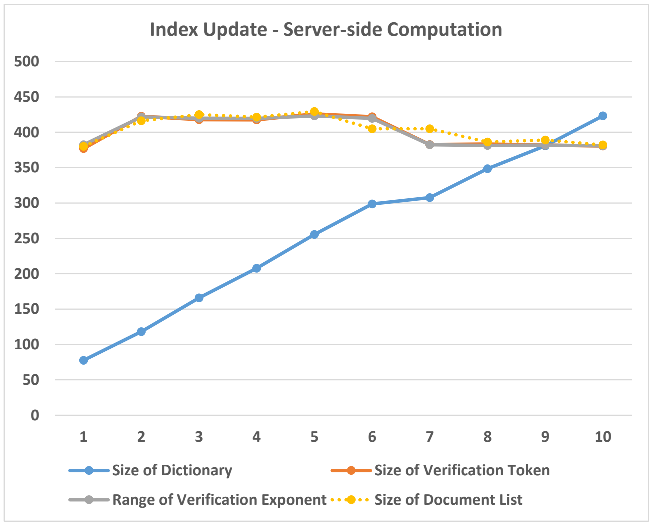


Figure 3.10: Update computation at server side

computation is mainly determined by the size of the document dictionary. Because the input of the server-side computation is the output of the user-side computation, the server-side computation is mainly determined by the number of the user-side outputs, i.e., the number of the dictionary.

Simulation results using the Enron data set

In this simulation, we use Lucene [5] to generate the inverted index for the Enron data set. The inverted index contains 5094 document lists. The total length of the document lists is 230618. The total length of the secure index, i.e., the encrypted document lists, is 12878368. We show the computation cost for the Enron data set in Table 3.2.

Table 3.2: Computation on Enron data set

# of Doc	Index generation	Update: user-side	Update: server-side	Search
5097	241.6 s	177.2 ms	395.5 ms	24 s

Table 3.3: Dataset Statistic

# of Documents	2000	4000	6000	8000	1000	12000
# of Keywords	1517	2506	3253	3934	4572	5097

3.7 Summary

In this chapter, we proposed a novel construction of a public-key searchable encryption scheme based on the inverted index. Compared with the existing works, our scheme achieves stronger privacy guarantee by lifting the one-time-only search limitation, protecting the search pattern and the access pattern, providing both forward privacy and backward privacy for secure index update. The proposed scheme also advances in functionality. First of all, the proposed scheme supports genetic multi-keywords conjunctive search which is left unsolved in the existing schemes. Secondly, the data owner can update the secure index when the underlining document set has changed. Finally, the proposed scheme allows the user to verify the correctness of the search result returned from the cloud server. Compared with the exiting public-key based schemes that heavily rely on expensive pairing operations, our scheme is more efficient by using only multiplications and exponentiations. We validated the practicality of the proposed scheme by implementing a prototype of the scheme and evaluated the performance using the Enron email dataset. The results showed that our scheme features reasonable index construction time for a large document set. Furthermore, our scheme scales well when handling large document set, which makes our scheme ideal for the real-world scenario.

Chapter 4

Privacy-Preserving Pattern Matching over Encrypted Genetic Data in Cloud Computing

Personalized medicine has been gaining popularity and is recognized as the health care model in the future. In this model, genetic testings are employed for selecting effective and optimal treatment based on the context of a patient's genetic information, such as DNA sequence, molecular, and cellular analysis. Compared with the traditional practice which takes a "one-size-fits-all" approach, personalized medicine is moving us to more precise, predictable and powerful health care which is customized for each patients. According to NIH [78], genetic testing, which could be used as diagnosis testing, prenatal testing, or predictive testing, is to search the existence of certain gene mutations over a patient's genetic data. For instance, certain mutations in the genes *BRCA1* and *BRCA2* are related to high risk of breast cancer [8]. By testing the existence of those gene mutations in a patient's DNA, early intervention measures can be taken to prevent breast cancer from development. Due to the massive storage and computation requirement, many genetic services including personalized medicine are outsourced to or provided by third-party service providers, for example, Google Genomics[1]. Recently, architectures of performing genetic computation in cloud computing have been proposed in [79, 80].

While it is promising to have customized health care for each, there are many security and privacy risks which could thwart its wide adoption in cloud computing. The main concern is whether the patient's genetic information is exposed to unauthorized parties during testings, especially when the testings are performed by a third-party such as cloud providers. In reality, personal genomics companies enforce their privacy policies mainly relying on legislation such as Health Insurance Portability and Accountability Act (HIPAA). However, such an approach is ineffective against information leakage caused by system failures or hackers. On

[1]https://cloud.google.com/genomics/

the other hand, de-identification, which removes or marks personal identifiers, is another widely adopted privacy-preserving technique. Unfortunately, many researchers (e.g. [11,12]) have found that attackers can re-identify participants using genetic data alongside with some public records.

A feasible and promising approach to protect the genetic data privacy is to encrypt the data before outsourcing. A challenging task is to perform genetic test over the encrypted data. An authorized party, such as doctors, should be able to perform genetic test over the encrypted DNA data while any revealed information such as the test result should leak no information about the patient to third parties such as cloud providers. Although the fully homomorphic encryption (FHE) [81] is a solution of the problem, its efficiency is still far from practical. Searchable encryption which builds secure indexes based on keywords is not applicable either because there are no explicitly given keywords in DNA sequences. Existing secure pattern matching schemes [82–86] are interactive protocols. The protocols assume the authorized party is remaining online during the entire testing process, which imposes usability and scalability issues. On the other hand, the communication overhead incurred by the interaction limits the practicability of those schemes. Another scheme [87] has limited privacy guarantee due to the leakage of the search pattern.

In this chapter, we aim to study the privacy-preserving genetic testing in cloud computing and focus on communication efficiency and strong privacy guarantee. To achieve genetic testing over the encrypted data while providing strong privacy guarantee, we adopt predicate encryption (PE) as the main cryptographic primitive. Using PE, the decryption of the ciphertext depends on not only the secret key but also a pre-defined function. However, to integrate PE into a privacy-preserving sequence matching scheme is nontrivial. To design a secure and well-functioning scheme, there are three important design challenges, i.e., 1) data structures to support sequence matching over encrypted data, 2) security and privacy mechanism to prevent information leakages such as search pattern and result privacy, and 3) protocol design to achieve efficient communication.

4.1 Related Work

Privacy-preserving computation over genetic data is always considered under two different system models. The first model assumes each party possesses its data and would like to compute certain functions over its input along with other parties' data. Secure multi-party computation is the main technique used under the first model. The second model is the secure outsourcing of computation model. In this model, the genetic data is stored in a semi-trusted party, usually in a cloud. The computation task is mainly carried out by the cloud server over encrypted data. We review the related works under each system model respectively.

4.1.1 Secure multi-party computation model

Atallah and Li [82] proposed a privacy-preserving protocol to compute the edit distance between two sequences based on dynamic programming. The protocol requires two non-colluding servers, each of them possessing one input sequence, to engage an interactive process. A secure look-up protocol is used to exchange the computation results of the servers in each iteration. Because the number of the iterations is the product of the lengths of the two input sequences, the computation, and the communication overhead are considerable. Jha *et al.* [84] improve the computation efficiency of [82]. Yet, it shares the same communication complexity since it is an iterative protocol as well. It is worth mentioning that these schemes can be applied to other problems that dynamic programming could solve. In [85], Wang *et al.* proposed a distributed framework for privacy-preserving genetic computing, which applies program specialization to partitioning genetic computation to different sensitivity levels. The expensive computation involving secure multi-party computation is only performed for the higher sensitivity level data. However, with our current knowledge, it is unclear whether the "insensitive" DNA data will be important in the future. Therefore, leaking these DNA data is not a satisfying practice. Troncoso-Pastoriza *et al.* [83] proposed a protocol to calculate edit distance with an encrypted input sequence through finite state machine (FSM). In their scheme, the server possesses the FSM for the target DNA sequence while the input of the FSM is the client's query sequence. The server and the client participate in an interactive protocol using oblivious transfer protocols to calculate the state transition of the FSM. Although the payload of each communication is smaller compared to the dynamic programming based approaches, the communication overhead is still considerable because the number of the iterations is linear in the product of the lengths of the two sequence lengths. It is worth mentioning that the computation used to generate the FSM can also be huge when the size of the sequence grows. Therefore, the poor scalability limits the usability of those schemes. In [86], Blanton and Aliasgari proposed a scheme which outsources the computation and the communication of [83] to multiple servers to improve efficiency in practice. All of the aforementioned schemes are based on interactive algorithms that require all the participants remain online during the entire process. However, in the genetic testing application scenario, it is impractical to require the patient or the authorized party such as doctors to remain online all the time as the process could be quite long for some tests. Wang *et al.* [88] proposed an privacy-preserving protocol to estimate the edit distance between two genome sequence. They estimate the edit distance by transforming the edit distance computation problem to the set intersection size approximation problem. The computation of the set intersection size is done through multi-party computation. Their scheme is extremely efficient as the edit distance computation time for two whole genome sequences can be finished in seconds with a relatively small error.

4.1.2 Secure outsourcing of computation model

Under the secure outsourcing of computation model, the cloud server is always considered as an honest-but-curious adversary. Therefore, the security objective is to perform certain kinds of computation in cloud without leaking private information.

