ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

SDCA: A SECURE AND PRIVACY PRESERVING DATA RETRIEVAL ARCHITECTURE ON CLOUD COMPUTING

M.Sc. THESIS

Büşranur BÜLBÜL

Department of Computer Engineering

Computer Engineering Programme

DECEMBER 2019

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

SDCA: A SECURE AND PRIVACY PRESERVING DATA RETRIEVAL ARCHITECTURE ON CLOUD COMPUTING

M.Sc. THESIS

Büşranur BÜLBÜL
(504161553)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Assoc.Prof.Dr. Deniz Turgay ALTILAR

DECEMBER 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SDCA: BULUT BİLİŞİMDE GÜVENLİ VE MAHREMİYET KORUMALI VERİ GETİRME MİMARİSİ

YÜKSEK LİSANS TEZİ

Büşranur BÜLBÜL
(504161553)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Assoc.Prof.Dr. Deniz Turgay ALTILAR

ARALIK 2019

Büşranur BÜLBÜL, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 504161553 successfully defended the thesis entitled “SDCA: A SECURE AND PRIVACY PRESERVING DATA RETRIEVAL ARCHITECTURE ON CLOUD COMPUTING ”, which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc.Prof.Dr. Deniz Turgay ALTILAR** Istanbul Technical University	.....
Jury Members :	**Assis.Prof.Dr. Mehmet Tahir SANDIKKAYA** Istanbul Technical University	.....
	Assoc.Prof.Dr. Muhammed Ali AYDIN Istanbul University	.....

Date of Submission : **14 Novemver 2019**
Date of Defense : **13 December 2019**

To my family,

FOREWORD

I would like to start with my deepest gratitude to my thesis supervisor Assoc. Prof.Dr.D.Turgay Altılar for his guidance and support all the time. He always encouraged me throughout the progress of my thesis study.

I would like to thank Dr.Mohanad Dawoud for his help. In this thesis, his labor and guidance are more.

I would like to thank my father, my mother and my sisters special for their supports me. Thanks also to my fiance motivated me with love.

December 2019 Büşranur BÜLBÜL

TABLE OF CONTENTS

Page

ABBREVIATIONS

AHEE	**:** Algebra Homomorphic Encryption Scheme
HEADA	**:** A Low Cost RFID Authentication Technique Using Homomorphic Encryption for Key Generation
NIST	**:** National Institute of Standards and Technology
RFID	**:** Radio-Frequency Identification
TF	**:** Term-Frequency
TF-IDF	**:** Term-Frequency-Inverse Document Frequency

LIST OF TABLES

Page

LIST OF FIGURES

Page

SDCA: A SECURE AND PRIVACY PRESERVING DATA RETRIEVAL ARCHITECTURE ON CLOUD COMPUTING

SUMMARY

In recent years,the interest to the cloud computing increases reasonably. The reason of that, the cloud computing provides important advantages like dynamic allocation of cloud resources. It can be accessed to the resources and their use can be increased or decreased according to requirements. Besides various advantages of cloud computing, privacy and security issue which are basically inherited from distributed system are yet to be addressed.

Even in our daily use of WEB, we are all using browsers for searching. Although the result for the searches will be appearing on the browser immediately, it is also logged and tagged with some kind of identity such as username, or machine MAC id. That ties the search keys and users or machines. The cloud can use the information and share with the other company relative from the user's requirements. Although the sharing is look like advantage in terms of accessing the needs, this case creates a huge privacy deficit. So that, the cloud should store the data as encrypted and protect the privacy of the user. When it provides security and privacy, it should be able to make the system efficient.

In the proposed architecture, the new servers (ranking server, private server, data server, authentication server and encryption server) are added on the defined basic cloud system. The new architecture uses the Fully Homomorphic Encryption for making calculation process on the encrypted data, the AES symmetric encryption to store the data. For providing the data index privacy, Term Frequency-Inverse Document Frequency table and Query Term Frequency vector are normalized before encryption. For anonymity, the system applies HEADA authentication protocol. With the methods, the new system supplies the main security and privacy requirements.

The proposed technique is applied on using two different datasets. The results of the simulation show that the proposed architecture provides security and privacy requirements as well efficiency.

SDCA: BULUT BİLİŞİMDE GÜVENLİ VE MAHREMİYET KORUMALI VERİ GETİRME MİMARİSİ

ÖZET

Bilgi Teknolojilerinin gelişmesi ile beraber daha fazla veri depolama ve daha hızlı işlemler yapabilecek kaynak gereksinimleri artmıştır. Bu ihtiyacı gidermek için Bulut Bilişim kavramı ortaya çıkmıştır. Bulut bilişim kullanıcılarına esnek bir yapı içinde teorik olarak sınırsız depolama alanı ve hesaplamalar yapmak için yüksek performanslı kaynaklar sunar. Kullanıcı bu kaynakları ve depolama alanını kullanacağı kadar öder, kaynak ve hizmet kullanımını kullanım süresi içinde artırabilir ya da azaltabilir. Kullanıcı bulut kaynaklarına her yerden istediği zaman erişebilir. Bu açıdan bulut bilişim kullanıcılara ve geliştiricilere avantajlı bir ortam sağlar. Kullanım kolaylığı açısından kullanıcıya sistemin işleyişi, kaynak kullanımı gibi konularda bilgi verilmez.

Bulut bilişimin avantajları yanında dezavantajları da mevcuttur. Sınırsız kaynak ve depolama alanı kullanımı önemli güvenlik açıklarına yol açabilir. Örneğin; kullanıcının verileri bulut hizmet sağlayıcılarında depolanırken kullanıcı bu verilerin nasıl saklandığı hakkında bilgi sahibi değildir. Kullanıcının verileri depolanır, çözümlenir ve yorumlanabilir, kullanıcı bu süreçleri kontrol edemez. Bu durumda kullanıcı sadece hizmet sağlayıcı ile arasında güven ilişkisi sağlar. Bu ilişki uygun ve yeterli görülse bile bir dağıtık sistem olarak bulut verileri ataklara açıktır. Bu nedenle hizmet sağlayıcısıyla güven ilişkisi kurulsa bile bulut bilişimde ciddi güvenlik ve mahremiyet sorunları oluşabilir. Örneğin; günlük hayatımızda sıkça kullandığımız banka işlemleri, internet üzerinden yaptığımız alışverişler, hasta bilgileri, arama motorlarına yapılan aramalar vb. gibi işlemler ile kullanıcıların hassas verileri kolaylıkla bulut bilişim tarafından saklanabilir ve bu bilgiler üçüncü taraflar ile paylaşılabilir. Arama motorlarında arattığımız herhangi bir sorgu ile ilgili reklamların çıkması bu duruma örnektir.

Bulut bilişimde birçok güvenlik ve mahremiyet problemleri mevcuttur. Bu tezde güvenlik, mahremiyet ve mahremiyet korumalı veri çağırma problemine çözüm için çalışmalar yapılmıştır. Literatürde bu problem ile ilgili yapılan birçok çalışma mevcuttur. Bu çalışmalarda da görüleceği gibi güvenliği sağlamak için yapılacak en önemli adım veriler üzerinde şifreleme yapmaktır. Veri sahibi verilerini bulut hizmet sağlayıcılarına gönderirken veriler şifrelenmeli, kullanıcılar verilerini sorguladığı zaman da istediği verilere ulaşabilmelidir. Kullanıcı bulut sunucular üzerinde arama yaparken kimliğini gizli tutabilmeli, bulut üzerinde bir iz bırakmamalıdır. Bunun yanında, kullanıcının verilerine başka bir kullanıcı (gerçek kullanıcı ya da sahte kullanıcı) erişim yapamamalıdır. Bunlar bulut bilişimde güvenlik ve mahremiyet için gereken en temel durumlardır.

Literatürde tanımlanan temel bulut bilişim mahremiyet korumalı veri getirme mimarisinde eleman olarak veri sahibi, kullanıcı ve bulut sunucu bulunur. Bulut sunucu depolama, hesaplama, veri gönderme gibi işlemleri sadece kendisi yapar bu da

bulut sunucunun kullanıcı verilerinden çok rahatlıkla çıkarımlar yapmasını sağlar.Aynı zamanda bu sunucu dürüst ama meraklı (honest but curious) olarak tanımlanmıştır. Bunun anlamı bulut sunucusu sisteme zarar vermez, ancak veri toplayıp çıkarımlar yapabilir.

Bu tezde temel mahremiyet korumalı veri çağırma mimarisinde bulut sunucunun yerine birbirleri ile bilgi alışverişi yapan yeni sunucular eklenmiştir. Bunlar; arama sunucusu, sıralama sunucusu, özel sunucu, şifreleme sunucusu, kimlik doğrulama sunucusu ve veri sunucusudur. Bu sunucuların her birinin ayrı görevi vardır. Yine bu sunucular da dürüst ama meraklı olarak tanımlanmış olmalarına rağmen sahip oldukları verilerden bir çıkarımda bulunamayacak şekilde tasarlanmışlardır.

Tezde verilerin şifrelenmesi için homomorfik şifreleme yöntemi kullanılmıştır. Bu yöntem şifreleme yöntemi şifreli veriler üzerinde işlem yapmaya izin verir. Homomorfik şifreli veriler üzerindeki işlem sonucu, şifresiz veriler üzerindeki işlem sonucunun homomorfik şifrelenmesi ile elde edilen sonucun aynısıdır. Homomorfik şifreleme yöntemi ile terim sıklığı-ters belge sıklığı (TF-IDF) tablosu ve sorgu terim sıklığı (QTF) vektörü şifrelenir ve bu şifreli değerlerden dokümanların kosinüs benzerlik değerleri hesaplanır. Bu benzerlik değerleri de homomorfik şifrelemeden dolayı şifrelidir ve sadece homomorfik şifreleme anahtarına sahip olanlar bunu çözebilir. Bunun dışında dokümanların şifrelenip saklanmasında simetrik şifreleme algoritması olan AES kullanılır.

Bulut sunucu tarafından herhangi bir frekans saldırısını engellemek için TF-IDF tablosu ve QTF vektörü şifrelemeden önce normalleştirilirler. Normalleştirme işlemi ile tablo ve vektördeki bütün değerler birbirine yakın ancak farklı değerler olurlar, şifrelemeden sonra ise bu yakın değerler birbirinden tamamen uzaklaşarak eşsiz değerler haline gelirler.

Bu tezde kullanılan bir diğer yöntem HEADA adı verilen ve RFID sistemleri için tasarlanmış olan kimlik doğrulama protokolüdür. Bu protokol ile kimlik doğrulama sunucusu ve kullanıcı arasında sadece ikisinin bildiği bir iletişim vardır. Kimlik doğrulama sunucusu anahtar seti üretir ve kullanıcı ile paylaşır. Kullanıcı bu anahtar setinden sadece kendisinin ve kimlik doğrulama sunucusunun bildiği oturum anahtarı, kimlik doğrulama biti, kendisine ait kimlik bilgisi ve anahtar üretir. Kimlik doğrulama sunucusu kullanıcıyı oturum anahtarına göre kabul eder ya da reddeder. Bu protokol ile her kullanıcı her sorguda eşsiz anahtarlar kullanır böylelikle anonimlik sağlanmış olur.

Tezde tanımlanan yöntemler ve mimari indekslenebilir veri seti üzerinde gerçekleştirilmiştir. Gerçekleştirme için gerekli ön işlemler yapılmış, anahtar kelimeler belirlenmiştir. Bu anahtar kelimelere göre indeksleme, normalleştirme, şifreleme, işlemleri gerçeklenmiştir ve kullanıcı sorgu sonuçları benzerliğe göre sıralı alarak alınmıştır.

Tez çalışmasında iki durum için sonuçlar vermiştir. İlk durumda tek seferde bir kullanıcının sorgu yapmasına izin veren mimari kullanılmıştır. Bu mimaride şifreleme yöntemleri ve normalleştirme sisteme ne gibi avantajlar ya da dezavantaj sağladığı, kullanıcının sorgu sonuçlarına nasıl etkisi olduğu gözlemlenmiştir. İkinci durumda ise kullanıcılar için belirleyici ama kimlik mahremiyetine zarar vermeyen değişkenler eklenmiş ve aynı anda birden fazla kullanıcının sorgu yapmasına izin veren mimari oluşturulmuştur.

Bu mimari ile de çalışmada sağlanması amaçlanan güvenlik ve mahremiyet gereksinimleri olan veri mahremiyeti, kimlik mahremiyeti, indeks mahremiyeti ve sorgu mahremiyeti sorunlarına çözüm amaçlanmıştır. Literatürde olan çalışmalarda ise bu mahremiyet gereksinimlerinden sadece bir ya da birkaçı için çözüm üretilmiştir.

Tasarlanan mimari ve gerçekleştirme çalışmalarının sonucunda, sistemde güvenlik açığı oluşturabilecek değişkenler aynı sunucuda bulundurulmadığından çalışmanın sonunda önerilen modelin bulut bilişimde güvenlik ve mahremiyet korumalı veri çağırma problemine çözüm getirdiği gözlemlenmiştir.

1. INTRODUCTION

With increasing the usage of the Information Technologies is appeared resources, data storages and performance requirements. More times, technological devices are inadequate for data storage and computational performance. Even if the devices are become effective, the cost of them increases. Because of the requirements, the cloud computation approach comes up. The cloud computing provides unlimited data storage and effective computation resources for users theoretically. Although the cloud computing has the advantages, the cost of it is more available because the characteristic of the cloud computing is “pay as you go”.

1.1 Cloud Computing

One of the definitions of the cloud computing is given by NIST (National Institute of Standards and Technology) [1] that is *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.”*

The NIST also presented the five cloud computing characteristics that are:

1. The characteristics of the cloud computing

   "On-demand self-service: *A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider."*

   "Broad network access: *Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations)."*

*"**Resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth."*

*"**Rapid elasticity:** Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time."*

*"**Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service."*

2. Delivery (Service) Models

SaaS: The service provides software programs on the cloud that users need them. Gmail, Salesforce.com and SAP Business ByDesign are examples of SaaS providers.

PaaS: The service provides software and hardware layer for program developers. The service presents platforms like operating system, database, the developers only manage provided system. Examples of the service providers are Google App Engine, Microsoft Windows Azure and Force.com .

IaaS: The Service is the most essential Service provider. The provider builds virtual machine and allocates the infrastructural resources to user. Amazon EC2, GoGrid and Flexiscale are some IaaS providers .

3. Delivery (Service) Models

Public Cloud: In the model, services are provided by service providers publicly on the Internet. The model requires a high level of security and an efficient management control.

Private Cloud: The model is built for a single organization and provides high security with efficient management control. The model is more expensive than the public model.

Hybrid Cloud: The model is composed a combination of two or more private and public clouds in the same organization.

Community Cloud: The model provides services for more than one organization needed the same requirements.

According to Kim [2], the major advantages of the cloud computing from the perspective of users are as follows;

- All the computing resources (servers, software, storage and networking) and electricity needed for the services are owned and managed by the 3rd party provider.
- The usage of the computing resources and services can be decreased or increased by the users flexibly and easily.
- The users pay only for the computing resources and services they use.
- The users can access to cloud for services anytime from anywhere.

1.2 Security and Privacy on the Cloud Computing

Although cloud computing comes with the above mentioned advantages, security, privacy, performance and availability issues are still problematic. Figure 1.1 is shown the challenges and issues of the cloud computing at 2009, the survey is researched by IDC [3]. Figure 1.2 is shown the challenges and issues of the cloud computing at 2019 [4]. Even after 10 years, the privacy and security have been the top concern in cloud computing. Although the security and privacy of data and applications are vitally important, it might not be provided efficiently by the cloud.

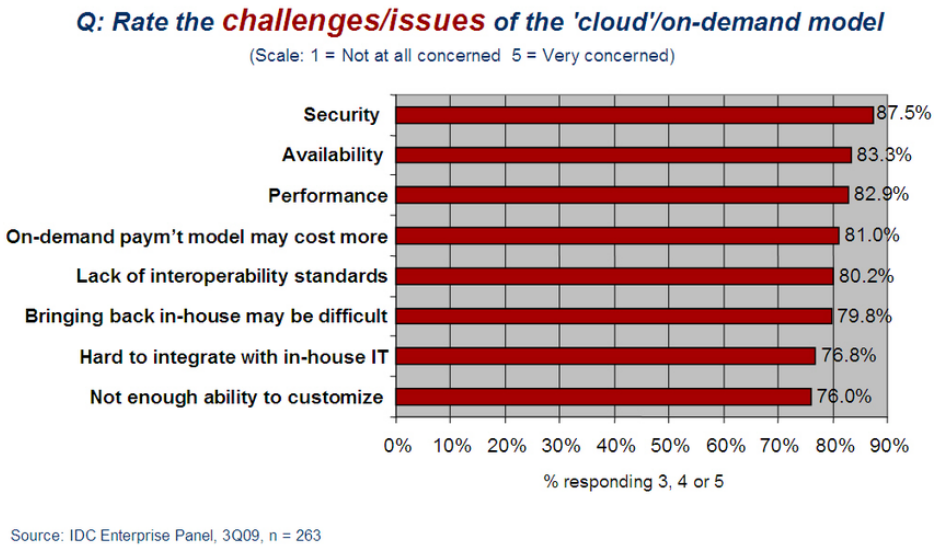

Figure 1.1 : The issues of the cloud computing in IDC survey at 2009[3].

Figure 1.2 : The issues of the cloud computing at 2019 [4].

In the research [5], the security threats, requirements and solutions are explained. According to the research, two kinds threats are occurred; threats from external attackers and threats from internal attackers.

