# PRACTICAL AND FULLY SECURE MULTI KEYWORD RANKED SEARCH OVER ENCRYPTED DATA WITH LIGHTWEIGHT CLIENT

by

TOLUN TOSUN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University
July 2019

# PRACTICAL AND FULLY SECURE MULTI KEYWORD RANKED SEARCH OVER ENCRYPTED DATA WITH LIGHTWEIGHT CLIENT

Approved by:

Prof. Dr. Erkay Savaş ..............................................................
(Thesis Supervisor)

Prof. Dr. Yücel Saygın ..............................................................

Asst. Prof. Dr. Ahmet Onur Durahim ..............................................

Date of Approval: July 19, 2019

# ABSTRACT

PRACTICAL AND FULLY SECURE MULTI KEYWORD RANKED SEARCH OVER ENCRYPTED DATA WITH LIGHTWEIGHT CLIENT

TOLUN TOSUN

COMPUTER SCIENCE AND ENGINEERING M.A. THESIS, MAY 2019

Thesis Supervisor: Prof. Dr. Erkay Savaş

Keywords: cloud computing; homomorphic encryption; secure document similarity; searchable encryption; search pattern; access pattern; secure $k$-NN

Cloud computing offers computing services such as data storage and computing power and relieves its users of the burden of their direct management. While being extremely convenient, therefore immensely popular, cloud computing instigates concerns of privacy of outsourced data, for which conventional encryption is hardly a solution as the data is meant to be accessed, used and processed in an efficient manner. Multi keyword ranked search over encrypted data (MRSE) is a special form of secure searchable encryption (SSE), which lets users to privately find out the most similar documents to a given query using document representation methods such as tf-idf vectors and metrics such as cosine similarity. In this work, we propose a secure MRSE scheme that makes use of both a new secure $k$-NN algorithm and somewhat homomorphic encryption (SWHE). The scheme provides data, query and search pattern privacy and is amenable to access pattern privacy. We provide a formal security analysis of the secure $k$-NN algorithm and rely on IND-CPA security of the SWHE scheme to meet the strong privacy claims. The scheme provides speedup of about two orders of magnitude over the privacy-preserving MRSE schemes using only SWHE while its overall performance is comparable to other schemes in the literature with weaker forms of privacy claims. We present implementations results including one from the literature pertaining to response times, storage and bandwidth requirements and show that the scheme facilitates a lightweight client implementation.

# ÖZET

## ŞIFRELENMIŞ VERI ÜZERINDE TÜMÜYLE GÜVENLI, UYGULANABILIR, DERECELENDIRILMIŞ VE ÇOKLU ANAHTAR KELIME DESTEKLEYEN ARAMA METODU

TOLUN TOSUN

BİLGİSAYAR MÜHENDİSLİĞİ VE BİLİMİ YÜKSEK LİSANS TEZİ, TEMMUZ 2019

Tez Danışmanı: Prof. Dr. Erkay Savaş

**Anahtar Kelimeler:** bulut programlama; homomorfik şifreleme; güvenli döküman benzerliği; sorgulanabilir şifreleme; arama örüntüsü; erişim örüntüsü; güvenli $k$-NN

Veri depolama, programlama ve benzeri bulut hizmetleri, kullanıcılarını bahsi geçen işlemlerin doğrudan yerine getirilmesinin yükünden kurtarmayı amaçlar. Bulut hizmetleri, oldukça popüler ve zahmetsiz olmasının yanı sıra, depolanan verinin mahremiyeti dikkat edilmesi gereken önemli bir husustur. Bu noktada, verinin pratik ve verimli bir şekilde işlenebilmesi için, klasik şifreleme yöntemleri bir çözüm teşkil etmez. Şifrelenmiş veriler üzerinde derecelendirilmiş Çoklu anahtar kelime ile arama(MRSE), sorgulanabilir şifrelemenin(SSE) özel bir dalıdır. Bu yöntem, kullanıcılarının şifrelenmiş veriler üzerinde, sorgularına karşılık gelen en yakın sonuçları güvenli bir şekilde bulabilmelerini sağlar. Benzerlik hesaplamasında kosunüs benzerliği ve tf-idf sembolizasyonu gibi araçlar kullanılır. Bu çalışmada, $k$-NN ve kabaca homomorfik şifreleme(SWHE) gibi tekniklerden yardım alan özgün bir MRSE metodu sunuyoruz. Metodumuz veri, sorgu ve sorgu örüntüsü mahremiyetini sağlarken, erişim mahremiyetinin sağlanmasına da olanak sağlar. Bahsi geçen güçlü güvenlik ve mahremiyet seviyelerine ulaşabilmek için, s$k$-NN algoritmasının usule uygun bir güvenlik analizini sunarken, kullanılan SWHE şemasının sağladığı IND-CPA güvenlik seviyesinden de yararlanılıyor. Şema, benzer mahremiyet seviyeleri sağlayan ve sadece SWHE kullanan klasik MRSE modellerine nazaran 100 kattan fazla bir performans artışı sağlarken, daha düşük seviyelerde mahremiyet seviyeleri vad eden çalışmalarla karşılaştırılabilir performanslar ortaya koyuyor. Uygulamamız ile şemamızı, yanıt süresi, depolama ve kota kullanımı cinsinden kıyaslarken, hafif bir istemci tanımlandığı ortaya koyuluyor

# ACKNOWLEDGEMENTS

I would like to start with thanking to my thesis advisor Prof.Dr. Erkay Savaş. I met with Cryptography in his introductory class. Later on, he became my supervisor on research and we have worked with him for four years. During this time, he has always supported me academically. He has "always" been accesible, shown respect to my ideas, motivated me to challenge myself.

The lion's share of this page is reserved for my family. I send my love to them. From my pre-school education to today, my parents have shown great sacrifice for both my academic and personal evolution. They have always gave me the free space to make my own decisions while supporting any decision I make. Although her young age, my sister has been the only one who I can talk about Computer Science and Cryptography at home, as she always motivates me towards my dreams. I believe she is going to be a very successful computer scientist in the future.

Lastly, I want to mention all my friends. They are all amazing and I am very lucky to have them in my life.

*to my mother, father and lovely sister*

## TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF ALGORITHMS

## LIST OF FIGURES

# Chapter 1

# INTRODUCTION

Cloud computing is one of the most important and popular technologies in contemporary times and its eminence is ever intensifying, in particular with the emergence of big data and its applications. Organizations now hoarding immense amount of data prefer outsourcing their data onto cloud, which reduces their management and maintenance costs. Such data, however, may potentially contain sensitive information; e.g., e-mails, personal health records, financial reports, income tax etc. Confidentiality of remotely stored sensitive data can be ensured by any symmetric key encryption scheme. Nevertheless, a greater challenge is to provide facilities that allow processing of encrypted data such as a advanced search for similar documents. A trivial solution is to download the entire data set, decrypt and process it locally. This is generally infeasible since it requires high bandwidth usage and violates the basic premises of cloud service, which are preferred mainly due to the difficulty of reliable management of data on in-house facilities.

The fully homomorphic encryption (FHE) scheme by Gentry (2009) allows the computation of algebraic operations directly on encrypted data without decrypting it. While more practical schemes than Gentry's solution have since been introduced such as somewhat homomorphic encryption(Brakerski, Gentry & Vaikuntanathan (2014); Cheon, Kim, Kim & Song (2017); Fan & Vercauteren (2012)) which allows batch processing of ciphertexts and they can be used for privacy-preserving document similarity applications, FHE solutions come with a heavy computational overhead and are not feasible in most cases. Consequently, researchers also seek more practicle alternatives to process encrypted data.

A multi keyword ranked search over encrypted data (MRSE) scheme(Cao, Wang, Li, Ren & Lou (2011)) is used to to compute similarity scores between encrypted documents from a set and a given encrypted query, rank them and find the $k$ most similar documents to the query. Both the query and the documents are represented as a collection of (weighted) keywords, where weights can be binary or *tf-idf* values.

Particularly, they are *tf-idf* vectors for which similarity can be measured using one of the metrics such as Euclidean distance and cosine similarity. Alternatively, as a query can be a document itself MRSE is sometimes referred as secure document similarity scheme(Orencik, Alewiwi & Savas (2015)). As the query aims to find out $k$ most similar documents, which are simply vectors, MRSE is also equivalent to secure $k$ nearest neighbor ($k$-NN) algorithm. Following this argument, Wong, Cheung, Kao & Mamoulis (2009) propose a secure $k$-NN (S$k$-NN) algorithm based on symmetric key cryptography to find $k$ most similar documents to a query.

Although KPA security is provided by S$k$-NN(Cao et al. (2011); Wong et al. (2009)) and MRSE variants ensures the confidentiality of outsourced data set, searchable index and query; privacy of *access pattern*, which concerns the data access frequency, is intentionally not addressed. They, on the other hand, utilize query randomization to protect the *search pattern* privacy which prevents linking queries. Nevertheless, the randomization fails to ensure uniformly distributed queries and thus they are highly influenced by the underlying plaintext. Moreover, access and pattern privacy are complementary as they can leak information about each other. Schemes that protect both search and access patterns in addition to query, index and data confidentiality are identified as *fully secure* in Pham, Woodworth & Salehi (2018).

In this work, we propose a new efficient MRSE scheme, MRSE_SWHE_SKNN, that ensures query, index and data confidentiality as well as search pattern privacy. Both documents in the data set and queries are represented as encrypted normalized *tf-idf* vectors and we use the secure dot product by Cao et al. (2011) (referred as S$k$-NN algorithm henceforth) to compute their pairwise cosine similarity. We encrypt queries using our own modified variant of S$k$-NN from Cao et al. (2011) while the searchable index is encrypted with both S$k$-NN and SWHE. Since queries are not encrypted with SWHE, the homomorphic computation of the dot product is considerably simplified resulting in short user response time. It also allows a very light client side implementation and decreases the storage requirements.

Finally, the recent advances in private information retrieval (PIR) techniques (cf. Aguilar-Melchor, Barrier, Fousse & Killijian (2016); Angel, Chen, Laine & Setty (2018)) help ensure access pattern privacy. Consequently, when combined with a PIR scheme, MRSE_SWHE_SKNN provides full security.

The **main contributions** of this work can be summarized as follows:

1.1 We propose a novel idea to combine SWHE and S$k$-NN, which simplifies the homomorphic computation of cosine similarity resulting low client response time.

1.2 We revisit the security claims of S$k$-NN algorithm in Cao et al. (2011); Wong et al. (2009) and provide their more detailed analysis. We then propose a new, modified variant of S$k$-NN (mS$k$-NN) used only to encrypt queries, which facilitates lightweight client side implementation with low storage and computation requirements. We also show that mS$k$-NN is IND-CPA secure.

1.3 We provide the implementation results of the proposed scheme, MRSE_SWHE_SkNN, as well as two other schemes; one is based only on S$k$-NN and the other only on SWHE. The implementation results show that MRSE_SWHE_SkNN is superior to the pure SWHE based implementation while it is comparable to the S$k$-NN only implementation.

# Chapter 2

# RELATED WORK

A practical method for searching over encrypted data, first introduced by Dawn Xiaoding Song, Wagner & Perrig (2000) comes to be referred as secure searchable encryption (SSE). Theirs is a symmetric key solution which uses two layers of encryption, scans entire data base to respond to the query and it is shown to be secure. Curtmola, Garay, Kamara & Ostrovsky (2006) proposed the first sub-linear searchable encryption protocols (SSE-1 and SSE-2) with respect to the number of documents in 2006. Their protocols meet the security requirements of both IND-CKA1 (indistinguishability against non-adaptive chosen keyword attacks) and IND-CKA2 (indistinguishability against adaptive chosen keyword attack). Both studies support only single keyword search.

Wong et al. (2009) propose a scheme that supports more than one keywords and ranking in S$k$-NN queries. Their method is able to find nearest neighbors of a query point given in Euclidean Space. Afterwards Cao et al. (2011) coin the term multi keyword ranked search over encrypted data (MRSE) and propose an efficient solution therein. They adapt and improve the original scheme by Wong et al. (2009) and use dot product as a similarity metric. Moreover, their solution is secure in *known background model* while the original solution is only KPA secure. Subsequently, several alternative MRSE solutions appeared in the literature; e.g. Chen, Qiu, Li, Shi & Zhang (2017); Dhumal & Jadhav (2016); Jiang, Yu, Yan & Hao (2017); Li, Xu, Kang, Yow & Xu (2014); Liu, Guan, Du, Wu, Abedin & Guizani (2019); Orencik, Kantarcioglu & Savas (2013); Strizhov & Ray (2016); Xia, Wang, Sun & Wang (2016); Zhao & Iwaihara (2017). Most of these solutions are based on the generation of a secure searchable index extracted from the data set, using *tf-idf* representation for documents in the data set. Xia et al. (2016) constructed a special, tree-based index to achieve sub-linear searching time. The works (Chen, Zhu, Shen, Hu, Guo, Tari & Zomaya (2016); Jiang et al. (2017); Strizhov & Ray (2016)) also investigate techniques for searching the data set in sub-linear time by comparing the query with only a portion of the data set, which pertains to the features in the query. This is implemented using a secure filtering mechanism.

Different requirements have been defined and provided by MRSE variants from 2014 onwards. Chen et al. (2017) place emphasis on allowing efficient dynamic updates on stored data. The scheme of Li et al. (2014) takes users search history into account while Zhao & Iwaihara (2017) presents a deep learning based approach. The scheme by Liu et al. (2019) supports *cross-lingual* search environment while the traditional *tf-idf* representation depends on a single global dictionary. The works by Liu et al. (2019) as well as Kim, Kim & Chang (2017) employ homomorphic encryption (Pailler's crypto system), which only provides additive homomorphic property. Strizhov & Ray (2016) uses both homomorphic addition and multiplication of SWHE for secure computation of dot products of the search query and documents in the data set. Since the homomorphic computations are overly complicated in their work, Strizhov & Ray (2016) report timings that are hardly practicle even though they propose a sub-linear solution. Their work among other shows clearly that SWHE should be utilized very carefully and depth of the circuit used for homomorphic computations must be minimized. In our work, we address this problem and mainly focus on simplifying homomorphic operations.

Existing MRSE schemes in the literature generally do not intend to protect the access pattern, except for Liu et al. (2019) as it is considered either a separate problem or prohibitively expensive. In the original S$k$-NN scheme(Wong et al. (2009)) or similar works, query randomization is introduced to protect the search pattern. Nevertheless, access pattern and search pattern are complementary in providing full security. Besides, the recent impressive progress in Oblivious RAM (ORAM) and PIR schemes suggest that access pattern privacy can be satisfied efficiently. From Goldreich & Ostrovsky (1996) onwards, several solutions have been proposed for private access to encrypted data using ORAMs (cf. Moataz, Mayberry, Blass & Chan (2015); Stefanov, van Dijk, Shi, Fletcher, Ren, Yu & Devadas (2013)). The main technique in ORAM is to shuffle the stored data after each access to prevent user from accessing the same data in the same storage block. Solutions utilizing ORAM require several rounds of interaction between the client and the server which can make it inefficient in terms of bandwidth usage. Although Naveed (2015) raises concerns as to whether ORAM completely hides the access pattern, it is still a promising technology(Kushilevitz & Mour (2019)). The studies in (Banawan & Ulukus (2018); Boneh, Kushilevitz, Ostrovsky & Skeith (2007); Mayberry, Blass & Chan (2013)) utilize PIR to ensure access pattern privacy in the searchable encryption context. Liu et al. (2019) profitably use PIR to set up an MRSE system without access pattern leakage. Works of (Kim et al. (2017), Elmehdwi, Samanthula & Jiang (2014)) hide the access pattern in S$k$-NN queries by PIR, but not in the context of MRSE. The state of art PIR techniques are quite efficient (cf. Aguilar-Melchor et al. (2016); Angel et al. (2018)), they can be used in many real life scenarios. By providing perfect search pattern privacy our proposal is amenable to full security in MRSE when used together with a secure and efficient ORAM or PIR.

