ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

PRIVACY PRESERVING SEARCH AND DATA RETRIEVAL FROM DATA CLOUDS

Ph.D. THESIS

Mohanad DAWOUD

Department of Computer Engineering

Computer Engineering Programme

MONTH YEAR OF DEFENSE

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE

ENGINEERING AND TECHNOLOGY

PRIVACY PRESERVING SEARCH AND DATA RETRIEVAL FROM DATA CLOUDS

Ph.D. THESIS

Mohanad DAWOUD
(504102503)

Department of Computer Engineering

Computer Engineering Programme

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

MONTH YEAR OF DEFENSE

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

VERİ BULUTLARINDA MAHREMİYET KORUMALI ARAMA VE VERİ GETİRME YONTEMİ

DOKTORA TEZİ

Mohanad DAWOUD
(504102503)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

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

TEZİN SAVUNULDUĞU AY YIL

Mohanad DAWOUD, a Ph.D. student of ITU Graduate School of Science Engineering and Technology 504102503 successfully defended the thesis entitled **"PRIVACY PRESERVING SEARCH AND DATA RETRIEVAL FROM DATA CLOUDS"**, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. D. Turgay ALTILAR** Istanbul Technical University	....
Jury Members :	**Prof. Dr. Bülent ÖRENCİK** Beykent University	....
	Assoc. Prof. Dr. Sıddıka Berna Örs YALÇIN Istanbul Technical University	....
	Prof. Dr. Albert LEVİ Sabancı University	....
	Asst. Prof. Dr. Tolga OVATMAN Istanbul Technical University	....

Date of Submission : 03 February 2017
Date of Defense : 15 June 2017

To my parents,
To my spouse and children,
To the martyrs of honor, dignity, and democracy,

FOREWORD

I would like to express my gratitude to the staff of Computer Engineering department in Istanbul Technical University.

The first days in the university were very easy with the help of Zuhal Yılmazer, the dean’s secretary. Special thanks also to Ahmet Dinçr from the registration office for his assistant.

I would like also to thank all the faculty members, particularly whose I worked with; Dr. D. Turgay Altılar, Prof. Muhittin Gökmen, Dr. S. Berna Ors Yalcin, Dr. Şule Gündüz, and Dr. A. Cüneyd Tantuğ.

Special thanks again to my supervisor Dr. D. Turgay Altılar, I learned too much from you. I am forever grateful for all the opportunities that have been provided for me. And thank you for never giving up on me.

I would like to thank Türkiye Bursları and Yurtdışı Türkler ve Akraba Topluluklar Başkanlığı for their financial support. I hope the best for these remarkable organizations.

I would not have made it this far without the support of my dad, everything have started by you. To my mother, whatever happens in this crazy world, your words always have a magic effect that make me calm. Thanks also to my siblings for their reviews, and special thanks to my wife and children for their patience.

Month year of defense Mohanad DAWOUD
Privacy-Preserving Search and Data Retrieval from Data Clouds

TABLE OF CONTENTS

ABBREVIATIONS

AHEE	: Algebra Homomorphic Encryption Scheme
APV	: Average precision Value
BaaS	: Backup as a Service
BPaaS	: Business Process as a Service
CaaS	: Communications as a Service
CRC	: Cyclic Redundancy Check
DBaaS	: Database as a Service
DMaaS	: Data Mining as a Service
DoS	: Denial of Service
DWaaS	: Data Warehousing as a Service
GC	: Green Cloud
GCC	: Green Cloud Computing
HEADA	: A Low Cost RFID Authentication Technique Using Homomorphic Encryption for Key Generation
HTML	: HyperText Markup Language
IaaS	: Infrastructure as a Service
IT	: Information Technology
IoT	: Internet of Things
LSH	: Locality Sensitive Hashes
NIST	: National Institute of Standards and Technology
OPSE	: Order-Preserving Symmetric Encryption
PaaS	: Platform as a Service
PPSED	: Privacy Preserving Search on Encrypted Data
PRNG	: Pseudo-Random Number Generator
QAA	: Query Anonymous Authentication
RFID	: Radio-Frequency Identification
SaaS	: Software as a Service
SOA	: Service-Oriented Architecture
TF	: Term-Frequency
TF-IDF	: Term Frequency-Inverse Document Frequency
XaaS	: Everything as a Service

LIST OF TABLES

LIST OF FIGURES

PRIVACY PRESERVING SEARCH AND DATA RETRIEVAL FROM DATA CLOUDS

SUMMARY

Recently, cloud computing systems are considered as one of the most important advances in the field of Information Technology (IT) applications. However, security and privacy still form the main concern that slower the widespread use of these systems in the sensitive applications that need high level of privacy and security on their data.

Outsourcing the data to unknown locations in the cloud needs securing these data by encrypting them. On the other hand, retrieving these data (or part of them) may need revealing sensitive data to unauthorized parties. Therefore, many techniques are proposed to handle this problem, which is known as Privacy-Preserving Data Retrieval (PPDR). These techniques tried to minimize the sensitive data that need to be revealed, which negatively affects the security and privacy of the data and/or the quality (or accuracy) of data retrieval.

In this thesis, 9 security requirements are defined to satisfy a high level of security and privacy in a PPDR system based on the reported techniques in the literature. Together with these 9 security requirements, a retrieval efficiency requirement is defined to keep the retrieval efficiency high. Therefore, a new technique is proposed to satisfy these 9 security and 1 efficiency requirements (which are noted as 9+1 requirements in this thesis). The proposed technique utilizes Query Anonymous Authentication -which is derived from another newly proposed Radio Frequency Identification (RFID) anonymous authentication technique called HEADA- together with a multi-server setting to satisfy these requirements. The technique provides an efficient ranking-based data retrieval by using the cosine similarity of the TF-IDF vectors. The accuracy and ranking of the data retrieval is tested on three different datasets and shown to be high compared to other techniques as well as to the un-encrypted indexes and data. The analysis of the proposed technique also shows that the technique is able to satisfy all the 9 security requirements, which are unsatisfied completely in the techniques reported in the literature.

VERİ BULUTLARINDA MAHREMİYET KORUMALI ARAMA VE VERİ GETİRME YONTEMİ

ÖZET

Son zamanlarda, bulut bilişim sistemleri, Bilgi Teknolojisi (BT) uygulamaları alanındaki en önemli gelişmelerden biri olarak görülüyor. Bununla birlikte, güvenlik ve mahremiyet endişeleri, veri üzerinde yüksek seviyede gizlilik ve güvenlik gerektiren, özel kişisel bilgiler içeren veya güvenlik açıklarının maddi kayıplara yol açabileceği hassas uygulamalarda bulut bilişim sistemlerinin daha yaygın şekilde kullanılmasına engel olmaktadır.

Verileri bulut bilişim sistemleri üzerinde, fiziksel olarak bilinmeyen konumlarda veya üçüncü taraf sağlayıcılar aracılığı ile saklamak , bu verilerin şifrelenerek güvence altına alınmasını gerektirir. Öte yandan, bu verilerin veya bir bölümünün bulut systemi üzerinden çağırılması, hassas verilerin, üçüncü partilere açıklanmasına, üçüncü taraflar tarafından erişilebilir hale gelmesine neden olabilir. Bu da çeşitli mahremiyet ve güvenlik sorunlarına yol açar.

Dolayısıyla, bu sorunun üstesinden gelmek için, Gizliliği Koruyarak Veri Çağırma (PPDR – Privacy Preserving Data Retrieval) olarak bilinen birçok teknik halihazırda önerilmektedir. Bu teknikler, üçüncü taraflarca okunur hale gelen verilerin güvenliği ve mahremiyetini sağlamayı amaçlar. Ancak tüm Gizliliği Koruyarak Veri Çağırma teknikleri, veri kalitesi ve veri çağırma işinin verimliliği, tutarlılığı ve doğruluğu üzerinde bir takım negatif etkilere sahip. Bu teknikler veri çağırma işleminin güvenliğini sağlarken, olumsuz etkileri de en aza indirmeyi amaçlar.

Bu tez çalışmasında, literatürde bildirilen teknikler temel alınarak, bir PPDR sisteminde yüksek seviyede güvenlik ve gizliliği sağlamak için 9 güvenlik noktası tanımlanmıştır. Bu 9 güvenlik noktası ile birlikte, çağırılan verilerin doğruluğunu ve tutarlılığını yüksek tutmak için bir çağırma verimliliği kriteri tanımlanmaktadır. Bu nedenle, bu 9 güvenlik ve 1 verimlilik gereksinimini karşılamak için yeni bir teknik önerilmektedir (bu tezde 9 + 1 kriteri olarak belirtilmiştir).

Veri çağırma işleminin güvenlik ve mahremiyetini 9+1 kriterini sağlayarak gerçekleştirmek için önerilen teknik, HEADA isimli henüz geliştirilen bir Radio Frekansı ile Tanımlama (RFID) gizliliği sisteminden türetilmiş, çoklu sunucu yapısı ile gerçekleştirilmiştir. Önerilen teknik, TF-IDF vektörlerinin kosinüs benzerliklerini kullanarak verimliliği yüksek, etkin ve gizliliği koruyan bir sıralı veri çağırma işlemi gerçekleştiriyor.

Tekniğin verimliliği ve veri cağırma işleminin derecesi, üç farklı veri kümesi ile test edilmiştir. Bu testler, önerilen tekniğin diğer gizlilik hedefleyen şifrelenmiş tekniklerden, hatta şifrelenmemiş tekniklerden daha yüksek sonuçlar aldığını göstermektedir.