- The external attackers are malicious and they use some attack techniques that are network eavesdropping, vulnerability scanning and malware attacking e.g. They may access unauthorized; they may delete the data outsourced to cloud and infer private behaviors of the user.

- The internal attackers are happened by cloud system participants in the cloud system. the cloud service provider (CSP) and third party auditor (TPA) are not

fully trusted participants. CSP and TPA are considered honest but curious, and CSP is considered vulnerable or greedy provider.

The threats cause some security and privacy problems [6, 7], such as data leakage or disclosure, unauthorized access, data corruption or loss and user privacy.

As shown in the literature [5, 8–10] the efficient security and privacy, can be provided by ensuring five characteristics that are availability, confidentiality, data integrity, data access controllability and privacy preservability.

Availability: Achieving users of the cloud computing can use the application and infrastructures at any time and at anywhere.

Confidentiality: Providing user's data hiding on the cloud. The data are not made available or disclosed to unauthorized users.

Data integrity: Protecting information integrity is to prevent loss of data or modifications. when any undesirable operations corrupt or delete the data, the data owner should be able to detect them.

Data access controllability: Controlling the access authorization, the data should be accessed by only authorized users.

Privacy preservability: Providing that the access behaviors and habits of the user should not be infered by any other parties in the cloud.

The unique security and privacy requirements on data clouds are shown in [11]. Five important privacy requirements are defined by the researches. The requirements are identity privacy, location privacy, data privacy, output privacy and function privacy. The researches express the importance of the requirements for users and primitive solutions.

For the security and privacy requirements in cloud, different data and privacy protection solutions have been developed [5], the solutions are shown in Table 1.1.

Table 1.1 : Security and privacy solutions.

Requirements	Solutions
Secure data search	Searchable Encryption (SE)
Secure data computition	Homomorphic Encryption (HE)
Secure data sharing	Selective Encryption Attribute-Based Encryption (ABE)
Secure data storage	Provable Data Possession (PDP) Proof of Retrievability (POR)
Privacy preservation data access	Access Pattern Protection Query Privacy Protection Identity Privacy Protection

1.3 Purpose of Thesis

In the thesis, the SaaS service model security is researched. There are several highlighted security issues in SaaS such as data security, network security, data locality, data integrity, data segregation, data access, authentication and authorization. The purpose of the thesis is providing privacy preserving search on data clouds. Privacy preserving search has been researched for the last five years. Some of the representative works are briefly mentioned in Section 1.4. However although the works solve the some security and privacy problems, they are inadequate for the cloud systems.

In the literature, the problems have been investigated extensively in the [12]. A novel architecture addressing 9+1 security and privacy requirements is designed. The thesis is based on his thesis architecture [13], new entities are added on the existing architecture to improve the key security by only sending the key to the encryption server, the performance of the new architecture and validation of the protocol is tested by simulation. The previous architecture have been designed for a single user although multiple users may use it exclusively. We have also designed the architecture which is concurrently.

Fully Homomorphic Encryption (FHE) and AES symmetric encryption algorithm are used on the data for the data security on the cloud computing. FHE is applied on the normalization TF-IDF table that includes the unique values thanks to normalization process. AES is applied on the documents directly. An anonymous authentication protocol called HEADA [14]is used for privacy preserving. The protocol provides

anonymity and uniquely for all user. The algorithms and protocol are explained in the following sections.

1.4 Literature Review

In this subsections, some essential researches about security and privacy issue on cloud computing are represented. The researchers in [15] propose protecting the user sensitive data on cloud and they use fully homomorphic encryption with different public key for this purpose. They apply a machine learning technique (classification) on the encrypted data and return encrypted result. In [16] deep learning technique is applied on the encrypted sensitive data with full homomorphic encryption. The authors provide the sensitive data privacy on the cloud by encrypting these data twice with user public key and CSP public key.

Location privacy on cloud is tackled in [17]. A framework which is called "PROPHET" is developed. The framework includes a user, the PROPHET that is the honest middleware server and location based server(LBS) that is an honest-but-curious server. The PROPHET offers warning service, analyzing, mining user's behavior pattern from location history and location anonymity. The researches have used Markov Chains and e-indistinguishable anonymity mechanism (New dummy locations are generated with Anonymous Space Region (ASR).) for providing location privacy in the system.

In the research [10], a problem of multi-keyword ranked search over encrypted cloud data is explained and solved. The research establishes a variety of privacy requirements. In the work, the symmetric public key encryption algorithm is used. Among various multi-keyword semantics, they choose the efficient similarity measure of "coordinate matching" to effectively capture the relevance of outsourced documents to the query keywords, and use "inner product similarity" to quantitatively evaluate such similarity measure.

The purpose of the authors in [18] is searchable encryption and privacy preserving. For that, searchable encryption (SE) is allows encrypted data to be queried without decryption. The authors created Dynamic Forward Privacy (DFP) scheme, which can achieve both forward-looking privacy and multiuser controlled setting. This scheme

ensures that sensitive stored files and data users in the cloud repository are protected to search for encrypted files under the control of the data owner.

A new medical cloud computing approach that addresses privacy issues with the cloud provider has been developed in [19]. The approach makes use of the FHE, which allows calculations of encrypted health information without complying with basic data. The authors defined three performance metrics related to the operation of the FHE: Γ, Λ, and γ determined the computational, storage, and bandwidth requirements of processes.

The [20] research is proposed an Effective and Secure Privacy Protection Approach (ESPPA) based on possible public key cryptography and multiple keyword search. They improve the efficiency of the program properly by using the probable public key encryption technique, rather than other encryption techniques for file encryption and the integrity of the data is verified. With the security and performance analysis of ESPPA provides the security and efficient requirements is proved. ESPPA is a mechanism that allows the user to search with keywords ranked in the encrypted data. It provides a way to protect the confidentiality of the owner's outsourced data and allow the user to search efficiently without decrypting the encryption text.

1.5 The Organization of The Thesis

The thesis consist of 7 chapters. In Chapter 1, the essential problem that is and relative literature are introduced. Chapter 2 includes the methodologies that are used in the research. Chapter 3 gives information about the problems of the basic secure searching system for a cloud system and describes the proposed solutions. Its simulation results are shown in Chapter 4. The shortcomings of the structure and proposed new solution are explained in the Chapter 5. The results and contributes of the new system are indicated in Chapter 6 In the last chapter of the thesis conclusions are explained.

2. METHODOLOGY

2.1 Homomorphic Encryption

Homomorphic encryption is an encryption method that allows performing operation on the encrypted data such that the result of the operation is the same as the result produced by applying the some operation to raw data.

$$D(E(x) \bullet E(y)) = x \bullet y$$

Figure 2.1 shows the use of the homomorphic encryption. x and y are plaintexts, a and b are ciphertexts that are generated with homomorphic key K_h. $\bullet$ operation is applied on the x and y and obtain z. Similarly, $\bullet$ operation is applied on the a and b encrypted data and c is obtained. $\bullet$ operation is can be addition, subtraction, multiplication or division operations. If c is decrypted with K_h, z is obtained thanks to the property of the homomorphic encryption.

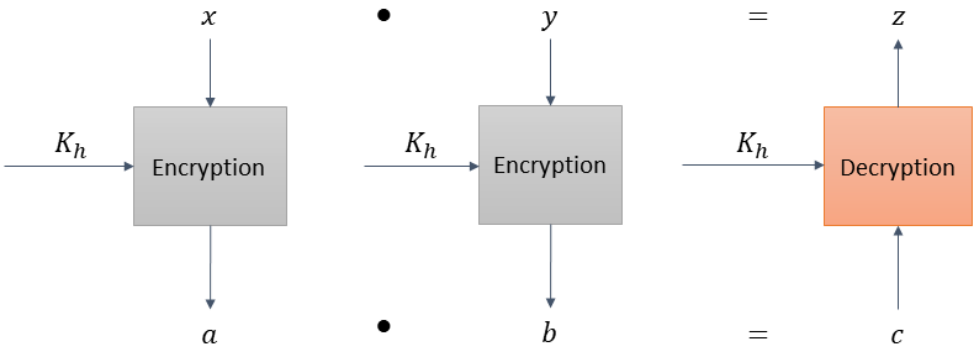

Figure 2.1 : The property of the Homomorphic Encryption.

The homomorphic encryption is categorized as partially homomorphic encryption and fully homomorphic encryption according to their capabilities.

2.1.1 Partially homomorphic encryption

RSA [21], EL-Gamal [22] and Pailler [23] algorithms are the well-known homomorphic encryption. These algorithms allow only one kind of operations; RSA and El-Gamal allows multiplication operation and Pailler allows addition operation so these algorithms are named Partially Homomorphic Encryption.

2.1.2 Fully homomorphic encryption

Fully Homomorphic Encryption is based The Gentry's scheme allows user to perform multiple types of operations on encrypted data [24]. Therefore, a system can carry out an analysis without knowing.

$$Dec(Enc(a)+Enc(b)) = a+b$$
$$Dec(Enc(a)*Enc(b)) = a*b$$

With updating to ElGamal algorithm, Algebra Homomorphic Encryption Scheme (AHEE) is proposed by G.Xiang et al. [25] which is other fully homomorphic encryption algorithm and allows addition and multiplication operations on encrypted data.

In the thesis, the fully homomorphic encryption is used, since when similarity values are found from the normalized $TF-IDF$ table, addition and multiplication operations are applied on encrypted data. FHE allows the operations on the encrypted data. The details of the generating normalized $TF-IDF$ table are explained in Section 2.2.

2.2 TF-IDF Normalization

$TF-IDF$ is a table that is used mostly on the text mining and data retrieval researches. The table bases on document frequency and term frequency. It uses to find similarity between documents or between documents and query. TF is term frequency, IDF is inverse document frequency [26]. The TF-IDF weight value computation is showed in Eq. 2.1.

$$tf-idfweight = tf*idf = tf*log(N-n)/n \tag{2.1}$$

Where;

N is the number of documents

n is the number of documents containing a query term

It means that if the more documents a term appears, the term is less important and the weight of the term is less.

In the $TF-IDF$ table, there are many same values (e.g. zero values), so the frequency attacks can happen easily. With the normalization process, the weights are made different each other, so the attacks are prevented. Table 2.1 is an example of the $TF-IDF$ table without normalization [13, 27].

In the table, it is known that some documents are similar. For example, id1 and id3 are similar documents. Even if homomorphic encryption is applied on the table, their values will be the same with each other. Therefore, the similar documents can be understood. To prevent the case, the normalization process is applied on the raw $TF-IDF$ table.

Table 2.1 : Example of the $TF-IDF$ table without normalization[13].

Document ID	w1	w2	w3	w4
id1	86	94	0	0
id2	0	0	0	56
id3	86	54	0	0
id4	0	58	0	0
id5	0	0	64	0
id6	90	0	0	58
id7	0	64	0	90
id8	58	0	46	0
id9	49	80	0	64
id10	0	0	64	0

The normalization process explained in detail in [13, 27],

- The unique TF-IDF values and the count of the values are ordered ascending by the TF-IDF values.
- A scaling factor is defining for determining the size of the difference between the original TF-IDF value and the normalized TF-IDF value as given in Eq.2.2.

$$e_q = u'_{q+1} - u'_q / h'_q * k \tag{2.2}$$

Where;

k is the scaling factor

e_q is the increasing value with k

u'_q is the ordered unique TF-IDF value

h'_q is the count of u'_q

- u''_{qs} unique normalized values are calculated by Eq. 2.3 for all values of S ($TF-IDF$ table) from 0 to S_q.

$$u''_{qs} = u'_q + (s * e_q) \tag{2.3}$$

Where S_q is computed by Eq.2.4

$$S_q = h'_q - 1 \tag{2.4}$$

- Folling the normalization process the set of new values u''_{qs} are all unique. The set of u''_{qs} is randomly distributed to the $TF-IDF$ table given in Table 2.2 and ends in normalization $TF-IDF$ table as given in Table 2.1 .

Table 2.2 : Example of the TF-IDF table with normalization[13].

Document ID	w1	w2	w3	w4
id1	87	94	17	8
id2	10	4	15	56
id3	86	54	19	0
id4	7	59	18	22
id5	3	14	68	13
id6	91	1	20	60
id7	12	66	6	90
id8	58	21	48	9
id9	46	80	11	64
id10	16	5	74	2

The new normalized TF-IDF table is encrypted by FHE. Although the values of the table are very similar each other, because of the encryption, the similarity between them becomes quite different.

2.3 HEADA Authentication Protocol

The HEADA was designed in [14] that has three components; an authentication server, a reader and a tag. The protocol was developed preventing replay attacks, allowing only authorized tag to create valid query. The HEADA is composed three phase; key generation, sub-keys combination selection and authentication protocol.

2.3.1 Key generation

The key generation process takes 4 steps as defined [14].

Step 1: Selecting parameters: W (the number of tags), G (the set of groups), I (the number of group), M (subkeys) parameters (described in [14]) are selected to satisfy the needed level of security and efficiency.

Step 2: Raw sub-keys generation: The raw sub-keys (it is represented by D in Figure 2.2) are generated using the selected parameters.

Step 3: Homomorphic encryption for irreversibility: The generated raw sub-keys are encrypted with homomorphic encryption so the sub-keys become irreversibly.

Step 4: Keys assignment to the tags: For each key k_n, the server selects randomly without duplication.

2.3.2 Sub-keys combination selection

- Each combination is selected only once.
- It has to be easy for the parties that have the selection key key(ks_n) to find the order of the combinations.
- It has to be infeasible, if not impossible, for the parties that do not have the selection key to find or predict the order of the combinations.

The Figure 2.2 shows selecting of the sub-keys combinations. The server selects (ks_n) randomly for every tag, ks_n where $0 < ks_n < max$ and $gcd(ks_n, max) = 1$. max is given in Eq.2.5,

$$max = 2^{\omega} - 1 \tag{2.5}$$

ω is the minimum number of binary bits that represent all the elements in each group. To select subkeys combinations, Eq.2.6 is applied for each iterations.

$$\varphi = (\varphi + ks_n) \bmod max \tag{2.6}$$

The process detail is explained in the research [14].

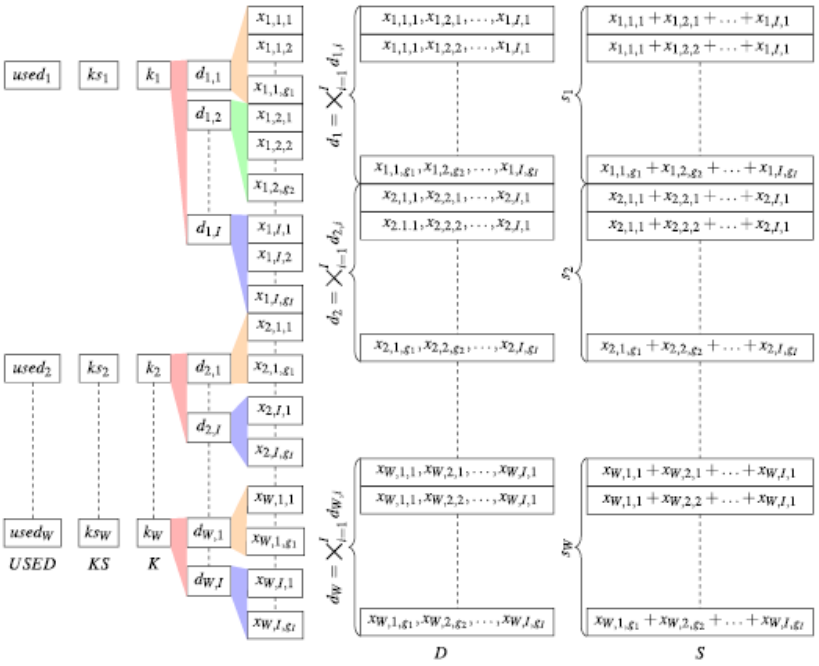

Figure 2.2 : Sub-keys selection showing.[27]

2.3.3 Authentication protocol

The authentication protocol between the tag and the server is shown in Figure 2.3. The protocol is designed for RFID based systems [14]. The tag selects $m+2$ (Note that bigger values of m will increase the security since it yield a larger larger group of the subkey.) combinations in the its sub-keys. The first m combinations are used for determining the ID of the tag. If the server finds the tag, the other $+2$ combinations are used by the server and tag for verifying each other.

In the thesis, the protocol is used for authenticating users and servers. The user generates $m+2+1+1$ combinations, new added $+1+1$ combinations are represented the *clientid* and *clientkey* in the SCDA-MU which is defined in Chapter 5

The new $+1+1$ combinations are selected like $m+2$ combinations but they do not use for authentication process. *clientid* and *clientkey* are generated to send to the other servers. When multiple client use our architecture, the servers know their data by using

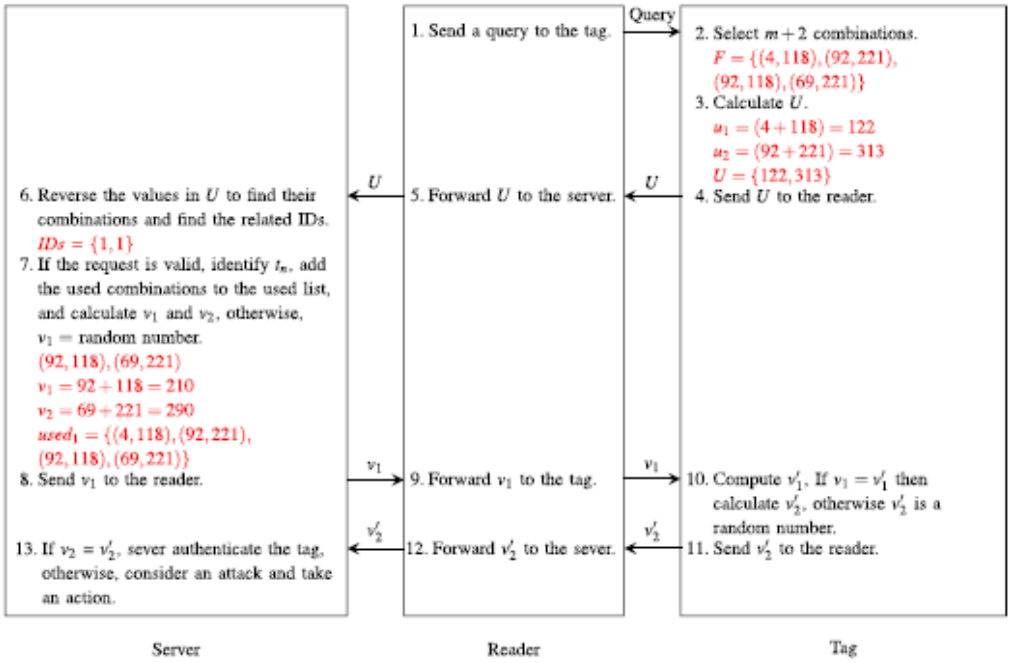

Figure 2.3 : Authentication protocol in [27]

the *clientid*. The *clientkey* is used to encrypt the results of the client by the server and to decrypt them by the client.