# Chapter 3

# PROBLEM FORMULATION AND PRE-ELIMINARIES

## 3.1 Notations

We use boldface lowercase letters for vectors and boldface uppercase letters for matrices. We use the notation $\boldsymbol{q}[j]$ to refer to the $j$-th element of the vector $\boldsymbol{q}$, where vector index starts at 0. The notation $\boldsymbol{q}[i:j]$ stands for all the elements in the range from $\boldsymbol{q}[i]$ to $\boldsymbol{q}[j]$ being inclusive of border elements, where $j \geq i$. While calligraphic letters are used for sets, a subscripted calligraphic letter denotes a set element; e.g., $\mathcal{I}_i$ is $i$th element of the set $\mathcal{I}$. A superscript in a mathematical object is used to represent vectors over the object; e.g., $\mathbb{Z}^\ell$ stands for vector of integers of dimension $\ell$. A regular lowercase letter is used in other cases when the type is unspecified or not important.

## 3.2 Problem Formulation

The problem is to find the $k$ most similar documents from database of documents to a given set of keywords as query, where both data and query privacy are of the primary concern. Our setting involves three parties: *data owner (DO)*, *data user (DU)* and *cloud server (CS)*. DO has a set of documents $\mathcal{D}$ and outsources it to a cloud server after protected against privacy violations such as applying encryption. DU is a party who is authorized by DO to query the data set through the use of tokens generated by DO. CS offers its professional services to store and manage data sets and process incoming queries such as multi-keyword ranked search. Figure 3.1 illustrates the overall system architecture. DO and DU can be the same entity. The protocols between DO and DU, namely search and access controls, are out of the scope of this work.

As CS sees neither the data nor the query content in our setting, our problem is essentially the document similarity search over encrypted data. In the literature it is frequently referred as multi-keyword ranked search over encrypted data (MRSE), which is a special case of document similarity and our proposal trivially solves the problem of document similarity search over encrypted data.

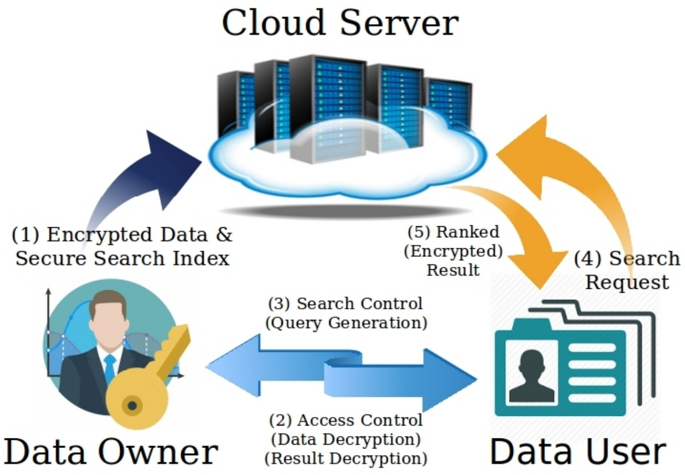

Cloud Server
(1) Encrypted Data & Secure Search Index
(5) Ranked (Encrypted) Result
(4) Search Request
(3) Search Control (Query Generation)
(2) Access Control (Data Decryption) (Result Decryption)
Data Owner
Data User

Figure 3.1 System Architecture Overview

## 3.3 TF-IDF

In order to compute a similarity score between two documents, a data representation methodology and a similarity metric are needed. We use the the most commonly used technique for data representation: term frequency - inverse document frequency (*tf-idf*) vector. In this technique there is a global dictionary of words and each document is represented as a *tf-idf* vector, whose elements are *tf-idf* values of the words for the document. The $tf$ value of a word in a document is directly proportional to the number of its occurrences in the document, while its *idf* is inversely proportional to the number of its occurrences in the whole data set. Finally, *tf-idf* value of a word in a document is the multiplication of its *tf* and *idf* values. Note that there is a single, global *idf* vector for the entire data set while each document has its separate *tf* vector. Therefore, a *tf-idf* vector of a document can also be seen as the component wise multiplication of the global *idf* vector and the *tf* vector corresponding to the document. All *tf-idf* vectors in a data set compose the *tf-idf* table. For a word $\mathcal{W}_i$ appearing in a document $\mathcal{D}_j$, *tf*values can be computed as

$$tf_{\mathcal{W}_i,\mathcal{D}_j} = \frac{\text{number of times } \mathcal{W}_i \text{ appears in } \mathcal{D}_j}{\text{number of words in } \mathcal{D}_j}$$

*idf* for $\mathcal{W}_i$ is computed as

$$idf_{\mathcal{W}_i} = ln\left(\frac{n}{\text{number of documents with } \mathcal{W}_i}\right)$$

*tf-idf* values are multiplicaton of *tf* and idf

$$tf\text{-}idf_{\mathcal{W}_i,\mathcal{D}_j} = tf_{\mathcal{W}_i,\mathcal{D}_j} \cdot idf_{\mathcal{W}_i}$$

The *tf-idf* vectors are always normalized throughout the work. This means the length[1] of *tf-idf* vectors are always 1.

Before constructing a *tf-idf* table for a data set, a pre-processing step is usually applied to eliminate the stop words. Moreover, stemming is performed to merge the words that own the same root. For example, a single token is generated for "consult", "consultor", "consulting", "consultant" and "consultantative".

## 3.4 Cosine Similarity

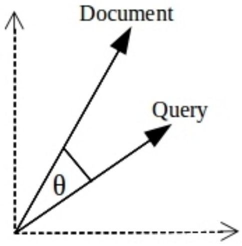


Figure 3.2 Illustration of cosine similarity of two vectors in 2-dimensional vector space

Cosine similarity is a metric for distance between two vectors, which is used to compare two documents represented by their $tf\text{-}idf$ vectors in this work. It basically calculates the cosine of the angle between the vectors. The similarity score calculated is in the range [0.0, 1.0]. Figure 3.2 visualizes the cosine similarity of two vectors in 2-dimensional vector space. As $\theta$ decreases, the similarity increases. For two input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, it is calculated using the formula

$$\frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||},$$

[1] length of a vector is also referred as the L2-norm

where $||.||$ stands for the $L_2$ norm. Note that when the $L_2$ norm of the vectors (length of the vectors) is equal to 1, computation of cosine similarity reduces to computation of dot product of two vectors. Therefore, we normalize vectors and computing dot product is always equivalent to calculating their cosine similarity for the rest of the document.

## 3.5 Fully Homomorphic Encryption

Homomorphic encryption allows computation over encrypted data without decryption. In particular, fully homomorphic encryption (FHE) schemes enable both multiplication and addition operations over encrypted data. As dot product requires only additions and multiplications, homomorphic computation of cosine similarity of two vectors can be performed efficiently. The first FHE scheme is proposed by Craig Gentry Gentry (2009) followed by more practical schemes such as BGV(Brakerski et al. (2014)) and FV(Fan & Vercauteren (2012)). Today, most homomorphic encryption schemes provide, in fact, “somewhat" homomorphic encryption (SWHE) capability, which simply means the number of homomorphic operations that can be applied on ciphertext is limited and the decryption is not possible if the limit, which is often referred as the *noise budget*, is exceeded. In particular, a ciphertext contains a noise term, which increases after every homomorphic operation over the ciphertext. For instance, homomorphic multiplication of two ciphertexts result in the multiplication of noise terms of operands while ciphertext addition leads to their addition. As homomorphic multiplication is much more costly than homomorphic addition so far as the noise budget is concerned, we are interested in calculating the number of sequential multiplications that are required for homomophic computations, which is also called as the circuit depth in the literature.

In this work, we consider FV(Fan & Vercauteren (2012)), which is a SWHE variant. Security of FV relies on the difficulty of ring learning with errors (R-LWE) problem, which is the ring variant of learning with errors problem(Regev (2009)). In FV, ciphertext is an element of polynomial ring, $R_t = \mathbb{Z}_t/f(x)$, where $q$ is the ciphertext modulus and $f(x)$ is a monic irreducible poynomial of degree $m$. In our setting, $f(x)$ is chosen as $x^m + 1$ where $m$ is a power of two, which is the $2m$-th cyclotomic polynomial, as proposed by modern applications. The scheme provides efficient batch encoding, where it is possible to encode $m$ numbers into what is known as plaintext *slots* and encrypt all of them into a single ciphertext; then homomorphic operations over the ciphertext will process the numbers in the slots in *SIMD* fashion. Batch encoded integers are in $\mathbb{Z}_p$, where $p$ is known as plaintext modulus. Moreover, it is possible to circularly rotate (or permute) $m$ numbers in the plaintext slots homomorphically.

In this work, we see the plain text slots as $m$-dimensional vector space(i.e., $\mathbb{Z}_p^m$). For the sake of simplicity, we assume that SWHE is able to encrypt vectors of artbitrary dimension and when the number of slots is less than the dimensionality, the encryption results in multiple ciphertexts. A ciphertext is a polynomial or a polynomial vector and we use a special font style (i.e., fraktur letters) to denote ciphertext; for example $\mathfrak{ct} \in R_t$ denotes a ciphertext. Finally, the numbers (e.g., $x_0, x_1, \ldots, x_{m-1}$) encrypted in ciphertext slots are denoted as $\mathfrak{ct}(x_0;\ x_1;\ \ldots;\ x_{m-1})$.

We list the SWHE operations and functions used in this thesis as follows:

- SWHE.$\textsc{KeyGen}(\lambda) \to \{pk, sk\}$: The key generation function takes the security parameter $\lambda$ and generates public and secret key pair of the SWHE scheme, $pk, sk$.

- SWHE.$\textsc{Encrypt}(pk, \boldsymbol{pt}) \to \mathfrak{ct}$: The encryption function takes the plaintext message $pt$ and the public key $pk$ and returns the ciphertext, $\mathfrak{ct}$.

- SWHE.$\textsc{Decrypt}(sk, \mathfrak{ct}) \to \boldsymbol{pt}$: The decryption function takes the ciphertext message $\mathfrak{ct}$ and the secret key $sk$ and returns the plaintext, $\boldsymbol{pt}$. Note that decryption works correctly only if the noise budget is not exceeded.

- SWHE.$\textsc{Add}(pk, \mathfrak{ct}_1, \mathfrak{ct}_2) \to \mathfrak{ct}$: The homomorphic addition function takes $pk$, and two ciphertexts, $\mathfrak{ct}_1 = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_1)$, $\mathfrak{ct}_2 = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_2)$ and returns the ciphertext $\mathfrak{ct} = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_1 + \boldsymbol{pt}_2)$.

- SWHE.$\textsc{AddPlain}(pk, \mathfrak{ct}_1, \boldsymbol{pt}_2) \to \mathfrak{ct}$: The homomorphic plain addition function takes $pk$, a ciphertext $\mathfrak{ct} = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_1)$, and a plaintext $\boldsymbol{pt}_2$ and returns the ciphertext $\mathfrak{ct} = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_1 + \boldsymbol{pt}_2)$.

- SWHE.$\textsc{Multiply}(pk, \mathfrak{ct}_1, \mathfrak{ct}_2) \to \mathfrak{ct}$: The homomorphic multiplication function takes $pk$, and two ciphertexts, $\mathfrak{ct}_1 = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_1)$, $\mathfrak{ct}_2 = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_2)$ and returns the ciphertext $\mathfrak{ct} = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_1 \cdot \boldsymbol{pt}_2)$.

- SWHE.$\textsc{MultiplyPlain}(pk, \mathfrak{ct}_1, \boldsymbol{pt}_2) \to \mathfrak{ct}$: The homomorphic plain multiplication function takes $pk$, a ciphertext $\mathfrak{ct} = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_1)$, and a plaintext $\boldsymbol{pt}_2$ and returns the ciphertext $\mathfrak{ct} = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}_1 \cdot \boldsymbol{pt}_2)$.

- SWHE.$\textsc{Rotate}(pk, \mathfrak{ct}, k) \to \mathfrak{ct}'$: The homomorphic rotation function takes $pk$, a ciphertext $\mathfrak{ct} = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt})$, and a rotation amount and returns the ciphertext $\mathfrak{ct}' = \text{SWHE.}\textsc{Enc}(pk, \boldsymbol{pt}')$ such that $\boldsymbol{pt}'[i] = \boldsymbol{pt}[i + k \pmod m]$ for $i \in [0, m-1]$.

Note that arithmetic operations on vectors are applied component-wise; e.g.

$$\begin{aligned}\boldsymbol{pt}_1 + \boldsymbol{pt}_2 = \boldsymbol{pt}_1[0] + \boldsymbol{pt}_2[0] \bmod p;\ \ldots;\\ \boldsymbol{pt}_1[m-1] + \boldsymbol{pt}_2[m-1] \bmod p\end{aligned}$$

where $p$ is the plaintext modulus.

The security level provided by the underlying R-LWE instances used in FV depends, to a lager extent, on $m$ and $t$. The scheme is IND-CPA secure as it provides probabilistic encryption. Higher noise budget required to compute deeper circuits necessitates increasing $t$, which, in turn, leads to using larger ring degrees $m$ to adjust the security level. Table 3.1 illustrates the relation between $m$ and $t$. The security estimations are made through the online lwe-estimator on `https://bitbucket.org/malb/lwe-estimator/src/master/`.

Table 3.1 Security levels of R-LWE instances for varying $m$ and $t$

| $m$ | $\log_2 t$ | | | |
|---|---|---|---|---|
| | 54 | 75 | 109 | 146 |
| 1024 | 63 | 49 | 45 | 45 |
| 2048 | 129 | 90 | 62 | 50 |
| 4096 | 282 | 194 | 128 | 93 |
| 8192 | 629 | 433 | 281 | 201 |

## 3.6 Secure kNN Computation

The secure $k$NN (S-$k$NN) algorithm is proposed by Wong et al. (2009) to find the $k$ nearest neighbors of encrypted data points, where the similarity metric is the Euclidean distance. Cao et al. (2011) show how to use S-$k$NN algorithm to calculate “dot product similarity". Next paragraphs explain the adaption of S-$k$NN which works in a vector space as similarity is measured by dot product similarity.

The aim of the S-$k$NN algorithm is to find the $k$ nearest neighbors of a query in a data set $\mathcal{D}$. The data set consists of n vectors, which will referred as *data vector*s henceforth. Namely, $\mathcal{D} = \{\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-1}\}$ and $\boldsymbol{d}_i \in \mathbb{R}^w$. The query is also a $w$-dimensional vector over real numbers, $\boldsymbol{q} \in \mathbb{R}^w$. Since the $k$ nearest neighbors will be detected over the encryptions of the data vectors and queries, the algorithm defines a dot product preserving encryption such that $E_q(\boldsymbol{q}) \cdot E_d(\boldsymbol{d}_i) = \boldsymbol{q} \cdot \boldsymbol{d}_i$ for $0 \leq i < n$. Then, it simply calculates the dot product of a data vector $\boldsymbol{q}$ and the query $\boldsymbol{d}_i \in \mathbb{R}^w$

using $E_q(\boldsymbol{q})$ and $E_d(\boldsymbol{d}_i)$, and returns the ids of $k$ documents with highest similarity scores.