Önerilen tekniğin analizi, tekniğin literatürde bildirilen tekniklerden hiç birinin aynı anda gerçekleştiremediği, 9 kriterin bir arada sağlandığını gösterir.

1. INTRODUCTION

Information Technology (IT) systems are used increasingly in all life aspects. Therefore, the size of the data that need to be stored, processed, and transferred through different public, private, or hybrid network systems is increasing rapidly. The production of the enterprise systems as well as the need for competition with highly supported and resource-allocated systems make the clouds essential in the IT industry [4].

Cloud Computing System was defined by Foster in [5] as "A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet." Buyya defined the cloud computing system in [6] as "a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers." According to these definitions, any new or small system can have the same capabilities of the resources (computing, storage, etc) as the enterprise systems with a reasonable cost and scalable resources. Moreover, the enterprise systems can benefit from the clouds by increasing capacity or adding capabilities, by pay-per-use service, according to their current needs.

Nowadays, there are many cloud systems that offer different services with high potentials such as Amazon's EC2 [7], IBM Cloud [8], Microsoft's Azure [9], and Google Cloud Platform [10]. Pastaki Rad et al. [11] showed that many requirements should be satisfied to realize a reliable cloud. They surveyed many platforms by comparing their arrangements, foundation and infrastructure services and their main capabilities used in some leading software companies. Dikaiakos et al. [12] defined the main requirements to be: suitable software/hardware architecture, data management, cloud interoperability, security, privacy, service provisioning and cloud economics. However, these requirements can be extended into many more specific requirements.

Despite the advantages of using clouds to reduce costs and to improve the productivity, security issues should be handled carefully; they may inhibit wide adoption of the cloud model [13]. Jansen and Grance [14] provided an overview of the security and privacy challenges pertinent to public cloud computing. They pointed out considerations that organizations should consider when they outsource their data, applications and infrastructure to a public cloud environment [14].

1.1 Privacy Preserving Search on Data Clouds

The working mechanism of the cloud systems requires the transfer of users data to unknown, and probably insecure, locations for storing, processing, or both. This may cause a threat to the data's security and privacy. The physical security systems which have been used to secure the physical documents become the responsibility of the data security management protocols which are used in these cloud systems. Therefore, finding such security protocols is considered as a big challenge toward the efficient usage of cloud computing in the critical and private systems.

Storing the data in the cloud systems can be secured by the traditional symmetric or asymmetric encryption algorithms [15]. However, any data mining or retrieval processes need the data, or part of them, to be revealed to the cloud system, which may break the security and privacy rules proposed to keep/transfer data as well. Many techniques have been proposed to enable the cloud to apply searching processes on the data without revealing them, or, revealing as little as security and privacy rules, defined by the system security administrator, allow. This is known as Privacy Preserving Search on Encrypted Data (PPSED). Figure 1.1 shows the essential model of such a typical data storage/retrieval system that is taken into consideration in this thesis. The system mainly consists of three parts: data owner, a server in a cloud (cloud), and client (user). Data owner has a large number of documents that need to be indexed, searched, and partially retrieved by the user, but he does not have the processing and storage capabilities. Cloud has the processing and storage capabilities needed to serve the system, but it is assumed to be "honest-but-curious". "Honest-but-curious" means that the cloud follows the designated protocol honestly, but curious to infer useful

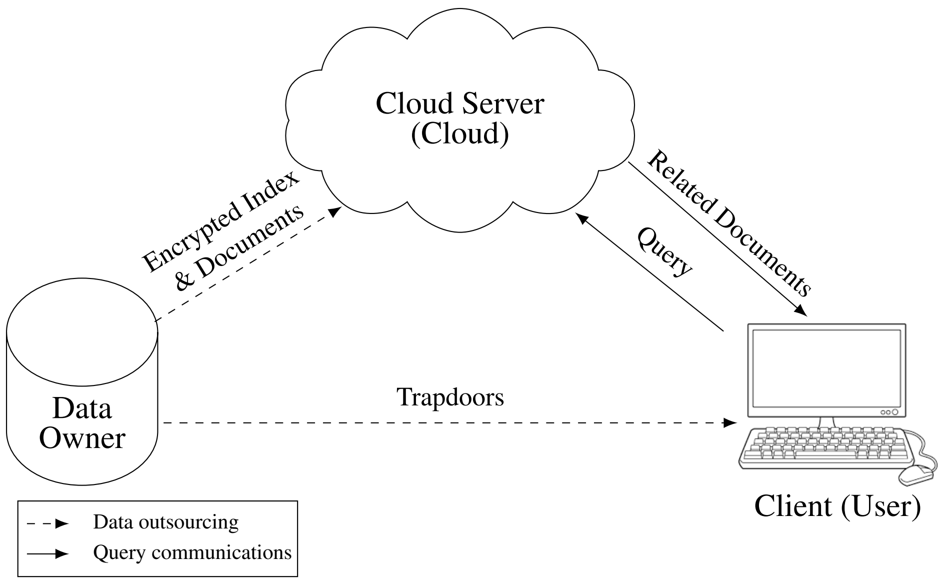

Figure 1.1: The simplest model of privacy preserving data retrieval system.

information by analysing the data flow during the protocol. User needs to retrieve documents related to a queried document (or keywords). Data owner creates indexes for the documents and store the indexes as well as the documents in the cloud in an encrypted form. He also creates trapdoors, which may have different forms according to the used technique, and sends them to the user. User uses these trapdoors together with the index of the queried document to create a query. He sends the query to the cloud which in turn is replied with the related documents.

1.2 Motivation of the Thesis

The wide adoption of cloud computing systems arose new privacy and security requirements different than the traditional ones. For example, the traditional encryption, which is used to satisfy the confidentiality property, may not be suitable in the cloud systems, where the encrypted data need to be processed by untrusted parties. Moreover, the storing server is not considered completely trusted anymore where all kinds of cryptanalysis attacks are considered probably applicable by the server. Another example is the authentication process in the cloud systems which may require the anonymity of some parties in the system. Otherwise, some sensitive information can be inferred. These examples and others show how the security and privacy requirements are extended, and some times changed, in the cloud systems. However, many commercial and business cloud systems still consider some of the traditional definitions of security and privacy requirements toward the users. These traditional definitions are used for many reasons, such as decreasing the processing cost and keeping control over the data by the service providers. The lack of experience, or in most cases the lack of required infrastructure and technology, may push the users to use these cloud services and make their data threatened [4].

In this thesis, privacy and security essential requirements needed in the data clouds, from the user perspective, are redefined. In data clouds, users outsource their data (in documents format) and retrieve them later using search queries. Moreover, a new practical model for secure data storage and retrieval in data clouds is proposed. The foundation of such a model simplifies the adoption of secure and privacy-preserving data clouds to the users.

1.3 Definitions of Privacy and Security Requirements in Data Cloud

Suppose that $features(\gamma)$ and $index(\gamma)$ are the features and index of a document γ, respectively. $query(\theta,\tau)$ is the query generated from the query document θ and the trapdoor τ. Any privacy-preserving data retrieval system should satisfy the following requirements:

1. **No Index Pattern:** For any two different documents α and β where $features(\alpha) = features(\beta)$, $index(\alpha) \neq index(\beta)$. Otherwise, the cloud can relate these documents to each other.

2. **No Query Pattern:** For any two query documents δ and θ, $query(\delta,\tau) \neq query(\theta,\tau)$. Otherwise, the data owner and the cloud can relate the users who are sending similar queries.

3. **No Documents Pattern:** For any party other than the user who generated a query $query(\delta,\tau)$, it is not possible to know the retrieved documents or the rank of the documents for the query $query(\delta,\tau)$. Otherwise, other parties can relate the retrieved documents together, or, relate the query $query(\delta,\tau)$ to some documents and disrelate it to others.

4. **No Index Frequency:** There is no pattern in the index that can be used to infer any information about the distribution of the data in the documents .

5. **No Query Frequency:** There is no pattern in the query that can be used to infer any information about the distribution of the data in the query.

6. **No Replay Attack:** For any valid query, it can not be used later by any party even the retrieved data are encrypted.

7. **Query Privacy:** Neither the cloud nor the data owner is allowed to know or to be able to infer anything about the contents of the user's queries. Also, only authorized users can make queries.

8. **Index Privacy:** For the cloud and the user, it is infeasible to know or to be able to infer anything about the contents of the index. Also, in case cloud can add fake indexes (even random ones), it can't get any useful information.

9. **Documents Privacy:** It is infeasible for the cloud or the unauthorized users to know or to infer anything about the contents of the encrypted documents.

These 9 security requirements together with the **high efficiency of data retrieval** are called the 9+1 requirements in the rest of this thesis. The efficiency of data retrieval is considered to be the accuracy of the retrieval based on a query, which is discussed in details in Chapters 4, 5, and 6.

1.4 Purpose of the Thesis

To the best of our knowledge, there is no technique that satisfies the 9+1 requirements efficiently in the current state of the art. Unsatisfying any of the 9 security requirements poses a threat on the privacy of the data, which is not negotiable in most of the current applications. On the other hand, high efficiency of data retrieval is needed to achieve the main aim of the data retrieval system. Satisfying some of the security requirements on cost of the retrieval efficiency decreases the reliability of the system.