3. SDCA-SU : A SECURE AND PRIVACY PRESERVING DATA RETRIEVAL ARCHITECTURE ON CLOUD COMPUTING FOR SINGLE USER

3.1 Problem Statement

A simple architecture and communication essentials are depicted in Figure 3.1. In this architecture, the data owner generates encrypted data and sends them to the cloud server. It also sends the trapdoors to the client to enable encryption of a query and decryption of results. The basic architecture given in Figure 3.1 ensures the security of the data stored on the cloud. The cloud server might follow client behaviours and make inferences. Moreover, cloud server may not decide whether the client is real or fake.

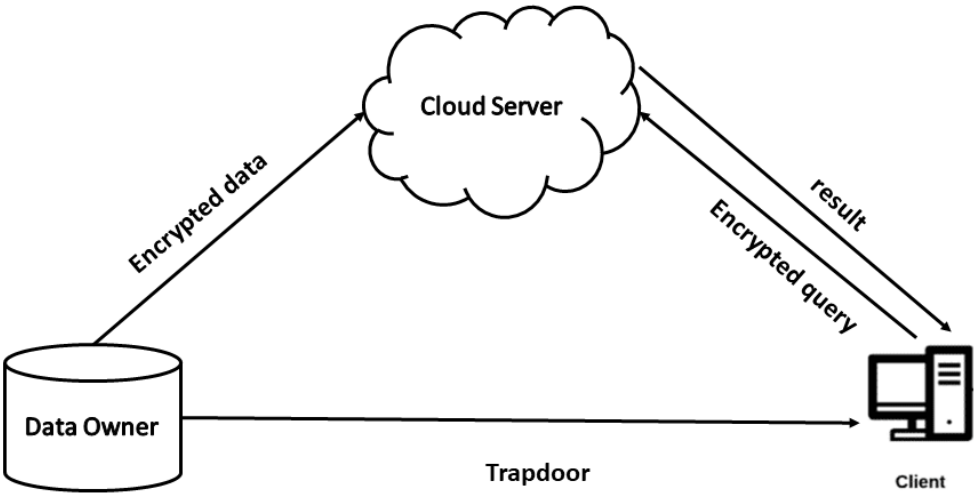

Figure 3.1 : The basic architecture of privacy preserving data retrieval system.[24]

Dawoud and Altılar proposed a technique to overcome the frequency attacks in [27]. They have used the FHE and normalized TF-IDF method for preventing multiple instances of zero. The researchers have developed the new system in research [12] and explained theoretically 9+1 security and privacy requirements: No index pattern, no query pattern, no documents pattern, no index frequency, no query frequency, no replay attack, query privacy, index privacy (9) and high efficiency of data retrieval (+1). The architecture they proposed in [12, 13] is shown in Figure 3.2. To accomplish

the requirements, they they introduced a data server, an authentication server, a ranking server and a private server to overcome the problems depicted for the basic architecture given in Figure 3.1 .

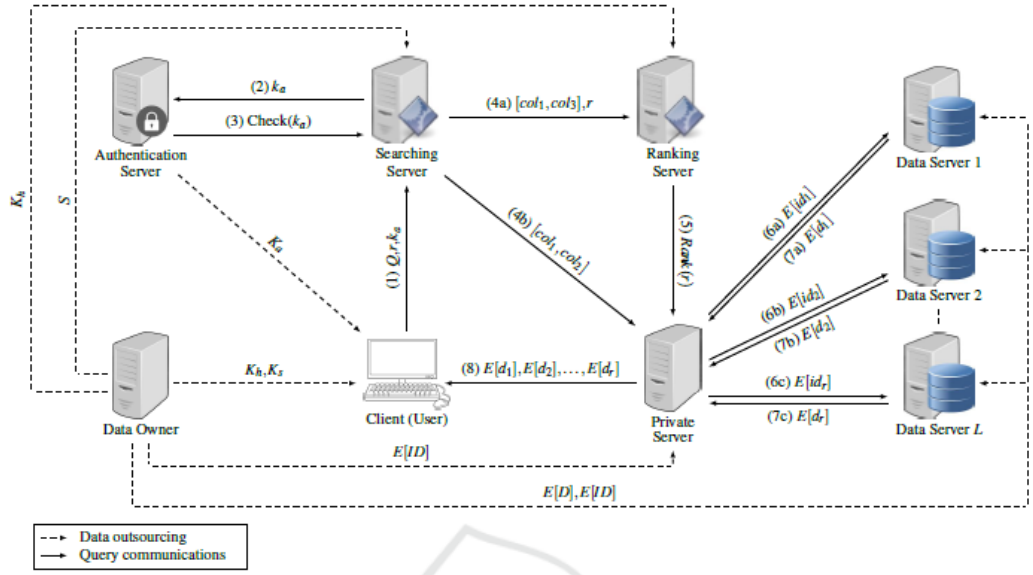

Figure 3.2 : The proposed architecture in [12,13].

Although their proposed architecture fulfils the requirements of the (9+1) system. There are some additional shortcoming as explained follows.

- Since the data owner sends to all user the same homomorphic key for the encryption, the results can be decrypted buy every other client in the loop as well . Additionally, when the data owner changes the homomorphic key, it must send the key to all users which produces additional traffic that yields inefficiency.
- Encrypted document ids are sent to the private server by both of the searching server and the data owner. This causes unnecessary data traffic as well.
- The ranking server sends the real similarity values of the documents to the private server. The private server infers from the values.
- The system provides the security and privacy requirements for only single user at a time.
- The proposed architecture is not evaluated through simulation.

3.2 The SDCA-SU Architecture

Given the shortcomings of architecture proposed in [12, 13], an encryption server is included in the architecture to know the homomorphic encrypted key from all user and system efficiently. The data outsourcing from the data owner to the private server is removed from the architecture efficiently, information of sending from ranking server to private server is reduced for data security and the new proposal system is simulated. The new architecture still provides the security and privacy requirements by only single user at the same time and the architecture is shown in Figure 3.3. The servers of the architecture are assumed "honest-but-curious" and do not collaborate with each other.

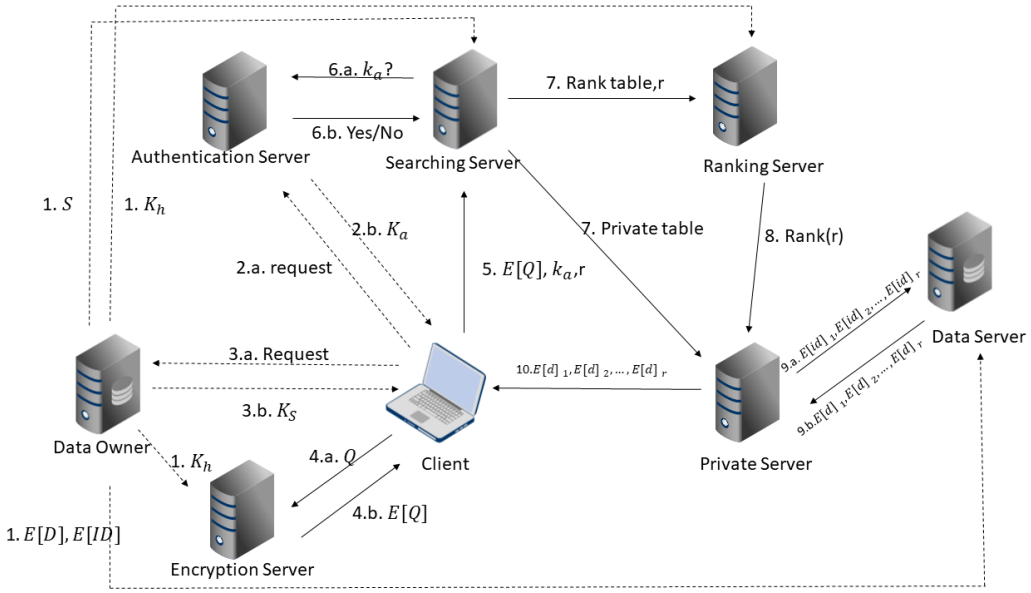

Figure 3.3 : The architecture of the SDCA-SU.

The essential differences between the previously proposed architecture and SDCA-SU are;

- Encryption server is introduced.
- Client is not responsible for homomorphic encryption. It similarly asks for encryption server to platform this dash.
- Data owner does not outsource own information to the private server any more.
- The constant of data send from ranking server to private server is limited to only random numbers produced for a search by ranking server.

3.2.1 Definitions

The variables used in further explanations and figures on the SDCA-SU are shown in Table 3.1.

Table 3.1 : Definitions of the architecture variables for SDCA-SU.

Variables	Definitions
K_h	Homomorphic encryption key
K_s	Symmetric encryption key
K_a	Authentication key set
k_a	Authentication session key
S	Normalized $TF-IDF$ table
$E[id]$	Encrypted document id by K_s
$E[d]$	Encrypted document by K_s
Q	Query
$E[Q]$	Encrypted QTF query by K_h
r	The number of documents to retrieve

3.2.2 Entities of SDCA-SU

3.2.2.1 Data owner

The data owner has the most vulnerable information of the system; it is accepted as secure. The data owner calculates $TF-IDF$ tables from the keywords of documents, the table is normalized and encrypted with FHE. The data owner also generates K_S (symmetric encryption key), K_h (FHE key) and shared them with relative entities; K_S is sent to the client, K_h is sent to the encryption server and ranking server. Also, the data owner encrypts documents and their ids, they are sent to the data server. Having all this sensitive information makes it the key entity of the system.

The data owner does not have the data about the client. Normalized $TF-IDF$ table and K_h are generated the same for each client. The table and K_h are sent once to relevant servers but K_S is sent to each client. The communication of the data owner is shown in Figure 3.4.

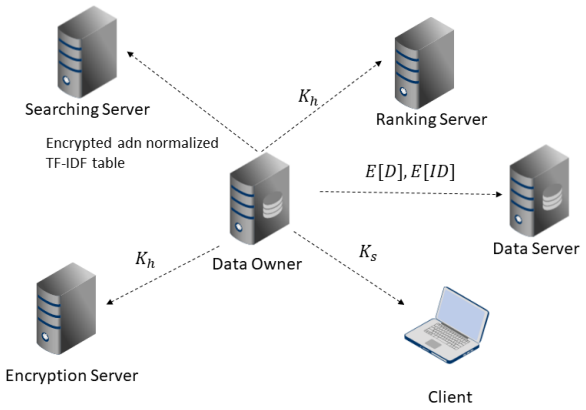

Figure 3.4 : The data owner and its communication with the other entities.

3.2.2.2 Searching server

The searching server is designed mainly for calculating similarity values between client query keywords and document keywords. The similarity values are found by using the Cosine Similarity Method.

At first, the searching server has only encrypted and normalized $TF-IDF$ table. The server cannot know any information about the documents from the table thanks to encryption. The server also knows $E[id]s$ with the table, but it is not any problem in terms of security.

After starting a client to search, the server has QTF generated like $TF-IDF$ table, client authentication key (K_a) and rank(r), they are sent by the client. The server cannot be informed from these data because QTF is first normalized so the values of them are unique and then it is encrypted FHE. Therefore, the server cannot estimate and make inferences about the searching query. Besides, although the server knows the authentication key, it is not authenticated by the authentication server. There is a protocol between the client and the authentication server and though the searching server wants to use the key, it is not accepted by the authentication server because they do not provide the authentication protocol in [14]. The authentication server accepts only verified clients and sends the "*accepted message*" to the searching server. The r is not important value; it presents the number of the documents wanted by clients. The communication with other entity of the server is represented in Figure 3.5.

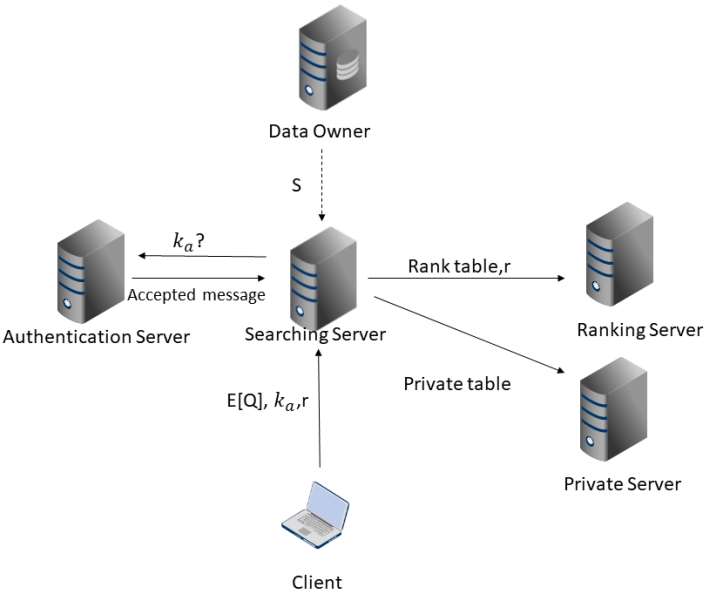

Figure 3.5 : The searching server and its communication with the other entities.

The searching server calculates the similarity values between $TF-IDF$ and QTF. The similarity calculation is implemented on the encrypted data so the similarity values are encrypted too and searching server is not informed from the values.

With the result of the similarity calculation, the server generates a table including three columns; first column presents random numbers that are assigned by the searching server for every different searching, second column presents E[id] coming with $TF-IDF$ table from the data owner and third column presents the similarity values of the documents with the searching query. The random numbers column play key role to match the tables in the private server. The searching server generates two new table for sending private server and ranking server.

Rank table: The table is sent to ranking server. It consists of two columns; first column presents the random numbers and second column presents similarity values.

Private table: The table is sent to private server. It consists of two columns; first column presents the random numbers and second column presents E[id]s.

3.2.2.3 Authentication server

The authentication server is an entity that provides identity privacy for clients. The server generates a set of K_a with respect to the parameters explained in Section 2.3. The number of elements in the set of K_a is expected to be greater than the number of clients. Therefore, a unique K_a will be produced per client.

A client sends a request to the authentication server, the server finds the first available (unassigned) K_a and sends it to client. Therefore, the authentication server cannot verify the identity of the client without requiring its actual ids but through k_a's generated via K_a. Within the protocol, the client sends k_a generated from K_a to the searching server, the searching server will extract k_a from the message and send it to the authentication server. Eventually the authentication server verifies without identity information [14] (Figure 3.6).

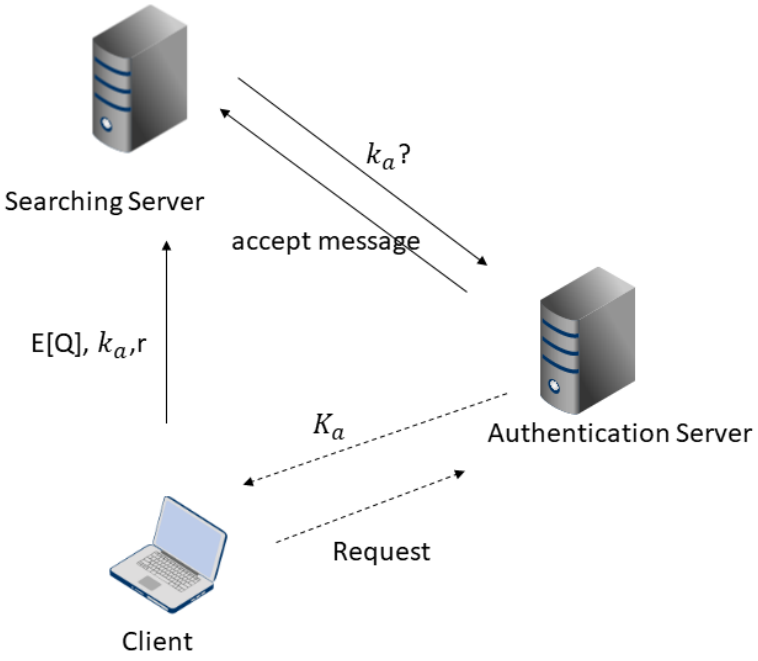

Figure 3.6 : The authentication server and its communication with the other entities.

3.2.2.4 Encryption server

The encryption server communicates with only the data owner and client as shown in Figure 3.7. The server receives K_h and hashed document keywords from the data owner. This communication occurs only once at the initial stage of system start app.