Lu *et al.* [89] proposed a secure outsourcing scheme for genome-wide association study (GWAS). The GWAS aims to discover the association between gene mutations and certain diseases. The major computation of the GWAS is based on the statistic information of the genetic data. In [89], the computation is performed over the encrypted statistic in the cloud to protect the data privacy. Barman *et al.* constructed a privacy-preserving computation scheme to calculate the health risk based on the known association between gene mutations and the diseases. Given a patient's gene mutation information as a vector and the known association mapping with a weight between a gene mutation and a disease, the cloud computes the aggregated risk. Chen *et al.* [90] proposed a secure DNA alignment scheme utilizing a hybrid cloud. The scheme first locates an approximated location by comparing the ciphertext of the genetic data in a public cloud. Then the alignment is performed in a private cloud under the plaintext form of the genetic data. Baldi *et al.* [91] implement the secure paternity tests utilizing the private set intersection (PSI) technique. Those applications work on the short tandem repeats which are the number of the repeats of a specific nucleotides pattern. A secure index is built based on the genetic signatures, i.e., the short tandem, and the cloud calculates the occurrence for a specific short tandem over the encrypted genetic data. Ayday *et al.* [92] proposed a private DNA sequence retrieving scheme based on the order-preserving encryption. The scheme builds an index using the sequence position information instead of the genetic sequence in the SAM file. Kantarcioglu *et al.* [93] proposed a scheme to securely query an SNP database using an additive homomorphic encryption scheme. Those schemes focus on a specific application that involves little or no genetic sequence data, and thus, have different challenges compared to our problem.

Very recently, Chase and Shen [87] proposed a symmetric searchable encryption scheme supporting subsequence matching. Similar to our approach, their construction is based on suffix tree. However, the search pattern is leaked to the cloud because deterministic encryption algorithm is used in their scheme. Also, approximate sequence matching is not supported.

Secure DNA sequence matching is also related to searchable encryption which allows the user to query the encrypted documents with the encrypted keyword(s). A considerable amount of the searchable encryption schemes (e.g. [16, 17, 23, 25, 26]) has been proposed. The core technique of searchable encryption is to build a secure index for the keywords extracted from a document set. However, there is no explicitly given keyword in genetic sequence data. A possible solution is to treat meaningful subsequences as keywords. However, the amount of the possible subsequence is huge as the DNA sequence consists of millions of base pairs. On the other hand, subsequences with various length are used in the DNA sequence matching problem. Therefore, the searchable encryption schemes cannot be applied to the secure DNA sequence matching problem.

4.2 Problem formulation

Human DNA contains important genetic instructions that define and function each. The length of a DNA sequence is usually huge. For example, human DNA contains about 3 billion nucleotide base pairs. Luckily, only a small portion, i.e., 2%, is biological important as they are used to encode protein sequences, which play important roles functioning our body. According to NIH [78], genetic testing is mainly used to search for abnormal gene mutations that lead to genetic disorder. In medical practice, diagnosis testing is to search the existence of the known gene mutations that relate to certain genetic diseases. Therefore, in personalized medicine, one of the fundamental genetic computation is to perform sequence matching.

4.2.1 System model

We consider a personalized medicine model where there are four entities in the system, i.e., *Patient, Provider, Authorized Party (AP)*, and *Cloud*. The *provider* such as a laboratory or a personal genomics service company sequences the *patient*'s DNA from cell samples. The DNA sequence is then outsourced into the *cloud* in the encrypted form to protect the data privacy. When an *authorized party*, such as a doctor, would like to perform a genetic testing for the patient, he can submit a testing request to the cloud and gets the result back. To ensure the patient privacy, the testing request, i.e., a DNA sequence pattern, should be encrypted as well. The proposed system model is shown in Fig. 4.1. The patient sends his cell to the provider to get the DNA sequenced. The provider first extracts the DNA sequences and then encrypts them. The encrypted DNA data is outsourced to the cloud. When the doctor would like to perform a diagnostic testing in the patient's DNA data, he first gets a trapdoor from the patient. This enforces the patient's authorization on the test because of the sensitivity of the DNA data. After getting the trapdoor, the doctor submits it to the cloud, and the cloud performs the matching between the secure indexes and the trapdoor. The result is returned to the doctor. Finally, the doctor finishes the diagnosis after the result is decrypted. The core computation for privacy-preserving genetic testing is secure sequence matching. Based on the system model, we define secure sequence matching as follows.

Definition 8 (Secure sequence matching)**.** *A secure sequence matching (SSM) is a collection of five polynomial-time algorithms, i.e., Setup, DataEnc, ReqEnc, Search, and Assert such that*

- $(k_S, k_T) \leftarrow Setup(1^\lambda)$*: is a probabilistic key generation algorithm run by the provider.*

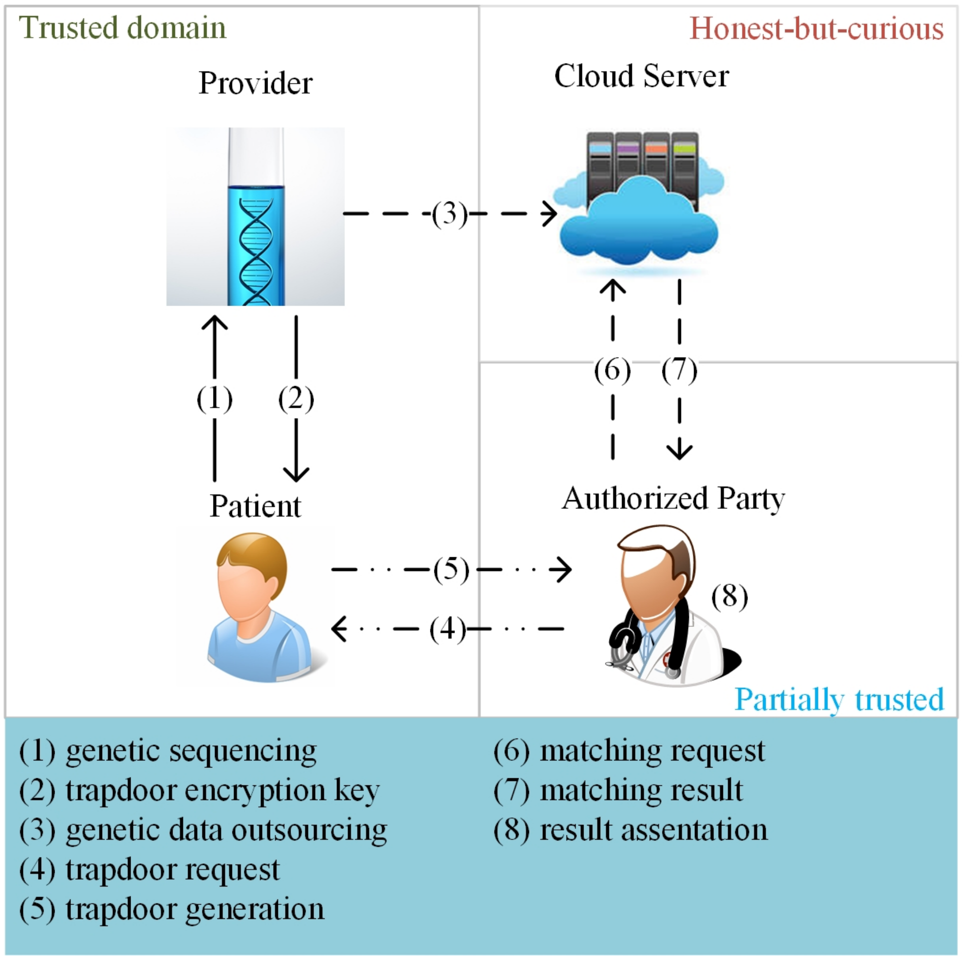


Figure 4.1: System model of personalized medicine in cloud computing.

It takes a security parameter λ *as the input and outputs a data encryption key* k_S *and a request encryption key* k_T*.*

- $(\tilde{\mathcal{S}}, \Omega_S) \leftarrow DataEnc(k_S, \mathcal{S})$*: is a probabilistic algorithm run by the provider. It takes* k_S *and a sequence* $\mathcal{S}$ *as the inputs, and outputs the ciphertext* $\tilde{\mathcal{S}}$ *and an assertion token* Ω_S*.*

- $(T, \Omega_T) \leftarrow ReqEnc(k_T, \Delta)$*: is a probabilistic algorithm run by the patient. It takes* k_T *and a query sequence* Δ *as the inputs, and outputs the trapdoor* T *as the encrypted query and the other assertion token* Ω_T*.*

- $R \leftarrow Search(\tilde{\mathcal{S}}, T)$*: is a probabilistic algorithm run by the cloud. It takes* $\tilde{\mathcal{S}}$ *and the trapdoor* T *as the inputs, and outputs the encrypted search result* R*.*

- 1 *or* $\perp \leftarrow Assert(\Omega_S, \Omega_T, R)$*: is a deterministic algorithm run by the authorized party. It takes both the assertion tokens and* R*. Output 1 if the query* Δ *is a subsequence of* *the sequence $\mathcal{S}$; otherwise, it outputs $\perp$.*

4.2.2 Security model

Because the DNA data involved in the health care application is highly sensitive, we assume secure communication channels are used to defend against the outside attacker. Therefore, we focus on the possible inside privacy leakage of the system.

Among the four entities, the patient and the provider are fully trusted because they have the original DNA data. We assume the cloud is "honest-but-curious" because of the possible compromise caused by the system failure or hacking. At last, we allow the authorized party to learn only the testing result. On the other hand, the authorized party must not be able to generate any valid trapdoors unless the testing request is granted by the patient.