Encryption of data vectors and queries are asymmetric. They are encrypted in a similar fashion, but the process diverges at specific points. The secret key comprises a bit string $\boldsymbol{s}$ of size $w$ and two invertible, randomly generated $w \times w$-dimensional matrices, $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$. Let us continue the discussion with data vector encryption. The first step is the *random splitting* operation. Random splitting aims to randomly share the information in $\boldsymbol{d}_i$ to $\boldsymbol{d}_i' \in \mathbb{R}^w$ and $\boldsymbol{d}_i'' \in \mathbb{R}^w$. The splitting operation is performed as follows for $j = 0, \ldots, w-1$: if $\boldsymbol{s}[j]$ is 1, then $\boldsymbol{d}_i'[j] \leftarrow \boldsymbol{d}_i[j]$ and $\boldsymbol{d}_i''[j] \leftarrow \boldsymbol{d}_i[j]$. Otherwise, $\boldsymbol{d}_i'[j]$ and $\boldsymbol{d}_i''[j]$ are randomly selected so that

$$\boldsymbol{d}_i[j] = \boldsymbol{d}_i'[j] + \boldsymbol{d}_i''[j].$$

Encryption of $\boldsymbol{d}_i$ is then concluded as

$$E_d(\boldsymbol{d}_i) = \{\boldsymbol{M}_1{}^T \boldsymbol{d}_i', \boldsymbol{M}_2{}^T \boldsymbol{d}_i''\}.$$

The query $\boldsymbol{q}$ is encrypted in a similar fashion. The same splitting operation is applied to the extended $\boldsymbol{q}$ except the roles of 1s and 0s in $\boldsymbol{s}$ are reversed. Finally, the encryption of $\boldsymbol{q}$ is performed as

$$E_q(\boldsymbol{q}) = \{\boldsymbol{M}_1{}^{-1}\boldsymbol{q}', \boldsymbol{M}_2{}^{-1}\boldsymbol{q}''\}.$$

The dot product can be computed over the encryptions of query and data vectors as follows:

$$\begin{aligned} E_q(\boldsymbol{q}) \cdot E_d(\boldsymbol{d}_i) &= (\boldsymbol{M}_1^T \boldsymbol{d}_i') \cdot (\boldsymbol{M}_1^{-1}\boldsymbol{q}') + (\boldsymbol{M}_1^T \boldsymbol{d}_i'') \cdot (\mathbf{M_2^{-1}}\boldsymbol{q}'') \\ &= (\boldsymbol{d}_i'^T \boldsymbol{M}_1)(\boldsymbol{M}_1^{-1}\boldsymbol{q}') + (\boldsymbol{d}_i''^T \boldsymbol{M}_2)(\boldsymbol{M}_2^{-1}\boldsymbol{q}'') \\ &= \boldsymbol{d}_i'^T \boldsymbol{q}' + \boldsymbol{d}_i''^T \boldsymbol{q}'' = \boldsymbol{d}_i' \cdot \boldsymbol{q}' + \boldsymbol{d}_i'' \cdot \boldsymbol{q}'' \end{aligned}$$

Note that we have

$$\boldsymbol{d}_i'[j]\boldsymbol{q}'[j] + \boldsymbol{d}_i''[j]\boldsymbol{q}''[j] = \boldsymbol{d}_i[j]\boldsymbol{q}[j] \qquad \text{for } 0 \leq j < w$$

Then, we obtain

$$\boldsymbol{d}_i' \cdot \boldsymbol{q}' + \boldsymbol{d}_i'' \cdot \boldsymbol{q}'' = \boldsymbol{d}_i \cdot \boldsymbol{q} \tag{3.1}$$

### 3.6.1 First MRSE scheme utilizing S-$k$NN Algorithm: MRSE$_{IITF}$

Cao et al. (2011) defines the problem of Multi Keyword Ranked Search over Encrypted Data(MRSE) for the first time and solves it utilizing Secure $k$NN algorithm. Here, we briefly explain the problem and their final solution(MRSE$_{IITF}$) from the paper. MRSE$_{IITF}$ is compared with MRSE_SWHE_SkNN in chapter 6.

The *data owner (DO)* has a set of documents $\mathcal{D}$ with size $n$. $\mathcal{D}$ is represented as *tf-idf* vectors. The dictionary for $\mathcal{D}$ is denoted by $\mathcal{W}$ while $|\mathcal{W}| = w$. *DO* extracts a secure searchable index $\mathcal{I}$ from $\mathcal{D}$ and stores it in *cloud server (CS)*. The *data user (DU)* is authorized to search $\mathcal{D}$ with a selected keywords of interest $\overline{\mathcal{W}}$. *CS* calculates and returns the top-$k$ results among $\mathcal{D}$ to $DU$, using $\mathcal{I}$. Overall secure search protocol of MRSE$_{IITF}$ consists of four stages: *Key generation*, *Index Generation*, *Query Generation*, *Search*.

- MRSE$_{IITF}$.KeyGen$(U) \to \boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{s}$: The key generation function generates secret keys of MRSE_TF_II scheme, $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{s}$ such that $|\boldsymbol{M}_i| = (w+U+1) \times (w+U+1)$ and $|\boldsymbol{s}| = w+U+1$. $U$ can be thought as a security parameter.

- MRSE$_{IITF}$.BuildIndex$(\mathcal{D}, \boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{s}) \to \mathcal{I}$: Index generation is done by encrypting every $\boldsymbol{d}_i$ by S-$k$NN algorithm with some modifications. For each document, $\boldsymbol{d}_i$ is extended to $(w+U+1)$ dimension first. During the dimension extension, $(w+j)$-th dimensions are set to random numbers $\epsilon_j$ for $1 \geq j \geq U$ while the last dimension is set to 1. Denote the extended vector by $\boldsymbol{d}^E{}_i$. Then the splitting operation is applied to $\boldsymbol{d}^E{}_i$ w.r.t $\boldsymbol{s}$ as explained in the previous section. Finally, the sub index $\mathcal{I}_i$ is $\{\boldsymbol{M}_1^T \boldsymbol{d}^E{}_i{}', \boldsymbol{M}_2^T \boldsymbol{d}^E{}_i{}''\}$.

- MRSE$_{IITF}$.GenerateQuery$(\overline{\mathcal{W}}, \boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{s}) \to \overline{\boldsymbol{q}}$: Let $\boldsymbol{q}$ is $w$-dimensional vector and $L = \left|\overline{\mathcal{W}}\right|$. The function sets $\boldsymbol{q}[i] = 1/\sqrt{L}$ if $\mathcal{W}_i \in \overline{\mathcal{W}}$ and $\boldsymbol{q}[i] = 0$ otherwise. Note that length of $\boldsymbol{q}$ is 1. For encryption, first a binary vector of length $U$ is generated by choosing $V$ dimensions randomly and setting $V$ out of $U$ dimensions to 1 while the other dimension are 0. Let $\boldsymbol{b}$ denotes the random vector. $\boldsymbol{q}$ is concatenated with $\boldsymbol{b}$, and then the resulting vector is multiplied by a random factor $r$. The result is again extended to $w+U+1$ dimension by setting the last dimension to a random number, $\tau$. Let $\boldsymbol{q}^E$ denotes the extended $\boldsymbol{q}$. Secure search query $\overline{\boldsymbol{q}} = \{\boldsymbol{M}_1^{-1} \boldsymbol{q}^{E'}, \boldsymbol{M}_2^{-1} \boldsymbol{q}^{E''}\}$ is returned.

- MRSE$_{IITF}$.Search$(\mathcal{I}, \overline{\boldsymbol{q}})$: The dot product between $\overline{\boldsymbol{q}}$ and each $\mathcal{I}_i$ is calculated as an approximation of cosine similarity between the query and the documents. Then documents with top-$k$ scores are returned.

Note that, the final similarity score for each document is equal to

$$r(\boldsymbol{q} \cdot \boldsymbol{d}_i + \sum_{j=1}^{U} \boldsymbol{b}[j]\epsilon_j) + \tau \tag{3.2}$$

According to Cao et al. (2011), the scheme is secure against scale analysis attack, which is a known background attack model, as similarity scores are randomized by adding artificial dimensions. For more information about scale analysis attack, please refer to Cao et al. (2011). On the other hand, the intentionally added error term defines a trade-off between the accuracy and privacy. This way, CS does not learn the actual similarity scores or exact document rank in the sorted similarity list, but can still find top-$k$ documents with relatively high accuracy.

The weigthining of data and query vectors in the above scheme is slightly different than the original explanataion of Cao et al. (2011). Particularly, Cao et al. (2011) use *tf* vectors instead of *tf-idf* for data vectors and *idf* values of the selected keywords for the query vector. We applied the same weighting logic with our proposed solution to MRSE$_{IITF}$ and explained in the above functions to perform a fair comparison.

# Chapter 4

# PROPOSED SOLUTION

In this section, we explain our proposal for efficient computation of the cosine similarity between a document and a query, both of which are encrypted. We combine the S-$k$NN algorithm and SWHE schemes to provide enhanced security and privacy for both documents and query. We first introduce our framework as a collection of functions. Then, we provide the privacy model that outlines the privacy and security requirements the solution aims to satisfy.

We introduce two algorithms for secure multi-keyword ranked search; namely multi keyword ranked search encryption with SWHE (MRSE_SWHE) and multi keyword ranked search encryption with both S$k$NN and SWHE (MRSE_SWHE_SkNN). The former is a trivial solution, which utilizes a SWHE scheme to homomorphically compute dot product similarity between two encrypted *tf-idf* vectors, and inefficient. The latter is our most important contribution and proves to be much more efficient and practicable.

## 4.1 Framework

Table 4.1 The notation used in the solutions

| | |
|---|---|
| $\mathcal{D}$ | Data set of documents $\lvert\mathcal{D}\rvert = n$ |
| $\mathcal{W}$ | Dictionary and $\lvert\mathcal{W}\rvert = w$ |
| $\boldsymbol{d}_i$ | *tf-idf* vector of document $\mathcal{D}_i \in \mathcal{D}$ |
| $\mathcal{I}$ | Secure searchable index for $\mathcal{D}$ |
| $\overline{\mathcal{W}}$ | Set of interested keywords to search in $\mathcal{D}$, $\left\lvert\overline{\mathcal{W}}\right\rvert = L$ |
| $\boldsymbol{q}$ | Query vector for $\overline{\mathcal{W}}$ |
| $\overline{q}$ | Secure search query, encryption of $\boldsymbol{q}$ |
| $\lambda$ | Security parameter |
| $\boldsymbol{M}$ | secret key of mS-$k$NN algorithm $\lvert\boldsymbol{M}\rvert = \eta \times \eta$ |
| $pk, sk$ | public and secret keys of the SWHE scheme, respectively |
| $m$ | degree of the polynomial modulus in SWHE, number of slots in batch mode for our settings |
| $p$ | plaintext modulus in SWHE |
| $t$ | ciphertext modulus in SWHE |
| $\mathbb{Z}_p^m$ | plain text space defined in SWHE scheme |
| $\mathrm{H} : \mathbb{R} \to \mathbb{Z}_p$ | Scale function $\mathrm{H}(x) = \lfloor \varsigma x \rceil$ |

The proposed framework consists of six functions as explained in the following.

- KeyGen($\lambda$) $\to \{SK, PK\}$: DO runs the key generation function that takes security parameter $\lambda$ selected for a desired security level and returns a secret and public key pair $\{PK, SK\}$.

- BuildIndex($PK(SK), \mathcal{D}$) $\to \mathcal{I}$: DO runs the function that takes the data vectors in $\mathcal{D}$ and public (and/or secret) keys and returns a secure searchable index $\mathcal{I}$.

- GenerateQuery($PK(SK), \overline{\mathcal{W}}$) $\to \overline{q}$: DU (with the help of DO) runs the function that takes a subset of words vector and public (and/or secret) keys and returns a secure search query $\to \overline{q}$.

- CalculateSimilarity($PK, \overline{q}, \mathcal{I}$) $\to ss$: CS runs the function that takes $PK$, the secure index $\mathcal{I}$, secure search query $q$; and returns the encrypted list of similarity scores $ss$.

- DecryptAndRank($sk, ss$) $\to \{ID_1, \ldots, ID_K\}$: After getting the encrypted results $ss$ from CS, DU decrypts them using the secret key and ranks the most similar $k$ documents IDs, $ID_i \in \mathbb{Z}_n$.

- PIR$(ID_i) \to \mathcal{D}_{ID}$: Run between DU and the CS to privately retrieve the documents in top-$k$ list using a state-of-art PIR protocol (e.g., Angel et al. (2018)).

Table 4.1 lists some of the notations used in the framework and solutinos. Note that PIR is beyond the scope of this work. Note that PIR is beyond the scope of this work.

## 4.2 Privacy Model

In this section, privacy goals are defined. We are using a honest-but-curious adversary model and our proposed scheme is designed to reveal nothing to the adversary, which is essentially CS that stores encrypted document set $\mathcal{D}$ and secure searchable index $\mathcal{I}$, observes queries $q$, and performs the CalculateSimilarity function. In honest-but-curious adversary model, the operations are performed as specified in the protocol definitions while the adversary can do some extra analyses to extract any meaningful information regarding the data set and queries. Confidentiality of the document collection $\mathcal{D}$ can be provided by any symmetric key cryptographic scheme and it is out of the scope of this thesis. We focus on the following security definitions:

**Definition 1** (Index Confidentiality)**.** *No polynomial time adversary can decrypt the secure searchable index $\mathcal{I}$.*

**Definition 2** (Query Confidentiality)**.** *No polynomial time adversary can decrypt any secure search query $q$.*

**Definition 3** (Search Pattern Confidentiality)**.** *Let $\overline{\mathcal{Q}} = \{\overline{q}_1, \overline{q}_2, \ldots, \overline{q}_\tau\}$ be a set of secure search queries. Search pattern $S_p(\overline{\mathcal{Q}})$ is a symmetric binary matrix of dimensions-$(\tau \times \tau)$ such that its element in i-th row and j-th column is 1 if and only if $\boldsymbol{q}_i = \boldsymbol{q}_j$ and 0 otherwise. No polynomial time adversary is able to guess any bit of $S_p(\overline{\mathcal{Q}})$ with chance better than $1/2$ probability.*

**Definition 4** (Similarity Pattern Confidentiality)**.** *Let $\overline{\mathcal{Q}} = \{\overline{q}_1, \overline{q}_2, \ldots, \overline{q}_\tau\}$ be a set of secure search queries. Similarity pattern $Sim_p(\overline{\mathcal{Q}})$ is a 3-D binary matrix of dimensions-$(\tau \times \tau \times w)$ such that its element with index $(i, j, k)$ is 1 if and only both if $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ contain keyword $\mathcal{W}_k$ or none of them does, and 0 otherwise. No polynomial time adversary is able to guess any bit of $Sim_p(\overline{\mathcal{Q}})$ with chance better than $1/2$ probability.*

**Definition 5** (Access Pattern Confidentiality)**.** *Let $\mathcal{D}(\overline{q}_i)$ be the subset of documents* *that match the secure search query $\overline{q}_i$ and $\overline{\mathcal{Q}} = \{\overline{q}_1, \overline{q}_2, \ldots, \overline{q}_\tau\}$. Then, access pattern of $\overline{\mathcal{Q}}$ is defined as $A_p(\overline{\mathcal{Q}}) = \{\mathcal{D}(\overline{q}_1), \mathcal{D}(\overline{q}_2), .., \mathcal{D}(\overline{q}_\tau)\}$. Any polynomial time adversary is unable to guess any document from $A_p(\overline{\mathcal{Q}})$ with a chance better than $1/n$ probability.*

Many MRSE schemes in the literature such as (Cao et al. (2011); Chen et al. (2017); Jiang et al. (2017); Orencik et al. (2013); Strizhov & Ray (2016); Xia et al. (2016)) employ query randomization to hide search or similarity patterns. However, search pattern can still leak when a response to a query is in plain. Unless the document collection changes frequently the matching documents to two queries $\dot{q}$ and $\ddot{q}$, that are different randomizations of the same query $\boldsymbol{q}$, will be the same. Also, most of the schemes in the literature do not protect the access pattern due to its high cost as a private access requires processing all documents in the whole document collection (e.g., utilizing a PIR (private information retrieval) scheme). Therefore, protecting the search pattern without protecting the response and the access pattern is meaningless since the search results will immediately enable an observer to link search queries.

Consequently, our solution hides both the query response and access pattern in order to realize stronger security claims for search and similarity pattern confidentiality. This requires that all similarity scores have to be calculated homomorphically by CS on encrypted queries and the resulting ciphertext containing similarity scores, $ss$, are sent to DU. While this is costly as far as the bandwidth and client side computation are concerned, the batching technique in SWHE helps enable efficient implementation. Also, recent efficient state-of-the-art PIR schemes such as (Aguilar-Melchor et al. (2016); Angel et al. (2018)) can be profitably used to hide access pattern, which is a concern when the actual similar documents are retrieved from CS. In this work, we provide implementation results to demonstrate that all our security requirements are met in an efficient manner.

## 4.3 Multi keyword ranked search with SWHE: MRSE_SWHE

Multi keyword ranked search using only a SWHE scheme (MRSE_SWHE) is a straightforward solution for secure document similarity, which trivially satisfies security requirements. It essentially employs an efficient SWHE scheme to perform homomorphic computations of the similarity scores between a query and documents in the data set, all of which are homomorphically encrypted and finds most similar $k$ documents securely.

Algorithms 1, 2, 3, 4, and 5 illustrate all the stages of MRSE_SWHE from the key generation to the client side decryption operations excluding the PIR stage, which is necessary for access pattern privacy. The key generation function in Algorithm 1, given the security parameter $\lambda$, generates a public and secret key pair for encryption of the index as well as queries under the chosen SWHE scheme.