The communication between the encryption server and clients occur per query for encrypting it. The encryption server receives query from the client. Since the client sends its query by hashing, the server does not know the content of it. The server also does not know the keywords of the documents as they were already hashed by the data owner. Thus privacy of client query and data owner keywords are provided considering the encryption server. The encryption server generates normalized QTF for the hashed query, encrypts it and sends it to the client.

Note that the encryption server does not exist in the previous proposal [12,13] as shown in Figure 3.2. In that proposal single client was considered and client had the K_h for the encryption. In case a new client appears to use the system, K_h will be sent to it as well. Moreover once a K_h is received, it can be used further by the client. This indicates that K_h will be distributed to many clients without any clients without any protection. Which eventually causes security vulnerabilities.

By the introduction of the encryption server to the system, K_h will be secured and client will still get the query encrypted. Without revealing any information about the query to the encryption server. The encryption server also provides a transparent use homomorphic encryption since the change of the key (K_h) will be confused within the encryption server. The clients are not affected from this change.

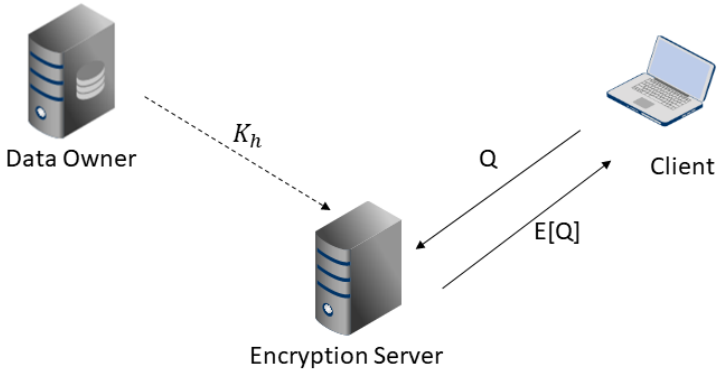

Figure 3.7 : The encryption server and its communication with the other entities.

3.2.2.5 Ranking server

The ranking server communicates with three entities; data owner, private server, searching server, as shown in Figure 3.8.

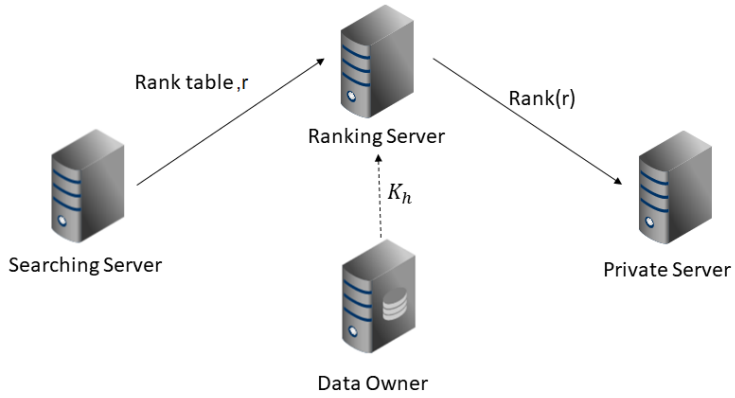

Figure 3.8 : The ranking server and its communication with the other entities.

The ranking server receives rank table and r (the number of documents to retrieve) from the searching server. The rank table has two columns are random numbers and encrypted similarity values that will be decrypted with K_h as explained in Subsection 3.2.2.2. The server cannot have any idea about the documents with the similarity values since these values are not useful without the id of the documents.

Top-r-Similarity table is produced by selecting the top r high similarity values in the table. This table contains the random values previously associated with the encrypted similarity values. The ranking server send the list of random numbers, i.e. rank(r), to the private server.

3.2.2.6 Private server

The private server is shown in Figure 3.9, it communicates with the client, the data server, the searching server and the ranking server. The server receives the private table which includes random numbers and $E[id]s$ from the searching server, the $rank(r)$ which includes random numbers ordered according to similarity values from the ranking server the table (Figure 3.13). The server merges the two tables received from two different servers into a single table by matching the random numbers to produce ranked $E[id]s$. From the operational perspective, private server concludes with a ranked table of $E[id]s$. Therefore, the privacy of the search and the documents are fulfilled for this server.

For every query in the system, unique set of random numbers are generated by the searching server, so the private server cannot define any relation between random numbers and $E[id]s$. Also note that the ids are encrypted by the data owner.

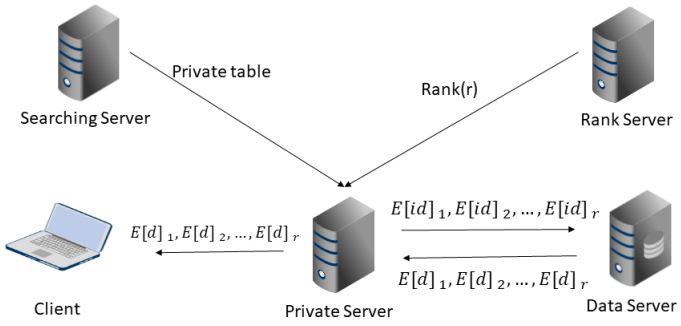

Figure 3.9 : The private server and its communication with the other entities.

When the private server establishes the list of ranked $E[id]s$, it sends it to the data server. After the private server receives the relevant $E[d]_1, E[d]_2, \ldots, E[d]_r$ from the data server, the $E[d]s$ are sent to the client. (Note that the data server includes randomly selected additional files as response. Therefore the private server cannot draw a conclusion between the data files and encrypted ids.) Moreover since the documents are encrypted, the server cannot know any information.

3.2.2.7 Data server

Data server communicates with private server and data owner as represented in Figure 3.10.

The data server keeps all documents and their ids encrypted. The encrypted data ($E[D]$ and $E[ID]$) is received from the data owner only once at the data outsourcing stage. Keeping data and ids encrypted ensures data security for documents and their ids.

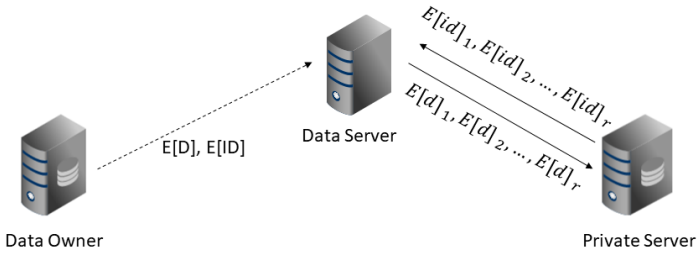

Figure 3.10 : The data server and its communication with the other entities.

When the private server sends $E[id]_1, E[id]_2, ..., E[id]_r$, the data server finds relevant $E[d]s$ by matching $E[ID]$ in the data server to $E[id]s$ coming from the private server. Then the set of $E[d]$s is sent to the private server.

3.2.3 Stages of the flow of SDCA-SU

3.2.3.1 Data outsourcing

When the system starts to execute, the data owner generates a homomorphic key K_h and a the symmetric key K_S. Normalized TF-IDF table is generated and encrypted with K_h and documents (D) and document ids (ID) are encrypted with K_S so $E[D]$ and $E[ID]$ are generated. The homomorphic encryption algorithm is described in Chapter 2. The data owner also works up the keyword table and hashes it. The hashed table is sent to the encryption server for generating query $TF(QTF)$.

The generated data are sent to relevant servers: S, the encrypted normalized $TF-IDF$ table, is sent to the searching server; K_h is sent to the encryption server and ranking server; $E[D]$ and $E[ID]$ are sent to the data server (As depicted with 1 in Figure 3.3).

3.2.3.2 Query creating

Once the data outsourcing operations are completed. The client can start operating. The client sends the request to the data owner and the authentication server to be accepted as a client in the system (2.a-3.a in Figure 3.3). The data owner sends K_S to the client to use it in decrypting the documents (3.b in Figure 3.3). The authentication server sends K_a (authentication key) to the client. This two messages are indicating that the client candidate became a client. K_a is used to generate unique session keys (k_a) used in query authentication (2.b in Figure 3.3). The ways of generating both K_a and k_a are described in Chapter 2.

After hashing a query, the client sends it to the encryption server (4.a in Figure 3.3). The encryption server generates the normalized TF vector(QTF) for the query. The normalization process is also described in Chapter 2. The normalized TF vector is encrypted with homomorphic key, K_h. The encrypted QTF, k_a i.e. session key and r i.e. the number of the documents to retrieve are send to the client and the client sends the encrypted query to the searching server (4-5 in Figure 3.3).

3.2.3.3 Query authentication

Before processing the query, the searching server sends k_a coming from the user to the authentication server. If the authentication server finds the valid user for the given k_a, the server sends an accept message to the searching server and the query process keeps on working. If the authentication server does not find the user or detected that the k_a was used before, the process started for the query is terminated (6.a-6.b in Figure 3.3).

3.2.3.4 Query searching

When the searching server receives the accept message from the authentication server, it starts to perform the query. The server has the homomorphic encrypted $TF-IDF$ table of the documents (S) and the homomorphic encrypted QTF table. The server uses the cosine similarity measure for finding similarity values between documents vector and query vector [28]. The cosine similarity value is found between the query vector $Q = [q_1, q_2, \ldots, q_M]$ and the n_{th} index vector $S_n = [s_{n,1}, s_{n,2}, \ldots, s_{n,M}]$ by Eq. 3.1.

$$cs_n = \frac{\sum_{m=1}^{M}(q_m \times s_{n,m})}{\sqrt{\sum_{m=1}^{M}(q_m)^2} \times \sqrt{\sum_{m=1}^{M}(s_{n,m})^2}} \tag{3.1}$$

The similarity vector between Q and S is showed in:

$$CS = [cs_n \mid 1 \leq n \leq N] \tag{3.2}$$

Figure 3.11 shows using $TF-IDF$ table and QTF table the calculation of cosine similarity values. The similarity values between each document and query are calculated separately.

The searching server generates a table which has three columns; the first column $col1$ represents the random number, second column $col2$ is the encrypted ids ($E[id]s$) and third column $col3$ is the similarity values of the $E[id]s$. A unique random number is given for each document and each searching, also the similarity values are still encrypted in the server. The server generates two new tables as shown Figure 3.12. The first table includes $col1$ and $col3$ is sent the ranking server, while the second table includes $col1$ and $col2$ is sent to the private server with r (7 in Figure 3.3).

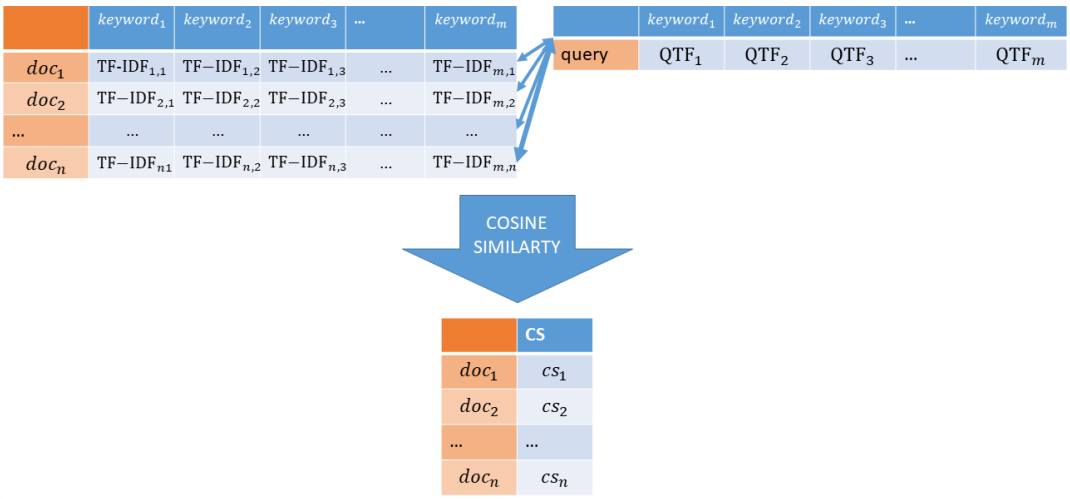

Figure 3.11 : Using cosine similarity measure on the system.

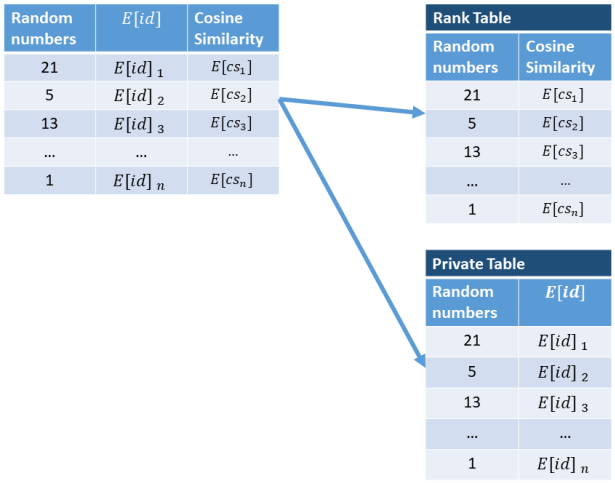

Figure 3.12 : Generating two tables for private and ranking server by the searching server.

3.2.3.5 Documents retrieval

The ranking server decrypts the similarity column i.e. $col3$ with K_h and sets the table in descending order according to the decrypted values of similarity, i.e. the actual similarity values. As it is shown in Figure 3.13 that a new table is created from the highest r rows $[col1]$ and it includes random number column by the ranking server and is sent to private server (8 in Figure 3.3). In the previous research [12, 13], the random number column is sent to the private server with real similarity values. However, learning the cosine similarity matched with random number values by private server

may cause a threat for data security since private serve also receives another table from searching server.

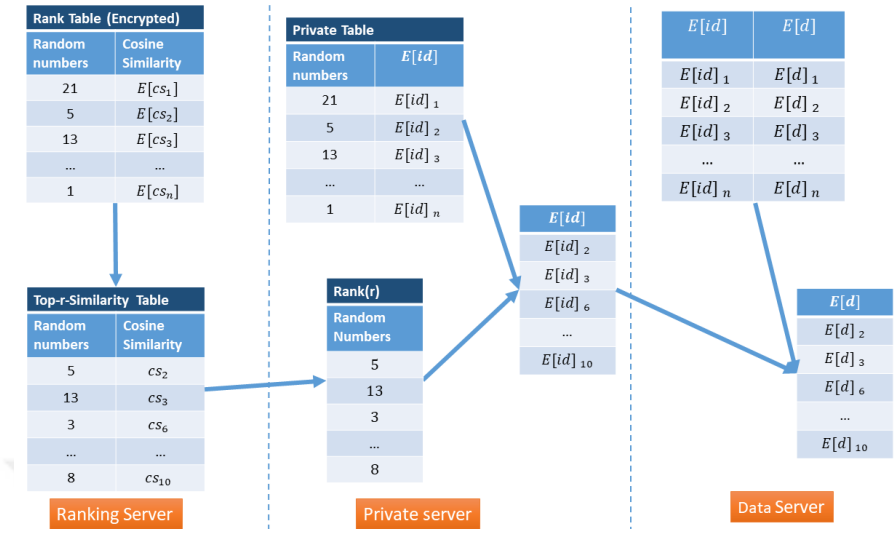

Figure 3.13 : Finding documents to retrieve.

The private server matches two tables coming from the searching server and the ranking server according to random numbers. Therefore, the server finds the $E[id]s$ of the related documents, and sends them to data server (9.a in Figure 3.3). The process over the tables Rank(r) and the Private tables shown in the Figure 3.13.

The data server matches the document ids initially sent by the data owner and related document ids coming from the private server (9.b in Figure 3.3). The set of encrypted documents $E[d]_1, E[d]_2, ..., E[d]_r$ is found (the last table in the Figure 3.13) and sent to the private server, the set of encrypted documents is forwarded to the client (10 in Figure 3.3). The client decrypts $E[d]_1, E[d]_2, ..., E[d]_r$ by using symmetric key K_S and ends up with the decrypted documents.

4. SIMULATION AND RESULTS FOR SDCA-SU

4.1 Simulation

In this chapter, the SDCA-SU is simulated and the results of the simulation are explained.

4.1.1 Dataset information

4.1 In this research, uw-can-data [29] and mini-news-group [30] dataset were used for the testing the proposed system. The list of the keywords per document in datasets have been produced though the 5 stages explained below.

1. The html documents were parsed using htmlparser1.6 [31] to extract the all words.
2. The words were cleared from the stop words list according to the long list, the short list and google list, these are listed in Appendix A.
3. Porter stemmer [32] is used to stem the keywords.
4. The words which are shorter than three characters in length, were removed.
5. The number of the words are less than two in one document, they were not used as keyword.

Statistical information for the datasets that are used in this research are shown in Table 4.1 The datasets are used separately.

Table 4.1 : Dataset Information for SDCA-SU.

dataset	the number of the document	the number of the keyword	the number of the class
uw-can-data	200	3167	10
mini-news-group_20	400	3153	20

For the documents and document IDs encryption, AES symmetric encryption algorithm [33] was used. For the simulation Java JDK 10.1 platform and Netbeans 8.2 application editor were used. For sending packets between servers and clients, TCP (Transmission Control Protocol) was used.

In the simulation, the servers (data server, private server, rank server, authentication server, searching server, encryption server), the data owner and the user were defined on the "localhost". The communication of the entities was provided by using the Java Socket Programming Tools.

The flow diagram of the SDCA-SU is shown in Figure 4.1.