4.2.3 Design objective

To achieve privacy-preserving genetic testing, the core requirement is that the cloud cannot deduce any useful information about the patient's genetic data. The requirement must be enforced even the cloud has collected an abundance of trapdoors and the corresponding matching results. We summarize the security and the performance objectives as follows.

- **Data confidentiality.** The cloud should not be able to recover any useful information from any encrypted data, which includes the encrypted genetic sequence, the encrypted request, and the encrypted matching result.
- **Trace indistinguishability.** Denote the trapdoor and its corresponding match result as a *trace*, the cloud should not be able to distinguish two traces. In other words, it is impossible for the cloud to link an encrypted request with a previously submitted one. It is also called search pattern privacy.
- **Trapdoor unforgeability.** The authorized party should not be able to forge a legit genetic testing request without the help of the patient.
- **Efficiency and usability.** As genetic testing may not require a real-time result, the computation time at the cloud side can be tolerated in certain extend. However, the computation at the patient and the authorized party side must be constrained because their computation resource is usually limited. On the other hand, the interaction among the entities should be minimized to enjoy usability.

4.3 Secure sequence matching scheme

In this section, we present our secure sequence matching scheme that fulfills the requirements of the privacy-preserving genetic testing problem. Then we further hide certain information regarding the DNA sequence to meet the higher privacy protection requirement.

4.3.1 Scheme overview

The main goal of our scheme is to securely matching two genetic sequences. The key idea of our scheme is to compare a query sequence Δ with a target sequence $\mathcal{S}$ in a character-by-character manner. If they match perfectly, we can assert that the query sequence Δ is a subsequence of $\mathcal{S}$, which confirms the existence of a genetic pattern in a patient's genetic data. Because the query pattern can appear at any position of a target sequence, we adopt the suffix tree structure to represent the target sequence. Given a sequence $\mathcal{S}$ of length n, a suffix $S_i, 1 \leq i \leq n$, is a subsequence of $\mathcal{S}$ from the position i to n. For example, S_2 of the sequence $ATGC$ is TGC. The suffices are very useful in subsequence matching because every subsequence of $\mathcal{S}$ must be a prefix of $S_i, \exists i \in [1, n]$. The suffixes of $\mathcal{S}$ are often organized as a tree which has exactly n leaves. Every edge is labeled by a subsequence of $\mathcal{S}$. The concatenation of the string-labeled edges from the root to a leaf represents a suffix of $\mathcal{S}$. To encrypt the sequence, we utilize a predicate encryption scheme [94] which supports a secure inner product computation between two encrypted sequences. The encryption algorithm is based on a composite-order of three distinct primes bilinear group. The bilinear group is defined as follows.

Definition 9. *$\mathbb{G}$ is a composite-order of three distinct primes bilinear group such that*

1. *$\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of finite order $N = pqr$, where p, q, r are distinct primes.*
2. *e is a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, i.e.,*
    - $\forall g, h \in \mathbb{G}, a, b \in \mathbb{Z}_N, e(g^a, h^b) = e(g, h)^{ab}$,
    - *if g is a generator of $\mathbb{G}$, $e(g, g)$ is a generator of $\mathbb{G}_T$ as well.*
3. *e can be calculated in polynomial time.*

However, to achieve secure and efficient sequence matching using PE, there are three main technical challenges.

1. The same character has to be encrypted differently to ensure indistinguishability while the ciphertext must be able to match correctly to properly perform genetic testing.

2. As most of the gene mutations are single-nucleotide polymorphisms (SNP), the matching algorithm should be tolerant of character mismatches to correctly handle gene mutations.

3. The character-by-character matching results must be aggregated efficiently and properly to meet the requirement of the communication efficiency.

In addition, the original PE in [94] does not support a character-by-character comparison functionality. Therefore, another challenge we need to address is to modify the scheme of [94] to support error-tolerant character-by-character comparison for two sequences.

4.3.2 Details of our scheme

First of all, we modify the original PE scheme in [94] to support our objective function. The sequence comparison function can be expressed using the following equation.

$$f = \sum_{i=1}^{n}(x_i - y_i)^2 = \sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} y_i^2,$$

where $x_i, y_i > 0$ are the characters of the sequences. Now we are ready to present our secure sequence matching scheme.

Following Def. 8, our scheme contains five algorithms, i.e., *Setup, DataEnc, ReqEnc, Search, Assert.*

- **Setup**$(1^\lambda, \mathcal{S})$ generates a public key $pk = (g_p, g_r)$ and a master secret key $msk = (p, q, r, g_q)$ given the security parameter λ. The $k_T = \{h_{1,i}, h_{2,i}\}_{i=1}^{n}$, and the $k_S = (g_q \cdot R_0, \{H_{1,i}, H_{2,i}\}_{i=1}^{n})$, where n is the length of the sequence $\mathcal{S}$, $H_{1,i} = h_{1,i} \cdot R_{1,i},\ H_{2,i} = h_{2,i} \cdot R_{2,i}$, and $R_0, R_{1,i}, R_{2,i}, \in \mathbb{G}_r$ for i from 1 to n.

- **DataEnc**$(k_S, \mathcal{S})$ encrypts each suffix $S_i, 1 \le i \le n$ of the sequence $\mathcal{S}$ and generates an assertion token Ω_S. The encrypted sequence is generated as follow,

  1. Randomly choose $s, \alpha, \beta \in \mathbb{Z}_N$.
  2. For each suffix $S_i = (x_i, x_{i+1}, \cdots, x_n)$ and $S'_i = (x'_i, x'_{i+1}, \cdots, x'_n), 1 \le i \le n$, choose random $R_{3,j}, R_{4,j}, R_{5,j}, R_{6,j} \in \mathbb{G}_r$ for j from 1 to $n - i + 1$.
  3. Encrypt each $x_j \in S_i, x'_j \in S'_i$ as a 4-tuple $(C^i_{1,j}, C^i_{2,j}, C^i_{3,j}, C^i_{4,j}) = (H^s_{1,j} \cdot k_T^{\alpha \cdot \mathcal{H}(x_j, j)} \cdot R_{3,j}, H^s_{2,j} \cdot k_S^{\beta \cdot \mathcal{H}(x_j, j)} \cdot R_{4,j}, H^s_{1,j} \cdot k_T^{-\alpha \cdot \mathcal{H}(x_j, j)^2} \cdot R_{5,j}, H^s_{2,j} \cdot k_S^{-\beta \cdot \mathcal{H}(x_j, j)^2} \cdot R_{6,j})$, where $\mathcal{H}$ is a cryptographic collision-free hash function.
  4. Denote $\tilde{S}_i = \{C^i_{1,j}, C^i_{2,j}, C^i_{3,j}, C^i_{4,j}\}_{j=1}^{n-i}$ as the ciphertext for S_i.

  The ciphertext of the sequence is $\tilde{\mathcal{S}} = \{\tilde{S}_1, \tilde{S}_2, \cdots, \tilde{S}_n\}$. The assertion token is $\Omega_S = g_p^s$.

- **ReqEnc**(k_T, Δ) encrypts the query sequence and generates an trapdoor assertion token Ω_T. Denote the character at the wildcard positions as $\star$. The trapdoor is encrypted as follow,

  1. Randomly choose $R_7 \in \mathbb{G}_r$, $f_1, f_2 \in \mathbb{Z}_q$, and $R_8 \in \mathbb{G}_q$.
  2. let $\Delta = (y_1, y_2, \cdots, y_m), m = |\Delta|$, randomly choose $r_{1,i}, r_{2,i}, r_{3,i}, r_{4,i} \in \mathbb{Z}_N$ for i from 1 to m.
  3. Encrypt each $y_i \in \Delta$ as a 4-tuple $(T_{1,i}, T_{2,i}, T_{3,i}, T_{4,i}) = (g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot 2 \cdot \mathcal{H}(y_i,i)}, g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot 2 \cdot \mathcal{H}(y_i,i)}, g_p^{r_{3,i}} \cdot g_q^{f_1}, g_p^{r_{4,i}} \cdot g_q^{f_2})$ if $y_i \neq \star$. Otherwise, encrypt $\star$ as $(T_{1,i}, T_{2,i}) = (g_p^{r_{1,i}}, g_p^{r_{2,i}}, g_p^{r_{3,i}}, g_p^{r_{4,i}})$. Generate the trapdoor assertion token Ω_T as a 2-tuple

$$(R_7 {\cdot} R_8 \cdot \prod_{i=1}^{m} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}},$$
$$e(g_q, g_q)^{-(\alpha f_1 + \beta f_2 \mod q) \sum_{i=1}^{m} \mathcal{H}(y_i,i)^2}).$$

- **Search**$(\tilde{\mathcal{S}}, T)$ generates the search result γ_i for each of $\tilde{S}_i \in \tilde{\mathcal{S}}$ for i from 1 to n as follow

$$\gamma_i = \prod_{j=1}^{m} e(C_{1,j}^i, T_{1,j}) \cdot e(C_{2,j}^i, T_{2,j}) \cdot e(C_{3,j}^i, T_{3,j}) \cdot e(C_{4,j}^i, T_{4,j})$$

- **Assert**$(\Omega_S, \Omega_T, \Gamma = \{\gamma_i\}_{i=1}^n)$ outputs 1 iff

$$e(\Omega_I, \Omega_T[1]) \cdot \gamma_i \cdot \Omega_T[2] \overset{?}{=} 1$$

  stands for at least one $i \in [1, n]$.