**Algorithm 1** MrseSwhe.KeyGen

```
Require: $\lambda$
Ensure: $\{pk, sk\}$
  $\{pk, sk\} \leftarrow$ SWHE.KeyGen($\lambda$)
```

The index and query generation operations, given in Algorithm 2 and Algorithm 3, respectively, are quite straightforward. The *tf-idf* vectors of the documents in the data set and of query are encrypted by the SWHE scheme in batch mode. Before the encryption operation in both index generation and query generation, real valued *tf-idf* values are mapped to integers in $\mathbb{Z}_p$ that are encoded as SWHE plaintexts. Steps 5-7 of Algorithm 2 and Steps 10-12 of Algorithm 3 illustrates the operation. The function $\mathrm{H} : \mathbb{R} \rightarrow \mathbb{Z}_p$ is a simple function defined as $\mathrm{H}(x) = \lfloor \varsigma x \rceil$, where $x \in [0.0, 1.0]$, the scale $\varsigma \in \mathbb{Z}$, and $\lfloor \cdot \rceil$ stands for rounding to the nearest integer. Depending on the values of the scale $\varsigma$, there will be a loss of accuracy because the precision of the *tf-idf* values is now bounded. The original cosine similarity score between a document and a query is bounded by 1. Therefore, setting scale parameter $\varsigma < \sqrt{p}$ guarantees that the similarity scores do not exceed $p$. Affect of $\varsigma$ is discussed in the section 6.2. Generally speaking, setting sigma as close to $\sqrt{p}$ as possible gives the best results.

**Algorithm 2** MrseSwhe.BuildIndex

```
Require: $pk$, $\mathcal{D}$
Ensure: $\mathcal{I}$
1: for $j$ from $0$ to $n-1$ do
2:     $\boldsymbol{d}_j \leftarrow$ tf-idf vector of $\mathcal{D}_j$.
3: end for
4: for $i$ from $0$ to $w-1$ do
5:     for $j$ from $0$ to $n-1$ do
6:         $\boldsymbol{d}^H[j] = \mathrm{H}(\boldsymbol{d}_j[i])$
7:     end for
8:     $\mathcal{I}_i \leftarrow$ SWHE.Encrypt($pk, \boldsymbol{d}^H$)
9: end for
```

Consequently, $\mathcal{I}_i$ is a SWHE encryption of $\{\boldsymbol{d}_1[i], \boldsymbol{d}_2[i], .., \boldsymbol{d}_n[i]\}$ as a result of Algorithm 2 while Algorithm 3 returns $\mathfrak{q}$ as a SWHE encryption of $\boldsymbol{q}$ by the SWHE scheme.

**Algorithm 3** MrseSwhe.GenerateQuery

```
Require: $pk$, $\overline{\mathcal{W}} \in \mathbb{R}^w$
Ensure: $\bar{\mathfrak{q}} \in R_q$
$L \leftarrow \left|\overline{\mathcal{W}}\right|$
$\boldsymbol{q} \in \mathbb{R}^w$, plain query vector
for $i$ from $0$ to $w-1$ do
    if $\mathcal{W}_i \in \overline{\mathcal{W}}$ then
        $\boldsymbol{q}[i] = 1/\sqrt{L}$
    else
        $\boldsymbol{q}[i] = 0$
    end if
end for
for $i$ from $0$ to $w-1$ do
    $\boldsymbol{q}^H[i] \leftarrow \mathrm{H}(\boldsymbol{q}[i])$.
end for
$\bar{\mathfrak{q}} \leftarrow \mathrm{SWHE.Encrypt}(pk, \boldsymbol{q}^H)$
```

In Algorithm 4, the cosine similarity of $\boldsymbol{q}$ to each $\boldsymbol{d}_i$ is homomorphically computed with the help of $\mathcal{I}$ and $\bar{\mathfrak{q}}$. Thus, the main loop iterates exactly $w$ times. Steps 2-11 homomorphically compute a new ciphertext $\mathfrak{q}'_i$, each slot of which encrypts $\boldsymbol{q}[i]$. $\boldsymbol{mask}_i$ is a simple binary vector in which only the i-th dimension is 1 while the rest is 0. It is used to choose $\boldsymbol{q}[i]$ among the underlying plaintext slots of $\bar{\mathfrak{q}}$ by the plaintext multiplication. Then, $\mathfrak{q}'_i$ is homorphically multiplied with $\boldsymbol{d}_j[i]$ for $0 \leq j < n$ in Step 12, which are encrypted in $\mathcal{I}_i$. Thanks to batch mode of SWHE, this is done in $\lceil n/m \rceil$ homomorphic multiplications. The ciphertext $\mathfrak{t}_i$, containing a partial result of the cosine similarity between the query and all documents, is then homomorphically added to similarity scores ciphertext $\mathfrak{ss}$ in Step 13. After the main loop is completed, each slot of $\mathfrak{ss}$ contains the similarity score between the query and a document. Namely, the i-th slot contains an approximation of the cosine similarity between $\boldsymbol{q}$ and $\boldsymbol{d}_i$ as $\mathrm{H}(\boldsymbol{q}) \cdot \mathrm{H}(\boldsymbol{d}_i)$.

For sake of simplicity, we can think the number of slots $m$ is greater than the number of words and the number of documents, namely $m > w, n$; therefore a single ciphertext can encrypt a query $\boldsymbol{q}$, an index element $\mathcal{I}_i$ or the similarity scores $\mathfrak{ss}$. In case this does not hold, we need to use more than one ciphertext to provision for the necessary number of slots. For instance, when $n > m$, then we need $\ell = \lceil n/m \rceil$ ciphertexts to hold similarity scores $\mathfrak{ss}$. On the other hand, $w > m$ requires $\ell = \lceil w/m \rceil$ ciphertexts to represent $\bar{\mathfrak{q}}$. However, the loop in Step 12 of Algorithm 4 iterates $\lceil \log_2(m) \rceil$ times in the worst case as this many iterations are enough to produce a ciphertext with $\boldsymbol{q}[i]$ at every slot.

The final stage of the scheme is straightforward as demonstrated in Algorithm 5; the

**Algorithm 4** MrseSwhe.CalculateSimilarity

```
Require: $pk, \bar{\mathfrak{q}} \in R_q$, $\mathcal{I} \in R_q^w$
Ensure: $\mathfrak{ss} \in R_q$
$\mathfrak{ss} \leftarrow$ SWHE.Encrypt($pk, \{\mathbf{0}\}^n$)
for $i$ from 1 to $w$ do
    for $j$ from 0 to $w$ do
        $\boldsymbol{mask}_i[j] \leftarrow 0$
    end for
    $\boldsymbol{mask}_i[i] \leftarrow 1$
    $\mathfrak{q}'_{\mathfrak{i}} \leftarrow$ SWHE.MultiplyPlain($pk, \boldsymbol{mask}_i, \bar{\mathfrak{q}}$)
    for $k$ from 0 to $\lceil \log_2(n) \rceil - 1$ do
        $\mathfrak{t}' \leftarrow$ SWHE.Rotate($pk, \mathfrak{q}'_{\mathfrak{i}}, 2^k$)
        $\mathfrak{q}'_{\mathfrak{i}} \leftarrow$ SWHE.Add($pk, \mathfrak{q}'_{\mathfrak{i}}, \mathfrak{t}'$))
    end for
    $\mathfrak{t}_i \leftarrow$ SWHE.Multiply($pk, \mathfrak{q}'_{\mathfrak{i}}, \mathcal{I}_i$)
    $\mathfrak{ss} \leftarrow$ SWHE.Add($pk, \mathfrak{ss}, \mathfrak{t}_i$)
end for
```

client decrypts $\mathfrak{ss}$ using secret key $sk$ and extract the ids of the documents that are in the top-$k$ in the list. This is done by finding the $k$-th best score($\top_k$), partitioning around $\top_k$ to find out the list of top-$k$ similar documents and sorting it. Note that computational complexity of overall ranking is $O(n + k \log k)$, which is very close to $O(n)$ as $k$ is generally a small integer in practice. Note that, the search result $\mathfrak{ss}$ is composed of $\lceil n/m \rceil$ SWHE ciphertexts each of which carries similarity information for disjoint parts of the data set and can be processed individually. Therefore, most of the workload of Algorithm 5 can be overlapped with Algorithm 4 for $n >> m$.

**Algorithm 5** MrseSwhe.DecryptandRank

```
Require: $sk$, $\mathfrak{ss}$
Ensure: Indices of top-$k$ elements in $\boldsymbol{r}$ after sorting
    $\boldsymbol{r} \leftarrow$ SWHE.Decrypt($sk, \mathfrak{ss}$)
    $\top_k \leftarrow$ $k$-th largest score in $\boldsymbol{r}$
    top-$k$ $\leftarrow$scores greater than or equal to $\top_k$
    sort top-$k$
```

As mentioned previously, the implementation of the MRSE_SWHE scheme is quite straightforward while it is not efficient as we demonstrate in this work. A much more efficient scheme is explained in the next section.

## 4.4 Second Scheme: MRSE_SWHE_SkNN

MRSE_SWHE_SkNN is an improved version of MRSE_SWHE that combines SWHE scheme with our modified version of the secure $k$-NN (mS-$k$NN) encryption algorithm introduced in Section 3.6.1. In particular, the new scheme significantly accelerates the CalculateSimilarity algorithm. In the MRSE_SWHE scheme, both searchable index and query are encrypted under a SWHE scheme. In MRSE_SWHE_SkNN, we use both SWHE and S-$k$NN schemes for encryption. We first explain the new, modified secure $k$-NN (mS-$k$NN) encryption algorithm in the next section.

### 4.4.1 Modified Secure-$k$NN Encryption Scheme: mS-$k$NN

The modified S-$k$NN (mS-$k$NN) algorithm uses a secret key, which is a non-singular matrix $\boldsymbol{M}$ of $\eta \times \eta$, whose elements are uniformly randomly sampled in the field $\mathbb{Z}_p$ as illustrated in Algorithm 6. Here, the modulus $p$, which is also the plaintext modulus in the SWHE scheme, is appropriately chosen for similarity computations. We explain as to how the matrix dimension $\eta$ is determined based on our security analysis in Section 5.1.2.

**Algorithm 6** mSkNN.KeyGen

```
Require: $\eta$
Ensure: $\boldsymbol{M} \in \mathbb{F}_p^{\eta\times\eta}$
  $\boldsymbol{M} \leftarrow \mathbb{F}_p^{\eta\times\eta}$, if $\boldsymbol{M}$ is singular, sample again
```

We have two encryption algorithms; one for the data set and the other for queries as given in Algorithm 7 and Algorithm 8, respectively. Algorithm 7 takes the secret matrix $\boldsymbol{M}$ and a $w$-dimensional data vector $\boldsymbol{d}$, and outputs a $2w$-dimensional vector $\bar{\boldsymbol{d}}$ as encryption of the input. The input $\boldsymbol{d}$ is originally a vector of real numbers and therefore it is mapped to $\mathbb{Z}_p^w$ using the function $\mathrm{H} : \mathbb{R} \to \mathbb{Z}_p$. Algorithm 7 is repeated for every document in $\mathcal{D}$. Here we assume $w = k \cdot \eta$ for some integer $k$ without loss of generality.

**Algorithm 7** mSkNN.EncryptData

```
Require: $\boldsymbol{M} \in \mathbb{Z}_p^{\eta\times\eta}$, $\boldsymbol{d} \in \mathbb{R}^w$
Ensure: $E_d(\bar{\boldsymbol{d}}) = \bar{\boldsymbol{d}} \in \mathbb{Z}_p^{2w}$
for $j$ from $0$ to $w-1$ do
    $\boldsymbol{d'}[2j] \leftarrow \mathrm{H}(\boldsymbol{d}[j])$
    $\boldsymbol{d'}[2j+1] \leftarrow \mathrm{H}(\boldsymbol{d}[j])$
end for
for $j$ from $0$ to $2w/\eta - 1$ do
    $\bar{\boldsymbol{d}}[j\eta : (j+1)\eta - 1] = {\boldsymbol{M}^{-1}}^T \boldsymbol{d'}[j\eta : (j+1)\eta - 1]$
end for
```

Algorithm 8, similarly takes $\boldsymbol{M}$ and a $w$-dimensional query vector $\boldsymbol{q}$ over real numbers, and outputs a $2w$-dimensional vector over $\mathbb{Z}_p$ as encryption of the input. Note that the query is randomized by a uniformly chosen $\rho \in \mathbb{Z}_p$, that supports query unlinkability.

**Algorithm 8** mSkNN.EncryptQuery

```
Require: $\boldsymbol{M} \in \mathbb{Z}_p^{\eta \times \eta}$, $\boldsymbol{q} \in \mathbb{R}^w$
Ensure: $E_q(\boldsymbol{q}) = \bar{\boldsymbol{q}} \in \mathbb{Z}_p^{2w}$
1: for $j$ from $0$ to $w-1$ do
2:     $\rho \leftarrow \mathbb{Z}_p$
3:     $\boldsymbol{q'}[2j] \leftarrow \rho$
4:     $\boldsymbol{q'}[2j+1] \leftarrow \mathrm{H}(\boldsymbol{q}[j]) - \rho$
5: end for
6: for $j$ from $0$ to $2w/\eta - 1$ do
7:     $\bar{\boldsymbol{q}}[j\eta : (j+1)\eta - 1] = \boldsymbol{M}\boldsymbol{q'}[j\eta : (j+1)\eta - 1]$
8: end for
```

The main idea in the modified version remains the same as the original S-$k$NN algorithm, and both data vectors and query vectors are splitted first, then multiplied by the transpose and the inverse of a secret matrix, respectively. The splitting is deterministic for data vectors (see Steps 2-5 of Algorithm 7) while a random splitting is applied to queries (see Steps 2-6 of Algorithm 8). As the output of Algorithm 7 will be re-encrypted as shown in Algorithm 10 for index construction, the splitting is based on simple repetition of each dimension. All operations are done in the field $\mathbb{Z}_p$ which is identical to the plaintext space of the SWHE scheme.

For correctness, we can show the dot product of encrypted query and tf-idf vector of a document gives an approximation of cosine similarity of these two vectors:

$$\begin{aligned}
E_d(\boldsymbol{d}) \cdot E_q(\boldsymbol{q}) &= \sum_{j=0}^{2w/\eta - 1} (\boldsymbol{M}^{-1}\boldsymbol{d'}[j\eta : (j+1)\eta - 1]) \cdot (\boldsymbol{M}^T \boldsymbol{q'}[j\eta : (j+1)\eta - 1]) \\
&= \sum_{j=0}^{2w-1} \boldsymbol{d'}[j]\boldsymbol{q'}[j] \\
&= \sum_{j=0}^{w-1} (\mathrm{H}(\boldsymbol{d}[j])\rho + \mathrm{H}(\boldsymbol{d}[j])((\mathrm{H}(\boldsymbol{q}[j]) - \rho)) \\
&= \sum_{j=0}^{w-1} \mathrm{H}(\boldsymbol{d}[j])\mathrm{H}(\boldsymbol{q}[j]).
\end{aligned}$$

The security of modified S-$k$NN scheme is meaningful when it is used in conjunction with the homomorphic encryption. Therefore, we defer its thorough security analysis to Section 5 after we provide all algorithmic details of the second scheme in next section.

### 4.4.2 Combining SWHE Scheme and Modified Secure-$k$NN Algorithms

In the proposed scheme, mS-$k$NN algorithm is used to encrypt queries as well as the index. The index is also encrypted under the SWHE scheme before sent to CS. Therefore, the key generation operation generates both a public and secret key pair for SWHE scheme and secret key $SK.\boldsymbol{M}$ for S-$k$NN scheme as shown in Algorithm 9. Here, the secret key comprises the secret key for SWHE, $SK.sk$, and the secret key for mS-$k$NN, $SK.\boldsymbol{M}$.

**Algorithm 9** MrseSwheSkNN.Keygen

```
Require: $\lambda$
Ensure: $\{PK, SK\}$
$\{pk, SK.sk\} \leftarrow$ SWHE.KeyGen($\lambda$)
$SK.\boldsymbol{M} \leftarrow$ mSkNN.KeyGen($\lambda$)
```

For index generation, *tf-idf* vectors of documents are encrypted first using mS-$k$NN algorithm as in Steps 1-4 of Algorithm 10. Then, the resulting ciphertexts $\boldsymbol{c}_i$ are encrypted again under SWHE in Step 9 of Algorithm 10.