The aim of this thesis is to design a complete data retrieval technique that satisfies the 9+1 requirements. The technique utilizes Query Anonymous Authentication (QAA) together with a multi-server setting. The technique provides an efficient ranking-based data retrieval by using the Cosine Similarity [16] of the Term Frequency-Inverse Document Frequency (TF-IDF) [17] vectors. A normalization technique for the Term-Frequency (TF) and the TF-IDF values is proposed to hide any frequency in the indexes as well as the queries. Moreover, an anonymous authentication technique is proposed using homomorphic encryption in key generation. The authentication technique is applied on the Radio-Frequency Identification (RFID) systems to examine its capabilities and features. However, the technique is shown more than suitable to be utilized in the proposed PPSED technique.

1.5 Literature Review

One of the first techniques proposed to handle the privacy preserving search on encrypted data was proposed by Song et al. [18]. Boneh et al. [19] proposed a single user technique that uses public key encryption. It enables a gateway to test whether a specific keywords are found in an email. Liu et al. [20] improved Boneh et al. technique by simplifying its operations. However, both Boneh et al. and Liu et al. techniques do not satisfy the requirements 1-7. Also, they are suitable only for single user applications which is not the case in many popular applications. Li et al. [21] proposed Authorized Private Keyword Search technique. The technique gives different privileges to the users to search on the data. It uses specific keywords and relations which is not suitable for many applications. Moreover, it does not satisfy the requirements 1-6. ChinnaSamy and Sujatha [22] proposed two rounds search technique. Although the technique insures the integrity of the retrieved documents, it does not satisfy the requirements 1-8. Kuzu et al. [23] proposed a single keyword technique that uses an encrypted Locality Sensitive Hashes (LSH) to prevent any unauthorized party from creating a query. The technique still does not satisfy the requirements 2,3, and 6. Tseng et al. [24] proposed a single keyword search technique using authentication. The technique does not satisfy the requirements 1-4 and 6. The techniques [19–24] are examples of binary keyword-based search techniques. This kind of search uses binary checking to find whether the keywords in the query are found in the document or not. Therefore, it misses a lot of similarity details and decreases the efficiency of data retrieval.

The techniques in [25–27] provided the capability of fuzzy keyword search. However, they may fail to satisfy the requirements 1-6. These techniques also share the same weaknesses with the keyword-based search techniques regarding the efficiency of data retrieval.

To overcome the weaknesses of binary keyword-based search, many techniques provided results ranking. However, these techniques compromise some key data to unauthorized parties in the system in order to provide this ranking. Wang et al. [28] used inner product similarity together with an Order-Preserving Symmetric Encryption (OPSE) to provide ranking. However, the technique leaks the order of the results, but not the exact similarity values. Leaked information maybe used in frequency attacks. Moreover, the technique provides only single keyword search queries. Therefore, the technique may fail to satisfy the requirements 3,6, and 8 efficiently. Cao et al. [29] proposed a multi-keyword ranked search technique that also uses the inner product similarity. The technique may fail to satisfy the requirements 1-3 and 6 efficiently. Both [28, 29] need the user to know the list of all valid keywords as well as their exact position to be able to generate a query. This may affect the scalability of the techniques. Wang et al. [30] utilized TF-IDF together with OPSE. The technique still has the weaknesses found in [28] regarding the compromising of results order to unauthorized parties and the single keyword search queries. Also, it may fail to satisfy the requirements 1-6 and 8 efficiently. Sun et al. [31] tried to improve Cao et al. [29] technique by creating a searchable index tree. Each leaf node in the tree contains the documents that have the keywords found in the path from root node to that leaf node. In case the query has only one keyword, the normalized TF distribution of this keyword is exposed directly. Grouping the documents in the leaf nodes of the index tree, together with some frequency analysis may compromise information about the contents of the documents and the queries. Therefore, this technique may fail to satisfy the requirements 1-9 efficiently. Orencik and Savas [32] improved some security issues of Liu et al. [20] technique. They used dummy indexes and keywords to handle index pattern and query pattern problems. These dummy indexes and keywords can be known by the cloud since they are not queried as much as the actual indexes and keywords. Authors suggested changing encryption keys periodically by the data owner or asking trusted proxy server to query these dummy indexes and keywords. The first solution is costly, while the second solution can be detected by the cloud by access frequency analysis. On the other hand, the ranking technique proposed in the technique requires adding extra index in the search cloud. This index compromises some information about the relation between documents. Therefore the technique may fail to satisfy the requirements 1 and 3 efficiently. Also, the used similarity measure misses a lot of similarity information which affects the efficiency of data retrieval. Chen et al. [33] proposed a multi-keyword ranked search scheme that supports latent semantic search. Similar to [28, 29], any user needs to know the list of all valid keywords as well as their exact position to be able to generate a query. Moreover, the technique may fail to satisfy the requirements 1-3 and 6 efficiently.

Gopal and Singh [3] used homomorphic encryption of TF-IDF values to provide a multi-keyword ranked search. The techniques uses the cosine similarity of TF-IDF vectors. The technique may fail to satisfy the requirements 1-6.

1.6 Thesis Organization

This Thesis is composed of 7 chapters including this chapter (Chapter 1) which introduces the problem considered in this thesis and the motivation for this work. It also defined the 9+1 requirements for secure and efficient data retrieval and search in data clouds. Chapter 2 discusses the cloud computing systems and their characteristics, service models and deployment models. Chapter 3 reviews briefly the homomorphic encryption and its characteristics which is used in both index & query generation and anonymous key generation. Chapter 4 presents the normalization technique used to hide the data frequencies, and therefore, to achieve 6+1 of the 9+1 requirements. To achieve the "No Replay Attack" requirement, a lightweight anonymous authentication technique is proposed in Chapter 5. The anonymous authentication technique needs the operations complexity in the server and user sides, during the authentication process, to be relatively low. Therefore, the proposed authentication technique is verified by implementing it on the RFID systems which are good examples of constrained resources systems. Chapter 6 discusses the integration of the normalization technique, the lightweight anonymous authentication technique, and a multi-server setting into one privacy-preserving document retrieval system that satisfies all the 9+1 requirements. Finally, Chapter 7 summarizes the contribution of this thesis and discusses the future directions of research based on it.

2. CLOUD COMPUTING

Although the Cloud computing (or simply cloud) utilization is growing rapidly in all life aspects, neither the concept nor the technology behind it is new. Cloud computing can be considered as an advanced application of the grid computing. It is generally defined as any system that provides hosted services over the internet. These services are provided through standards-based interfaces [34]. The main concept of cloud computing is the sharing of computing resources rather than depending on local resources. These shared resources host and run the services and applications for the users. Therefore, simple users or even enterprise companies are able to benefit from these shared computers as a utility, such as electricity, water, etc. without the need for establishing a computing infrastructure locally. Establishing local computing infrastructure maybe costly or even unavailable for small or emerging businesses. The efficient use of the shared resources within a cloud, which is known as Green Cloud Computing (GCC) or simply Green Cloud (GC), has many advantages, such as:

- Reducing environmental damage by minimizing the power consumption, and accordingly, the carbon emissions. It also minimizes the waste during infrastructure updating processes [35, 36].
- Minimizing upfront infrastructure costs.
- Minimizing operational time and costs.
- Minimizing communications between cloud servers [37].

However, implementing such a cloud requires the utilization of the **Virtualization** technology. Virtualization allows one physical device to be logically separated into one or more logical devices (virtual devices). Each of these virtual devices can be allocated to a different task. Therefore, resources of idle virtual devices can be used to enhance the performance of other working virtual devices.

2.1 Definition of Cloud Computing

There are many detailed definitions for the cloud computing. However, the most convincing ones share the following concepts:

1. The cloud consists of a network of remote servers.
2. These servers are hosted on the Internet.
3. The cloud provides an on-demand services.

For example, Foster [5] defined the cloud computing system as "A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet." Similar concepts can also be found in Buyya [6] definition of the cloud computing system, where he defined it as "a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers." National Institute of Standards and Technology (NIST) [38] defined the cloud as "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 promotes availability and is composed of five essential characteristics, three service models, and four deployment models."

2.2 Cloud Computing Characteristics

NIST presented the cloud computing characteristics in five essential characteristics [38]. These characteristics are:

1. *"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."*

2. *“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).”*

3. *“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.”*

4. *“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.”*

5. *“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.”*

However, these five characteristics can be expanded into many detailed characteristics. For example [38–40]:

1. **Agility:** The cloud provides reliable system deployment in a relatively short time and easy steps. It supports tools and utilities that optimize the use of resources and budget. Moreover, it is easy adaptable in responding to changing conditions.

2. **Cost:** As the cloud supports the Service-Oriented Architecture (SOA) as well as the pay-per-use model, the user does not need to purchase the computing infrastructure to start working. On the other hand, virtualization in the cloud servers improves the utilization and the sharing of the resources which reflects on the cost of the services.

3. **Device and location independence:** Users may use the standard protocols implemented in web browsers. However, some cloud systems may require special softwares or platforms to run their services, these softwares and platforms can be implemented on the current devices starting from mobile phones up to supercomputers. Therefore, users are able to run the services, which are hosted and provided by a third off-site party, regardless of their locations if they have connection to the Internet.

4. **Maintenance:** To maintain or update a service in a cloud, only the copy installed in the cloud will be affected. There is no need to re-install or update individual installations for each user.

5. **Multitenancy:** Resources and costs are shared across different users. These resources are located in centralized locations with lower costs. The centralization of resources improves their utilization and efficient usage.