Discussion. In our scheme, each character is encrypted along with a different random element each time. The introduced randomness hides the relationship of the underlying characters while has no side-effect when performing sequence matching. We add wildcard support during query request encryption algorithm to handle possible gene mutations. At last, the matching result for each character is combined to a single value to enable efficient communication. Therefore, our scheme fulfills the technical challenges. Note that the *Setup* and the *DataEnc* both require the sequence as the inputs. Both the algorithms are run by the provider, and we only need the length of the sequence in the *Setup*.

4.3.3 Correctness

We first prove the correctness of our scheme. We have the following claim.

Theorem 6. *Assert outputs* 1 *if and only if the query sequence* Δ *is a prefix of at least one suffix of the sequence* $\mathcal{S}$.

Proof. Assert calculate the following equation.

$$\begin{aligned}
&e(\Omega_S, \Omega_T[1]) \cdot \gamma_i \cdot \Omega_T[2]\\
=&e(g_p^s, R_7 \cdot R_8 \cdot \prod_{i=1}^{m} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}) \cdot \prod_{i=1}^{m} e(C_{1,j}^i, T_{1,j}) \cdot e(C_{2,j}^i, T_{2,j}) \cdot e(C_{3,j}^i, T_{3,j}) \cdot e(C_{4,j}^i, T_{4,j})\\
=&e(g_p^s, \prod_{i=1}^{m} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}) \cdot \prod_{i=1}^{m} e(H_{1,i}^s k_T^{\alpha \cdot \mathcal{H}(x_i,i)} R_{3,i}, g_p^{r_{1,i}} g_q^{2 \cdot f_1 \cdot \mathcal{H}(y_i,i)}) \cdot e(H_{2,i}^s k_S^{\beta \cdot \mathcal{H}(x_i,i)} R_{4,i}, g_p^{r_{2,i}} g_q^{2 \cdot f_2 \cdot \mathcal{H}(y_i,i)}) \cdot\\
&\cdot \prod_{i=1}^{m} e(H_{1,i}^s k_T^{-\alpha \cdot \mathcal{H}(x_i,i)^2} R_{5,i}, g_p^{r_{1,i}} g_q^{f_1}) \cdot e(H_{2,i}^s k_S^{-\beta \cdot \mathcal{H}(x_i,i)^2} R_{6,i}, g_p^{r_{2,i}} g_q^{f_2}) \cdot \Omega_T[2]\\
=&e(g_p^s, \prod_{i=1}^{m} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}) \cdot \prod_{i=1}^{m} e(h_{1,i}^s g_p^{\alpha \cdot \mathcal{H}(x_i,i)}, g_p^{r_{1,i}} g_q^{2 \cdot f_1 \cdot \mathcal{H}(y_i,i)}) \cdot e(h_{2,i}^s g_p^{\beta \cdot \mathcal{H}(x_i,i)}, g_p^{r_{2,i}} g_q^{2 \cdot f_2 \cdot \mathcal{H}(y_i,i)})\\
&\cdot \prod_{i=1}^{m} e(h_{1,i}^s g_p^{-\alpha \cdot \mathcal{H}(x_i,i)^2}, g_p^{r_{3,i}} g_q^{f_1}) \cdot e(h_{2,i}^s g_p^{-\beta \cdot \mathcal{H}(x_i,i)^2}, g_p^{r_{4,i}} g_q^{f_2}) \cdot \Omega_T[2]\\
=&e(g_q, g_q)^{(\alpha f_1 + \beta f_2) \sum_{i=1}^{m} 2\mathcal{H}(x_i,i)\mathcal{H}(y_i,i)} \cdot e(g_q, g_q)^{-(\alpha f_1 + \beta f_2) \sum_{i=1}^{m} \mathcal{H}(y_i,i)^2} \cdot e(g_q, g_q)^{-(\alpha f_1 + \beta f_2) \sum_{i=1}^{m} \mathcal{H}(x_i,i)^2}\\
=&e(g_q, g_q)^{\sum_{i=1}^{m} (\alpha f_1 + \beta f_2 \mod q)(2\mathcal{H}(x_i,i)\mathcal{H}(y_i,i) - \mathcal{H}(y_i,i)^2) - \mathcal{H}(x_i,i)^2)},
\end{aligned}$$

Whether the above equation results 1 is determined by

$$\sum_{i=1}^{m} (\alpha f_1 + \beta f_2 \mod q)(2\mathcal{H}(x_i,i)\mathcal{H}(y_i,i) - \mathcal{H}(y_i,i)^2 - \mathcal{H}(y_i,i)^2 - \mathcal{H}(x_i,i)^2) \tag{4.1}$$

Clearly, if the query sequence Δ is a prefix of one of the suffix of the sequence $\mathcal{S}$, Eq. 4.1 equals 0, which means the output of *Assert* is 1.

Now we focus on the proof of sufficiency. Note that because $\mathcal{H}$ is a cryptographic collision-free hash function, the probability of $\mathcal{H}(x_i, i) = \mathcal{H}(y_j, j), i \neq j$ is negligible. If for i from 1 to m, $\mathcal{H}(x_i, i) = \mathcal{H}(y_i, i)$, i.e. $x_i = y_i$, then the above equation evaluates to 0. If there exits $i \in [1, m], h(x_i, i) \neq \mathcal{H}(y_i, i)$, i.e., the query sequence Δ is not a subsequence of $\mathcal{S}$, there are two cases. If $\sum_{i=1}^{m}(\mathcal{H}(x_i, i) - \mathcal{H}(y_i, i))^2 \neq 0 \mod q$, then the above equation evaluates to 0 with a negligible probability. The other case is that $\sum_{i=1}^{m}(\mathcal{H}(x_i, i) - \mathcal{H}(y_i, i))^2 = 0 \mod q$. However, this reveals a non-trivial factor of $N = pqr$. Because finding a non-trivial factor of N is hard according to the subgroup decision assumption, the probability of the second case is also negligible. Therefore, if *Assert* outputs 1, the query sequence Δ is a prefix of one of the suffix of the sequence $\mathcal{S}$.

Combining the proof of necessity and sufficiency, we prove that our scheme correctly reveals the fact whether a query sequence Δ is a subsequence of $\mathcal{S}$.

□

4.3.4 Security analysis

Since we utilize predicate encryption in [94] as our cryptographic primitive, we first briefly introduce its security. The security of the predicate encryption scheme is based on the following hardness assumption over a bilinear group.

Definition 10 (Subgroup decision problem). *Given two cyclic groups $\mathbb{G}, \mathbb{G}_T$ of finite order $N = pqr$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, we randomly choose*

1. $P_0 \in \mathbb{G}_p, R_0 \in \mathbb{G}_r$,
2. $T_0 \in \mathbb{G}_{pq}, T_1 \in \mathbb{G}_p$,

where $\mathbb{G}_p, \mathbb{G}_r, \mathbb{G}_q, \mathbb{G}_{pq}$ are subgroups of $\mathbb{G}$ of orders p, r, q, pq, respectively.

Given $D = (N, \mathbb{G}, \mathbb{G}_T, e, P_0, R_0, T_b)$ where $b = 0$ or 1, we define the advantage of an algorithm $\mathcal{A}$ to solve the subgroup decision problem to be

$$Adv_{\mathcal{A}} = |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|.$$

The assumption is that for any PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}$ is negligible in the security parameter. The hardness of the subgroup decision problem relies on the hardness of factoring N. The proof of the above assumption is given in [94] and we refer our readers to that paper for more detail.

We now are ready to present our formal security definition of the secure sequence search scheme. Our definition is based on an indistinguishable game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Definition 11 (Indistinguishability security for SSM). *Let SSM=(Setup, DataEnc, ReqEnc, Search, Assert) be a secure sequence match scheme. An adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ to play the following game.*

Setup

$\mathcal{C}$ creates an alphabet Σ and generates a sequence $\mathcal{S}$ from Σ. $\mathcal{C}$ first runs Setup to generate the encryption keys, then runs DataEnc to encrypt $\mathcal{S}$ as $\tilde{\mathcal{S}}$. Finally, $\mathcal{C}$ publishes $\tilde{\mathcal{S}}$.

Queries

$\mathcal{A}$ is allowed to request trapdoor for any query sequence of his choice. $\mathcal{A}$ can submit the trapdoor to $\mathcal{C}$ and gets Γ as the matching result. However, $\mathcal{A}$ is not allowed to access Assert.

Challenge

At some point, $\mathcal{A}$ chooses two non-empty query Δ_1, Δ_2 such that $\Delta_1 \neq \Delta_2, |\Delta_1| = |\Delta_2|$. Then submit them to $\mathcal{C}$.

After receiving Δ_1, Δ_2, $\mathcal{C}$ randomly chooses $b \leftarrow \{0, 1\}$ and generates the trapdoor T_b for Δ_b. $\mathcal{A}$ is allowed to search with T_b. The challenge for $\mathcal{A}$ is to decide b. After the challenge is issued, $\mathcal{A}$ may continue request trapdoors.

Response

$\mathcal{A}$ outputs her guess for b as b'. The advantage of the adversary winning the game is

$$ADV_{\mathcal{A}} = |Pr[b = b'] - 1/2|$$

If for any probabilistic polynomial-time adversary $\mathcal{A}$, the $ADV_{\mathcal{A}}$ is a negligible function of the security parameter, we say the SSM is secure in terms of indistinguishability.