**Algorithm 10** MrseSwheSkNN.BuildIndex

```
Require: $pk$, $sk$, $\mathcal{D}$
Ensure: $\mathcal{I} \in R_q^{2w}$
 1: for $i = 1$ to $n$ do
 2:     $\boldsymbol{d}_i \leftarrow$ tf-idf vector of $\mathcal{D}_i$
 3:     $\boldsymbol{c}_i \leftarrow$ mSkNN.EncryptData($sk.\boldsymbol{M}$, $\boldsymbol{d}_i$)
 4: end for
 5: for $i$ from $0$ to $2w-1$ do
 6:     for $j$ from $0$ to $n-1$ do
 7:         $\boldsymbol{t}_i[j] = \boldsymbol{c}_j[i]$
 8:     end for
 9:     $\mathcal{I}_i \leftarrow$ SWHE.Encrypt($pk, \boldsymbol{t}_i$)
10: end for
```

Query generation is explained in Algorithm 11, where only the mS$k$NN algorithm is used for encryption. Simply speaking, the probabilistic nature of Algorithm 8 provides both security and unlinkability of queries.

The similarity calculation operation in the new scheme, performed as described in Algorithm 12, differs from the method in Algorithm 4 in two important aspects. Firstly, as the query is not encrypted under SWHE, all the expensive homomorphic operations over the query in Steps 7-11 of Algorithm 4 are now eliminated. Especially, there may be up to $\lceil \log_2 m \rceil$ sequential homomorphic rotations and additions in Steps 8-11. This can be prohibitively expensive even for data sets with

**Algorithm 11** MrseSwheSkNN.GenerateQuery

```
Require: $pk$, $\overline{\mathcal{W}} \in \mathbb{R}^w$
Ensure: $\bar{\mathfrak{q}} \in R_q$
$L \leftarrow \left|\overline{\mathcal{W}}\right|$
$\boldsymbol{q} \in \mathbb{R}^w$, plain query vector
for $i$ from $0$ to $w-1$ do
    if $\mathcal{W}_i \in \overline{\mathcal{W}}$ then
        $\boldsymbol{q}[i] = 1/\sqrt{L}$
    else
        $\boldsymbol{q}[i] = 0$
    end if
end for
$\bar{\boldsymbol{q}} \leftarrow$ mSkNN.EncryptQuery($SK.\boldsymbol{M}, \boldsymbol{q}$)
```

moderately many documents. For instance, $m = 4096$ requires 12 such operations performed sequentially, all of which are eliminated in the new scheme.

**Algorithm 12** MrseSwheSkNN.CalculateSimilarity

```
Require: $\bar{\boldsymbol{q}} \in \mathbb{Z}_p^{2w}$, $\mathcal{I} \in R_q^{2w}$
Ensure: $\mathfrak{ss} \in R_q$
1: $\mathfrak{ss} \leftarrow$ Swhe.Encrypt($pk, \{\mathbf{0}\}^n$)
2: for $i = 1$ to $2w$ do
3:     $\mathfrak{t} \leftarrow$ Swhe.MulPlain($\bar{\boldsymbol{q}}[i], \mathcal{I}_i$)
4:     $\mathfrak{ss} \leftarrow$ Swhe.Add($\mathfrak{ss}, \mathfrak{t}$)
5: end for
```

Secondly, compared to the homomorphic multiplication in Step 12 of Algorithm 4, the homomorphic multiplication in Step 3 of Algorithm 12 is inexpensive as one of its operand is not homomorphically encrypted. As this multiplication increases the noise only linearly (in comparison to quadratic increase in Step 12 of Algorithm 4), we need much lower noise budget. This, in turn, leads to much smaller ciphertext modulus $t$. Consequently, the new similarity calculation operation turns out to be much faster than done in Algorithm 4 as shown in the subsequent sections.

The similarity calculation in Algorithm 12 is also different from MRSE$_{IITF}$(Cao et al. (2011)) algorithm in several fundamental aspects. Firstly, in the MRSE$_{IITF}$ algorithm, results of the cosine similarity calculations are not encrypted as can be observed in Eq. (3.2), where they are only translated by random factor $r$ and partially hidden by the noise term $\epsilon_i$. As $r$ is the same for all documents in the data set, their ranks in the similarity list are not expected to be affected by the translation operation. Therefore, while two randomized queries corresponding to the same *tf-idf* vector appear to be unrelated they will most probably result in identical similarity lists; hence the original S-$k$NN algorithm leaks the search pattern to CS unless the data set changes. The similarity list $\mathfrak{ss}$ in Algorithm 12, however, is homomorphically encrypted that can be decrypted only with the secret key of the SWHE scheme. Consequently, CS has not access to the similarity list, and therefore the privacy leak is trivially eliminated. However, $\mathfrak{ss}$ comprises the similarity of each document to the query that needs to be sent to DU, which leads to an increase in the bandwidth. The overhead is somewhat alleviated due to the fact that one SWHE ciphertext can encrypt $m$ similarity result. The number of ciphertext needed for all similarity scores is then $\ell = \lceil n/m \rceil$.

Secondly, when $\mathfrak{ss}$ is decrypted by DU via

$$\boldsymbol{r} \leftarrow \text{SWHE.}\textsc{Decrypt}(sk.sk, \mathfrak{ss})$$

we obtain the similarity list in plaintext, in which each element approximates (depending on the function H) the exact computation of the dot product of the query vector and the *tf-idf* vector of the corresponding document

$$\boldsymbol{r}[i] = \sum_{j=0}^{2w-1} \mathrm{H}(\boldsymbol{q}[j])\mathrm{H}(\boldsymbol{d}_i[j]) \approx \mathrm{H}(\textsc{CosSim}(\boldsymbol{d}_i, \boldsymbol{q}))$$

for $i = 0, 1, \ldots, n-1$. This is in contrast to the MRSE$_{IITF}$ algorithm that computes $r \cdot (\textsc{CosSim}(\boldsymbol{d}_i, \boldsymbol{q})) + \epsilon_i$, where the similarity score is translated by random factor $r$ and perturbed by $\epsilon_i$, (see Eq. (3.2)). The randomization is needed in the original S-$k$NN algorithm as a partial protection against leakage to CS that has access to $\boldsymbol{r}$. The translation operation by the random factor and the noise introduced by $\epsilon_i$ is eliminated in the new scheme, where CS cannot decrypt $\mathfrak{ss}$ and access $\boldsymbol{r}$.

# Chapter 5

# SECURITY AND PRIVACY ANALYSIS

In this section, we discuss the security and privacy of the proposed scheme. First, the security of the original S-$k$NN scheme will be revisited and the security claims in the literature will be re-examined. Then, we provide the security analysis of the new mS-$k$NN scheme in Section 4.4.1. In particular, we show that Algorithm 8 used for query encryption is secure. Finally, we prove that the security and privacy claims in Section 4.2 are satisfied by the proposed scheme.

## 5.1 Security of the Original S-$k$NN Scheme Revisited

(Wong et al. (2009), Cao et al. (2011)) claim that the original S-$k$NN scheme is KPA-secure; i.e., secure against known plaintext attacks. The security analysis assumes that adversary has access to $\mu$ pairs of plaintext and ciphertext for the query vector, $\boldsymbol{q}$ (plaintext) and corresponding $\boldsymbol{c}$ (ciphertext), respectively. Recalling that $\boldsymbol{q'}$ and $\boldsymbol{q''}$ are the random splits of $\boldsymbol{q}$ with dimension of $w$ each in the encryption phase we have $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2) = (\boldsymbol{M_1^{-1}q'}, \boldsymbol{M_2^{-1}q''})$. Since the secret binary vector $\boldsymbol{s}$ of $w$ is used for the splitting operation, the analysis take the elements of the vectors $\boldsymbol{q'}$ and $\boldsymbol{q''}$ as unknowns. Consequently, adversary can obtain $2\mu w$ equations from $\mu$ pairs of $\boldsymbol{q}$ and $\boldsymbol{c}$, but the total number of unknowns is $2w^2 + 2\mu w$. As the number of unknowns will always be larger than the number of equations independent of number of plaintext, ciphertext pairs, adversary can never solve the equation system and find the secret key.

One potential issue with this analysis is about the number of unknowns. Recall that the splitting operation is performed as

$$\begin{aligned} \boldsymbol{q}'[i] &= (1-\boldsymbol{s}[i])\boldsymbol{q}[i]+\boldsymbol{s}[i]\rho \\ \boldsymbol{q}''[i] &= (1-\boldsymbol{s}[i])\boldsymbol{q}[i]+\boldsymbol{s}[i](\boldsymbol{q}[i]-\rho), \end{aligned} \tag{5.1}$$

for $i = 0,\ldots,w-1$ and where $\rho$ is randomly chosen. Then, the actual number of unknowns can be computed as $2(w)^2 + w + \mu(w+1)$, where $2(w)^2$ stands for the unknowns coming from $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$, $w$ stands for the confidentiality of $\boldsymbol{s}$ and $\mu(w)$ stands for the random numbers generated during the encryption. Consequently, we can construct an equation system where the number of unknowns is less than or equal to the number of equations for certain value of $\mu$. However, the equation system becomes more complicated resulting in a costly solution. Therefore, we can sketch another security analysis based on hypothesis testing explained in the next section.

### 5.1.1 Hypothesis Testing on Secret Key $\boldsymbol{s}$

Following the security analysis given by Wong et al. (2009), we assume that there are $2\mu w$ equations and $2w^2 + 2\mu w$ unknowns in the equation system. However, we can decrease the number of unknowns by hypothesis testing on one bit of $\boldsymbol{s}$.

Let our hypothesis be $\boldsymbol{s}[i] = 1$ for $i = [0, w-1]$. Then Eq. (5.1) simplifies to $\boldsymbol{q}'[i] = \boldsymbol{q}''[i] = \boldsymbol{q}[i]$, which will reduce the number of unknowns by one to $2w^2 + 2\mu(w-1)$. Then, the following inequality holds

$$2w^2 + 2\mu(w-1) \leq 2\mu w$$

for $w^2 \leq \mu$. This indicates that if adversary collects at least $w^2$ pairs of plaintext and ciphertext, the equation system can be solved and thus secret keys are recovered. For testing our hypothesis, additional control pairs are used to check if the solution is correct. If the control pair is consistent with the solution, then the hypothesis is correct and $\boldsymbol{s}[i]$ is indeed 1; otherwise 0. This procedure can be repeated for all bits of $\boldsymbol{s}$.

This result suggests that the S-$k$NN scheme requires a more rigorous security analysis that should be based on the difficulty of solving the equation system. Once the security analysis helps quantify the security level we can optimize the key sizes, which are prohibitively large in S-$k$NN. For instance, we can eliminate $\boldsymbol{s}$ as secret key if the secret matrices $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ already suffice for a desired security level.

Indeed, the equation system is non-linear and can be very difficult to solve even for moderate sizes of the secret matrices. In the next section, we provide a security analysis of our modified version of S-$k$NN scheme using the results from the work (Courtois, Klimov, Patarin & Shamir (2000)) that provides an analysis for the computational complexity of solving multivariate polynomial equations.

### 5.1.2 Security of the Modified S-$k$NN Scheme

As the index and the similarity scores in the proposed scheme are always encrypted using SWHE, which provides IND-CPA security, we only discuss the query security in this section.

The secret binary vector $\boldsymbol{s}$ does not play an essential role in the security as shown in Section 5.1.1. Also in the original scheme, the dimensions of $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are $w \times w$, which can be prohibitively large as $w$ is the number of words in the dictionary $\mathcal{W}$ and usually very large (e.g. $w > 1000$). Thus, to decrease the secret key size the modified S-$k$NN scheme eliminates $\boldsymbol{s}$ and employs only a single, sufficiently large secret key matrix $\boldsymbol{M}$ of dimension $\eta \times \eta$. It turns out that $\eta$ can be much smaller than $w$ and taken as $w = k\eta$ for some positive integer $k$.

In query encryption by Algorithm 8, the random splitted query vector $\boldsymbol{q'}$ is multiplied by $\boldsymbol{M}$ to obtain the ciphertext vector

$$\bar{\boldsymbol{q}}[i] = \sum_{j=0}^{\eta-1} \boldsymbol{M}[i \bmod \eta][j]\boldsymbol{q'}[\lfloor i/\eta \rfloor \eta + j] \tag{5.2}$$

for $0 \leq i \leq 2w-1$. Eq.(5.2) can be generically written as a system of multivariate quadratic polynomial equations

$$b_k = \sum_{1 \leq i \leq j \leq \nu'} a_{ijk} x_i x_j \tag{5.3}$$

for $k = 1, 2, .., \mu'$, which are investigated by Courtois et al. (2000). The problem of solving quadratic polynomial equations system is known to be NP-hard over any field. Here, the system has $\mu'$ equations for $\nu'$ unknowns; namely $x_1, x_2, .., x_{\nu'} \in \mathbb{Z}_p$ ($\mathbb{Z}_p = \mathbb{F}_p$ when $p$ is prime). In case $\mu' \approx \nu'$, the so called *working factor* (WF) for solving Eq.(5.3) is given as $WF \geq e^{\alpha\sqrt{\nu'}(\ln \nu'/2+1)}$), where $\alpha$ in the exponent is the complexity of solving linear equation systems(Courtois et al. (2000)). For instance, $\alpha = 3$ for Gaussian elimination and $\alpha = 2.3766$ for improved methods(Courtois et al. (2000)).

| $\eta$ | security level ($\lambda$) | # of multiplications in $\mathbb{F}_p$ |
|---|---|---|
| 8 | 57 | 16384 |
| 12 | 102 | 24576 |
| 16 | 152 | 32768 |
| 20 | 205 | 40960 |
| 24 | 261 | 49152 |
| 1024 | 24336 | 2097152 |

Table 5.1 Security level of Algorithm 8 for various values of $\eta$ with $\alpha = 2.3766$ and the number of multiplications in query encryption with $w = 1024$
.

For $\mu$ pairs of $\boldsymbol{q}$ and $\bar{\boldsymbol{q}}$ vectors in Eq. (5.2), the equation system results in $\eta^2 + \mu w$ unknowns and $2\mu w$ equations. When $2\mu w = \eta^2 + \mu w + 2$ (slightly over-determined), the equation system can be solved and $WF \geq \eta^{\alpha\eta}$ (see Appendix for the details). Consequently, the security levels provided by Algorithm 8 for query encryption depending on the matrix size $\eta$ can be approximated as shown in Table 5.1. As seen in the table, with even moderate dimensions of secret matrix $\boldsymbol{M}$ we can achieve sufficiently high security; e.g., $\eta = 16$ provides approximately 152 bit of security.

Here we provide only a sketch of security analysis to demonstrate the query confidentiality described in Def 2. A stronger security analysis for Def 2, we prove that Algorithm 8 is in fact CPA secure in Theorem 6 in Appendix. Similarly, for search and similarity pattern confidentiality (Definitions 3 and 4, respectively ) Algorithm 8 should be also IND-CPA, which is proved by Theorem 7 in Appendix.

Note that a smaller matrix size accelerates the query encryption operation significantly. Table 5.1 enumerates the number of multiplications in $\mathbb{F}_p$ during a query encryption operation (Step 7 of Algorithm 8). While the original S-$k$NN algorithm requires prohibitively large number of $\mathbb{F}_p$ multiplications even for moderate size dictionaries (e.g., 2,097,152 for $w = 1024$ in the last row of Table 5.1), mS-$k$NN requires much fewer number of such operations (e.g., only 32768 for $\lambda = 152$). Finally, mS-$k$NN reduces the storage requirement for secret key from $O(w^2)$ to $O(\eta^2)$.

### 5.1.3 Privacy Claims

This section shows that the proposed scheme in Section 4.4.1 (namely MRSE_SWHE_SkNN scheme) satisfies all the privacy goals (given in Definitions 1-4) in Section 4.2.

**Theorem 1.** *The MRSE_SWHE_SkNN scheme provides confidentiality for the index $\mathcal{I}$ of a document collection $\mathcal{D}$ in accordance with Definition 1.*

**Proof:** The index $\mathcal{I}$ of a document collection $\mathcal{D}$ is encrypted by the FV scheme (Fan & Vercauteren (2012)), which is one of the most efficient SWHE schemes and provides IND-CPA security. Therefore, no polynomial time adversary with access to index encryption oracle can decrypt $\mathcal{I}$ and learns the *tf-idf* values of documents in $\mathcal{D}$ without the secret key of the SWHE scheme except with negligible probability.