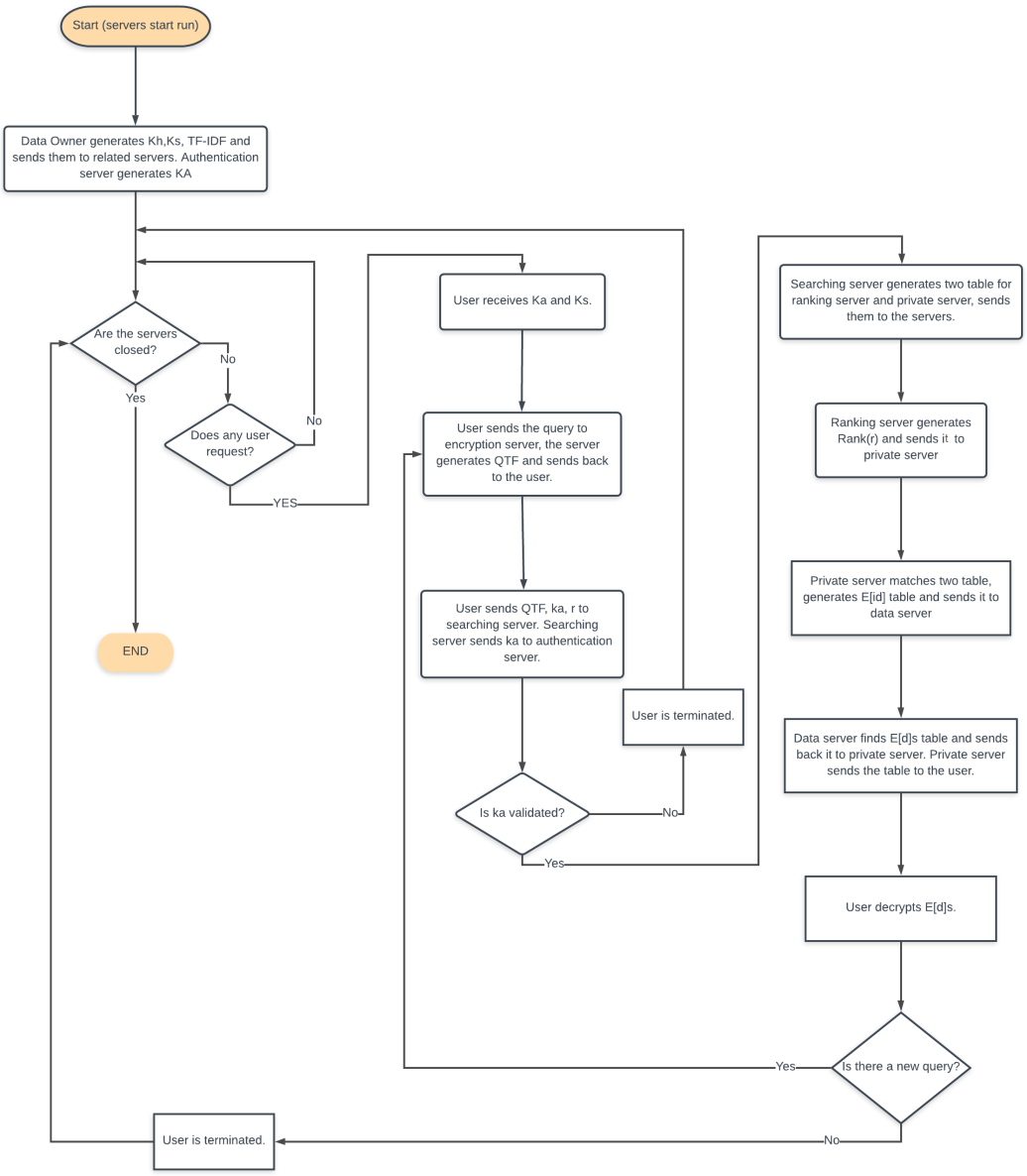

Figure 4.1 : The flow diagram of the SDCA-SU.

In the program, servers run in an infinite loop to provide the continuity. Users send their query one by one and they terminate after receiving data from the private server.

4.2 Results

1. **The variation of the similarity values for same queries:**

   When a client searches the same keyword ten times, the system retrieves the same documents with almost same rank but the similarity values of the retrieved documents are different. The reason of it, at the normalization process, the QTF values are became unique values and they are distributed to relatively locations randomly for every keyword search.

   In the experiment, the query was determined as 'bear' and searched ten times. The most related similarity results of the ten query was listed. The similarity values of the documents for every ten queries are shown in Figure 4.2. The values have changed for every query but the rank of the documents are almost same. Although the experiment was applied for single client, if more than one client search for the same keyword, the output would be similar to the first case (Figure 4.2).

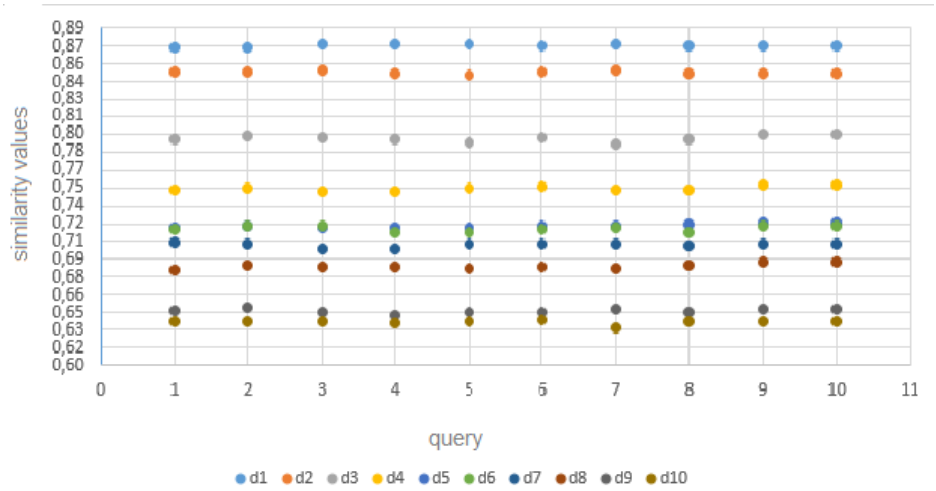

Figure 4.2 : Document similarity values for 10 repetition of the same query. (Every different color point represents different document.)

2. **The effects of the fully homomorphic encryption on the similarity values:**

   In Figure 4.3, the related document similarity values are viewed on the system with FHE and without FHE. Both of two case the real similarity values of the retrieved documents almost the same. Actually, the similarity values are considered the same

because the normalization process is applied on the data and a few differences made up. If the normalization process was not applied, the similarity values could be the same for two case.

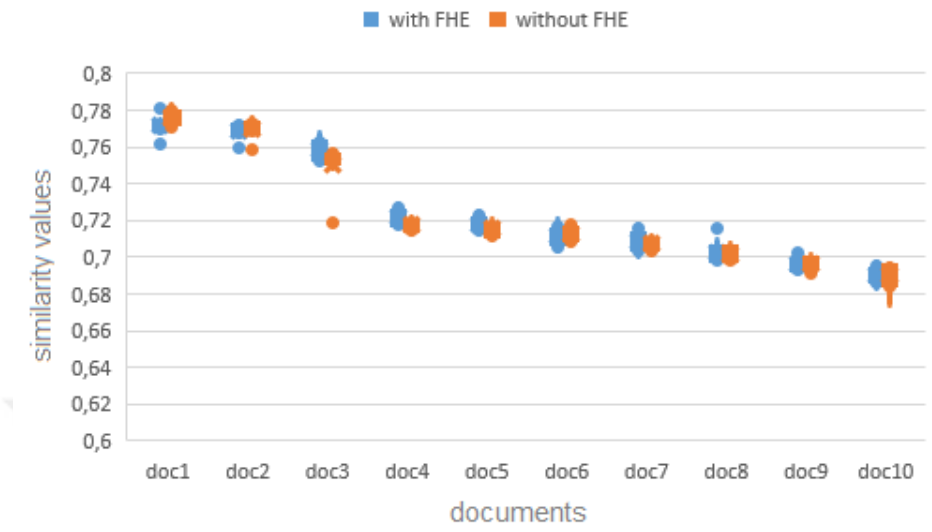

Figure 4.3 : The similarity values of the retrieved documents by the system with FHE and without FHE.

3. **Execution time differences between with FHE and without FHE:**

With FHE, the searching on the system are made more securely than without FHE. The time from the entry of the user's query to the response of the system is shown in Figure 4.4 for the two cases. The figure is prepared for the same 10 queries. The time difference between the two conditions is acceptable for this secured system.

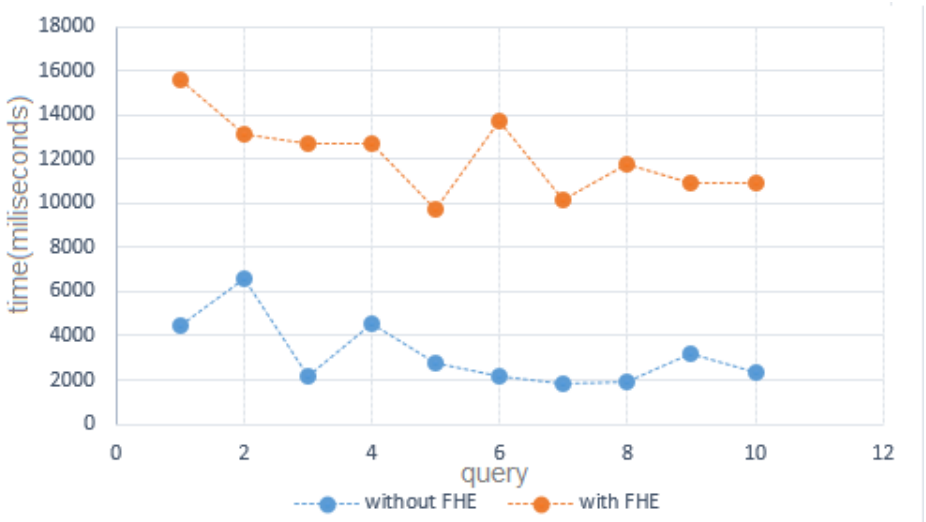

Figure 4.4 : Execution time differences of the system between with FHE and without FHE. (milliseconds)

4. **The similarity differences among the different copies of a document:**

   In this experiment, an original document and its modified copies are used. The modified copies have been generated manually. Their properties;

   original: a document in the uw-can-data

   copy1: The same with the original document

   copy2: Only one query keyword is deleted in the original document

   copy3: Two query keywords are deleted in the original document

   copy4: Three query keywords are deleted in the original document

   copy5: Three irrelevant keywords are deleted in the original document

   The five documents are generated and added in along with the other documents on the "uw-can-data" dataset. The TF-IDF matrix is generated. In the first case, the similarity values without normalization are calculated and in the second case, the values with normalization are calculated for the same query.

   In the first case, as there is no normalization, the result always gives the same similarity values and the same sequences, because TF-IDF values of the documents and QTF values of the query are the same every time. Figure 4.5 is shown the results of the first case.

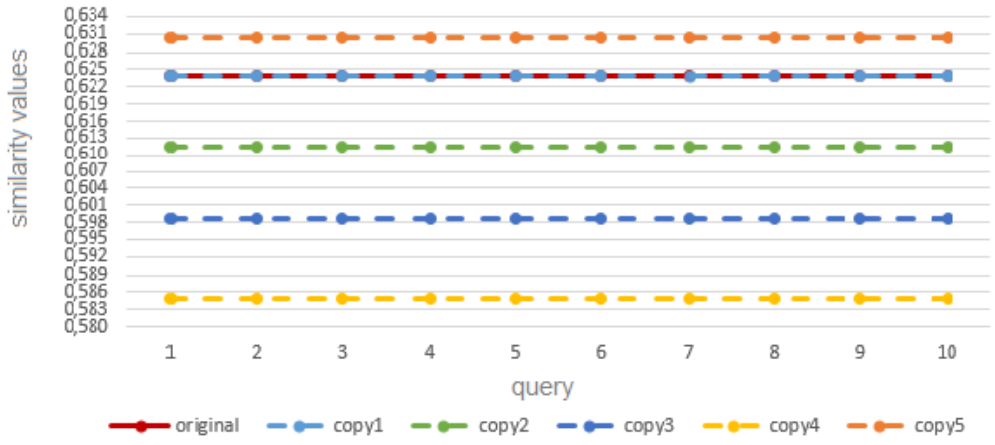

Figure 4.5 : The similarity values without normalization of the original document and its modified copies.

In the second case, because of the normalization process, the similarity values become slightly different. As defined in Chapter 2, a k value is defined for

the normalization process. The k value is a parameter to determine the distance between the same values in the $TF-IDF$ matrix. New $TF-IDF$ values are calculated based on this distance value and located randomly in the $TF-IDF$ matrix. The same process is applied on QTF matrix as well. Since the new values are located randomly for every query searching, the similarity values may different but very close to each others for every ten same query searching in the case of the normalization.

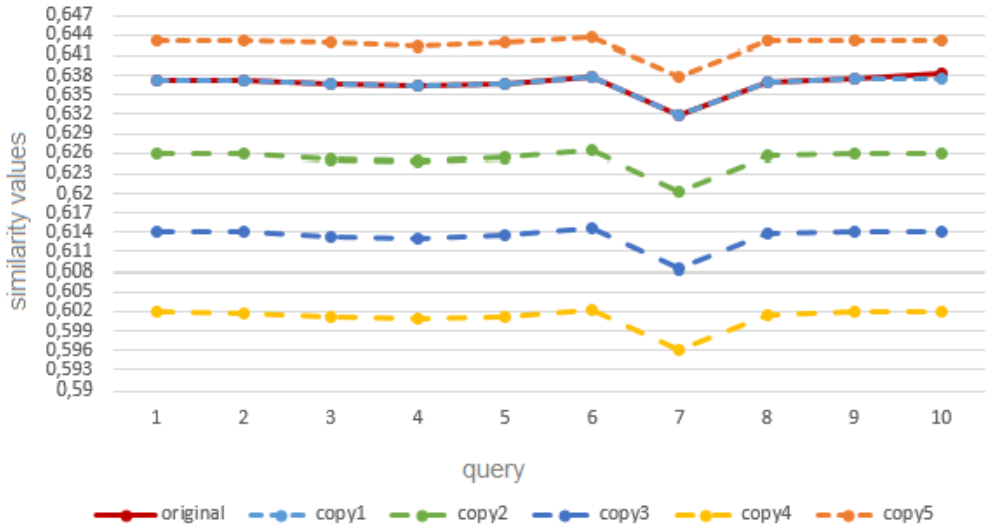

Figure 4.6 : The similarity values with normalization of the original document and its modified copies.

In Figure 4.5, original document and its copy "copy1" are shown. The documents have the same similarity value. Because normalization process is not applied on the $TF-IDF$ matrix, their similarity values and orders are the same for ten query searching.

The query keyword is removed in the original documents; one query keyword in the copy2, two query keywords in the copy3 and three query keywords in the copy4. Because their $TF-IDF$ values for the query keyword decrease, their similarity values and ranking decrease also.

For the last document copy5, any three unrelated keywords are deleted in the document so the $TF-IDF$ value for the query keyword increases and consequently similarity value increases.

In the previous approach for this experiment, different k values was used for TF-IDF table and QTF table. Although it was considered that the results would not change,

the rank of documents are similar to the non-normalized case but similarity values of them are different from the case. Figure 4.7 shows the similarity values and ranks of the documents according to the same query with above.

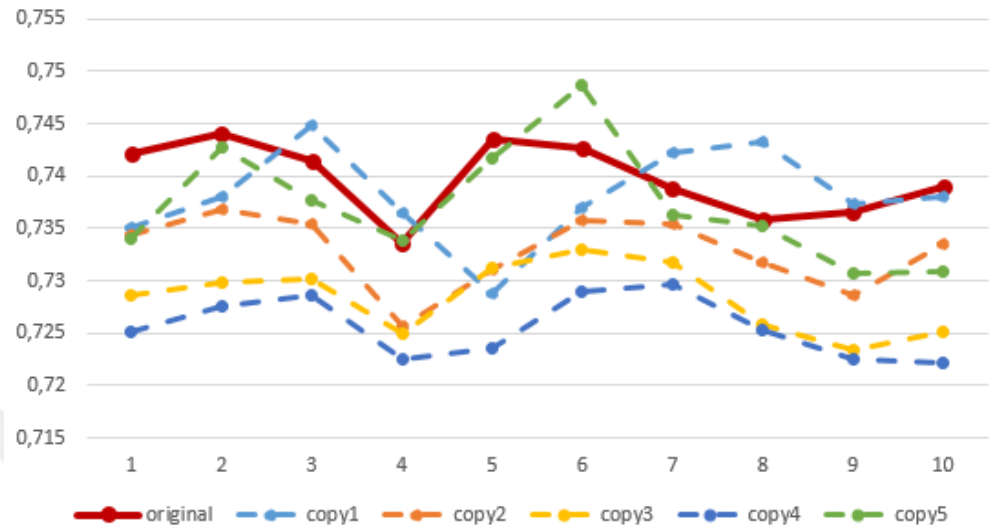

Figure 4.7 : The similarity values with normalization of the original document and its modified copies according to previous approach.

5. **The effect of the normalization process to the similarity values:**

   The normalization process is applied on the dataset uw-can-data for different k values ($k = 10, 20, 50, 100, 1000, 10000, 100000$). The query keyword "bear" is searched by the client. The ordered documents and their similarity values are shown in Table 4.2.

   The second column is represented the similarity values without normalization process, so they are considered original similarity values. The other columns are represented similarity values with normalization process depended on the k values. The results show that when k value is increased, the similarity values and document ranks get closer to the original similarity values and document ranks respectively. When k value is decreased, the similarity values of the relevant documents decrease and the similarity values of irrelevant documents increase. Although the values of the relevant documents decrease, the retrieved documents are the same with the non-normalized search.

Table 4.2 : The similarity values of the documents based on the k values.