We denote an SSM that satisfies the above definition as an IND-CTA (indistinguishable under chosen-trapdoor attack) SSM scheme.

Theorem 7. *Our scheme is an IND-CTA secure sequence matching scheme.*

Proof. The security of our scheme relies on the predicate encryption of [94]. We assume the predicate encryption scheme is indistinguishability under the *chosen-attribute attack* (IND-CAA). The attributes refer to the inputs of the function that is encoded in the encryption algorithm. In our scheme, the two input sequences correspond to the attributes. Therefore, the *chosen-attribute attack* is equivalent to the *chosen-trapdoor attack* under our application scenario. Due to the page limitation, we refer our readers to [94] for the proof detail of that assumption.

Assume that $\mathcal{A}$ can win the above security game with a non-negligible advantage $ADV_{\mathcal{A}}$, we can construct a probabilistic polynomial-time algorithm $\mathcal{B}$ such that $\mathcal{B}$ can break the indistinguishability of the PE in [94]. $\mathcal{B}$ can access to a random oracle O_f, which f could be either a random function or the PE algorithm. $\mathcal{B}$ modifies the *DataEnc* and the *ReqEnc* algorithms as follows.

1. In *DataEnc*, for each $x_j \in S_i, i \in [1, n-1]$, $\mathcal{B}$ gets $(C_{1,j} = g^{r_{1,j}}, C_{2,j} = g^{r_{2,j}}, C_{3,j} = g^{r_{3,j}}, C_{4,j} = g^{r_{4,j}}), j \in [1, i]$ and $\Omega_S = g_p^{r_\omega}$ from O_f.
2. In *ReqEnc*, for each $y_i \in \Delta, i \in [1, m]$, $\mathcal{B}$ gets $(T_{1,i}, T_{2,i}, T_{3,i}, T_{4,i})$ and Ω_T from O_f.

Then $\mathcal{A}$ interacts with $\mathcal{B}$ to play the security game. $\mathcal{B}$ guesses f is the PE algorithm if $\mathcal{A}$ outputs 1; otherwise, $\mathcal{B}$ guesses f is a random function. Because $ADV_{\mathcal{A}}$ is non-negligible, $\mathcal{B}$ also has non-negligible advantage to distinguish a random function from the PE algorithm. However, as the PE algorithm is IND-CAA secure, $\mathcal{B}$ does not exist. Therefore, $\mathcal{A}$ does not exist, which means our scheme is IND-CTA secure. □

The intuition behind the proof is that anything the adversary can learn from the interaction can be learned solely from running a simulator himself. In another word, the history of the trapdoors and the search results cannot help the adversary to distinguish other trapdoors, i.e., the definition of the IND-CTA.

Trapdoor unforgeability. To forge a valid trapdoor, the authorized party must be able to forge both T and Ω_T. As T is a vector that each element corresponds to a character, the authorized party can generate a valid T of his choice. However, Ω_T is generated through the trapdoor encryption key k_T which is kept by the patient. A single Ω_T is generated for a query sequence. Recall the trapdoor generation process, a collection of random numbers are introduced. Therefore, without knowing k_T, it is impossible to forge a valid verification token.

Discussion. Our scheme reveals the length of the sequence as well as the length of the query. Although this information is usually considered as less sensitive, we could apply random padding to hide the information. Adding random padding is quite easy in our scheme. We can add 0s at the end of the sequence and the query. The adversary cannot distinguish the pseudo characters from the real ones because of the other randomness and the subgroup decision assumption.

4.4 Performance analysis and evaluation

In this section, we theoretically analyze the performance of our scheme and compare it with the secure dynamic programming solution [84] and the secure finite automation machine solution [86]. Then we perform a simulation study to illustrate the cost of our scheme using a real-world DNA data.

4.4.1 Computation complexity

Our scheme relies on the bilinear pairing operation, which is computationally expensive. We calculate the numbers of the multiplication as well as the exponentiation. The computation complexity of each algorithm is summarized as follows.

- In *DataEnc*, each suffix of the sequence $\mathcal{S}$ is encrypted. For each character in a suffix, we need 8 multiplications and 4 exponentiations. Therefore, to encrypt all the suffices requires $4n^2 + 4n$ multiplications and $2n^2 + 2n$ exponentiations, where n is the length of $\mathcal{S}$. One exponentiation is needed to generate the assertion token.
- In *ReqEnc*, $4m$ multiplications and $4m$ exponentiations are used to generate the trapdoor, where m is the length of the query sequence. $2m + 1$ multiplications are needed

for the first part of the assertion token. The second part of the assertion token requires one pairing, one exponentiation, and $m + 3$ multiplications.

- In *Search*, $2m$ pairings are needed for each suffix. Therefore, it needs $2mn$ pairings in total.
- In *Assert*, one pairing and $n + 1$ multiplications are required.

In summary, our scheme requires $O(mn)$ pairings, $O(n^2)$ exponentiations, and $O(n^2)$ multiplications. The dominance of the computation is the bilinear pairing. According to the benchmark of the public available libraries such as JPBC [95] or PBC [96], the average time for a 1024-bit discrete log security is 13 ms. Therefore, for a genetic sequence with five thousand base pairs, it takes about three hours to test a query sequence with one hundred base pairs. The computation time is within the reasonable range in practice.

In [84], Jha *et al.* proposed a secure dynamic programming scheme to compute the edit distance between two sequences. Their scheme has three phase. *Phase 0* involves no heavy computation. In *Phase 1*, one player evaluates $n \times m$ instances of Yao's secure circuit evaluation over an equality circuit. The other player needs to initiate $n \times m \times q$ 1-out-of-2 oblivious transfers, where $q = \log |\Sigma|$ and Σ is the alphabet, i.e., $\{A, T, G, C\}$ in our case. *Phase 2* needs $n \times m$ iterations. Each iteration involves one evaluation of an instance of a minimum-of-three circuit and $3 \log(m + n)$ instances of 1-out-of-2 oblivious transfer. In summary, the scheme requires $2mn$ circuit evaluations, $3mn \log(n + m) + 2mn$ oblivious transfers. As described in [84], their scheme takes about 35 seconds to compute for a (25×25) problem. Since the complexity of their scheme is linear in nm, a (5000×100) problems will take much more time.

In [83], Troncoso-Pastoriza *et al.* presented a finite automaton machine based scheme to perform error resilient subsequence testing. The key idea of the scheme is to split the inputs of the FSM to two servers and to compute the state of the FSM interactive protocol through secret sharing. The computation of each iteration is small since the process only involves a constant number of the Paillier homomorphic encryption operations [71]. However, the number of the states of the FSM grows as the size of the sequence, which increases the computation overhead as well as the storage overhead to store the transition matrix for the FSM. Additionally, it also takes extra computation to construct the FSM for a specific sequence.

Discussion. Among the three approaches, [83] incurs the least computation overhead but requires a pre-process to generate the FSM. Although our scheme uses expensive bilinear pairing operation, the computation complexity is similar to the scheme in [84]. On the other hand, we push most of the computation to the cloud which has powerful and plenty computation resources. Additionally, parallel computing techniques such as MapReduce [97] can be effortlessly applied to our search algorithm which is to compute bilinear pairing between two characters.

4.4.2 Communication consumption

Our scheme advances the existing works in the communication complexity. Our scheme only needs one round communication. The cloud returns n group elements in $\mathbb{G}_T$ to the authorized party. According to [98], a group of the size 128-bit over an elliptic curve is secure enough. Therefore, the total communication to transmit the result is small, i.e., in the magnitude of kilobytes.

The communication overhead of the scheme in [84] is high due to the interactions in *Phase 2* and the interactions of the oblivious transfer. In *Phase 0*, the communication cost is $(m+n)\log(m+n)$ bits. In *Phase 1*, there are $n \times m$ iterations. Each iteration consumes 4 bits. In Phase 2, there are $n \times m$ iteration. Each iteration contains $3\log(m+n)$ instances of 1-out-of-2 oblivious transfer. As shown in [84], for a (200×200) problem, the most efficient protocol consumes more than 360 megabytes bandwidth. Since the communication overhead is linear in the product of the size of the sequences, i.e., mn, the bandwidth cost will be increased dramatically.

In [83], the communication overhead comes mainly from the oblivious transfer. There are $O(mn)$ oblivious transfers and $O(n|Q|)$ ciphertext transfers, where $|Q|$ is the number of the state of the FSM and it is linear in n. Since the communication complexity is the same as in [84], it implies that the scheme in [83] will consume similar amount of the bandwidth as in [84].

Discussion. Comparing with the other schemes, our scheme incurs much less communication overhead. Because the bandwidth is more valuable in cloud computing compared with the computation and the storage, the communication efficiency is an important factor for the schemes to be practical. Additionally, the existing schemes are interactive protocols, which assume the both parties, i.e., the cloud and the doctor, to be online the entire process to participate the interaction. However, the assumption is not reasonable in practice and raises usability issues in the personalized medicine application scenario.

4.4.3 Simulation study

We carry out several experiments to evaluate the computation performance of our scheme. We utilize the JAVA Paring-Based Cryptography Library (jPBC) [95] to implement the proposed scheme in this paper. All the experiments are conducted on a laptop equipped with an Intel 2.6GHz processor and 8GB memory. The operating system is Ubuntu 14.04. We use the human genome data from the Ensembl project [99] in the form of SAM files. SAM file format is one of the most popular file formats to store genetic sequences. Each SAM file contains short DNA sequences of which the lengths are from hundreds to thousands of base pairs.