**Theorem 2.** *The MRSE_SWHE_SkNN scheme provides confidentiality for encrypted query $\bar{\boldsymbol{q}}$ in accordance with Definition 2.*

**Proof:** A query vector in the MRSE_SWHE_SkNN scheme is encrypted using a secret matrix $\boldsymbol{M}$ of dimension $\eta \times \eta$ as described in Algorithm 8. As claimed in Section 5.1.2 and proved by Theorem 6 in Appendix, encryption in Algorithm 8 provides CPA security. Therefore, no polynomial time adversary with access to query encryption oracle can decrypt an encrypted query $\bar{\boldsymbol{q}}$ and learns the *tf-idf* values of the query $\boldsymbol{q}$ without the secret matrix $\boldsymbol{M}$ provided that $\eta$ is sufficiently large except with negligible probability.

**Theorem 3.** *The MRSE_SWHE_SkNN scheme provides search pattern confidentiality for an arbitrary set of query vectors $\mathcal{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_t\}$ in accordance with Definition 3.*

**Proof:** Theorem 7 in Appendix shows that query encryption scheme in Algorithm 8 is IND-CPA secure. Namely, given two arbitrarily chosen queries $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$ and the encryption of one of them $\bar{\boldsymbol{q}}_b = E(\boldsymbol{q}_b)$ for uniformly randomly chosen $b \in \{0,1\}$, adversary that has access to encryption oracle cannot guess $b$ with a probability significantly better than 0.5. Let $\bar{\mathcal{Q}} = \{\bar{\boldsymbol{q}}_1, \bar{\boldsymbol{q}}_2, \ldots, \bar{\boldsymbol{q}}_t\}$ be a randomly permuted set of encryption of queries in $\mathcal{Q}$. Suppose also $\boldsymbol{q}_i = \boldsymbol{q}_j$ for a pair of $i$ and $j$ $(i \neq j)$. Due to IND-CPA security, no adversary can tell which elements in $\bar{\mathcal{Q}}$ correspond to $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ except with the probability of random guess (which is $\frac{1}{t(t-1)}$). This is due the fact that every dimension of both $\bar{\boldsymbol{q}}_i$ and $\bar{\boldsymbol{q}}_j$ is uniformly distributed in $\mathbb{F}_p$ as explained in Theorem 7 in Appendix.

**Theorem 4.** *The MRSE_SWHE_SkNN scheme provides similarity pattern confidentiality for an arbitrary set of query vectors $\mathcal{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_t\}$ in accordance with Definition 4.*

**Proof:** The proof follows directly from Theorem 3. Let $\bar{\mathcal{Q}} = \{\bar{\boldsymbol{q}}_1, \bar{\boldsymbol{q}}_2, \ldots, \bar{\boldsymbol{q}}_t\}$ be a randomly permuted set of encryption of queries in $\mathcal{Q}$. Suppose also, for a pair of $i$ and $j$ $(i \neq j)$, two queries $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ does not contain a word $\mathcal{W}_k$ for an arbitrary $k$; namely $\boldsymbol{q}_i[k] = \boldsymbol{q}_j[k] = 0$. Due to IND-CPA security, no adversary can tell which elements in $\bar{\mathcal{Q}}$ correspond to $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ except with the probability of random guess (which is $\frac{1}{t(t-1)}$). We can make the same arguments for two queries having a word $\mathcal{W}_k$ in common with identical or different *tf-idf* values.

**Theorem 5.** *The MRSE_SWHE_SkNN scheme is amenable to access pattern con-*

*fidentiality for an arbitrary set of query vectors $\mathcal{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_t\}$ in accordance with Definition 5.*

**Proof:** Let $\bar{\mathcal{Q}} = \{\bar{\boldsymbol{q}}_1, \bar{\boldsymbol{q}}_2, \ldots, \bar{\boldsymbol{q}}_t\}$ denotes the secure search queries correspond to $\mathcal{Q}$. The access pattern is defined as $A_p(\bar{\mathcal{Q}}) = \{\mathcal{D}(\bar{\boldsymbol{q}}_1), \mathcal{D}(\bar{\boldsymbol{q}}_2), .., \mathcal{D}(\bar{\boldsymbol{q}}_\tau)\}$, where $\mathcal{D}(\bar{\boldsymbol{q}}_i)$ is the subset of documents that match the secure search query $\bar{\boldsymbol{q}}_i$. The similarity search results are encrypted by FV(Fan & Vercauteren (2012)) as SWHE scheme and it is IND-CPA secure. Therefore, any leakage can only happen during the retrieval of $\mathcal{D}(\bar{\boldsymbol{q}}_i)$. Any IND-CPA PIR protocol such as (Aguilar-Melchor et al. (2016),Angel et al. (2018)) avoids leakage from retrieval of resulting documents from the cloud server. Thus, no polynomial time adversary can tell any document in $A_p(\bar{\mathcal{Q}})$ except with the probability of random guess $1/n$.

# Chapter 6

# IMPLEMENTATION RESULTS AND EVALUATION

section, we investigate the second MRSE scheme (MRSE_SWHE_SkNN in Section 4.4) in terms of efficiency and accuracy. The method is compared with our first solution (i.e., MRSE_SWHE in Section 4.3) and one of the existing solutions in the literature (MRSE$_{IITF}$ Cao et al. (2011)). MRSE$_{IITF}$(Cao et al. (2011)) utilizes the secure $k$NN algorithm explained in Section 3.6.1. For test purposes, we use Enron E-mail data set(Shetty & Adibi (2004)). As the number of words depends on the number of documents used in our experiments, we include Figure 6.1 that illustrates the relation between $w$ (number of keywords extracted) and $n$ (number of documents) in the test data set.

## 6.1 Implementation Details

Implementing MRSE schemes including MRSE_SWHE, MRSE_SWHE_SkNN and MRSE$_{IITF}$( Cao et al. (2011)) requires programming using different software tech-

Table 6.1 Number of keywords ($w$) extracted from Enron data set for varying number of documents($n$)

| $n$ | $w$ |
|---|---|
| 2048 | 2445 |
| 4096 | 3605 |
| 6144 | 5545 |
| 8192 | 6315 |
| 10240 | 6862 |

nologies. The test data set is pre-processed using Python gensim[1] library to transform it from bag-of-words to *tf-idf* vector representation. The stop words are filtered out and rest of the dictionary is stemmed by nltk tools[2]. Homomorphic operations required by both methods are carried out in Microsoft SEAL[3] in C++. Operations over real valued matrices and vectors are performed using Apache Math Library[4], which provides necessary operations for matrix, vector multiplications, matrix inversions over real numbers. On the other hand, matrix operations over $\mathbb{Z}_p$ needed by mSkNN are implemented in-house using Java programming language. The testing framework for all three schemes is assembled using Scala programming language which connects all the modules of the project. As we work in $\mathbb{Z}_p$ in both of our schemes MRSE_SWHE and MRSE_SWHE_SkNN, we use the BFV implementation in the SEAL library. The following parameter selection ensures correct decryption after the homomorphic evaluation for MRSE_SWHE: $\mathcal{S}_1 = \{m = 8192, log_2(t) = 152, p = 65537\}$. As for the MRSE_SWHE_SkNN, the parameter selection is as follows: $\mathcal{S}_2 = \{m = 2048, log_2(t) = 54, p = 40961\}$, which meets the noise budget requirements of the circuit for the computations. For $\mathcal{S}_1$ and $\mathcal{S}_2$, the underlying R-LWE instance provides 192-bit and 128-bit security, respectively. The scale parameter in the function H (See Section 6.2) used to map *tf-idf* values to $\mathbb{F}_p$ is chosen as $\varsigma = 200$ since $\varsigma < \sqrt{p}$ guaranteeing that plaintext after homomorphic computations will never exceed the plaintext modulus $p$. As for mS-$k$NN configurations, $\eta$ is chosen as 15 that provides ≈138-bit security.

## 6.2 Accuracy

As pointed out in Section 5.1.3, the MRSE$_{IITF}$( Cao et al. (2011)) scheme claims to provide rank privacy and there is a trade-off between rank privacy and accuracy. Both the MRSE_SWHE and the MRSE_SWHE_SkNN schemes provide perfect rank privacy as the similarity scores are encrypted by SWHE.

The ranking of documents in the decrypted similarity results can be different than the grand truth due to the precision loss in MRSE_SWHE_SkNN. On the other hand, the trade-off between privacy and accuracy in MRSE$_{IITF}$ is controlled by a system parameter, denoted as $\sigma$, which is simply the standard deviation of the normal distribution used to sample the noise terms $\epsilon$ and $\epsilon_i$ (See Section 3.6.1) .

[1] See https://pypi.org/project/gensim/

[2] See https://www.nltk.org/

[3] See https://www.microsoft.com/en-us/research/project/microsoft-seal/

[4] See https://commons.apache.org/proper/commons-math/

Therefore, we investigate the accuracy of both methods in more detail and observe how they are affected by parameters $\varsigma$ and $\sigma$. First, we define a metric, used in our experiments, to better capture the performance of the MRSE schemes.

**Accuracy Metric:** Correctly measuring the accuracy of the system is important in order to properly optimize the system parameters. A straightforward approach is compare the top-k document IDs returned by a MRSE scheme with those in the *grand truth.* A better approach is to consider the documents returned by a MRSE scheme and use their *correct* similarity scores obtained in the ground truth. Thus, given a query $\boldsymbol{q}$, we formulate the accuracy of MRSE scheme as

$$\text{ACC} = \frac{\sum_{i=1}^{k} \boldsymbol{ss}[\mathcal{D}_{\boldsymbol{q},i}^{MRSE}]}{\sum_{i=1}^{k} \boldsymbol{ss}[\mathcal{D}_{\boldsymbol{q},i}^{GT}]} \times 100 \tag{6.1}$$

where $\boldsymbol{ss}[\mathcal{D}_i]$ denotes the correct similarity between the document $\mathcal{D}_i$ and the query $\boldsymbol{q}$. Also $\mathcal{D}_{\boldsymbol{q}}^{MRSE}$ and $\mathcal{D}_{\boldsymbol{q}}^{GT}$ are sorted sets of similar documents returned by the MRSE scheme and the ground truth, respectively.

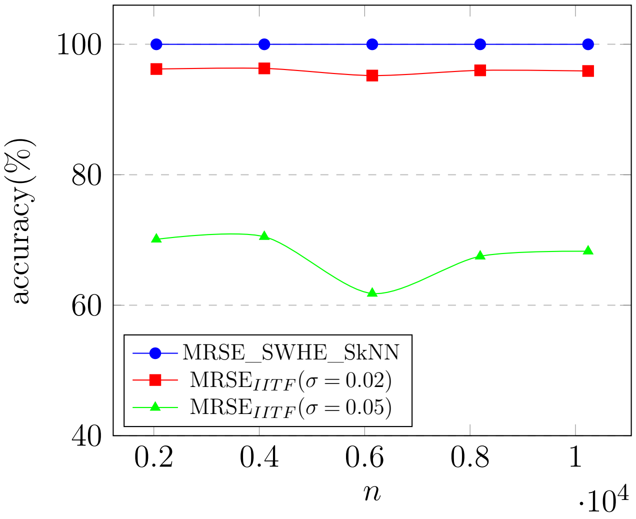
As mentioned in the Section 4.3, $\varsigma$ is chosen deterministicly depending on the value of $p$. To increase $\varsigma$, one should increase $p$, which requires increasing $t$ in order to meet the noise budget requirements of homomorphic evaluation circuit. However, setting $\varsigma = 200$ as mentioned in section 6.1 results in about %99.999 percent accuracy in our tests for varying $n$ and $w$ values as can be observed in Figure 6.1(for $w$ values, see Table 6.1). For MRSE$_{IITF}$, we use two values of standard deviation; namely $\sigma = 0.02$ and $\sigma = 0.05$. As results clearly suggest, MRSE_SWHE_SkNN is more accurate that MRSE$_{IITF}$ without compromising privacy, whereas the accuracy of MRSE$_{IITF}$ is highly sensitive to $\sigma$. Note that higher values of $\sigma$ that is employed against background attacks decrease accuracy while MRSE_SWHE_SkNN is not vulnerable to those attacks as it is CPA secure.

## 6.3 Performance

For performance evaluations, we use as metrics bandwidth usage, memory requirement and response times. The results are the average of 100 runs with randomly chosen input values whereby the test results with unusually high standard deviations are not taken into account for average computations.

In order to perform a better simulation of the client-server architecture, we have used a client and a server machine located in the university. The client has Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz as processing unit with up to 3GB RAM access. All the experiments regarding DU/DO is run by the mentioned machine.

Figure 6.1 Accuracy of MRSE$_{IITF}$ and MRSE_SWHE_SkNN with varying values of $n$ and $w$



On the other hand, server is a supercomputer with 16 Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz processors. The server program is allowed to use 50GB of RAM. The link between the client and the server is 10MB/s in practise, which is measured by SCP[5].

### 6.3.1 Time Performance

The most important operation is the secure computation of similarity scores given a query $\boldsymbol{q}$ as far as the execution times are concerned as CS performs it in response to potentially overly many queries submitted online. Therefore, we first obtain the execution times of Algorithm 12 used in MRSE_SWHE_SkNN for secure similarity calculations in server side and compare them against those of Algorithm 4 used in MRSE_SWHE in Table 6.2. The latter scheme, MRSE_SWHE, represents the approach taken by many works in the literature such as (Strizhov & Ray (2016)), which basically requires homomorphically encrypting the query as well. Table 6.2, which reports a speedup of at least 199 times, demonstrates the advantage of our approach. This is natural due to the fact that the query is not homomorphically encrypted in MRSE_SWHE_SkNN simplifying the homomorphic computations in Algorithm 12 in comparison with those of Algorithm 4.

We next compare the execution times of Algorithm 12 with that of MRSE$_{IITF}$(Cao et al. (2011)) used for similarity calculation. The latter scheme uses much weaker privacy claims and therefore provides faster computation of similarity scores in gen-

[5] Secure Copy Protocol

Table 6.2 Execution times of similarity score computation for MRSE_SWHE and MRSE_SWHE_SkNN in milliseconds with varying values of $n$ and $w$

| $n$ | $w$ | Algorithm 4 | Algorithm 12 | speedup |
|---|---|---|---|---|
| 8192 | 1000 | 172265 | 863 | **199** × |
| 8192 | 1500 | 252469 | 1263 | **199** × |
| 8192 | 2000 | 344143 | 1704 | **201** × |
| 8192 | 2500 | 410530 | 2039 | **201** × |
| 16384 | 1500 | 496932 | 2480 | **200** × |

Table 6.3 Execution times of similarity score computation for of MRSE$_{IITF}$ and MRSE_SWHE_SkNN in milliseconds for varying values of $n$ and $w$

| $\boldsymbol{n}$, $\boldsymbol{w} = \mathbf{2000}$ | **2048** | **4096** | **6144** | **8192** | **10240** |
|---|---|---|---|---|---|
| MRSE$_{IITF}$(Cao et al. (2011)) | 16 | 31 | 47 | 64 | 80 |
| Algorithm 12 | 419 | 830 | 1280 | 1627 | 2159 |
| $\boldsymbol{w}$, $\boldsymbol{n} = \mathbf{4096}$ | **2000** | **4000** | **6000** | **8000** | **10000** |
| MRSE$_{IITF}$(Cao et al. (2011)) | 31 | 63 | 92 | 122 | 152 |
| Algorithm 12 | 830 | 1618 | 2439 | 3271 | 4318 |

eral. Table 6.3, which lists the execution times of Algorithm 12 and the algorithm used in MRSE$_{IITF}$(Cao et al. (2011)) for similarity computation, shows the overhead incurred due to the stronger privacy claims in our proposal. Nevertheless, the server side execution times, while important, does not suffice to evaluate the overall performance of a scheme as the execution times of other operations such those spent on client side can be important. To this end, we measure the *client response time* as the time experienced by the client; i.e., DO and/or DU in our scenario. And we show that the performance of our scheme is comparable to that of Cao et al. (2011), if not better for certain parameters, as long as the client response time is concerned. This is due the fact that the performance of the query generation algorithm of Cao et al. (2011) is highly sensitive to the number of words in the dictionary $w$ as the dimension of secret matrices are determined solely by $w$.