Documents	without normalization	with normalization (k=100000)	with normalization (k=10000)	with normalization (k=1000)	with normalization (k=100)	with normalization (k=50)	with normalization (k=20)	with normalization (k=10)
doc1	0,910223483	0,910234373	0,910343129	0,91097183	0,87654954	0,782599501	0,507365875	0,300251523
doc2	0,867531737	0,867563778	0,867856987	0,870239289	0,853891399	0,778163958	0,53973005	0,349652461
doc3	0,805862328	0,805896804	0,806202051	0,809094543	0,799452759	0,732443353	0,516357061	0,344611691
doc4	0,755233453	0,755278621	0,755649614	0,759257032	0,758760167	0,706104311	0,508216464	0,344767153
doc5	0,734279182	0,734308913	0,734560843	0,736678704	0,721042782	0,666454898	0,463623779	0,307711881
doc6	0,730482263	0,730509243	0,73073575	0,732972414	0,717364006	0,658625936	0,450701408	0,290905248
doc7	0,727893525	0,727918536	0,728172493	0,730639973	0,716549873	0,665687839	0,458782895	0,300931907
doc8	0,691084014	0,691120647	0,6914534	0,694410723	0,691553663	0,641338173	0,465677391	0,315286613
doc9	0,653393479	0,653429452	0,653736808	0,656601305	0,653145263	0,612844974	0,440168596	0,30046437
doc10	0,639999087	0,64003595	0,640383585	0,643910573	0,641901912	0,61350215	0,445001155	0,310023626
doc11	0,6240509	0,624080138	0,62436765	0,627046316	0,622871096	0,576133808	0,416867374	0,279092583
doc12	0,615829202	0,615864879	0,61622322	0,619690117	0,622031458	0,593412445	0,432438083	0,301521144
doc13	0,597374194	0,597409915	0,597718463	0,600323468	0,597998771	0,557088012	0,403251935	0,184663935
doc14	0,593994222	0,594040587	0,594416125	0,598055972	0,608222611	0,575402525	0,426661611	0,320051895
doc15	0,593964676	0,594004932	0,594361413	0,597977537	0,602082684	0,564723584	0,419011982	0,296977634
doc16	0,505907153	0,505932151	0,506197546	0,508701434	0,505790807	0,472127934	0,340747755	0,236537143
doc17	0,47352476	0,473571582	0,473944428	0,477663571	0,489936778	0,470551836	0,37456638	0,266475347
doc18	0,452463943	0,452497032	0,452799732	0,455516965	0,462279853	0,434162529	0,319554134	0,229066491
doc19	0,437840301	0,43786849	0,438104439	0,440822749	0,444869456	0,426993584	0,314744736	0,234481529
doc20	0,430583194	0,430612701	0,430930309	0,43400912	0,440741914	0,424038222	0,326716929	0,237670256
	0,048169275	0,048217386	0,048579673	0,052600089	0,089989471	0,133634871	0,191775179	0,218187918
	0,030760184	0,030796263	0,031153016	0,034786783	0,080990099	0,119393456	0,186894311	0,214181459
irrelevant docs	0,02763329	0,027657803	0,027892269	0,029856275	0,068081159	0,116339368	0,184352858	0,208530121
	0,025623516	0,025658516	0,025980557	0,029299356	0,067884321	0,11534893	0,175114479	0,186229362
	0,017318302	0,017368586	0,017860276	0,02327915	0,067163346	0,112399208	0,174271492	0,186131737

6. **The average F score :**

In the experiment, the average *Average* F_{score} is calculated. The F_{score} is the harmonic mean of the precision and recall [34], where an F score reaches its best value at 1 (perfect precision and recall) and worst at 0. Precision (p) is the number of correct positive results divided by the number of all positive results returned by the classifier, and recall(r) is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

$$F_{Score} = \frac{2 * precision * recall}{precision + recall} \tag{4.1}$$

$$AvFscore = \frac{\sum_n Fscore}{Numberof Documents} \tag{4.2}$$

The averages $AverageF_{score}$ values of two datasets with normalization process and without normalization process are shown in Table 4.3. All documents in the datasets are searched as query. The same preprocesses in Subsection 4.1.1 are applied on the query documents. The results are the same for two case which are normalized and non-normalized on the datasets.

Table 4.3 : The average F scores of the datasets.

dataset	with normalization	without normalization
uw-can-data	0,6971	0,6971
mini-20newsgroups	0,2481	0,2481

Some classes of the mini-20newsgroups dataset are similar each other, so the result is lower than uw-can-data dataset.

With respect to the experiments defined under the effect ... values and The average F score headings. We can conclude that although the normalization process provides index and query privacy, it does not affect the results. Therefore, the normalization process can be considered as a very effective method for the research.

7. **The use a threshold to define relevance of the document:**

In the research, it is considered that a threshold is included in the ranking server to not to respond with documents with low relevance. When the ranking server

sends the random numbers (rank(r)) table to the private server, the ranking server sends the random numbers that are higher than the threshold according to the real similarity values. Thus, the private server sends the most relevant documents to the client. Determining of the threshold is very difficult, because similarity values are changeable from query to query and from dataset to dataset. Determining of the threshold can be considered as a new research topic. That is not covered within this research.

5. SDCA-MU : A SECURE AND PRIVACY PRESERVING DATA RETRIEVAL ARCHITECTURE ON CLOUD COMPUTING FOR MULTIUSER

5.1 Problem Statement

In Chapter 3, the novel approach for privacy preserving data retrieval in cloud is explained theoretically by new addition on the architecture of defined previously in [12, 13]. According to the results given in 4, the proposed method i.e. SDCA-SU, proposes solutions to the shortcomings of the once in the literature [12, 13]. Although SDCA-SU fulfills the requirements of such a data retrieval system, only one user can be active user at a time. In this section, the developments required to make the system for concurrent use is explored.Such a change imposes some new challenges given as follow;

- When more than one user send the query to searching server concurrently, searching server should differentiate between concurrent requests.Since the searching server generates two tables for private server and rank server, these tables must be generated per user and must be related to the particular user.
- The private server must know the tables coming from searching server and the rank server are belonging to which users because the server must match correct tables. The same approach, i.e. requiring multiple tables specially produced and tapped to a user, is a valid problem for the private server as well. Moreover, private server must have the capability of sending concept set of documents to relevant clients.

5.2 The Proposed Architecture

As mentioned before the proposed system in Chapter 3 is a solution for security and privacy issues when only one user is active at a time. For multiuser system that enables concurrency, a new system has to be proposed. In our proposal, SADCA-MU,

the same server architecture is used as given in SDCA-SU with different additional functionalities.(Figure 5.1)

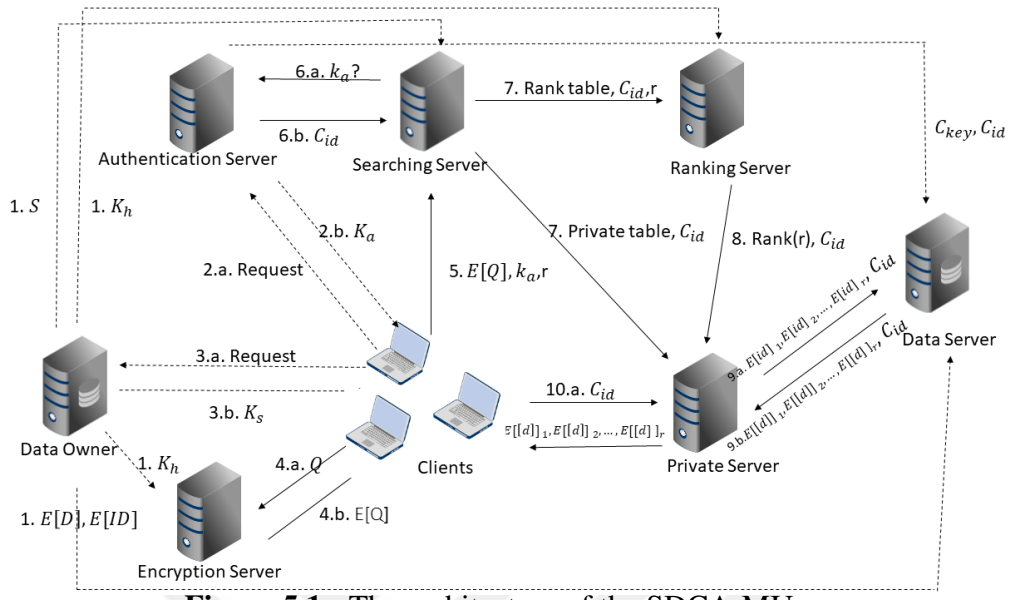

Figure 5.1 : The architecture of the SDCA-MU.

5.2.1 Definitions

The extended set of variables and their definitions for the SDCA-MU are shown in Table 5.1.

Table 5.1 : Definitions of the extended set of architecture variables for SDCA-MU.

Variables	Definitions
K_h	Homomorphic encryption key
K_s	Symmetric encryption key
K_a	Authentication key set
k_a	Authentication session key
C_{id}	Client id
C_{key}	Client key
S	Normalized $TF-IDF$ table
$E[id]$	Encrypted document id by K_s
$E[d]$	Encrypted document by K_s
$E[E[d]]$	Encrypted document by C_{key}
Q	Query
$E[Q]$	Encrypted QTF query by K_h
r	The number of documents to retrieve

5.2.2 Entities of SDCA-MU

5.2.2.1 Data owner

The data owner carries out the same operations as in the data owner of SDCA-SU (Fiqure 5.2). Increasing the number of the client and allowing them to run concurrently does not affect its operations because the server generates data (K_h, S) and sends them to other servers only one time. K_s is sent to clients who want to search a query.

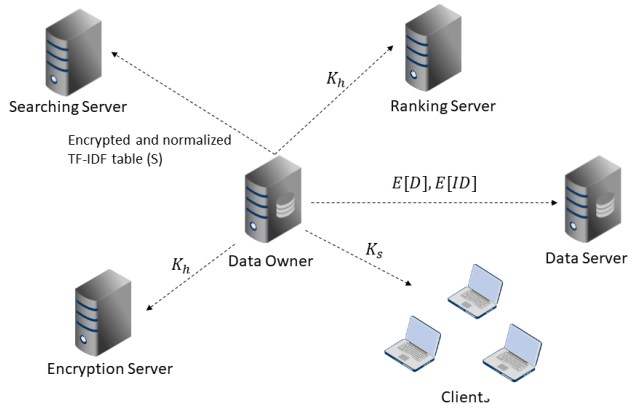

Figure 5.2 : The data owner and its communication with the other entities.

5.2.2.2 Searching server

The searching server is considered as similar to the searching server of the SDCA-SU. The only difference is that the server receives a client id (C_{id}) from the authentication server at the SDCA-MU (Figure 5.3). The authentication server accepts only verified clients and sends the C_{id} to the searching server. Because of this change the searching server receives an additional information i.e. the id of the user per query. However, this additional information would not be sufficient to infer any addition information. Therefore, knowing the C_{id} by the server is not problem in terms of identity privacy.

Like the searching server in the SDCA-SU, the server generates two tables for rank and private servers and sends these tables with C_{id} to the relevant servers.

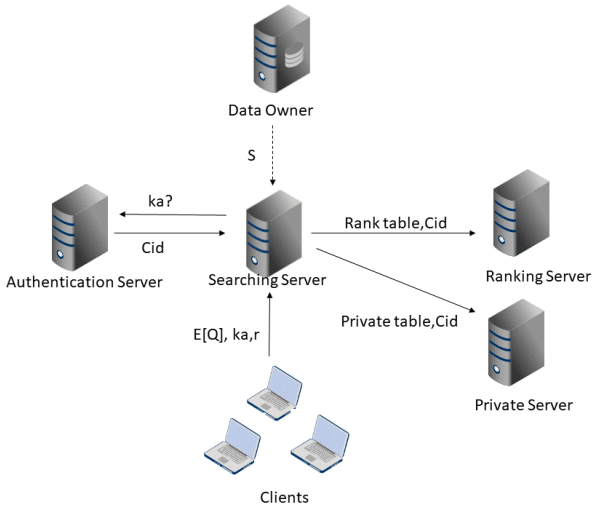

Figure 5.3 : The searching server and its communication with the other entities.

5.2.2.3 Authentication server

Unlike the SDCA-SU, the authentication server generates C_{id} and C_{key} for users. It is not problem in terms of security that the authentication server knows C_{id} and C_{key} because they are unique for each query of each client. The server sends both of them to the data server. Also, the server sent C_{id} to the searching server as an accept message if the server accepts the client. Figure 5.4 is shown the communication with other entity of the authentication server.

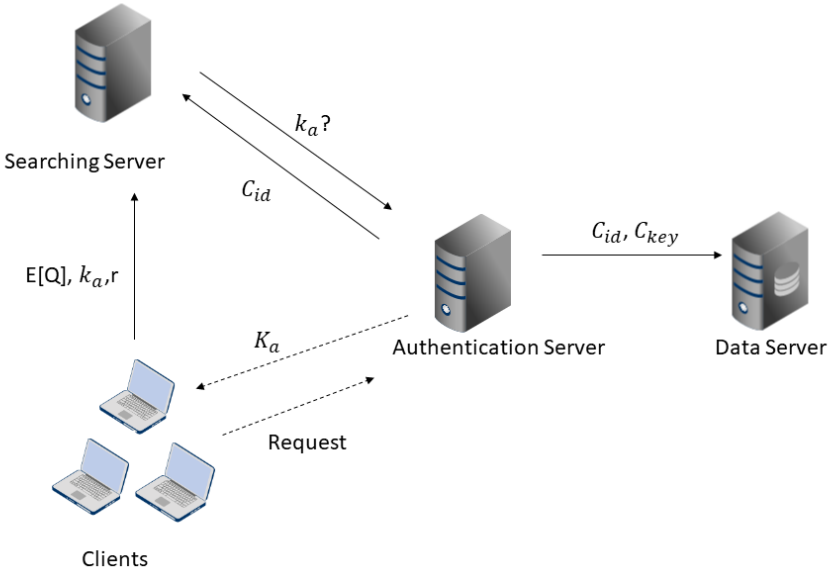

Figure 5.4 : The authentication server and its communication with the other entities.

5.2.2.4 Encryption server

The encryption server carries out the same process as one in the SDCA-SU and its communications are shown in Figure 5.5.

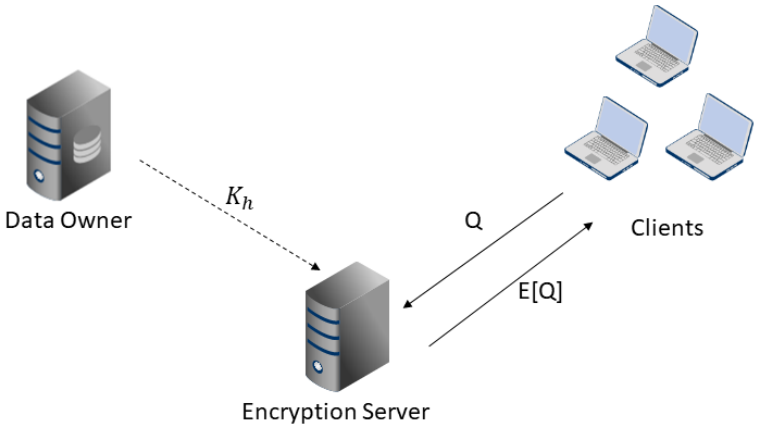

Figure 5.5 : The encryption server and its communication with the other entities.

5.2.2.5 Ranking server

The ranking server in SDCA-MU receives C_{id} with r from the searching server unlike the SDCA-SU. The ranking server finds the real similarity values and ranks them in descending order. According to ranks it creates a new list by selecting the r values from the top. After creating, the table and C_{id} are sent to the private server. As with other servers, the server does not know the client identity from C_{id} because of uniqueness of C_{id}. Therefore, there is not problem imposed because of this change from the perspective of security and privacy. The communications with other server of the ranking server is shown in Figure 5.6.

5.2.2.6 Private server

The private server keeps tables coming from other searching server, rank server and data server as well as clients essentially for book keeping.

The private server initially creates a table called SS_{table}, to save successive couples of private table and C_{id} coming from searching server in a single table. Whenever a private table along with a C_{id}, it will be appended to SS_{table} for further use. The private table has two columns that are random numbers and $E[id]$s. The server also keeps an

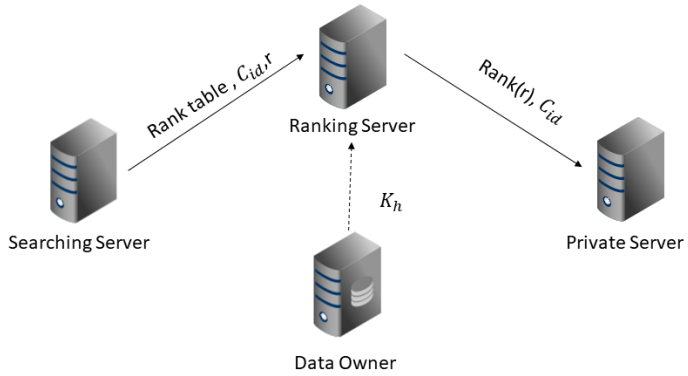

Figure 5.6 : The ranking server and its communication with the other entities.

RS_{table} for data coming from the ranking server that includes C_{id} and Rank(r) table. RS_{table}, similar to SS_{table}, is to keep consecutively received tables from ranking server. The Rank(r) table has the random numbers column. The private server starts with matching C_{id} coming from the searching server to C_{id} coming from the rank server. When the private server finds the matched private table and Rank(r) table, it matches random number column of the private table to random number column of the Rank(r) table to find $E[id]$s. The processed data are removed from SS_{table} and RS_{table} since there would be no additional process to be performed for this particular client.