6. **Reliability:** Distribution of the resources in different locations as well as the backup and recovery systems improves the business continuity of the systems. The device and location independency helps the cloud to efficiently recover the systems after disasters.

7. **Scalability and elasticity:** The cloud systems can provision the resources to the users on demand. Therefore, users can scale up and down based on their usage.

8. **Security and privacy:** One of the main concerns regarding outsourcing data to cloud computing service providers is security and privacy. Although Cloud systems may have the capabilities to apply security and privacy requirements on the data of their users, users still unable to completely rely on the honesty of these cloud systems. Therefore, users have to apply their own security and privacy requirements on their data before outsourcing them to the cloud systems. This property is discussed in details in Chapters 5, 4 and 6.

2.3 Service Models

Agility in cloud computing services plays an important role in the selection of service providers in the small and medium-sized enterprises, as well any the other organizations. The ease of on-demand service deployment decreases the set-up time and cost of the services. There are different cloud service models that offer a number of different options. Three general service models, which are deployed over the internet as a pay-per-use policy or using a subscription fee, are used by the providers to deliver these services [38]. These service models are:

1. Infrastructure as a Service (IaaS): In this service model, service provider provides processing, storage, networks, and other fundamental computing resources to the user. The user in his turn can deploy and run arbitrary software, which can include operating systems and applications through the Internet or dedicated virtual private networks. The service provider is responsible of managing the physical resources under the provisioned virtual machines. The user is unaware of the actual computing resources, location, data partitioning, scaling and backup details of the provisioned resources.

2. Platform as a Service (PaaS): In this service model, the service provider gives development environments and tools to the users, who are application developers, to implement and run their own systems. The services include providing the needed infrastructure as well as the operating systems, database systems, web servers, and programming language execution environments as a toolkit for the users. The service provider is also responsible of the installing, run, management and maintenance of the tools as well as their hardware and software infrastructures in the toolkit. The users customize these tools to meet their requirements with a low cost and in a short time. The users also may not need to buy their own licenses for these tools since they can be part of the provided services, which also decreases the set-up cost. However, in the PaaS model, users do not have the control over the underlying cloud infrastructure.

3. Software as a Service (SaaS): In this service model, the service provider is responsible of providing the software applications as well as managing the infrastructure and platforms that run the applications. The users access and run the software applications in the cloud servers through the Internet without the need to install these applications in their own machines. The software applications are generally pre-built and consumed using the provided functionality without

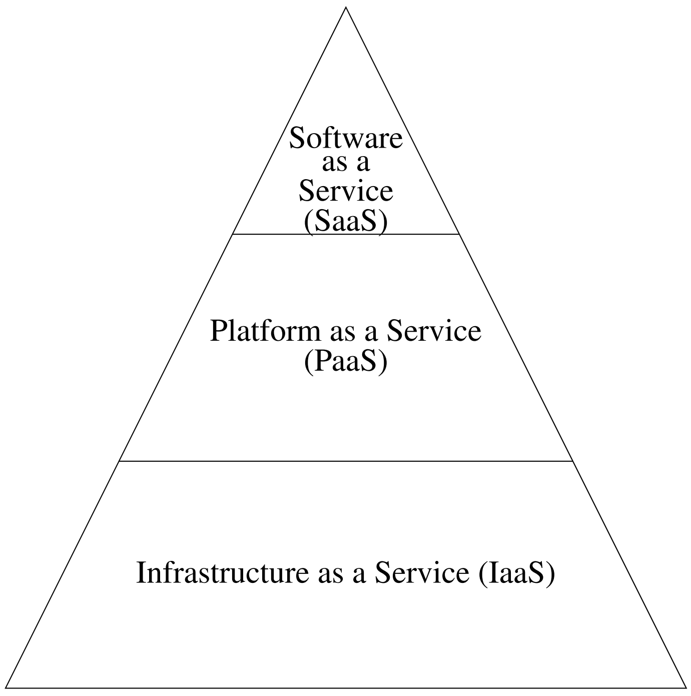

Figure 2.1: Cloud computing service models [1]
.

significant customization. This model simplifies (or even remove) the burden of the installing, run, and maintenance of these software applications as well as their infrastructure and platforms on the users side and puts the most (or all) of it on the cloud provider side. The number of cloud computing providers decreases rapidly as the level of user control over customization, configuration, and management of cloud resources increases.

Figure 2.1 [1] summarizes the relation between the three models. It can be seen that the SaaS service model includes both the PaaS and the IaaS services. PaaS service model also includes the IaaS services. Note that as we move from the bottom of the pyramid to the top, the user control over the resources decreases, which means less set-up cost and time but also less flexibility of use. For more flexibility, users have to go down in the pyramid for the lower service models which means higher set-up cost and time.

Beside the three general service models shown above, there are many derived service models such as Database as a Service (DBaaS), Communications as a Service (CaaS), Business Process as a Service (BPaaS), Data Mining as a Service (DMaaS), Data Warehousing as a Service (DWaaS), Backup as a Service (BaaS), or even Everything as

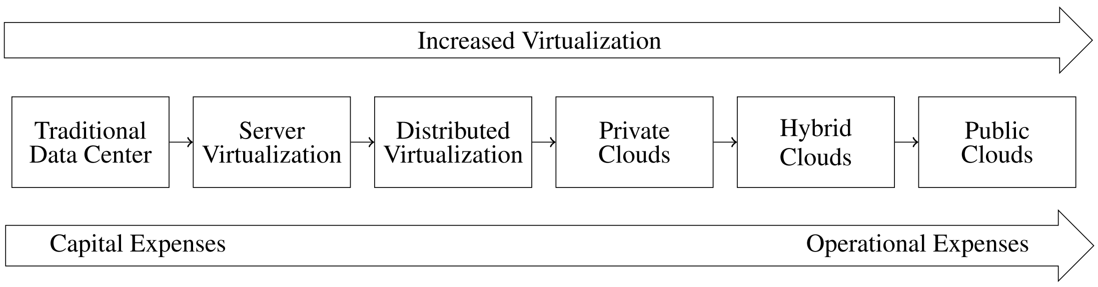

Figure 2.2: Evolving from Virtualization to the Cloud [1].

a Service (XaaS). However, these service models still being able to categorized under the IaaS, PaaS, or SaaS service models. They are defined for commercial purposes or for specialized service provisioning.

2.4 Deployment Models

More than a marketing term, Cloud Computing refers to flexible self-service network-accessible computing resource pools that can be allocated to meet demand. The high need for flexible computing resources produced a high demand for the automation of managing these resources. The current deployment models of cloud systems have evolved from the traditional data centres. However, the different requirements produced different deployment models. The most general deployment models are Private, Public, and Hybrid models as shown in Figure 2.2 [1] which also shows the evolution of the cloud systems.

1. Private Cloud: The private cloud consists of centrally accessed computing resources owned by an organization that gives its members the authority to access these resources from different locations with different authorities. The private cloud is implemented over the organization resources. The aim of implementing a private cloud is to utilize the local resources and provide the same characteristics as public clouds, which means less recurring costs. On the other side, the owner of the private cloud still has the control of the cloud resources as well as the usage of these resources by the users, which means more security. For example, the governmental organizations that have private data of their citizens use a private cloud to process these data to keep its security and privacy, and at the same time benefit from the cloud computing properties.

Although the use of private clouds may provide features such as easy scalability, flexible resource management, maximum hardware utilization, data center automation, chargeback metering, identity-based security, etc., it has an economical downside for the organization since it is still responsible for running and managing the resources instead of passing that responsibility to a cloud provider.

2. Public Cloud: The public cloud consists of a pool of computing resources managed, but not necessarily owned, by the cloud provider. These resources are organized to deliver services to the users over the Internet in different service models, such as SaaS, PaaS, and IaaS. These services are provided to the organizations in pay-per-usage model. The organizations benefit from this model to pay only for the used services and decrease the setup and management cost and time. They use the public clouds when they are less likely to need the level of infrastructure and security offered by private clouds. The public cloud provides on-demand scalability to the users where they can easily scale up/down by purchasing/freeing more resources from/to the cloud provider. Users don't need to purchase and implement hardware. Moreover, they can use Internet to access these services from anywhere which make it location independent. For example, Google, Amazon, Microsoft (Windows Azure), Apple (iCloud) etc. provide cloud services to the public users over the Internet.

3. Hybrid Cloud: The hybrid cloud is a composition of two or more private and public clouds to perform distinct functions within the same organization. They are mostly used by the organizations that need to apply security requirements on part of their data but still need scalable computing resources. The sensitive and confidential data are handled internally in the private clouds while the less sensitive and confidential data are outsourced to the public clouds. Some organizations may have their own private clouds that meet their requirements in the ideal situations, but still need to scale across the public clouds in the workload peaks using what is called "cloud bursting". In this way the organization does not have to purchase or setup any hardware which also decrease the costs and setup time.

However, there is another less known deployment model called "Community Cloud", where the infrastructure is shared between different organizations with the same concerns. The cost of the hardware and management of the cloud can also be shared between these organizations.