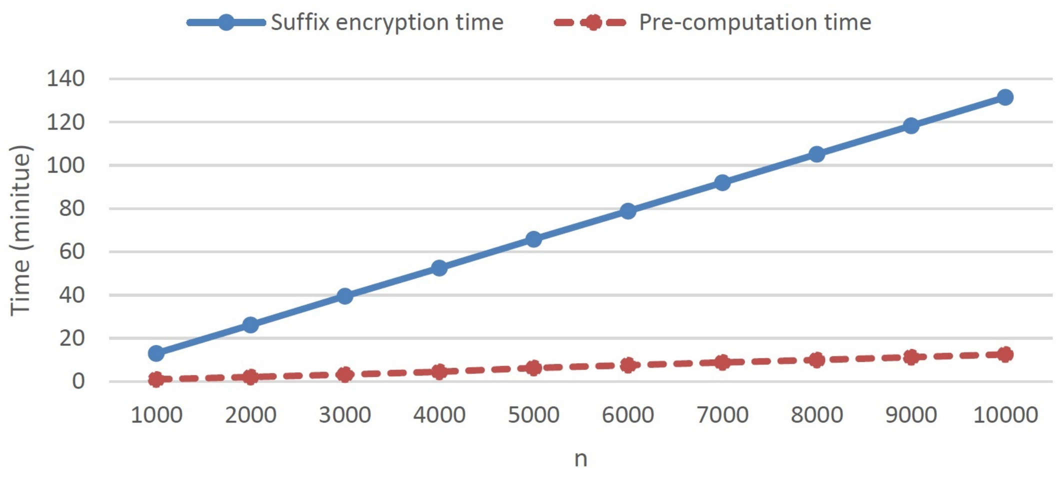


Figure 4.2: Sequence encryption time for a suffix.

Sequence encryption cost

The sequence time for a suffix is linear in the size of the suffix. Since there are n suffices in a sequence of length n, the encryption cost is quadratic in the size of the sequence $\mathcal{S}$. It is worth mentioning that the algorithm is only performed once. However, it is still computation expensive considering that the provider may have limited computation power. To that end, we further optimize the process. Note that the only terms in the ciphertext involving the input sequence is $k_S^{\alpha h(x_i,i)}, k_T^{\beta h(x_i,i)}$. Therefore, the other parts of the ciphertext can be pre-computed to reduce the computation. Meanwhile, we would like to reduce the amount of the exponentiation by using a hash function $\mathcal{H} : \Sigma \times \mathcal{Z}^+ \rightarrow \mathcal{Z}_N$, where Σ is an alphabet for the DNA base pair, i.e., {A,T,G,C}. We can encode each letter σ in Σ to a binary expression using a one-to-one mapping $\mathbb{B} : \Sigma \rightarrow \{0,1\}^{\log|\Sigma|}$. Then the h can be

$$\mathcal{H}(x_i, i) = \mathbb{B}(x_i)||F(i),$$

where $||$ is concatenation and $F : \{0,1\}^\star \rightarrow \{0,1\}^l$ is a cryptographic hash function such as SHA-1. The concatenation can be expressed as an addition as $B(x_i) \times 2^l + F(i)$. Because the size of the alphabet is limited, we can pre-compute $k_{Ind}^{\alpha \cdot B(\sigma) \cdot 2^l}$ and $k_{Ind}^{\alpha \cdot F(i)}$ for $\sigma \in \Sigma, i \in [1, n]$. Then during the index generation process, we only need to compute a multiplication instead of an exponentiation. We show the optimized encryption time in Fig. 4.2. The pre-computation time is shown as the red dash in Fig. 4.2. Note that this modification does not affect the correctness of our scheme because h utilizes a cryptographic hash function and is collision-free. We denote an alphabet of size n as Σ and define the following hash function $h : \Sigma \times \mathbb{Z} \rightarrow \mathbb{Z}_N$ as

$$\mathcal{H}(x, i) = \mathcal{B}(x)||F(i) \mod N, x \in \Sigma, i \in \mathcal{Z},$$

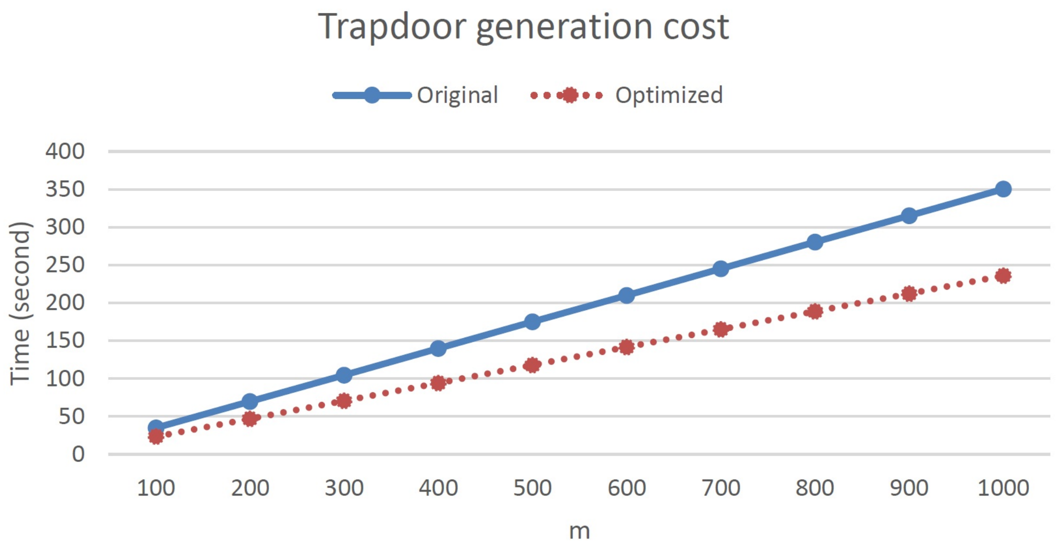


Figure 4.3: Pattern encryption time.

where $\mathcal{B} : \Sigma \rightarrow \{0,1\}^{\log n}$ is one to one mapping and $F : \{0,1\}^\star \rightarrow \{0,1\}^l$ is a collision-free cryptographic hash function. We now prove h is also a collision-free hash function.

Proof. We prove our claim by contradiction. We assume the h is not a collision-free hash function. Then there must exist $x, y \in \Sigma$ and $i, j \in \mathbb{Z}$ such that $\mathcal{H}(x,i) = \mathcal{H}(y,j)$. Because the hash result from $\mathcal{H}$ is a concatenated string, we have the following equations $\mathcal{B}(x) = \mathcal{B}(y), F(i) = F(j)$. However, because $\mathcal{B}$ is a one-to-one mapping and F is a collision-free cryptographic hash function, the above equations hold iff $x = y, i = j$, which contradicts with our assumption. Therefore, h is a collision-free hash function as well. □

Trapdoor generation cost

The trapdoor generation process is similar to the index generation process which involves only the multiplication and the exponentiation. The generation time is linear in the size of the query sequence because we encrypt each character only once. The patient can also pre-compute part of the terms in the trapdoor to optimize the computation. We show the trapdoor generation time for the both approaches in Fig. 4.3. Note that compared with the index generation which is run by the provider, the computation cost of the trapdoor is more important because the process might be run on the user's resource limited device. After optimization, our scheme generates a trapdoor for a 1000-long in about 4 minutes. Since our simulation is a proof-of-the-concept, we believe the overhead can be further reduced with the proper implementation and optimization.

Table 4.1: Pattern matching Cost in cloud

	Computation time (hours)			Cost in dollar amount		
n	m=100	m=200	m=300	m=100	m=200	m=300
500	1.724	3.510	5.311	0.25$	0.42$	0.67$
1000	3.484	6.948	10.440*	0.42$	0.84$	1.2$

The number with $\star$ is estimated value.
The dollar amount is based on the hourly rate of an Amazon EC2 M4.large instance.

Pattern matching cost

The search process is the most computation expensive algorithm due the amount of the pairing operation. As analyzed earlier, for a sequence $\mathcal{S}$ with length n and a query sequence with length m, the number of the pairing computation is linear in mn. We show the search cost in Table 4.1 when the size of $\mathbb{S}$ is fixed with $n = 500, 1000$, respectively. As shown in the table, the computation cost is linearly increasing on the size of the query sequence grows. Because the running time of the problem size (1000×300) takes more than 7 hours, we estimate that value based on our observations. Although the computation overhead seems high in our simulation, it is worth mentioning that the cloud should have better computation power than ours. For example, an Amazon EC2 M4.large instance which has 2 vCPU, 6.5 ECU and 8 GB memory costs 0.12$ for an hour. The computation power of the M4.large instance is roughly the same as our simulation platform. Therefore, the dollar amount cost to perform a matching is very cheap compared with most of the laboratory testings nowadays. Also, because the matching between the query and each suffix is independent, the computation task can be divided into multiple sub-tasks so that a parallel processing model such as MapReduce [97] is readily applied. On the other hand, in personalized medicine, the search process, i.e., the diagnosis test like a blood test, is not required to return the real-time result. Therefore, we consider our scheme is still reasonable.

4.5 Summary

Efficient secure sequence matching over the encrypted genetic data is the key challenge to design practical personalized medicine system in cloud computing. In this chapter, we propose a novel scheme to address the unique challenge brought by secure sequence matching over the encrypted data in cloud computing. We achieve wildcard-based sequence pattern matching through our novel modification of the predicate encryption scheme in [94]. Our proposed scheme is provable secure under the well-defined subgroup decision assumption. We greatly reduce the communication overhead to $O(1)$ round compared with the existing schemes. Although the computation complexity of our scheme is similar to the existing schemes, we further optimize our encryption algorithm to reduce the computation overhead. Through implementation and simulation, we show that our solution is feasible and practical.