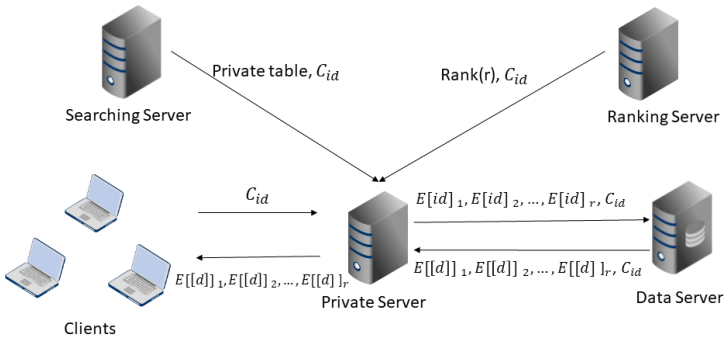

Figure 5.7 : The private server and its communication with the other entities.

The private server generates a PS_{table} that includes two data. One of the data is C_{id} that is received from the searching server and the rank server, the other is $E[id]$ list is found by matching Rank(r) table to private table. The private server sends the elements of the table, i.e. $E[id]$ and C_{id}, to the data server in the first come first serve basis to retrieve the relevant document. Document coming from data server are encrypted by the data server using C_{key}. Thus, data server sends $E[E[D]]$ to private server. The results are

kept that is generated in a DS_{table} that includes C_{id}s and encrypted $E[E[D]]$ list. Private server serves clients request for document through DS_{table} content.

When a client request for its query, the client sends its C_{id} to the private server. The server checks for C_{id} coming from the client in the DS_{table}. If the server finds C_{id}, it sends $E[E[d]]$ list to the client or if does not find it, depending on the policy it might be suspended for a present value long or return a negative acknowledgement reply or simply ignores. Upon successful response and after the sending $E[E[d]]$ to client, the relevant row is removed by the private server, Note that, the private server does not have any information about the content of the query done by the clients. Since the documents are encrypted by data server the content of the documents are not known by the private server as well. The matching process between the private table and the Rank(r) table is done by random number column and the numbers are changed for each query. Therefore, the private server does not infer any additional information between any client and any document.

Clients send their request to the private server with their C_{id} to make sure that only the requested documents are sent to clients by the server. Even though the server sends the wrong documents or shares all documents with all clients, clients cannot decrypt that they have not asked for because they do not know any C_{key} apart from their own.

5.2.2.7 Data server

In the SDCA-MU, the data server receives $E[D]$ list and $E[id]$ list from the data owner like the SDCA-SU. (Figure 5.8) In addition, as mentioned in subsection of the authentication server, the data server receives C_{id} and C_{key} from the authentication server and keeps them in a new table called AS_{table}. The C_{id} and C_{key} are unique for each query of each client. When the private server sends $E[id]$ list with C_{id}, the data server finds relevant $E[d]$ list by matching $E[id]$ in the data server to $E[id]$ coming from the private server. Additionally, the data server matches C_{id} coming from the private server to C_{id} column in the table and finds the C_{key} of the client. The data server encrypts the $E[d]$ list with C_{key} symmetrically to produce $E[E[d]]$ list and sent it to the private server.

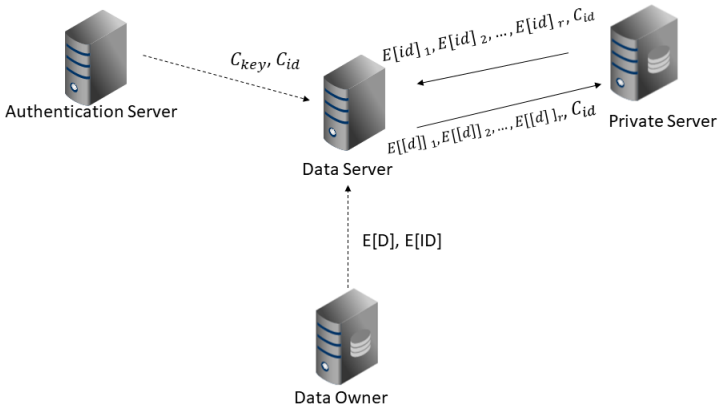

Figure 5.8 : The data server and its communication with the other entities.

The data server cannot infer any information about the identity of the client since the C_{id} and C_{key} are different for every query. Documents and their ids are encrypted by data owner. There is no security vulnerability in terms of the content of the documents.

5.2.3 Stages of the flow of SDCA-MU

In the section only the processes that are different from these in SDCA-SU. SDCA-MU processes are actually the same as SDCA-SU for data outsourcing (Subsection 3.2.3.1) and query creating (Subsection 3.2.3.2)

5.2.3.1 Query authentication

As explained in Chapter 2, the authentication protocol for SDCA-SU requires k_a combination for tags. However, in SDCA-MU 2 additional combination; one for C_{id} and the other for C_{key} is required. Therefore, before sending the query to searching server, each client generates own unique $k_a + 2$ combinations selected from their K_a. C_{id} is unique id for each client and it is generated for every query, C_{key} is unique symmetric encryption key for each client and it is generated for every query to encrypt $E[d]$ in the data server and to decrypt only related client.

After sending the query to searching server, searching server sends only the user k_a to the authentication server. According to Authentication Protocol [14], the authentication server can know C_{id} and C_{key} that are not sent by the client. If the authentication server finds the user, the server accepts the client and sends C_{id} to searching server, C_{id} and C_{key} are sent to the data server and the system continuous

working. If the authentication server does not find the user or detects the k_a has been received previously, the client request is terminated (6.a-6.b in Figure 5.1).

5.2.3.2 Query searching

When the searching server receives the accept message with C_{id}, it starts to search the query. The server has homomorphic encrypted $TF-IDF$ table of the documents (S) and homomorphic encrypted TF table of the query (QTF). The cosine similarity value is found between the query vector $Q = [q_1, q_2, \ldots, q_M]$ and the nth index vector $S_n = [s_{n,1}, s_{n,2}, \ldots, s_{n,M}]$ by Eq. 3.1.

The similarity between Q and S (index of all documents) is showed in:

$$CS = [cs_n \mid 1 \leq n \leq N] \tag{5.1}$$

The searching server generates a table which includes the similarity vector. The table has three columns; the first column, $col1$, keeps the random number, second column, $col2$, keeps the encrypted ids ($E[id]$s) and third column, $col3$, keeps the similarity values of the $E[id]$s. The server orders the table according to $col1$, then generates new two tables. One table includes $col1$ and $col2$ is sent private server with C_{id}, while second table includes $col1$ and $col3$ is sent to ranking server with r and C_{id}(7 in Figure 5.1).

5.2.3.3 Documents retrieval

The ranking server decrypts the similarity column i.e. $col3$ with K_h and sets the table in descending order according to the decrypted values of similarity, i.e. the actual similarity values. A new table is created from the highest r rows $[col1]$ and it includes random number column by the ranking server and is sent to private server (8 in Figure 5.1).

The private server has more than one table coming from searching server and ranking server. Firstly the server finds tables of having the same C_{id} than it compares two tables coming from the searching server and the ranking server according to $col1$. Therefore, the server finds $E[id]$s of the related documents, and sends them to data server. (9.a in Figure 5.1).

The data server matches $E[id]$s coming from the data owner and related $E[id]$s of coming from the private server (9.b in Fig.2), finds the relevant $E[d]$s and associates $E[d]$s with C_{id} coming from private server. The server has a table which includes C_{key}s and C_{id}s coming from the authentication server. $E[d]$s are encrypted again with related C_{key} by the data server to produce $E[E[d]]$ list. $E[E[d]]$ list are sent to the private server.

The private server keeps a list including C_{id} and $E[E[d]]$ list coming from the data server. When a client wants to retrieve results documents about searching query, it sends to C_{id} to the private server (10.a in Figure 5.1). If the server finds the client C_{id}, the server sends the relevant $E[E[d]]$s to the client (10.b in Figure 5.1).

Finally, the client has $E[E[d]]$ list that is encrypted with symmetric encryption twice one by the data owner and the other by the data server. The client has both encryption keys, K_s and C_{id}, to decrypt $E[E[d]]$ list to end up with the document.

6. SIMULATION AND RESULTS OF SDCA-MU

6.1 Simulations

In this chapter, the SDCA-MU is simulated and the results of the simulation are explained.

6.1.1 Dataset information

From the implementation point of view, SDCA-MU is exactly the same as SDCA-SU considering the use of dataset and relevant prepossessing operations. Note that the same environment is also used for simulation purposes.

The flow diagram of the SDCA-MU system is shown in Figure 6.1. In the program, servers have run infinitely. Users search their query and they terminates.

6.2 Results

The essential difference between SDCA-SU and SDCA-MU is from the perspective of security and privacy rather than data retrieve process. Therefore, in this section only the security and privacy issues are discussed. The system consisting of multiple servers is designed to data retrieval. Each server is considered as not malicious curious. Therefore, data forms documents, id of documents, similarity values, TF-IDF table and QTF vector are considered as sensitive data such that any combination at any server may not lead on inference on data. These data must be protected from the servers and the servers should not relate between data and client. Table 6.1 shows the servers and the variables that are known by the server in the system. The proposed technique provides the requirements of privacy preserved data retrieval.

Documents and their ids are encrypted separately with K_S by the data owner. Encrypted documents $E[D]$ list and encrypted documents id $E[ID]$ list are sent to the data server by data owner and stored in there. $E[ID]$ with the normalized $TF-IDF$ table (S),

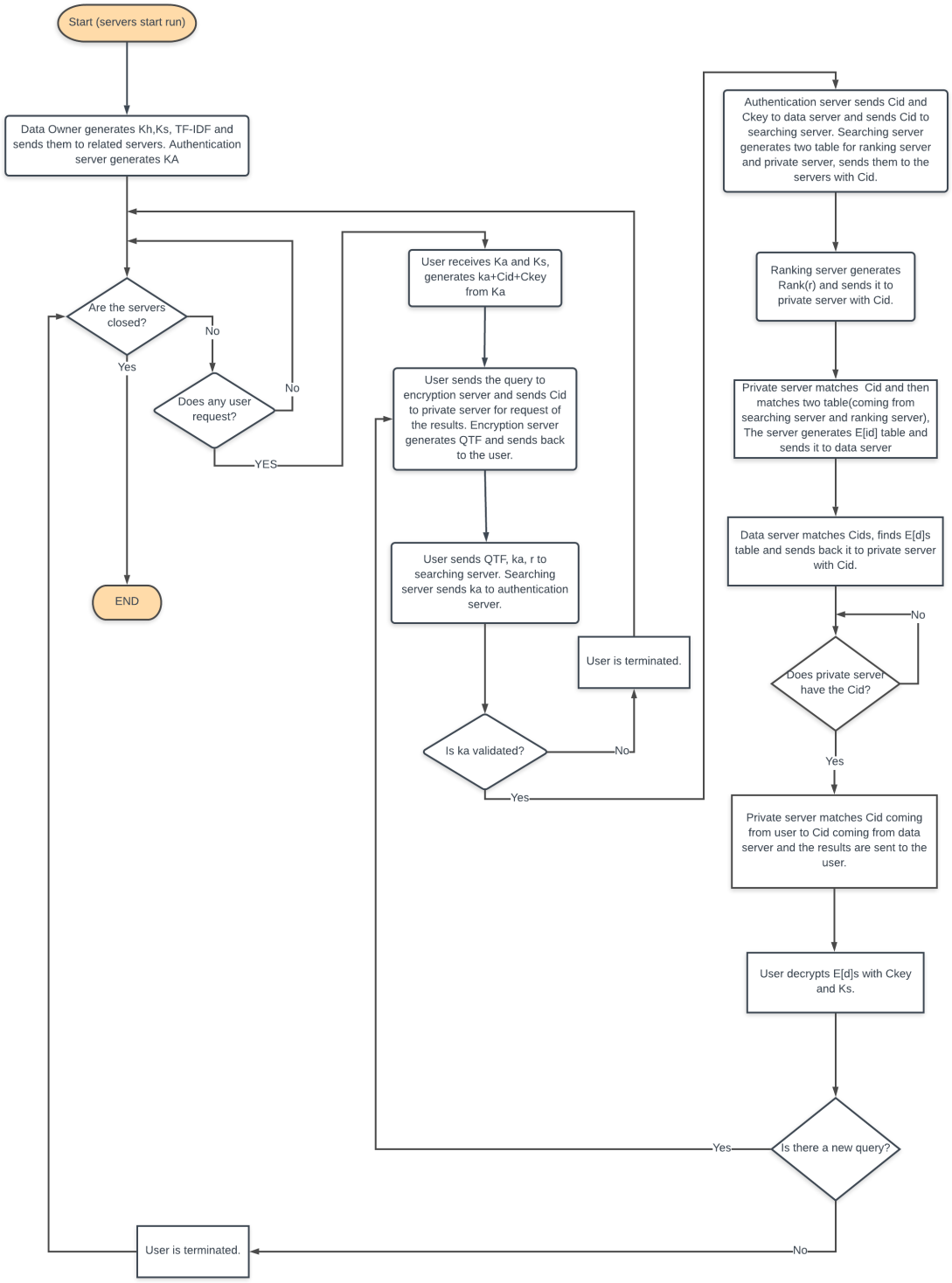

Figure 6.1 : The flow diagram of the SDCA-MU.

are sent to the searching server. The servers that know $E[D]$ and $E[ID]$ are shown in Table 6.1. In the system, $E[D]$ is known by only the data server, $E[id]$ is known by the private server, the searching server and the data server. The servers require symmetric encryption key K_S for decrypting the documents and their id. They cannot have the K_S so the document security is provided.

Table 6.1 : Variables known to servers in the system.

Variables	Authentication Server	Searching Server	Ranking Server	Private Server	Encryption Server	Data Server
K_h	$\times$	$\otimes$	$\surd$	$\times$	$\surd$	$\times$
K_s	$\otimes$	$\times$	$\times$	$\times$	$\times$	$\otimes$
K_a	$\surd$	$\times$	$\times$	$\times$	$\otimes$	$\times$
k_a	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$
C_{id}	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
C_{key}	$\surd$	$\times$	$\times$	$\times$	$\times$	$\surd$
S (TF-IDF)	$\times$	$\surd$	$\times$	$\times$	$\times$	$\times$
QTF	$\times$	$\surd$	$\times$	$\times$	$\surd$	$\times$
$Similarity$	$\times$	$\otimes$	$\surd$	$\otimes$	$\times$	$\times$
$E[Similarity]$	$\times$	$\surd$	$\surd$	$\times$	$\times$	$\times$
$E[id]$	$\times$	$\surd$	$\otimes$	$\surd$	$\times$	$\surd$
$E[d]$	$\times$	$\times$	$\times$	$\surd$	$\times$	$\surd$

The values of the $TF-IDF$ table are shown index frequency of keywords. $TF-IDF$ table is sparse which includes many zero and non-zero values are usually either same of close to each other. Although the values in the table are encrypted because of the poor distribution of the values, index pattern can be inferred. Since the encrypted values of the same values are also the same. In order to avoid such a problem, the values in the tablet are normalized as explained in the Section 2 before encryption. Therefore, the same values and zeros become different but close to each other. After the encryption, the values become unique values. The new table S is known by only the searching server. The server needs K_h for decrypting it. Thus, the index frequency cannot be known in the system and index privacy requirement is fulfilled.

Like $TF-IDF$ table, QTF vector is normalized and then encrypted with K_h. Although the normalized values are close to each other, with the encryption they became unique number with random difference. In the system, the QTF is known by the encryption server and the searching server as showing in Table 6.1. Since the searching server does not have K_h, the server cannot decrypt the vector values. Therefore, query frequency and query pattern cannot be known and the query index privacy requirement is provided.

The client, use a single-use key for each query. The authentication server shares authentication key K_a, with the client by using HEADA authentication protocol. According to the protocol, the client cannot use a previously used k_a for its query

to prevent replay attacks and the server accept or reject the client repairing the k_a. Moreover, although k_a is known by the searching server in the system, the server cannot search any query because the authentication protocol cannot be performed between the searching server and the authentication server and the authentication server recognizes as a fake entry.

To generate QTF, the encryption server must have keywords of documents and query so the query and the keyword list are sent to the encrypted server by hashed. Therefore, the documents as well as the contents of the keywords are protected from encryption server.

For SDCA-MU, C_{id} and C_{key} are included in the system. With the authentication protocol, the authentication server and client select $m+2+1+1$ combinations. $m+2$ combinations are used k_a generation and verification between the authentication server and the client (it is explained in Section 2), the other $+1$ combination is used C_{id} and last $+1$ combination is for C_{key}. Thus, C_{id} and C_{key} are generated unique for every client and every query. Because of the unique combinations, although all servers know the C_{id}, identity privacy is provided.

C_{key} encrypts and decrypts the query results of the clients by the data server that receives from the authentication server and the data server sends encrypted results to the private server. Thanks to C_{id}, the private server can know which query results are sent to which client. If any unverified client wants to request, the private server does not send to the client any result since its C_{id} is undefined. Even though the private server sends any query result to any undefined client, the client does not know right C_{key} and cannot decrypt the results. The authentication server and the data server know the C_{key} but they do not have the K_s for decrypting the results. Therefore, the query results are protected twice.

The private server matches C_{id} to $E[E[d]]$ encrypted by C_{key} and $E[id]$. In the system, every client has unique C_{id}, C_{key} for every query, so the server cannot relate them.

The similarity values of the documents on the searching server are not real results because $TF-IDF$ table and QTF vector are homomorphic encrypted with K_h. Thus, the similarity results are encrypted. If the server has K_h, it can calculate real similarity values and relate between $E[id]$ and similarity values.