2.5 Data Cloud Systems in Practice

Recently, IT systems witnessed a significant growth in cloud adoption. The huge sizes of transfered and processed data have emerged the importance of adopting data cloud systems. These data cloud systems can be implemented using any of the common three cloud service models (Saas, PaaS, IaaS). There are many examples of data cloud systems in our life, starting from simple web-pages hosting systems to enterprise storage systems. According to the State of the Cloud Report published by RightScale Universal Cloud Management in 2016, the optimizations in the access speed, scalability, availability and many other characteristics of the cloud systems in the year 2016 cause a high demand on the data cloud systems [4]. On the other side, according to the "Global Data Leakage Report 2016" [2], most of the data leakages in the year 2016 are coming from the networking and cloud systems as shown in Figure 2.3. Storing data in the cloud systems means trusting unknown parties, however, according to the same report [2], 61.8% of the leakages in the year 2016 are coming from internal offenders. This means that the cloud systems, which we are considered to be trusted, can be the main source of data leakages.

In this thesis, the focus is on the data storage and retrieval cloud systems where users data (in documents format) are outsourced to the clouds to be retrieved completely or partially later using search queries. The final model proposed in this thesis is a hybrid cloud model that combines both private and public cloud systems which is compatible with the current and future tends of cloud adoption [4].

2.6 Summary

In this chapter, different cloud definitions are discussed to show the evolving stages of the cloud systems from traditional data centers to the current form of cloud systems with their characteristics. These characteristics as well as the service models and deployment models are presented in this chapter to show the focus of this thesis and gives more details over the previous chapter. As homomorphic encryption is used in the proposed model, next chapter discusses the homomorphic encryption algorithms

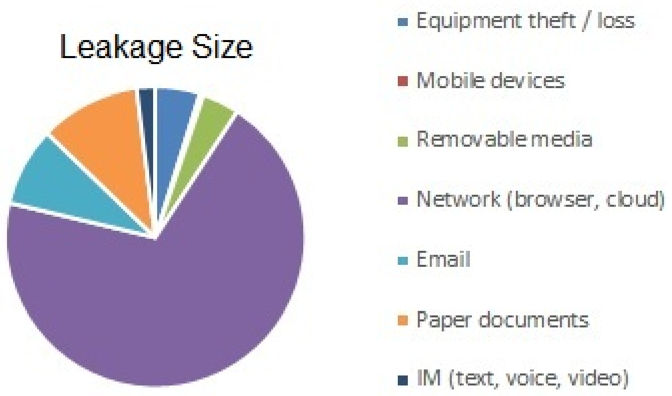

Figure 2.3: Leaks by channel 2016 [2].

as well as their categories and applications in the cloud systems, especially the data cloud systems.

3. HOMOMORPHIC ENCRYPTION IN CLOUD SYSTEMS

Homomorphic encryption is a form of encryption that allows computations to be applied on the encrypted data. The result is the ciphertext of the value resulting from applying the same operations (or other) on the unencrypted values. Figure 3.1 shows the homomorphic property of homomorphic encryption. Suppose that x and y are two plaintexts, and, a and b are the ciphertexts resulting form encrypting x and y, respectively, using the same homomorphic key K_H. z is resulting from applying the $\boxdot$ operation on x and y, while c is resulting from applying $\odot$ operation on a and b. $\odot$ and $\boxdot$ severally can be addition, subtraction, multiplication or division operations. The homomorphic property ensures that decrypting c using the key K_H results to z.

3.1 Categories of Homomorphic Encryption

The homomorphic encryption algorithms are categorized according to their capabilities, and therefore their implementations in the applications. The homomorphic encryption algorithms are categorized as follows:

1. **Partially homomorphic algorithms:** support only multiplication or addition. The most known partially homomorphic encryption algorithms are:

    - **Unpadded RSA**
      The RSA was proposed by R. Rivest et al. [41] as the first practical public key (asymmetric) cryptosystem. The cryptosystem uses two different keys, called public and private keys. Data encryption using one of them requires the other key to decrypt these data. Therefore, it is widely used for encryption as well as authentication and digital signature. RSA uses high exponents of large numbers which make it hard to be implemented in the low end processing units. It was mostly used to share the session symmetric keys between parties at the begining of communications.

$$\begin{tikzcd}
 & x \arrow[d] & \boxdot & & y \arrow[d] & = & & z \\
K_H \arrow[r] & \boxed{E} \arrow[d] & & K_H \arrow[r] & \boxed{E} \arrow[d] & & K_H \arrow[r] & \boxed{D} \arrow[u] \\
 & a & \odot & & b & = & & c \arrow[u]
\end{tikzcd}$$

Figure 3.1: Homomorphic property of the homomorphic encryption.

Given that m is the modulus of the public key and e is the exponent, the encryption of a message x is $E[x]$, where:

$$E[x] = x^e \; mod \; m$$

Then the multiplication homomorphic property of the RSA can be shown as follows:

$$E[x_1] \cdot E[x_2] = (x_1^e \cdot x_2^e) \; mod \; m = (x_1 \cdot x_2)^e \; mod \; m = E[x_1 \cdot x_2]$$

- **ElGamal**

  ElGamal [42] is another public key cryptosystem that satisfies the multiplication homomorphic property. Given that (G, q, g, h) is a public key in ElGamal cryptosystem where q is the order of a cyclic group G with generator g, if a is the secret key then $h = g^a$, and the encryption of a message x is $E[x]$, where:

  $$E[x] = (g^r, x \cdot h^r)$$

  for random $r \in \{0, \dots, q-1\}$. Then, the homomorphic property can be shown as follows:

  $$E[x_1] \cdot E[x_2] = (g^{r_1}, x_1 \cdot h^{r_1}) \cdot (g^{r_2}, x_2 \cdot h^{r_2}) = (g^{r_1+r_2}, (x_1 \cdot x_2) \cdot h^{r_1+r_2}) = E[x_1 \cdot x_2]$$

- **Other partially homomorphic cryptosystems**

  RSA and ElGamal are two examples of partially homomorphic cryptosystems. However, there are many other cryptosystems such as Goldwasser–Micali [43], Benaloh [44], Paillier [45], Okamoto–Uchiyama [46], and Naccache–Stern [47].

2. **Fully homomorphic algorithms:** support both multiplication and addition. The most known fully homomorphic encryption algorithms are:

- **Gentry cryptosystem**

  Gentry fully homomorphic cryptosystem was firstly proposed by Craig Gentry in 2009 [48]. It has been shown that the cryptosystem supports addition and multiplication. A key concept in the development of the Gentry cryptosystem is Gentry's bootstrapping technique. Schemes based on Gentry's blueprint are noise-based, which means that the plaintext is hidden by noise which can be removed by decryption. However, this noise increases with each homomorphic evaluation, and once it exceeds a certain threshold, decryption will fail. To overcome this problem, Gentry introduced the notion of recryption which works by encrypting a ciphertext anew (so that it becomes doubly encrypted) and then removing the inner encryption by homomorphically evaluating the doubly encrypted plaintext and the encrypted decryption key using the decryption circuit. The complexity of decryption increases in the Gentry cryptosystem as the operation on the encrypted data are increased, which make it inefficient in some applications.

- **Algebra Homomorphic Encryption Scheme (AHEE)**

  The AHEE is proposed by G. Xiang et al. [49] as an update to ElGamal [42] algorithm. The security of AHEE algorithm depends on problems from calculating discrete logarithm in finite field and decomposition of large numbers. Examples in [49] shows the additive and multiplicative homomorphism of AHEE. The computational cost of the encryption and decryption operations can be considered constant if the key is known, which make it efficient in many applications.

- **Other fully homomorphic cryptosystems**

  There are many other fully homomorphic cryptosystems which vary in their efficiency, security and complexity, such as Smart and Vercauteren [50], Dijk et al. [51], Brakerski and Vaikuntanathan [52] and Coron et al. [53].

3.2 Applications of Homomorphic Encryption in Cloud Systems

The homomorphic property of the homomorphic encryption make it a very practical and effective tool in the cloud systems. The need for making privacy-preserving calculations in untrusted or unsecured servers on the Internet become possible. There have been many applications of applying the homomorphic encryption to satisfy the privacy and security of the data stored and processed in the cloud systems. For example, privacy-preserving advertising [54, 55], medical applications [56, 57], forensic image recognition [58,59], data mining [60], financial privacy [57], protection of mobile agents [61], privacy preserving search in data [62, 63] and Watermarking & finger printing schemes [64, 65]. In this thesis, homomorphic encryption is used in two different methods to satisfy privacy-preserving data storage and retrieval based on queries. The first method is to encrypt the data that need to be stored in the cloud as shown in details in Chapter 4 using fully homomorphic encryption algorithm. The second method is to generate anonymous authentication keys for each query as shown in details in Chapter 5. Although partially (addition) homomorphic encryption algorithm can be used in the anonymous authentication keys generation, fully homomorphic encryption algorithm is considered more than enough to be used in the keys generation process.

3.3 Summary

In this chapter, homomorphic encryption algorithms, as well as their properties, categories, and applications are introduced. In this thesis, homomorphic encryption algorithm is used in two different, and separated, methods in the final privacy-preserving data retrieval model. More details about using the homomorphic encryption algorithms in the cloud systems is presented in the next two chapters.

4. TF-IDF NORMALIZATION

Chapter 1 introduced 9 security requirements as well as an efficiency requirement (9+1 requirements) for a privacy-preserving data storage and retrieval on data clouds. There have been many proposed techniques to satisfy some of these requirements, however, encrypting the data using deterministic encryption algorithms, which is the case in the most convenient encryption algorithms, cannot hide these frequencies and patterns. For example, Figure 4.1 shows the main stages of generating the index in Gopal et al. [3] technique. Note that frequencies and patterns should be removed from the data in all the system parts, otherwise, attackers can infer many important information about the data using frequency and pattern analysis attacks. In this chapter, a two-rounds technique that uses the model shown in Figure 1.1 is proposed. A normalization stage is added, as shown in Figure 4.2, to remove any frequency in the TF-IDF table before encrypting its values using the homomorphic encryption to generate the index. The technique also uses the cosine similarity between the TF-IDF vectors to find a ranked similarity vector of the documents according to a query. It satisfies the conditions 1,4-5, and 7-9 efficiently. The other requirements are satisfied in Chapter 6 using a multi-server setting as well as an anonymous authentication technique that is presented in Chapter 5.