Chapter 5

Conclusion and Future Research

In this dissertation, we have explored key points to protect user data privacy in various cloud computing applications. We designed effective and efficient secure index schemes to achieve various keyword search functions over encrypted data. We investigated secure sequence matching for genetic testings in personalized medicine. In this chapter, we summarize the research work and discuss future research directions.

5.1 Research summary

The rapid development of cloud computing has quickly moved everything into the cloud. Although cloud-based applications have been providing convenient, affordable, and productive services for its users, the privacy concern of the data stored in the cloud has also emerged as the most critical issue that hinders the further generalization of cloud computing. Although traditional cryptographic approaches have been applied for decades to provide data privacy protection, these approaches are not able to achieve the same effect in cloud computing because most of the cloud applications cannot be performed over the encrypted data. On the other hand, the success of cloud computing is driven by the economic gain from both the service provider and the end users. Therefore, it is an extremely challenging task to design efficient secure data computation schemes for cloud applications that can work over encrypted data. Our main findings and their implications can be concluded as follows.

- We address the multi-keyword fuzzy search over encrypted data problem with user data privacy protection. In contrast to previous fuzzy keyword search solutions [13–15], which require expanded storage for wild-card based fuzzy keyword set, our scheme exploits locality-sensitive hashing to provide efficient fuzzy search without expanding index files. Compared with the tag-encoding based scheme [36], our scheme provides a generic multi-keyword support which means the search result is generated through a single run of a multi-keyword trapdoor instead of multiple single keyword trapdoors. In contrast to previous solutions on multiple keywords search [38, 44], our scheme eliminates the need for a predefined dictionary and hence enables efficient file update. We theoretically analyze the trade-off between the search efficiency and the result accuracy for different choices of parameters to provide the deep insight of our scheme. We implemented our scheme and evaluated on various parameter choices, and the simulation results demonstrate that our scheme is accurate and efficient.

- We propose a dynamic searchable encryption scheme based on the inverted index. We design a probabilistic trapdoor generation algorithm to break the trapdoor linkability and to hide the search pattern, which is left unsolved in the existing works. Our scheme supports generic conjunctive multi-keyword search using only one trapdoor while the existing invert index based searchable encryption schemes only support single keyword search. In our scheme, the data owner can update the secure index when changes such as document addition and deletion have been made to the underlying document set. Also, our scheme features a probabilistic algorithm that allows the user to efficiently verify the correctness of the search result. Comparing with the existing public-key searchable encryption schemes which use expensive pairing operations, our scheme is more efficient as only multiplication and exponentiation are used. We provide a theoretical analysis and a simulation study. The simulation results show that our scheme is suitable for practical usage, and the overhead is moderate.

- We propose a novel scheme for privacy-preserving genetic testing in cloud computing. To address the genetic sequence matching challenge, we modified the predicate encryption (PE) scheme in [94] to achieve approximate sequence (wildcard-based) matching for genetic sequences. In particular, the genetic sequence is encrypted with PE and an authorized party such as doctors can submit a genetic testing request with a secure query sequence pattern to the cloud. Our scheme is provably secure under the well-defined subgroup decision assumption over a bilinear group. To provide strong privacy guarantee, search pattern and testing results are protected from the cloud server. Also, the authorized party only learns the information which the patient allows. We utilize suffix tree structure to pre-process the genetic sequence so that the sequence matching computation can be done in a single round of communication. We thoroughly analyze of the complexity of the different approaches for the secure DNA sequence matching problem. We then compare our scheme with them to illustrate the strength of our scheme in practice. We further perform a simulation study using the public available DNA data. The results show that the computation overhead for the problem is reasonable, and the communication cost is small.

5.2 Future research directions

Searchable encryption has been studied for more than a decade. A lot of schemes have been proposed. The main research directions are to provide rich search functions and to improve computation efficiency. We can further investigate the following open problems for practical searchable encryption in cloud computing.

- *Phrase search.* One of the missing functions in current secure indexes is phrase search. Current multi-keyword search schemes are able to test the existence of the query keywords but not able to tell the relative positions of the query keywords. Because of the importance of the phrase search, one of the future research objectives is to provide secure phrase search over encrypted data.
- *Result ranking* which returns a list of documents has become a necessary function as the data volume increases. Simple similarity measurement such as "coordinate matching" and "cosine similarity". Advanced similarity measurement such as normalized Google distance is more preferable as it suits better in a natural language sense. The technical challenge is to incorporate computation of the similarity score in the secure index.
- *Computation efficiency.* Finally, the computation efficiency is another research direction of searchable encryption. Symmetric searchable encryption which uses symmetric encryption algorithms is efficient but can only support a limited number of search functions. Asymmetric searchable encryption that utilizes functional encryption is more powerful on search functionality but is less efficient. Therefore, to design a secure index that satisfies both the requirements is still an open question.

We can extend our research effort in the following directions for the secure genetic sequence matching problem.

- Our secure genetic sequence works over the raw genetic data, i.e., sequence alignment/map (SAM) file format. There exists a scalability problem when the sequence matching is performed over a huge volume of DNA data. In plaintext genetic applications, compressed data format is usually used to save both storage and computation. A possible future direction is to develop secure schemes based on the compressed genetic data format such as binary alignment/map (BAM) or CRAM to improve computation efficiency.
- In the current scheme, the matching result is returned to the authorized party as the diagnosis. However, the cloud provider may return random value as the result to avoid expensive computation. To prevent such behaviors, the authorized party should be able to confirm the computation is properly performed over the intended data. An extension of the work could be to implement result verification into the scheme to detect malicious or lazy behaviors of the cloud provider.

- It is also possible to extend our scheme to support a more advanced genetic application. For instance, current scheme returns the result that indicates the existence of a particular genetic pattern. In certain scenarios such as paternity testings, the occurrence of a genetic pattern is also desired. It is worthwhile to study the security and function requirements in a broader set of application scenarios.

Chapter 6

Bibliography

[1] S. Johnston, "Cloud computing." [Online]. Available: https://commons.wikimedia.org/wiki/File:Cloud_computing.svg#/media/File:Cloud_computing.svg

[2] C. Strachey, "Time sharing in large fast computers," in *Communications of the ACM*, vol. 2, no. 7. ASSOC COMPUTING MACHINERY 1515 BROADWAY, NEW YORK, NY 10036, 1959, pp. 12–13.

[3] D. E. Standard, "National bureau of standards (us), federal information processing standards publication 46, national technical information service, springfield, va, april 1997," *Federal Register, March*, vol. 17, 1975.

[4] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pp. 19–22, 2001.

[5] Lucene, "The Lucene search engine," 2005.

[6] [Online]. Available: http://sphinxsearch.com/

[7] G. S. Ginsburg and J. J. McCarthy, "Personalized medicine: revolutionizing drug discovery and patient care," *TRENDS in Biotechnology*, vol. 19, no. 12, pp. 491–496, 2001.

[8] "Fact sheet: Brca1 and brca2: Cancer and genetic testing," April 2015.

[9] K. Wetterstrand, "Dna sequencing costs: Data from the nhgri genome sequencing program (gsp)." [Online]. Available: www.genome.gov/sequencingcosts

[10] T. Goetz, "23andme will decode your dna for $1,000: welcome to the age of genomics," *Wired Mag*, no. 15, 2007.

[11] X. Zhou, B. Peng, Y. Li, Y. Chen, H. Tang, and X. Wang, "To release or not to release: Evaluating information leaks in aggregate human-genome data," in *Computer Security âĂŞ ESORICS 2011*, ser. Lecture Notes in Computer Science, V. Atluri and C. Diaz, Eds. Springer Berlin Heidelberg, 2011, vol. 6879, pp. 607–627.

[12] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich, "Identifying personal genomes by surname inference," *Science*, vol. 339, no. 6117, pp. 321–324, 2013.

[13] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–5.

[14] C. Liu, L. Zhu, L. Li, and Y. Tan, "Fuzzy keyword search on encrypted cloud storage data with small index," *ICCCIS 2011*, pp. 269–273, 2011.

[15] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," *ICDCSW 2011*, pp. 273–281, 2011.

[16] B. Wang, S. Yu, W. Lou, and Y. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 2112–2120.

[17] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 79–88.

[18] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 965–976.

[19] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage." *IACR Cryptology ePrint Archive*, vol. 2014, p. 219, 2014.

[20] B. Wang, W. Song, W. Lou, and Y. Hou, "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee," in *INFOCOM, 2015 Proceedings IEEE*, April 2015.

[21] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," *TCC 2009*, pp. 457–473, 2009.

[22] N. Attrapadung and B. Libert, "Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation," *PKC 2010*, vol. 6056, pp. 384–402, 2010.

[23] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, Proceedings. 2000 IEEE Symposium on*, 2000, pp. 44–55.

[24] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '13. New York, NY, USA: ACM, 2013, pp. 71–82.

[25] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds. Springer Berlin Heidelberg, 2004, vol. 3027, pp. 506–522.

[26] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *INFOCOM, 2011 Proceedings IEEE*, April 2011, pp. 829–837.

[27] Y. Hwang and P. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," *Pairing 2007*, pp. 2–22, 2007.