The client response time includes the execution times of query generation (Algorithm 11), similarity calculation (Algorithm 12), decryption and ranking of the results (Algorithm 5) as well the time spent on communication. In particular, the query generation time in MRSE_SWHE_SkNN outperforms that of MRSE$_{IITF}$ as can be observed in Table 6.4. The table indicates that the client side computation in MRSE$_{IITF}$ does not scale well as the prohibitively large dimension of secret matrices are involved in the computation. We did not include the timing results of MRSE$_{IITF}$ for $w \geq 8000$ as operating with such large matrices is not practical; especially storing them as secret keys is infeasible for many real world scenarios as discussed subsequently. The dimension of the secret matrix in MRSE_SWHE_SkNN, on the other

Table 6.4 Query generation times of MRSE$_{IITF}$ and MRSE_SWHE_SkNN in milliseconds for various values of $w$

| $w$ | **MRSE**$_{IITF}$(Cao et al. (2011)) | **MRSE_SWHE_SkNN** |
|---|---|---|
| 2000 | 97 | 4 |
| 4000 | 463 | 9 |
| 6000 | 1243 | 13 |
| 8000 | NA | 18 |
| 10000 | NA | 22 |

hand, depends only the security level and thus remains the same for larger values of $w$.

The last operation that affects the client response times is the result decryption and ranking. This phase does not exist in MRSE$_{IITF}$ since the results are returned as plain texts. Figure 6.2 illustrates the result decryption and ranking overhead for various values of $n$ as it is in dependent of $w$. Although the timings increase linearly with $n$, most of it can be overlapped with the similarity calculation which is executed by CS. In the client response time experiments, the queries are processed by CS in overlapping fashion with the DU.

Figure 6.3 plots the client response times when the code for server side computations (i.e., similarity score computation) are parallelized and tasks are distributed over different number of cores in the server CPU. For client side computation, multi-core implementation is not used as we assume client hardware is resource constrained. We set $n = 2048$ and use the number of words returned by the tool set, which is $w = 2445$. As can be observed in Figure 6.3, for certain cases, the proposed scheme can outperform MRSE$_{IITF}$, which is the one of the fastest secure MRSE algorithm in the literature. For larger values of $n$, MRSE$_{IITF}$ can outperform the proposed scheme; but parallelization will always favor the proposed scheme.

Another operation, albeit less important than query generation and similarity score computation, is setup which will be needed in the setup phase or whenever the key is updated. More precisely, setup consists of Algorithms 6 and 7 that are used to find an invertible matrix $\boldsymbol{M}$ and encrypt the index, respectively. It is not explicitly shown in Algorithms 6, but the execution time of computing the inverse of $\boldsymbol{M}$ is also included in the setup time. We measured the setup times for both MRSE$_{IITF}$ and MRSE_SWHE_SkNN for different values of $n$ and $w$ and summarized the results in Tables 6.5. The proposed scheme time performance is superior to MRSE$_{IITF}$ so fast as setup is concerned, which is due to the fact that the latter employs extremely large secret matrices.

Figure 6.2 Result decryption and ranking times of MRSE_SWHE_SkNN with various values of $n$

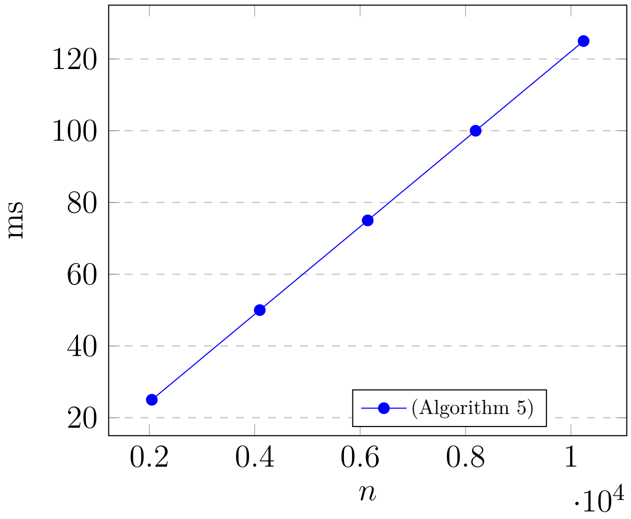
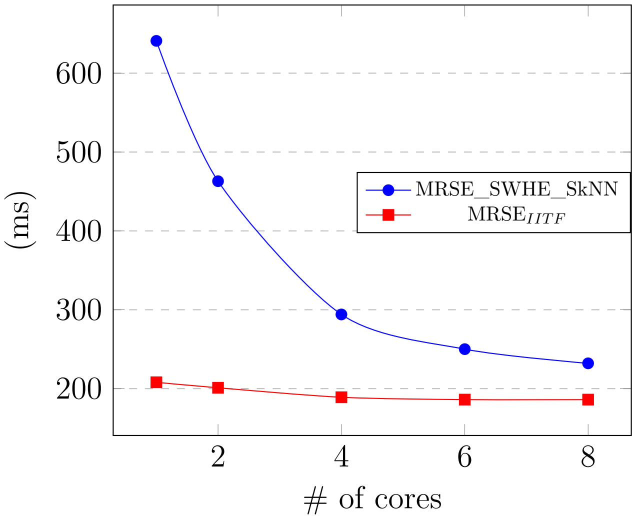


Figure 6.3 Client response times of MRSE$_{IITF}$ and MRSE_SWHE_SkNN for various number of cores used in the server with $n$=2048 and $w = 2445$



### 6.3.2 Storage Requirements

We also compare MRSE schemes in terms of storage requirements for index, secret key and query response. We first plot index sizes of MRSE$_{IITF}$ and MRSE_SWHE_SkNN for varying values of $n$ and $w$ in Figures 6.5 and 6.6. As figures clearly show, the index size of MRSE_SWHE_SkNN is larger, which is due to the ciphertext expansion in SWHE encryption. The increase in index size, however, can be tolerated as it is stored at server (CS) which is generally assumed to posses ample storage.

Next, we compare sizes of the secret keys needed for query generation, which do not include SWHE keys as they are not used in query generation. The results

Figure 6.4 Client response times of $\text{MRSE}_{IITF}$ and MRSE_SWHE_SkNN for various number of cores used in the server with $n$=4096 and $w = 3615$

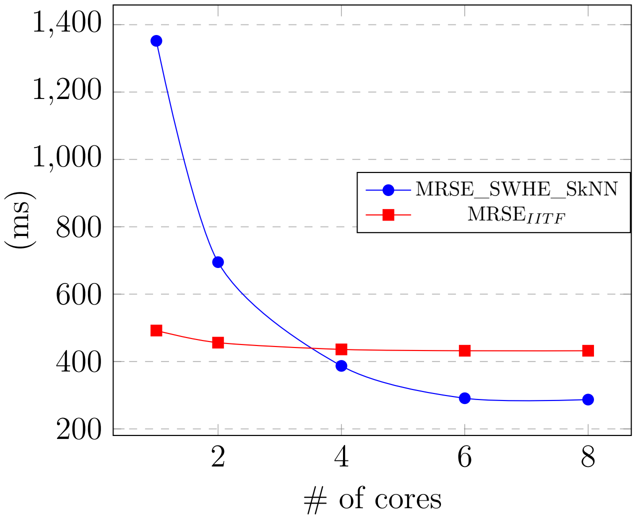


Table 6.5 Setup times of $\text{MRSE}_{IITF}$ and MRSE_SWHE_SkNN for varying values of $w$ and $n$

| $n$, $w$ = 2000 | $\text{MRSE}_{IITF}$(Cao et al. (2011)) | MRSE_SWHE_SkNN |
|---|---|---|
| 2048 | 9m | 6s |
| 4096 | 17m | 12s |
| 6144 | 25m | 18s |
| 8192 | 34m | 24s |
| 10240 | 43m | 31s |
| **$w$, $n$ = 4096** | **$\text{MRSE}_{IITF}$(Cao et al. (2011))** | **MRSE_SWHE_SkNN** |
| 4000 | 87m | 23s |
| 6000 | 256m | 34s |
| 8000 | NA | 46s |
| 10000 | NA | 58s |

are presented in Table 6.6, which shows that the key size of $\text{MRSE}_{IITF}$(Cao et al. (2011)) increases with the number of words in the dictionary. The key size of MRSE_SWHE_SkNN is affected by neither $w$ nor $n$; but only with security parameter. Considering that the secret keys are kept at client side (either DO or DU), key sizes in the order of mega bytes in $\text{MRSE}_{IITF}$ are prohibitively high.

Finally, we inspect the keys sizes for decrypting query results, which applies only in MRSE_SWHE_SkNN since the query results in $\text{MRSE}_{IITF}$ are not encrypted. The size of the decryption keys in MRSE_SWHE_SkNN is 17 KB, as $\mathcal{S}_2$ is used for SWHE purposes. For larger values of $w$, decryption key size can increase as the noise budget requirements will increase. However, $\mathcal{S}_2$ is sufficient for all our test scenarios as mentioned in Section 6.1.

Figure 6.5 Index sizes of MRSE$_{IITF}$ and MRSE_SWHE_SkNN for varying values of $n$ with $w = 2000$

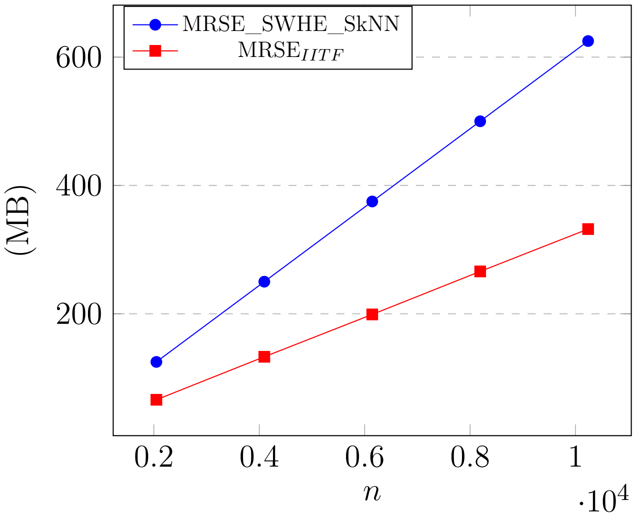
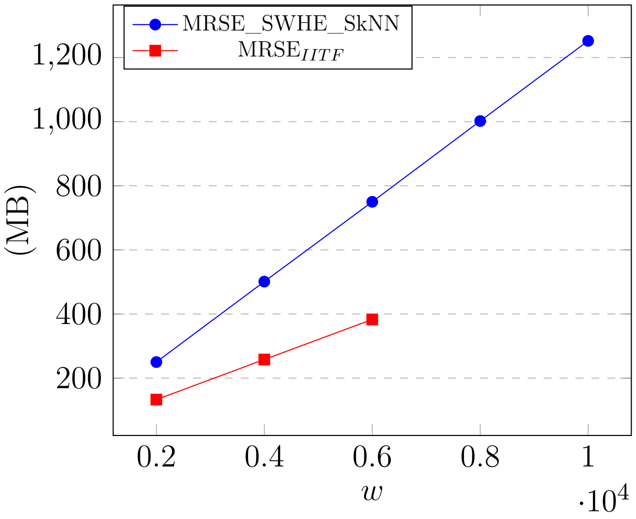


Figure 6.6 Index sizes of MRSE$_{IITF}$ and MRSE_SWHE_SkNN for varying values of $w$ with $n = 4096$



The key size is only affected by $w$ and the parameter set $\mathcal{S}_3$ is sufficient for all values of $w$ used in our experiments.

### 6.3.3 Bandwidth Requirements

We also investigate the sizes of data exchanged between CS and DU during the protocol that includes the query and the response. The results for various values of $n$ and $w$ are presented in Table 6.7. In MRSE$_{IITF}$ the query size is only affected $w$. Naturally, neither $w$ nor $n$ has any influence on response size. In MRSE_SWHE_SkNN, the query size depends on $w$, while the response size is heavily influenced by $n$. MRSE$_{IITF}$ outperforms MRSE_SWHE_SkNN due to the ciphertext expansion in

Table 6.6 Key sizes of MRSE$_{IITF}$ and MRSE_SWHE_SkNN, used for query generation for various values of $w$

| $w$ | **MRSE**$_{IITF}$(Cao et al. (2011)) | **MRSE_SWHE_SkNN** |
|---|---|---|
| 2000 | 70 MB | 1 KB |
| 4000 | 267 MB | 1 KB |
| 6000 | 586 MB | 1 KB |
| 8000 | NA | 1 KB |
| 10000 | NA | 1 KB |

homomorphic encryption. However, bandwidth usage of MRSE$_{IITF}$ grows faster with $w$ comparing to MRSE_SWHE_SkNN. Because each element in the query vector is a 8-byte double instead of a 4-byte integer. The bandwidth requirements of MRSE_SWHE_SkNN can be tolerated in practical applications.

Table 6.7 Bandwidth usage of MRSE$_{IITF}$ and MRSE_SWHE_SkNN for varying values of $n$ and $w$

| $\boldsymbol{n}$, $\boldsymbol{w}$= **2000** | **MRSE**$_{IITF}$(Cao et al. (2011)) | **MRSE_SWHE_SkNN** |
|---|---|---|
| 2048 | 33KB | 47KB |
| 4096 | 33KB | 79KB |
| 6144 | 33KB | 111KB |
| 8192 | 33KB | 143KB |
| 10240 | 33KB | 175KB |
| $\boldsymbol{w}$, $\boldsymbol{n}$ = **4096** | **MRSE**$_{IITF}$(Cao et al. (2011)) | **MRSE_SWHE_SkNN** |
| 2000 | 33KB | 79KB |
| 4000 | 64KB | 95KB |
| 6000 | 95KB | 110KB |
| 8000 | 127KB | 126KB |
| 10000 | 158KB | 142KB |

# Chapter 7

# CONCLUSION

In this work, we present an efficient, highly accurate and secure scheme for multi-keyword ranked searching over encrypted data. The documents in the data set as well as the search queries are represented as *tf-idf* vectors while the comparison metric is cosine similarity. The scheme utilizes a modified variant of secure $k$NN algorithm and SWHE as the most important building blocks. The proposed scheme proved to produce highly accurate results, and there is no trade-off between accuracy and privacy. Our implementation results shown that server side computations are two orders of magnitude faster that the existing schemes in the literature using only of SWHE.

We give an in-depth security analysis of S-$k$NN schemes in the literature and discuss a theoretical attack to better evaluate their security. Our analysis helps construct a simplified variant of S-$k$NN scheme that uses much shorter keys. As a result, a smaller key size and limited use of homomorphic cryptography (only decryption) allow much more lightweight client implementations with faster client side computations and reduced storage requirement. Our security analysis concludes that the new S-$k$NN scheme is IND-CPA secure.

As future work, we will concentrate on reducing the impact of the dictionary size and the number of documents on the performance of the proposed system. As *tf-idf* representation of documents are sparse vectors, the documents can be represented as encrypted *key-value* pairs, encrypted by SWHE. This may reduce the space complexity of both secure search query and the secure searchable index significantly while it potentially increases the time complexity of the search as string comparison is an expensive operation to perform in cipher text form. On the other hand, the bandwidth usage of the proposed protocol increases with the number of documents as space complexity of the search results are $O(n)$. By applying a clustering strategy on the outsourced data set, this can drop to $O(|C|_{max})$ where $|C|_{max}$ denotes the size of the largest cluster. Again it decreases the accuracy of the search in return.

# BIBLIOGRAPHY

Aguilar-Melchor, C., Barrier, J., Fousse, L., & Killijian, M.-O. (2016). Xpir : Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, *2016*(2), 155–174.

Angel, S., Chen, H., Laine, K., & Setty, S. (2018). Pir with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy (SP)*, (pp. 962–979).

Banawan, K. & Ulukus, S. (2018). The capacity of private information retrieval from coded databases. *IEEE Transactions on Information Theory*, *64*(3), 1945–1956.

Boneh, D., Kushilevitz, E., Ostrovsky, R., & Skeith, W. E. (2007). Public key encryption that allows pir queries. In Menezes, A. (Ed.), *Advances in Cryptology - CRYPTO 2007*, (pp. 50–67)., Berlin, Heidelberg. Springer Berlin Heidelberg.

Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, *6*(3), 13:1–13:36.

Cao, N., Wang, C., Li, M., Ren, K., & Lou, W. (2011). Privacy-preserving multi-keyword ranked search over encrypted cloud data. In *2011 Proceedings IEEE INFOCOM*, (pp. 829–837).

Chen, C., Zhu, X., Shen, P., Hu, J., Guo, S., Tari, Z., & Zomaya, A. Y. (2016). An efficient privacy-preserving ranked keyword search method. *IEEE Transactions on Parallel and Distributed Systems*, *27*(4), 951–963.