4.1 Introduction

In data mining, TF table is used to get feature vectors of the documents (especially text documents). In this chapter, we consider a dataset D consists of N documents where

$$D = (d_1, d_2, \cdots, d_N)$$

the set of the ID's of these documents is

$$ID = (id_1, id_2, \cdots, id_N)$$

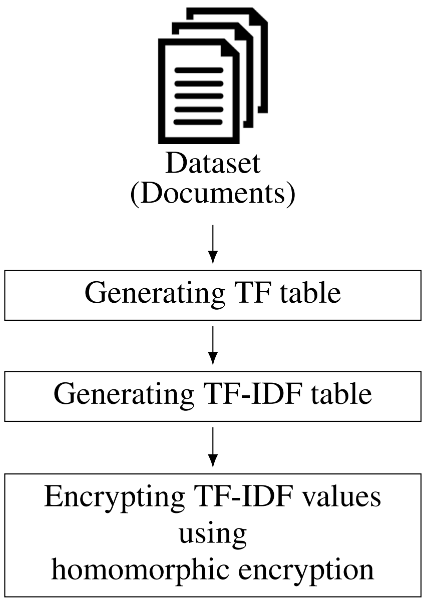

Figure 4.1: Index generation stages of Gopal et al. [3] technique.

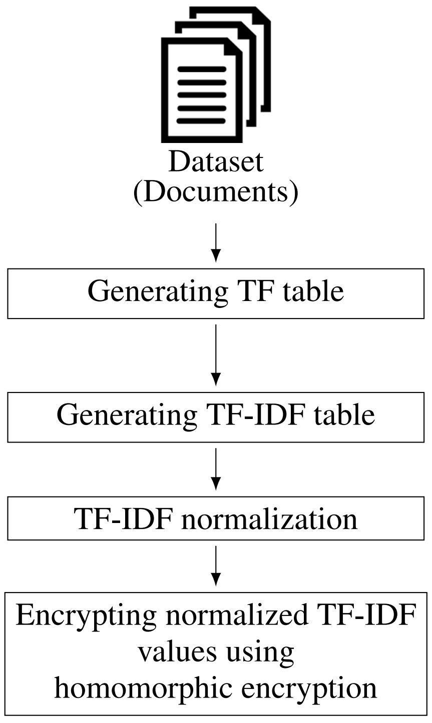

Figure 4.2: Index generation stages of the proposed technique.

The total number of the unique keywords in the entire documents is M, therefore, the set of all unique keywords is W, where

$$W = (w_1, w_2, \cdots, w_M)$$

For a TF table, the rows represent the documents while the columns represent the keywords, so

$$TF = [x_{n,m} \mid 1 \leq n \leq N \text{ and } 1 \leq m \leq M]$$

The value of $x_{n,m}$ represents how many times the m'th keyword is found in the n'th document. If the value of an entity $x_{n,m}$ is zero, this means that the n'th document doesn't include the m'th keyword, also, any equal values in one column means that the corresponding documents has the same keyword with equal number of occurrences. Creating TF table generates a lot of entities with zero value; this is because the table does not include units only for the found unique keywords, but also for the non-found unique keywords in each document. To show that, stop-words are removed from the documents of the "uw-can-data" dataset [66] using three lists of stopwords (for more details about the stopwords list, see APPENDIX A.2). Table 4.1 shows that the ratio of the non-zero entities to the zero entities in the TF table is 1.41%, which means large number of zeros in the table.

For efficient retrieval, TF-IDF [67] is used. However, the entities of the TF-IDF table need to be encrypted to hide their actual values, which is needed for the security and privacy in the retrieval system. In most efficient Privacy-Preserving Search techniques, the entities of the comparable parts of the index need to be encrypted individually, otherwise, the index need to be decrypted in each search process. Moreover, using homomorphic encryption make it possible to process the individually encrypted values without the need for decrypting them. Therefore, if the encryption has to be deterministic, the values in the TF-IDF table will be mapped to new values in the encrypted TF-IDF table. Therefore, a new table with the same statistics but different values is generated. This makes the dataset vulnerable to frequency analysis attacks. Whatever the value that appears with largest number of times in the encrypted TF-IDF table, it will be considered to represent the zeros in the TF table with a high probability.

Table 4.1: Statistics of the keywords in the "uw-can-data" dataset.

The total number of keywords in the documents	The total number of different keywords in the documents	The total number of non-zeros in the TF table	The total number of zeros in the TF table	The ratio of the non-zeros to the zeros in the TF table (non-zeros/ zeros)
91923	21014	91923	6506473	1.413

4.1.1 Frequency Analysis Attacks

The frequency attack problem is handled partially in [29, 31, 33] by using the matrix multiplication for index generation. Each element in a TF-IDF vector is affected by all the other elements in the same vector as well as the corresponding vector in the key matrix. Therefore, elements with zero or high frequently occurred values will have different values after encryption according to the randomness of the key. However, this is not the case with the techniques similar to Gopal et al. [3] technique since the entities of the features table are encrypted separately using the same key. Other techniques, such as [19, 20, 25], compare the encrypted keywords to find the matches. This means that similar keywords before encryption are still similar after encryption. These techniques are vulnerable to frequency analysis attacks. Therefore, a proposed technique has to be developed to prevent this frequency analysis attacks keeping in mind not to affect the retrieval efficiency.

4.1.2 Relations Between Documents

Including only the keywords with values greater than zero [3, 25] gives an idea about which keywords are found in specific documents and which keywords are not. This can be considered as threat where the documents can be related to each other. For example, assume that a dataset consists of 10 documents ($N = 10$), and the total number of unique keywords is 4 ($M = 4$), and the details of the documents are as follows:

1. $Keyword_1$ found in documents 1, 3, 6, 8 and 9 for 86, 86, 90, 58 and 46 times respectively.

2. $Keyword_2$ found in documents 1, 3, 4, 7 and 9 for 94, 54, 58, 64 and 80 times respectively.

3. $Keyword_3$ found in documents 5, 8 and 10 for 64, 46 and 64 times respectively.

Table 4.2: Example of a TF table.

Document ID	w_1	w_2	w_3	w_4
id_1	86	94	0	0
id_2	0	0	0	56
id_3	86	54	0	0
id_4	0	58	0	0
id_5	0	0	64	0
id_6	90	0	0	58
id_7	0	64	0	90
id_8	58	0	46	0
id_9	46	80	0	64
id_{10}	0	0	64	0

4. $Keyword_4$ found in documents 2, 6, 7 and 9 for 56, 58, 90 and 64 times respectively.

Then, the TF table of this example is shown in Table 4.2. Note that these values are not actual values, they are selected to simplify the example.

It can be seen that even if the keywords, document ID's, and frequencies are encrypted, one can end up with many deductions, such as:

- Zero is repeated 23 times, which can be detected even after encryption.
- Documents 1 and 3 are related: they include two common keywords with exact frequencies and another two common keywords with different frequencies. Moreover, based on the first induction, both of them do not have the keywords $Keyword_3$ and $Keyword_4$.
- Documents 4 and 8 are unrelated (have no common keywords).
- Based on the first induction, document 10 does not contain $Keyword_2$ and $Keyword_4$.
- Documents 5 and 10 are exactly the same.

Even though such a simple example, it is seen that including zeros is necessary to prevent such deductions. Including only non-zero values allows anyone to know which keywords are found in exact documents and which are not. On the other hand, including zeros need to hide their frequency. Fig. 4.3 shows the histogram for

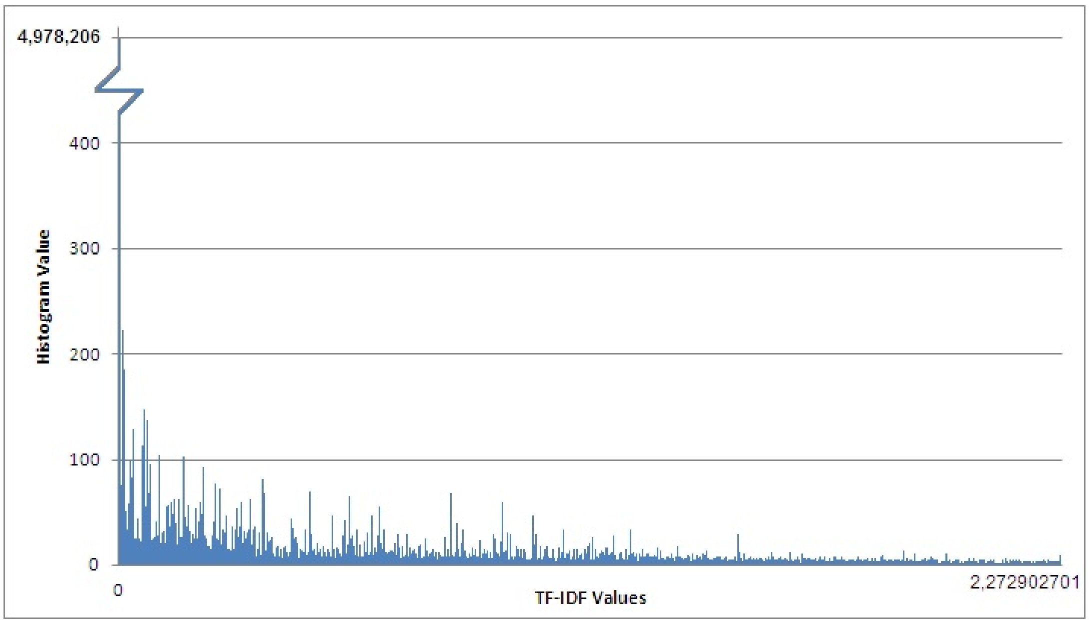

Figure 4.3: Histogram of the TF-IDF table of uw-can-data dataset.

the TF-IDF table of uw-can-data dataset [66], some TF-IDF values have frequencies more than others, which can be considered as indicators to them in the frequency analysis attacks even after encryption. So, the goal is to normalize these values before encrypting them.