[28] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," *Theory of Cryptography*, vol. 4392, pp. 535–554, 2007.

[29] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," *ACNS 2004*, vol. 3089, pp. 31–45, 2004.

[30] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," *ICDCS 2010*, pp. 253–262, 2010.

[31] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," *ACNS 2005*, vol. 3531, pp. 442–455, 2005.

[32] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *CCS 2006*, vol. 19, pp. 79–88, 2006.

[33] E.-J. Goh, "Secure indexes," *Cryptology ePrint Archive on October 7th*, pp. 1–18, 2003.

[34] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," *S&P 2000*, vol. 8, pp. 44–55, 2000.

[35] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *IEEE INFOCOM 2010, mini-conference*, San Diego, CA, USA, March 2010.

[36] A. Boldyreva and N. Chenette, "Efficient fuzzy search on encrypted data." *21st International Workshop on Fast Software Encryption*, 2014.

[37] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," *EUROCRYPTO 2004*, pp. 506–522, 2004.

[38] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *INFOCOM 2011*, pp. 829–837, 2011.

[39] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," *Proceedings of the 30th ACM symposium on Theory of computing*, vol. 126, pp. 604–613, 1998.

[40] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, pp. 422–426, 1970.

[41] E.-J. Goh *et al.*, "Secure indexes." *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[42] E. Shi, J. Bethencourt, T. h. Hubert, C. Dawn, and S. A. Perrig, "Multi-dimension range query over encrypted data," *S&P 2007*, pp. 350–364, 2007.

[43] Y. Lu, "Privacy-preserving logarithmic-time search on encrypted data in cloud," *NDSS 2012*, 2012.

[44] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *ASIACCS 2013*, May 2013.

[45] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," *28th International Conference on Data Engineering*, pp. 1156–1167, 2012.

[46] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings," in *SecureComm 2010*, Singapore, September 7-9 2010.

[47] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *IEEE INFOCOM 2010*, San Diego, CA, USA, March 2010.

[48] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation." in *NDSS*, vol. 20, 2012, p. 12.

[49] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia, "Privacy-preserving group data access via stateless oblivious ram simulation," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms.* SIAM, 2012, pp. 157–167.

[50] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in *Security and Privacy (SP), 2014 IEEE Symposium on.* IEEE, 2014, pp. 639–654.

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

[52] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," *SCG 2004*, 2004.

[53] S. M. Bellovin and W. R. Cheswick, "Privacy-enhanced searches using encrypted bloom filters," 2007.

[54] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in cryptology.* Springer, 1985, pp. 10–18.

[55] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing.* ACM, 1982, pp. 365–377.

[56] R. Wicklin, "The frequency of bigrams in an english corpus," *Stats-Blog*, 2014. [Online]. Available: http://www.statsblogs.com/2014/09/26/the-frequency-of-bigrams-in-an-english-corpus/

[57] M. N. Jones and D. J. Mewhort, "Case-sensitive letter and bigram frequency counts from large-scale english corpora," *Behavior Research Methods, Instruments, & Computers*, vol. 36, no. 3, pp. 388–396, 2004.

[58] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Applied Cryptography and Network Security*, ser. Lecture Notes in Computer Science, J. Ioannidis, A. Keromytis, and M. Yung, Eds. Springer Berlin Heidelberg, 2005, vol. 3531, pp. 442–455.

[59] M. Bellare, A. Boldyreva, and A. OâĂŹNeill, "Deterministic and efficiently searchable encryption," in *Advances in Cryptology - CRYPTO 2007*, ser. Lecture Notes in Computer Science, A. Menezes, Ed. Springer Berlin Heidelberg, 2007, vol. 4622, pp. 535–552.

[60] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized private keyword search over encrypted data in cloud computing," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, June 2011, pp. 383–392.

[61] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacy-preserving query over encrypted graph-structured data in cloud computing," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, June 2011, pp. 393–402.

[62] Y. Lu, "Privacy-preserving logarithmic-time search on encrypted data in cloud." in *NDSS.* The Internet Society, 2012.

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

[64] W. Sun, S. Yu, W. Lou, Y. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 226–234.

[65] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage." in *Network and Distributed System Security (NDSS) Symposium*, February 2014.

[66] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science and Its Applications âĂŞ ICCSA 2008*, ser. Lecture Notes in Computer Science, O. Gervasi, B. Murgante, A. LaganÃă, D. Taniar, Y. Mun, and M. Gavrilova, Eds. Springer Berlin Heidelberg, 2008, vol. 5072, pp. 1249–1259.

[67] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *Journal of Systems and Software*, vol. 83, no. 5, pp. 763 – 771, 2010.

[68] Y. Zhao, X. Chen, H. Ma, Q. Tang, and H. Zhu, "A new trapdoor-indistinguishable public key encryption with keyword search," *Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications*, vol. 3, no. 1/2, pp. 72–81, 2012.

[69] D. E. Knuth, *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms.* Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1997.

[70] M. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology - EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds. Springer Berlin Heidelberg, 2004, vol. 3027, pp. 1–19.

[71] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology âĂŤ EUROCRYPT âĂŹ99*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Springer Berlin Heidelberg, 1999, vol. 1592, pp. 223–238.

[72] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[73] P. Mohassel, "Fast computation on encrypted polynomials and applications," in *Cryptology and Network Security.* Springer, 2011, pp. 234–254.

[74] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–9.

[75] M. S. Islam, M. Kuzu, and M. Kantarcioglu, “Access pattern disclosure on searchable encryption: Ramification, attack and mitigation.” in *NDSS*, 2012.

[76] J. Von Zur Gathen and J. Gerhard, *Modern computer algebra.* Cambridge university press, 2013.

[77] P. Mohassel and M. Franklin, “Efficient polynomial operations in the shared-coefficients setting,” in *Public Key Cryptography - PKC 2006*, ser. Lecture Notes in Computer Science, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds. Springer Berlin Heidelberg, 2006, vol. 3958, pp. 44–57.

[78] L. H. N. C. for Biomedical Communications, “Handbook: Genetic testing,” December 2015.

[79] D. P. Wall, P. Kudtarkar, V. A. Fusaro, R. Pivovarov, P. Patil, and P. J. Tonellato, “Cloud computing for comparative genomics,” *BMC bioinformatics*, vol. 11, no. 1, p. 259, 2010.

[80] K. Krampis, T. Booth, B. Chapman, B. Tiwari, M. Bicak, D. Field, and K. E. Nelson, “Cloud biolinux: pre-configured and on-demand bioinformatics computing for the genomics community,” *BMC bioinformatics*, vol. 13, no. 1, p. 42, 2012.

[81] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing-STOC\’09.* ACM Press, 2009, pp. 169–169.

[82] M. J. Atallah and J. Li, “Secure outsourcing of sequence comparisons,” *International Journal of Information Security*, vol. 4, no. 4, pp. 277–287, 2005. [Online]. Available: http://dx.doi.org/10.1007/s10207-005-0070-3

[83] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, “Privacy preserving error resilient dna searching through oblivious automata,” in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS ’07. New York, NY, USA: ACM, 2007, pp. 519–528. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315309

[84] S. Jha, L. Kruger, and V. Shmatikov, “Towards practical privacy for genomic computation,” in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, May 2008, pp. 216–230.

[85] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong, “Privacy-preserving genomic computation through program specialization,” in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS ’09. New York, NY, USA: ACM, 2009, pp. 338–347. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653703

[86] M. Blanton and M. Aliasgari, "Secure outsourcing of dna searching via finite automata," in *Data and Applications Security and Privacy XXIV*, ser. Lecture Notes in Computer Science, S. Foresti and S. Jajodia, Eds. Springer Berlin Heidelberg, 2010, vol. 6166, pp. 49–64.

[87] M. Chase and E. Shen, "Substring-searchable symmetric encryption," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 263–281, 2015.

[88] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2015, pp. 492–503.

[89] W. Lu, Y. Yamada, and J. Sakuma, "Efficient secure outsourcing of genome-wide association studies," in *Security and Privacy Workshops (SPW), 2015 IEEE.* IEEE, 2015, pp. 3–6.

[90] Y. Chen, B. Peng, X. Wang, and H. Tang, "Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds." in *NDSS*, 2012.

[91] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering gattaca: Efficient and secure testing of fully-sequenced human genomes," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 691–702. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046785

[92] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J.-P. Hubaux, "Privacy-preserving processing of raw genomic data," in *Data Privacy Management and Autonomous Spontaneous Security.* Springer, 2014, pp. 133–147.

[93] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, "A cryptographic approach to securely share and query genomic sequences," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, no. 5, pp. 606–617, Sept 2008.

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

[95] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, Kerkyra, Corfu, Greece, June 28 - July 1, 2011, pp. 850–855.

[96] B. Lynn, "On the implementation of pairing-based cryptosystems," Ph.D. dissertation, Stanford University, 2007.

[97] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[98] E. Barker, W. Burr, A. Jones, T. Polk, S. Rose, M. Smid, and Q. Dang, “Recommendation for key management part 3: Application-specific key management guidance,” *NIST special publication*, vol. 800, p. 57, 2009.

[99] P. Flicek, M. R. Amode, D. Barrell, K. Beal, K. Billis, S. Brent, D. Carvalho-Silva, P. Clapham, G. Coates, S. Fitzgerald *et al.*, “Ensembl 2014,” *Nucleic acids research*, p. gkt1196, 2013.