Although the ranking server has the K_h and the real similarity values, the server cannot find any relation between the documents and the similarity values, because the server knows only random numbers and similarity values, it does not have any documents and their id information.

If the data owner changes the K_h, the client is not affected with this process, and the system continuous in the same way because of the encryption server. The number of the parties that know K_h decreases, therefore the system becomes more securely and user's working load is reduced.

In the SDCA-NU, if the servers generate any query and want to search the it, the servers cannot receive any result. The servers need K_a for generating C_{id}, C_{key} and k_a, so only the authentication server can search the query. Although the authentication server can send any query, receive result from the private server with C_{id} and decrypt the result with C_{key}, the server cannot decrypt the result because it does not have K_s.

7. CONCLUSIONS AND FUTURE WORKS

7.1 Conclusions

In this research, a number of secure and privacy preserving of data retrieval problems. That have not been sorted out in a previous research [12] on cloud computing are attacked and a novel solution for those is suggested. In the suggested model, new servers are included in the previously proposed basic privacy preserving data retrieval system. With the new servers, it is aimed to reduce the data information to prevent inferences on a cloud server, in order to provide security and privacy requirements of data retrieval system.

Privacy preserving data retrieval on the data clouds have been researched by many groups using different approaches and techniques. In this research, Fully Homomorphic Encryption is used to encrypt data and to make calculations on the encrypted data, the normalization process is applied on the TF-IDF and QTF tables to provide uniqueness before encryption. HEADA authentication protocol [14] is used to provide anonymity and verify the users.

In this research, two new architectures are designed; the first architecture is called SDCA-SU is for serving only single user at a time, the other is called SDCA-MU is for serving concurrent multiuser. These architectures are designed by basing the previous research [12].

The most significant contribution of the SDCA-SU is introducing a new server (Encryption server) to hide the homomorphic key from the all clients and managing the homomorphic key. Moreover, SDCA-SU overcome the other shortcomings of depicted on the previous research which are; the previous architecture has unnecessary data traffic and the private server knows critical information about documents. The SDCA-SU is evaluated by simulation and the results showed that the normalization process provides privacy and security on the data retrieval system and it is not affected the retrieved documents and their similarity. Similarly, not only fully homomorphic

encryption ensures operations on the encrypted data and provides documents and query security but also is not affected the retrieved documents and their similarity.

SDCA-SU is designed for only a single user at a time and if more than one user search a query, the servers should have more information about the clients and many infer through matching tables. In SDCA-MU, additional information about the clients are sent to the servers (searching server, data server, private server, ranking server, encryption server) of the architecture. Similar to the SDCA-SU, SDCA-MU are evaluated by the simulation and the results indicated that SDCA-MU provides document privacy, index privacy, query privacy, identity privacy ant it prevents replay attacks. Moreover, the retrieved documents can be decrypted by only related client.

As a result, in the thesis, titled *SDCA : Secure and Privacy Preserving Data Retrieval Architecture on Cloud Computing* novel solutions for the data retrieval system on the cloud computing have been suggested. Both of solutions are providing 9+1 requirements of a privacy preserving and secure data retrieval system. Those solutions are also tested through simulations. We believe that both architectures can be used a platform for various applications with low cost low capacity end users.

7.2 Future Works

Although the system is fulfilling the requirements of privacy preserving and secure data retrieval on clouds, there are some issues to be research. These issues are left out of the content of this for further research. SDCA has only one data server to match $E[id]$ and to find $E[d]$. That may cause to make inference by the data server in the long run. Similarly, private server knows $E[id]$ and $E[d]$ lists, the server can classify queries with time. Thus, distributed data servers can be used instead of single data server and it can be avoided from knowing whole $E[id]$ list in the data server. For the private server, the functionality of the private server can be provides another server or client, a new architecture can be developed for this.

Real applications can run to test the real-time performance of the architecture.

Although it is not considered within the content of this thesis, this research is an ongoing one with which we are aiming to implement IoT based low-end system for e-health.

REFERENCES

[1] **Mell, P.**, **Grance, T.** *et al.* (2011). The NIST definition of cloud computing.

[2] **Kim, W.** (2009). Cloud computing: Today and tomorrow., *Journal of object technology*, *8*(1), 65–72.

[3] **Omotunde, A.**, **Oludele, A.**, **Kuyoro, S. and Chigozirim, A.** (2013). Survey of Cloud Computing Issues at Implementation Level, *Journal of Emerging Trends in Computing and Information Sciences*, *4*, 91=96.

[4] **Url-1**, Survey Reveals Cloud Computing Benefits and Challenges, `https://www.datamation.com/cloud-computing/survey-reveals-cloud-computing-benefits-and-challenges.html`.

[5] **Tang, J.**, **Cui, Y.**, **Li, Q.**, **Ren, K.**, **Liu, J. and Buyya, R.** (2016). Ensuring security and privacy preservation for cloud data services, *ACM Computing Surveys (CSUR)*, *49*(1), 13.

[6] **Manel, M.**, **Saied, Y.B. and Saidane, L.A.** (2018). Privacy Preserving Approaches in Cloud Computing.

[7] **Neela, T.J. and Saravanan, N.** (2013). Privacy preserving approaches in cloud: a survey, *Indian Journal of Science and Technology*, *6*(5), 4531–4535.

[8] **Zhou, M.**, **Zhang, R.**, **Xie, W.**, **Qian, W. and Zhou, A.** (2010). Security and Privacy in Cloud Computing: A Survey, *2010 Sixth International Conference on Semantics, Knowledge and Grids*, pp.105–112.

[9] **Chakrabarti, A.** (2007). *Grid Computing Security*, Springer-Verlag, Berlin, Heidelberg.

[10] **Cao, N.**, **Wang, C.**, **Li, M.**, **Ren, K. and Lou, W.** (2014). Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data, *IEEE Transactions on Parallel and Distributed Systems*, *25*(1), 222–233, `http://www.sciencepublishinggroup.com/journal/paperinfo.aspx?journalid=132{\&}doi=10.11648/j.ajnc.20150403.11http://ieeexplore.ieee.org/document/6674958/`.

[11] **Zhou, J.**, **Cao, Z.**, **Dong, X. and Vasilakos, A.V.** (2017). Security and Privacy for Cloud-Based IoT: Challenges, *IEEE Communications Magazine*, *55*(1), 26–33.

[12] **Dawoud, M. and Altilar, D.T.** (2016). Privacy-preserving Data Retrieval using Anonymous Query Authentication in Data Cloud Services, *2*(Closer), 171–180.

[13] **Dawoud, M.** (2017). Privacy preserving search and data retrieval from data clouds, *Ph.D. thesis*, Istanbul Technical University.

[14] **Dawoud, M. and Altilar, D.T.** (2016). HEADA: a low cost RFID authentication technique using homomorphic encryption for key generation, *Security and Communication Networks*, *9*(17), 4182–4191, http://doi.wiley.com/10.1002/sec.1597.

[15] **Li, P.**, **Li, J.**, **Huang, Z.**, **Gao, C.Z.**, **Chen, W.B. and Chen, K.** (2018). Privacy-preserving outsourced classification in cloud computing, *Cluster Computing*, *21*(1), 277–286, http://link.springer.com/10.1007/s10586-017-0849-9.

[16] **Weng, L.**, **Amsaleg, L.**, **Furon, T.**, **Weng, L.**, **Amsaleg, L.**, **Furon, T.**, **Media, P.p.O.**, **Ieee, S.**, **Weng, L.**, **Amsaleg, L. and Furon, T.** (2016). Privacy-Preserving Outsourced Media Search To cite this version : Privacy-Preserving Outsourced Media Search.

[17] **Qu, J.**, **Zhang, G. and Fang, Z.** (2017). Prophet: A Context-Aware Location Privacy-Preserving Scheme in Location Sharing Service, *Discrete Dynamics in Nature and Society*, *2017*, 1–11, https://www.hindawi.com/journals/ddns/2017/6814832/.

[18] **Ocansey, S.K.**, **Ametepe, W.**, **Li, X.W. and Wang, C.** (2018). Dynamic searchable encryption with privacy protection for cloud computing, *International Journal of Communication Systems*, *31*(1), e3403, http://doi.wiley.com/10.1002/dac.3403.

[19] **Kocabaş, Ö.** Towards Privacy-Preserving Medical Homomorphic Encryption.

[20] **Pasupuleti, S.K.**, **Ramalingam, S. and Buyya, R.** (2016). An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing, *Journal of Network and Computer Applications*, *64*, 12–22, http://dx.doi.org/10.1016/j.jnca.2015.11.023.

[21] **Bleichenbacher, D.** (1998). Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1, *Annual International Cryptology Conference*, Springer, pp.1–12.

[22] **Tsiounis, Y. and Yung, M.** (1998). On the security of ElGamal based encryption, *International Workshop on Public Key Cryptography*, Springer, pp.117–134.

[23] **Paillier, P.** (1999). Public-key cryptosystems based on composite degree residuosity classes, *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, pp.223–238.

[24] **Gentry, C.** (2009). Fully homomorphic encryption using ideal lattices, *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09*, ACM Press, New York, New York, USA, p.169, http://portal.acm.org/citation.cfm?doid=1536414.1536440.

[25] **Xiang, G., Yu, B. and Zhu, P.** (2012). A algorithm of fully homomorphic encryption, *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, IEEE, pp.2030–2033.

[26] **Gopal, G.N. and Singh, M.P.** (2012). Secure similarity based document retrieval system in cloud, *Proceedings - 2012 International Conference on Data Science and Engineering, ICDSE 2012*, 154–159.

[27] **Dawoud, M. and Altilar, D.T.** (2014). Privacy-Preserving Search in Data Clouds Using Normalized Homomorphic Encryption, 62–72.

[28] **SALTON, G. and BUCKLEY, C.** (1988). Term-Weighting Approaches in Authomatic Text Retrieval, *Information Processing & Management*, *24*(5), 513–523.

[29] **Hammouda, K. and Kamel, M.**, Web Mining Data - UW-CAN-DATASET, http://pami.uwaterloo.ca/~hammouda/webdata/.

[30] **Mitchell, T.**, 20 Newsgroups, https://archive.ics.uci.edu/ml/machine-learning-databases/20newsgroups-mld/20newsgroups.data.html.

[31] **Url-2**, https://www.findjar.com/jar/org/htmlparser/htmlparser/1.6/htmlparser-1.6.jar.html?all=true.

[32] **Url-3**, https://tartarus.org/martin/PorterStemmer/java.txt.

[33] **Daemen, J. and Rijmen, V.** (1999). AES proposal: Rijndael.

[34] **Sasaki, Y.** *et al.* (2007). The truth of the F-measure, *Teach Tutor mater*, *1*(5), 1–5.

APPENDICES

APPENDIX A.1 : Stopwords Lists

APPENDIX A.1

Three lists of stopwords are used to remove stopwords from the documents. These lists are followed:

Long lists of stopwords : a; able; about; above; abst; accordance; according; accordingly; across; act; actually; added; adj; adopted; affected; affecting; affects; after; afterwards; again; against; ah; all; almost; alone; along; already; also; although; always; am; among; amongst; an; and; announce; another; any; anybody; anyhow; anymore; anyone; anything; anyway; anyways; anywhere; apparently; approximately; are; aren; arent; arise; around; as; aside; ask; asking; at; auth; available; away; awfully; b; back; be; became; because; become; becomes; becoming; been; before; beforehand; begin; beginning; beginnings; begins; behind; being; believe; below; beside; besides; between; beyond; biol; both; brief; briefly; but; by; c; ca; came; can; cannot; can't; cause; causes; certain; certainly; co; com; come; comes; contain; containing; contains; could; couldnt; d; date; did; didn't; different; do; does; doesn't; doing; done; don't; down; downwards; due; during; e; each; ed; edu; effect; eg; eight; eighty; either; else; elsewhere; end; ending; enough; especially; et; et-al; etc; even; ever; every; everybody; everyone; everything; everywhere; ex; except; f; far; few; ff; fifth; first; five; fix; followed; following; follows; for; former; formerly; forth; found; four; from; further; furthermore; g; gave; get; gets; getting; give; given; gives; giving; go; goes; gone; got; gotten; h; had; happens; hardly; has; hasn't; have; haven't; having; he; hed; hence; her; here; hereafter; hereby; herein; heres; hereupon; hers; herself; hes; hi; hid; him; himself; his; hither; home; how; howbeit; however; hundred; i; id; ie; if; i'll; im; immediate; immediately; importance; important; in; inc; indeed; index; information; instead; into; invention; inward; is; isn't; it; itd; it'll; its; itself; i've; j; just; k; keep; keeps; kept; keys; kg; km; know; known; knows; l; largely; last; lately; later; latter; latterly; least; less; lest; let; lets; like; liked; likely; line; little; 'll; look; looking; looks; ltd; m; made; mainly; make; makes; many; may; maybe; me; mean; means; meantime; meanwhile; merely; mg; might; million; miss; ml; more; moreover; most; mostly; mr; mrs; much; mug; must; my; myself; n; na; name; namely; nay; nd; near; nearly; necessarily; necessary; need; needs; neither; never; nevertheless; new; next; nine; ninety; no; nobody; non; none; nonetheless; noone; nor; normally; nos; not; noted; nothing; now; nowhere; o; obtain; obtained; obviously; of; off; often; oh; ok; okay; old; omitted; on; once; one; ones; only; onto; or; ord; other; others; otherwise; ought; our; ours; ourselves; out; outside; over; overall; owing; own; p; page; pages; part; particular; particularly; past; per; perhaps; placed; please; plus; poorly; possible; possibly; potentially; pp; predominantly; present; previously; primarily; probably; promptly; proud; provides; put; q; que; quickly; quite; qv; r; ran; rather; rd; re; readily; really; recent; recently; ref; refs; regarding; regardless; regards; related; relatively; research; respectively; resulted; resulting; results; right; run; s; said; same; saw; say; saying; says; sec; section; see; seeing; seem; seemed; seeming; seems; seen; self; selves; sent; seven; several; shall; she; shed; she'll; shes; should; shouldn't; show; showed; shown; showns;

shows; significant; significantly; similar; similarly; since; six; slightly; so; some; somebody; somehow; someone; somethan; something; sometime; sometimes; somewhat; somewhere; soon; sorry; specifically; specified; specify; specifying; state; states; still; stop; strongly; sub; substantially; successfully; such; sufficiently; suggest; sup; sure; t; take; taken; taking; tell; tends; th; than; thank; thanks; thanx; that; that'll; thats; that've; the; their; theirs; them; themselves; then; thence; there; thereafter; thereby; thered; therefore; therein; there'll; thereof; therere; theres; thereto; thereupon; there've; these; they; theyd; they'll; theyre; they've; think; this; those; thou; though; thoughh; thousand; throug; through; throughout; thru; thus; til; tip; to; together; too; took; toward; towards; tried; tries; truly; try; trying; ts; twice; two; u; un; under; unfortunately; unless; unlike; unlikely; until; unto; up; upon; ups; us; use; used; useful; usefully; usefulness; uses; using; usually; v; value; various; 've; very; via; viz; vol; vols; vs; w; want; wants; was; wasn't; way; we; wed; welcome; we'll; went; were; weren't; we've; what; whatever; what'll; whats; when; whence; whenever; where; whereafter; whereas; whereby; wherein; wheres; whereupon; wherever; whether; which; while; whim; whither; who; whod; whoever; whole; who'll; whom; whomever; whos; whose; why; widely; willing; wish; with; within; without; won't; words; world; would; wouldn't; www; x; y; yes; yet; you; youd; you'll; your; youre; yours; yourself; yourselves; you've; z; zero");

Short lists of stopwords : a; about; above; after; again; against; all; am; an; and; any; are; aren't; as; at; be; because; been; before; being; below; between; both; but; by; can't; cannot; could; couldn't; did; didn't; do; does; doesn't; doing; don't; down; during; each; few; for; from; further; had; hadn't; has; hasn't; have; haven't; having; he; he'd; he'll; he's; her; here; here's; hers; herself; him; himself; his; how; how's; i; i'd; i'll; i'm; i've; if; in; into; is; isn't; it; it's; its; itself; let's; me; more; most; mustn't; my; myself; no; nor; not; of; off; on; once; only; or; other; ought; our; ours; ; ourselves; out; over; own; same; shan't; she; she'd; she'll; she's; should; shouldn't; so; some; such; than; that; that's; the; their; theirs; them; themselves; then; there; there's; these; they; they'd; they'll; they're; they've; this; those; through; to; too; under; until; up; very; was; wasn't; we; we'd; we'll; we're; we've; were; weren't; what; what's; when; when's; where; where's; which; while; who; who's; whom; why; why's; with; won't; would; wouldn't; you; you'd; you'll; you're; you've; your; yours; yourself; yourselves

Google lists of stopwords : I; i; a; about; an; are; as; at; be; by; com; de; en; for; from; how; in; is; it; la; of; on; or; that; the; this; to; was; what; when; where; who; will; with; und; the; www

CURRICULUM VITAE

Name Surname: Büşranur Bülbül

Place and Date of Birth: Trabzon/Turkey, 11.11.1992

E-Mail: bulbulb17@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2015, Melikşah University, Faculty of Engineering and Architecture, Computer Engineering Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2017-..., Research Assistant at ITU Graduate School of Science Engineering and Technology
- 2016-2017, Research Assistant at RTEU Engineering Faculty

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Bulbul B., Altilar D.T., 2019: Privacy Preserving Data Retrieval on Data Clouds with Fully Homomorphic Encryption. *4th International Conference on Computer Science and Engineering UBMK'19*, September 11-15, 2019 Samsun, Turkey.