4.1.3 Retrieval Efficiency

Data retrieval quality depends on many different factors; one of these factors is the way of choosing feature vectors for the documents. According to [68], binary term vectors give lower efficiency than weighted term vectors. Note that [29, 31] use the binary vectors while [3] uses the weighted vectors in their techniques.

4.2 Normalization Technique

As mentioned in Section 4.1, the techniques in [29, 31] can hide zeros and high frequently occurred values. However, because of using the binary vectors as well as dummy keywords, the retrieval efficiency becomes lower than weighted term vector algorithms. Therefore, Gopal technique [3] has to be improved to be able to handle the three issues mentioned in Section 4.1. With reference to Fig. 1.1, the suggested model is working as follows:

1. Data owner creates the TF table; the keywords in this table are hashed.

2. The names of the documents and the documents themselves are encrypted separately using symmetric or asymmetric key (K_s).

3. TF-IDF is created from the TF table.

4. TF-IDF table is normalized using the technique which will be explained later in this section.

5. The entities of the normalized TF-IDF table are encrypted individually using homomorphic encryption with the same key (K_h). The encrypted TF-IDF table is the index that will be outsourced to the cloud (encrypted data & querying services).

6. K_h and K_s are sent from the data owner to the client (the trapdoors).

7. The client applies the same operations on the TF vector generated from the query document using K_h before sending it to the cloud.

8. The cloud calculates the similarity between the query and the documents using operations on the encrypted data without revealing them.

9. The similarity vector is sent to the client to decrypt it using K_h and find the best matches to be retrieved.

10. The client send the ID's of the selected documents to the cloud. The cloud replies by the encrypted documents which are decrypted by the client using the secret key K_s.

Assume that the number of the unique values in the TF-IDF table is Q, then

$$U = (u_1, u_2, \cdots, u_Q)$$

where U is the set of unique values in the TF-IDF table, in this case, the histogram of the TF-IDF table is

$$H = (h_1, h_2, \cdots, h_Q)$$

where h_q represents the number of times that u_q appears in the TF-IDF table for $1 \leq q \leq Q$. To normalize these values, we will start by the values in Table 4.2 as a TF-IDF

Table 4.3: Histogram of the example values in Table 4.2.

TF-IDF values (U)	Histogram (H)
86	2
94	1
0	23
56	1
54	1
58	3
64	4
90	2
46	2
80	1

Table 4.4: Ordered Histogram.

U'	H'
0	23
46	2
54	1
56	1
58	3
64	4
80	1
86	2
90	2
94	1

values of the example given in Subsection 4.1.1. The actual TF-IDF values are not calculated to simplify the example. Therefore, both U and H are shown in Table 4.3.

The normalization process goes as follows:

1. Order U increasingly in U'. Therefore,

$$U' = (u'_1, u'_2, \cdots, u'_Q)$$

Values of H are ordered corresponding to U' in H' as shown in Table 4.4. Therefore,

$$H' = (h'_1, h'_2, \cdots, h'_Q)$$

Table 4.5: Increasing value (e_q) with k=2.

U'	H'	e_q
0	23	(46-0)/(23× 2)=1
46	2	(54-46)/(2× 2)=2
54	1	(56-54)/(1× 2)=1
56	1	(58-56)/(1× 2)=1
58	3	(64-58)/(3× 2)=1
64	4	(80-64)/(4× 2)=2
80	1	(86-80)/(1× 2)=3
86	2	(90-86)/(2× 2)=1
90	2	(94-90)/(2× 2)=1
94	1	Minimum(e_q)=1

2. For each $u'_q \in U'$, calculate e_q, where

$$e_q = \frac{u'_{q+1} - u'_q}{h'_q \times k} \tag{4.1}$$

   k is a scaling factor that determines the size of difference between the original value and the normalized values (will be discussed later in Section 4.3). For e_Q, minimum e_q value is taken to be its value as shown in Table 4.5 where k is used as 2.

3. For each $u'_q \in U'$:

   (a) Define $S_q = h'_q - 1$

   (b) Generate a new set $u"_q = (u"_{q0}, u"_{q1}, u"_{q2}, \cdots, u"_{qS_q})$ as follows:

   - For $s = 0$ to S_q
     - $u"_{qs} = u'_q + (s \times e_q)$

   Table 4.6 shows $u"_q$ for $1 \leq q \leq Q$.

4. Replace all the entities in the TF-IDF table that have the u'_q value by the elements of the $u"_q$ randomly without repetition as shown in Table 4.7 and Table 4.8.

In this case all the TF-IDF values will be different. Also, even in small difference between the values will be hidden by the encryption process. For Example, documents 5 and 10 which are exactly the same in the TF-IDF table are different after the normalization.

Table 4.6: Normalized values (u''_q for $1 \leq q \leq Q$).

U'	u''_q	Frequency
0	0+(0×1)=0	1
	0+(1×1)=1	1
	0+(2×1)=2	1
	0+(3×1)=3	1
	0+(4×1)=4	1
	0+(5×1)=5	1
	0+(6×1)=6	1
	0+(7×1)=7	1
	0+(8×1)=8	1
	0+(9×1)=9	1
	0+(10×1)=10	1
	0+(11×1)=11	1
	0+(12×1)=12	1
	0+(13×1)=13	1
	0+(14×1)=14	1
	0+(15×1)=15	1
	0+(16×1)=16	1
	0+(17×1)=17	1
	0+(18×1)=18	1
	0+(19×1)=19	1
	0+(20×1)=20	1
	0+(21×1)=21	1
	0+(22×1)=22	1
46	46+(0×2)=46	1
	46+(1×2)=48	1
54	54+(0×1)=54	1
56	56+(0×1)=56	1
58	58+(0×1)=58	1
	58+(1×1)=59	1
	58+(2×1)=60	1
64	64+(0×2)=64	1
	64+(1×2)=66	1
	64+(2×2)=68	1
	64+(3×2)=70	1
80	80+(0×3)=80	1
86	86+(0×1)=86	1
	86+(1×1)=87	1
90	90+(0×1)=90	1
	90+(1×1)=91	1
94	94+(0×1)=94	1

Table 4.7: Substitution of TF-IDF values with the normalized values table.

Document ID	w_1	w_2	w_3	w_4
id_1	86→87	94→94	0 →17	0 →8
id_2	0 →10	0 →4	0 →15	56→56
id_3	86→86	54→54	0 →19	0 →0
id_4	0 →7	58→59	0 →18	0 →22
id_5	0 →3	0 →14	64→68	0 →13
id_6	90→91	0 →1	0 →20	58→60
id_7	0 →12	64→66	0 →6	90→90
id_8	58→58	0 →21	46→48	0 →9
id_9	46→46	80→80	0 →11	64→64
id_{10}	0 →16	0 →5	64→70	0 →2

Table 4.8: Normalized TF-IDF table.

Document ID	w_1	w_2	w_3	w_4
id_1	87	94	17	8
id_2	10	4	15	56
id_3	86	54	19	0
id_4	7	59	18	22
id_5	3	14	68	13
id_6	91	1	20	60
id_7	12	66	6	90
id_8	58	21	48	9
id_9	46	80	11	64
id_{10}	16	5	70	2

The final step in creating the index for the cloud is to encrypt the entities of the normalized TF-IDF table using Homomorphic encryption as in [3]; this hides the actual values, but operations on these values are still applicable.

To discuss the effect of applying this technique on the retrieval efficiency, the retrieval efficiency of the normalized TF-IDF table is compared with the original TF-IDF table. Average Precision Value (APV) is used to calculate the retrieval efficiency of the techniques as follows:

1. For each document $d_n \in D$, calculate the precision value pr_n as follows:

$$pr_n = \frac{\text{Retrieved Documents} \cap \text{Related Documents}}{\text{Retrieved Documents}} \tag{4.2}$$

   Where the number of retrieved documents is equal to the size of the cluster containing the document d_n in the original dataset.