Chen, L., Qiu, L., Li, K.-C., Shi, W., & Zhang, N. (2017). Dmrs: an efficient dynamic multi-keyword ranked search over encrypted cloud data. *Soft Computing*, *21*(16), 4829–4841.

Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In Takagi, T. & Peyrin, T. (Eds.), *Advances in Cryptology – ASIACRYPT 2017*, (pp. 409–437)., Cham. Springer International Publishing.

Courtois, N., Klimov, A., Patarin, J., & Shamir, A. (2000). Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Preneel, B. (Ed.), *Advances in Cryptology — EUROCRYPT 2000*, (pp. 392–407)., Berlin, Heidelberg. Springer Berlin Heidelberg.

Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2006). Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS ’06, (pp. 79–88)., New York, NY, USA. ACM.

Dawn Xiaoding Song, Wagner, D., & Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*, (pp. 44–55).

Dhumal, A. A. & Jadhav, S. (2016). Confidentiality-conserving multi-keyword ranked search above encrypted cloud data. *Procedia Computer Science*, *79*, 845 – 851. Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016.

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

Fan, J. & Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144. `https://eprint.iacr.org/2012/144`.

Gentry, C. (2009). *A fully homomorphic encryption scheme.* PhD thesis, Stanford University. `crypto.stanford.edu/craig`.

Goldreich, O. & Ostrovsky, R. (1996). Software protection and simulation on oblivious rams. *J. ACM*, *43*(3), 431–473.

Jiang, X., Yu, J., Yan, J., & Hao, R. (2017). Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data. *Information Sciences*, *403-404*, 22 – 41.

Kim, H.-I., Kim, H.-J., & Chang, J.-W. (2017). A secure knn query processing algorithm using homomorphic encryption on outsourced database. *Data & Knowledge Engineering*.

Kipnis, A. & Shamir, A. (1999). Cryptanalysis of the hfe public key cryptosystem by relinearization. In Wiener, M. (Ed.), *Advances in Cryptology — CRYPTO' 99*, (pp. 19–30)., Berlin, Heidelberg. Springer Berlin Heidelberg.

Kushilevitz, E. & Mour, T. (2019). Sub-logarithmic distributed oblivious ram with small block size. In *Public-Key Cryptography – PKC 2019*, volume 11442 of *Lecture Notes in Computer Science*, (pp. 3–33). Springer.

Li, R., Xu, Z., Kang, W., Yow, K. C., & Xu, C.-Z. (2014). Efficient multi-keyword ranked query over encrypted data in cloud computing. *Future Generation Computer Systems*, *30*, 179 – 190. Special Issue on Extreme Scale Parallel Architectures and Systems, Cryptography in Cloud Computing and Recent Advances in Parallel and Distributed Systems, ICPADS 2012 Selected Papers.

Liu, X., Guan, Z., Du, X., Wu, L., Abedin, Z. U., & Guizani, M. (2019). Achieving secure and efficient cloud search services: Cross-lingual multi-keyword rank search over encrypted cloud data. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, (pp. 1–6).

Mayberry, T., Blass, E.-O., & Chan, A. H. (2013). Efficient private file retrieval by combining oram and pir. Cryptology ePrint Archive, Report 2013/086. `https://eprint.iacr.org/2013/086`.

Moataz, T., Mayberry, T., Blass, E.-O., & Chan, A. H. (2015). Resizable tree-based oblivious ram. In Böhme, R. & Okamoto, T. (Eds.), *Financial Cryptography and Data Security*, (pp. 147–167)., Berlin, Heidelberg. Springer Berlin Heidelberg.

Naveed, M. (2015). The fallacy of composition of oblivious ram and searchable encryption. *IACR Cryptology ePrint Archive*, *2015*, 668.

Orencik, C., Alewiwi, M., & Savas, E. (2015). Secure sketch search for document similarity. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, (pp. 1102–1107).

Orencik, C., Kantarcioglu, M., & Savas, E. (2013). A practical and secure multi-keyword search method over encrypted cloud data. In *2013 IEEE Sixth International Conference on Cloud Computing*, (pp. 390–397).

Pham, H., Woodworth, J. W., & Salehi, M. A. (2018). Survey on secure search over encrypted data on the cloud. *CoRR*, *abs/1811.09767*.

Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, *56*(6), 34:1–34:40.

Shetty, J. & Adibi, J. (2004). The enron email dataset database schema and brief statistical report. Technical report.

Stefanov, E., van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., & Devadas, S. (2013). Path oram: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, (pp. 299–310)., New York, NY, USA. ACM.

Strizhov, M. & Ray, I. (2016). Secure multi-keyword similarity search over encrypted cloud data supporting efficient multi-user setup. *Transactions on Data Privacy*, *9*, 131–159.

Wong, W. K., Cheung, D. W.-l., Kao, B., & Mamoulis, N. (2009). Secure knn computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, (pp. 139–152)., New York, NY, USA. ACM.

Xia, Z., Wang, X., Sun, X., & Wang, Q. (2016). A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, *27*(2), 340–352.

Zhao, R. & Iwaihara, M. (2017). Lightweight efficient multi-keyword ranked search over encrypted cloud data using dual word embeddings. *ArXiv*, *abs/1708.09719*.

# APPENDIX A: SECURITY PROOFS

Our security claims rely on the difficulty of solving multivariate polynomial equations generated during the matrix-vector multiplications between the randomized query $\bar{\boldsymbol{q}}$ and the secret matrix $\boldsymbol{M}$. We use the results from Courtois et al. (2000) extensively in our security analysis.

**Theorem 6.** *The query encryption scheme* mSkNN.EncryptQuery *presented in Algorithm 8 is CPA secure.*

**Proof:** Assume adversary has access to $\mu$ pairs of plaintext of $\boldsymbol{q}$ and ciphertext $\bar{\boldsymbol{q}}$, namely $\mathcal{Q} = \{\boldsymbol{q}_0, \ldots, \boldsymbol{q}_{\mu-1}\}$ and $\bar{\mathcal{Q}} = \{\bar{\boldsymbol{q}}_0, \ldots, \bar{\boldsymbol{q}}_{\mu-1}\}$. Recall $\boldsymbol{q'}$ is a $2w$-dimensional vector, which is the random split of $\boldsymbol{q}$. $\mathcal{Q}' = \{\boldsymbol{q'_0}, \ldots, \boldsymbol{q'_{\mu-1}}\}$ denotes the set of random splits of $\bar{\boldsymbol{q}}_k$. A ciphertext vector $\bar{\boldsymbol{q}}_k$ can be formulated as

$$\bar{\boldsymbol{q}}_k[i] = \sum_{j=0}^{\eta-1} \boldsymbol{M}[i \bmod \eta][j]\boldsymbol{q'}_k[\lfloor i/\eta \rfloor \eta + j] \tag{A.1}$$

for $0 \leq k < \mu$ and $0 \leq i < 2w$. Note that exactly half of the dimensions of the vector $\boldsymbol{q'_k}$ are independently random due to the random splitting process and the other half are dependent (see lines 1-5 in Algorithm 8). In Eq.(A.1), there are $\eta^2$ unknowns due to $\boldsymbol{M}$ and $\mu w$ due to $\mathcal{Q}'$, resulting in a quadratic equation system with a total of $\eta^2 + \mu w$ unknowns and $2\mu w$ equations.

Courtois et al. (2000) investigate the best strategies for solving quadratic equations of the form,

$$b_k = \sum_{1 \leq i \leq j \leq \nu'} a_{ijk} x_i x_j$$

for $k = 1, 2, \ldots, \mu'$. As the equation system in(Courtois et al. (2000)), with $\mu'$ equations for $\nu'$ unknowns $x_1, x_2, .., x_{\nu'}$, is in identical form to ours in Eq.(A.1), we can directly adopt the results from Courtois et al. (2000) in our security analysis.

Courtois et al. (2000) propose a method known as *XL(extended linearization)*, which is the improved version of the *relinearization* technique proposed by Kipnis & Shamir (1999) and investigate the computation complexity of *XL*. Among various results given, we adapt those for the cases relevant in our analysis:

- **Case 1:** $\mu' > \nu' + C$ with $C \geq 2$ represents the case when the number of equations is slightly larger than the number of unknowns. The authors formulate the computational difficulty of solving the equation system as the *working factor* and give $WF_1 \geq e^{\alpha\sqrt{\nu'}(\frac{\ln\nu'}{2}+1)}$, where $\alpha$ stands for the complexity of solving linear equation systems. For instance, $\alpha = 3$ for Gaussian elimination and $\alpha = 2.3766$ for improved methods(Courtois et al. (2000)).
- **Case 2:** Alternatively, $\mu' > \epsilon\nu'^2$ for $\epsilon > 0$ represents the case when the number of equations is much larger than the number of unknowns. The goal here is to see whether the computational complexity decreases with insertion of new equations; which means in our case to observe more plaintext-ciphertext pairs. The formula given by the authors for the working factor is $WF_2 \approx \nu'^{\frac{\alpha}{\sqrt{\epsilon}}}$.

In our analysis for both cases the number of equations is $\mu' = 2\mu w$ and the number of unknowns is $\nu' = \eta^2 + \mu w$. In the first case, we set $2\mu w > \eta^2 + \mu w + 2$ with $C = 2$, smallest possible value for the number of equations. Then, we have $\mu > (\eta^2 + 2)/w$. As $\nu' = \eta^2 + \mu w$, we can obtain $\nu' > 2\eta^2 + 2$. Consequently,

$$WF_1 \geq e^{\alpha\sqrt{\nu'}(\frac{\ln\nu'}{2}+1)} > e^{\alpha\eta(\ln\eta+1)} > e^{\alpha\eta\ln\eta}$$

$$WF_1 > (e^{\ln\eta})^{\alpha\eta} = \eta^{\alpha\eta}. \tag{A.2}$$

Thus, $WF_1$ is lower bounded by $\eta^{\alpha\eta}$.

In the second case, $\mu' = \epsilon\nu'^2$ means $2\mu w = \epsilon(\eta^2 + \mu w)^2$. Expanding its right hand side, the equation becomes $2\mu w = \epsilon(\eta^4 + 2\eta^2\mu w + (\mu w)^2)$ leading to the following second degree equation

$$\epsilon(\mu w)^2 + (2\eta^2\epsilon - 2)\mu w + \eta^4\epsilon = 0. \tag{A.3}$$

Eq.(A.3) is a second power polynomial in the form of $ax^2 + bx + c$, where $x = \mu w$, $a = \epsilon$, $b = (2\eta^2\epsilon - 2)$ and $c = \eta^4\epsilon$. In order for Eq.(A.3) to have solution for $\mu w$, its discriminant must be larger than or equal to 0

$$\Delta = (2\eta^2\epsilon - 2)^2 - 4\epsilon^2\eta^4 = (2\eta^2\epsilon - 2)^2 - (2\eta^2\epsilon)^2.$$

Otherwise, adversary is not able to attack to the system through by collecting more plaintext-ciphertext pairs, which is not meaningful. Then,

$$4\eta^4\epsilon^2 - 8\eta^2\epsilon + 4 - 4\eta^4\epsilon^2 > 0$$
$$4 > 8\eta^2\epsilon \Rightarrow \epsilon < 1/(2\eta^2) \Rightarrow 1/\epsilon > 2\eta^2.$$

Finally, we obtain the inequality

$$1/\sqrt{\epsilon} > \sqrt{2}\eta \tag{A.4}$$

Consequently, for the working factor of the second case we have

$$WF_2 \approx \nu'^{\frac{\alpha}{\sqrt{\epsilon}}} = (\eta^2 + \mu w)^{\frac{\alpha}{\sqrt{\epsilon}}}$$
$$(\eta^2 + \mu w)^{\frac{\alpha}{\sqrt{\epsilon}}} > \eta^{\frac{2\alpha}{\sqrt{\epsilon}}} > \eta^{2\sqrt{2}\alpha\eta}.$$

Clearly, $WF_2$ is also lower bounded by $\eta^{\alpha\eta}$ and we can conclude that solving the equation system in Eq.(A.1) is infeasible for sufficiently large $\eta$ values.

Also observe that, our analysis suggests that liberty of using special queries (i.e., chosen plaintext) does not decrease the number of unknowns. Hence, we do not expect that solving Eq.(A.1) gets easier in chosen plaintext attack scenario. Consequently, we conclude that the query encryption algorithm mSkNN.EncryptQuery is CPA secure. ■

**Theorem 7.** *The query encryption scheme* mSkNN.EncryptQuery *presented in Algorithm 8 is IND-CPA secure.*

**Proof:** To show an encryption algorithm is IND-CPA secure is harder in general. To this end, the following IND_CPA security game is played between adversary and challenger, in which adversary has access to the encryption oracle (i.e., she can obtain *a* ciphertext corresponding to arbitrarily chosen plaintext).

```
Adversary:
    Picks two arbitrary queries $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$
Challenger:
    $b \leftarrow \{0,1\}$ (e.g., flips a coin)
    $\boldsymbol{c} \leftarrow$ mSkNN.EncryptQuery($\boldsymbol{M}, \boldsymbol{q}_b$)
Adversary:
    Outputs $b'$

Adversary wins if $b = b'$
```

As the adversary can always make a random guess of $b$, the advantage of the adversary (A) in the game can be defined as

$$\textsc{Adv}^{\text{IND-CPA}}_{\text{mSkNN.EncryptQuery}}(\text{A}) = 2 \cdot \left| \Pr[\text{A wins}] - \frac{1}{2} \right|$$

The encryption scheme is said to be IND-CPA secure if $\textsc{Adv}^{\text{IND-CPA}}_{\textsc{mSkNN.EncryptQuery}}(\mathrm{A})$ is “small" for all polynomial adversaries A.

Theorem 6 states that no polynomially bounded adversary is able to decrypt a ciphertext and learn to $b$ as mSkNN.EncryptQuery is CPA-secure. The adversary, on the other hand, can make use of the probability distributions of ciphertexts to increase its advantage. Here, we show that the ciphertexts generated by mSkNN.EncryptQuery are uniformly randomly distributed in $\mathbb{Z}_p^{2w}$ and independent of the corresponding plaintexts.

Suppose $\bar{\boldsymbol{q}} = \textsc{mSkNN.EncryptQuery}(\boldsymbol{q})$ for an arbitrary query vector $\boldsymbol{q}$. Let $\boldsymbol{\rho} = (\rho_0, \rho_1, \ldots, \rho_{w-1})$ be the vector, whose elements are random values sampled in Step 2 of Algorithm 8. Then, Eq.(5.2) can be re-arranged into the form

$$\begin{aligned}\bar{\boldsymbol{q}}[i] = & \sum_{j=0}^{\eta/2-1} \boldsymbol{M}[i \bmod \eta][2j+1]\boldsymbol{q}[\lfloor i/\eta \rfloor \eta/2 + j] \\ & + \sum_{j=0}^{\eta/2-1} \rho[\lfloor i/\eta \rfloor \eta/2 + j](\boldsymbol{M}[i \bmod \eta][2j] - \boldsymbol{M}[i \bmod \eta][2j+1])\end{aligned}$$

for $k = 0, \ldots, 2w-1$. Since each element of $\boldsymbol{\rho}$ is uniformly randomly distributed in $\mathbb{Z}_p$, the equation becomes

$$\bar{\boldsymbol{q}}[i] = \rho' + \sum_{j=0}^{\eta/2-1} \boldsymbol{M}[k][2j+1]\boldsymbol{q}[\lfloor i/\eta \rfloor \eta/2 + j],$$

where $\rho'$ is also uniformly randomly distributed in $\mathbb{Z}_p$ as $k\mathbb{Z}_p \simeq \mathbb{Z}_p$ for any $k \in \mathbb{Z}_p$, where $p$ is prime. Moreover, a variable $u \in \mathbb{Z}_p$ is uniform in $\mathbb{Z}_p$ if $u = \rho + \theta$ for an arbitrary $\theta \in \mathbb{Z}_p$ and uniformly random $\rho \in \mathbb{Z}_p$. In conclusion, $\bar{\boldsymbol{q}}[i]$ in Eq.(5.2) is uniformly randomly distributed in $\mathbb{Z}_p$ and therefore independent from the plaintext query $\boldsymbol{q}$. ■

Note that the proof of Theorem 7 does not hold if we work with real numbers in Algorithms 7 and 8 as they cannot always be sampled uniformly randomly in $\mathbb{R}$. Therefore, the function $\mathrm{H} : \mathbb{R}^w \to \mathbb{Z}_p^w$ in those algorithms plays an essential role in our proofs.