2. Calculate the APV as follows:

$$APV = \frac{\sum_n pr_n}{\text{Number of Documents}} \tag{4.3}$$

4.3 Simulations and Results

In order to test the suggested technique, three different datasets are used: uw-can-data [66], mini-classicdocs [69] and mini-20newsgroups [70]. Table 4.9 shows some details of these three datasets (for more details, see APPENDIX A.1). The datasets are prepared before being used as shown in Figure 4.4. The steps are as follows:

1. HyperText Markup Language (html) documents are parsed using htmlparser-1.6 to extract the data from them.

2. Stopwords are removed using three different lists of stopwords: Long list, Short list and Google list.

3. Porter stemmer is used to stem the keywords.

4. The datasets are classified using k-means classification with cosine similarity distance.

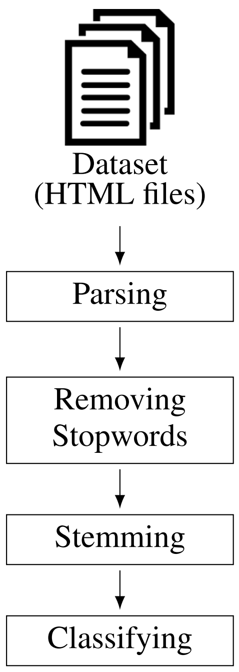

Figure 4.4: Dataset preparation stages.

Using the normalization technique will make all the histogram values of the normalized TF-IDF table equal to one. The number of different numbers of the TF-IDF table will be equal to: number of unique keywords $\times$ number of documents.

To know the effect of normalization on the retrieval efficiency, different values of the factor k are used. As mentioned before, the factor k determines the size of the difference between the original value (u'_q) and the expanded set of values ($u"_q$) in the normalization process. The technique was applied on the uw-can-data, and mini-classicdocs datasets separately as follows:

- For z=1 to 10000 increasing by 5:
  - Calculate APV_z= the APV where $k = z$.
  - Calculate AV= Average of APV_z over all z values.

Table 4.10 shows the APV's using the original TF-IDF tables (without normalization) for the three datasets in the first column, which is the case in [3] technique. The second column represents the APV's for the binary term tables also for the three datasets, which is the case in [29, 31]. Finally, the third column represents the average APV's (AV's) for the normalized TF-IDF tables with k = 1 to 10000 increased by 5 for the three datasets, which is the case in the suggested technique in this chapter.

Table 4.9: Details of the three datasets (uw-can-data, mini-20newsgroups and mini-classicdocs) used in the evaluation of the suggested technique.

Dataset	Number of Documents	Number of Classes	Description
uw-can-data	314	10	Web pages from various web sites at the University of Waterloo, and some Canadian websites
mini-20newsgroups	400	20	A collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups, the number of documents is minimized to 400 documents with the same number of classes
mini-classicdocs	800	10	Four different document collections: CACM, CISI, CRAN, and MED. the number of documents is minimized from 7095 documents to 800 clustered in 10 classes

Table 4.10: Comparison between normalized and unnormalized TF-IDF tables based on AVP and AV values.

Dataset	APV without normalization	APV With binary TF table	AV of the normalized TF-IDF table
uw-can-data	0.175935689	0.150279841	0.183681939
mini-20newsgroups	0.110836309	0.101195467	0.114799958
mini-classicdocs	0.110236005	0.107274797	0.111710287

4.4 Analysis of the Normalization Technique

The effectiveness of normalization technique is discussed in this section with regard to the results given in Section 4.3.

4.4.1 The effects of the normalization on retrieval efficiency

Results show that the retrieval efficiency does not decrease after normalization of the TF-IDF tables. As shown in Table 4.10, the average of the *APV*'s (*AV*) after normalization are higher than the precision values before normalization, which in turn higher than the *APV*'s of the binary features, for the three datasets. Although the similarity values can be affected by the normalization, precision values are shown to be unaffected in the same degree, which satisfies the retrieval efficiency requirement.

4.4.2 The effects of this technique on the time and memory costs

Time cost: The normalization technique will be done once in the setup of the system (which is offline process), all the steps can be done using parallel processors, ordering the histogram increasingly according to the TF-IDF values is $O(n \log n)$ for n unique keywords. Memory cost: Storing the different values after normalization needs: (number of unique keywords $\times$ number of documents $\times$ size of each unit).

4.4.3 The effects of the used normalization on privacy

Using normalization gives unique values for all the normalized TF-IDF elements. This prevents any kind of frequency attacks (discussed in 4.1.1 and 4.1.2 subsections). Although the difference between the values may be small before encryption, the Homomorphic encryption maps them to different values.

Table 4.11 summarizes the comparison between [3], [29, 31] and the proposed technique with regard to the first three problems have been introduced in Section 4.1 which are frequency analysis attacks, relations between documents, and retrieval efficiency.

4.5 Summary

Table 4.11: Comparison between different techniques.

Problem	Gopal [3]	Sun & Cao [29, 31]	Proposed technique
1- Frequency analysis attacks	Vulnerable	Invulnerable	Invulnerable
2- Relations between documents	Can be deduced	Hard to deduce	Hard to deduce
3- Retrieval efficiency	Higher than Sun & Cao	Lower than Gopal	Higher than both

Data frequencies and patterns in the index or the queries can be used to infer important private and secure information about users, index, or queries. Using deterministic encryption algorithms, even the homomorphic ones, to encrypt the index or the query values does not solve the problem. It has been shown that the index and the queries in the data retrieval systems are vulnerable to frequency attacks. Therefore, a new normalization technique is proposed in this chapter to hide any frequencies in the index or the queries. The first step is to normalize the values to make them all unique, then to encrypt these values using homomorphic encryption to allow calculations over them. Although the normalization technique is shown efficient in satisfying seven of the nine security requirements as well as keeping the retrieval and ranking efficiency high, it is still unable to satisfy two of the security requirements. One of these two requirements is the prevention of Replay Attacks, which requires an anonymous authentication for each query. In the next chapter, a lightweight anonymous authentication technique is proposed and verified on the RFID systems as an example of limited resources units. This authentication technique is integrated with the normalization technique presented in this chapter to find the complete model of privacy-preserving data storage and retrieval on data clouds shown in Chapter 6.

5. ANONYMOUS AUTHENTICATION

The wide use of cloud systems, which have different ownerships, arose new types of authentication processes. Although authentication is considered as a necessity in most of the applications to grant different levels of privileges and authorities, still there are some requirements to be met, such as the privacy of users. Revealing any information about the user, such as identity, location, ect. is considered as a break of the privacy of that user. For example, revealing the identity of a user in a mobile system allows the attacker to track that user by tracking the authentication points. Therefore, anonymous authentication techniques became a must in IT systems. Note that in any authentication system, and for generality, the user should be considered having constrained resources. On the other side the authentication server is considered to have enough resources to apply complex operations. The networks between the user and the authentication server are considered insecure. Anonymous authentication in RFID systems is considered as a good example to verify the proposed anonymous authentication technique. Therefore, a new anonymous authentication technique (Called HEADA) is explained in this chapter. Although the HEADA is designed and verified on the RFID systems, it is efficiently used, with some modifications, in the final model of privacy-preserving data retrieval as shown in Chapter 6.

5.1 Introduction

By the rapid integration of the IT in all life aspects, RFID technology started to play an important rule in the automatic identification of objects such as people, cargos, vehicles, goods, and many different assets [71–73]. There are many advantages of using RFID technology compared to other technologies that are being used for similar purposes such as magnetic strips and bar code [74]. The main advantage is the ability to store the identification data in the objects, then reading these data automatically without the need for direct contact between the object and the reader. RFID technology utilises radio signals generated by a network of readers distributed over a distance of

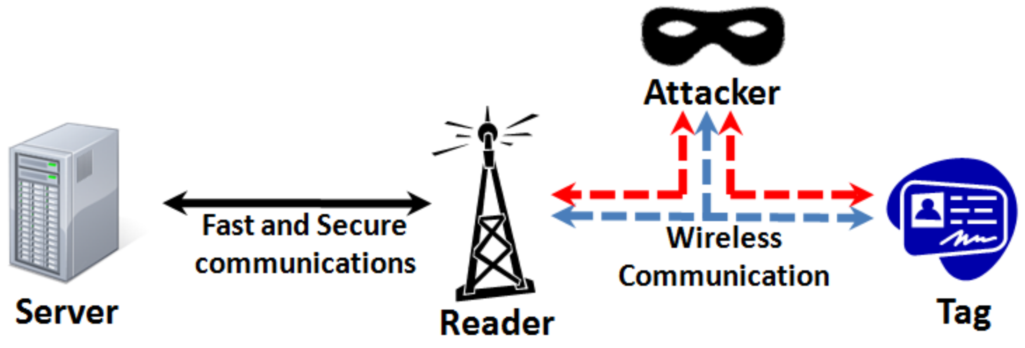

Figure 5.1: Considered model of RFID system with the main entities and the communications between them.

several meters to detect, sort, identify and track the objects. An RFID system consists of three essential entities: servers, readers and tags. The servers in the system process data received from readers using stored databases for authentication or any further processes. The servers also can be responsible of any key generations or data storage in the deployment course of readers or tags. Readers are devices which are connected to the servers with fast and secure networks. They are used to transfer data between servers and tags. Tags are devices that store a unique ID as well as different data that may vary for each application [75]. The RFID system that is considered in this chapter is depicted in Figure 5.1. For simplicity, the system consists of a server, a reader and a tag. The reader sends a low-level radio frequency magnetic field that energizes the tag in its range. The tag responds by its identification data via radio waves. The reader sends data to a server to check them through fast and secure communication channel. Server grants different access levels to a tag based on the received identities and the data stored in the database [76]. Figure 5.1 also shows the behaviour of an expected attacker. An attacker can: listen to the communications between the reader and the tag, communicate with the reader as a tag, and communicate with the tag as a reader regardless of being the tag in the range of the reader or not.

The variation of RFID applications led to the foundation of different types of tags with different properties and capabilities. Tags can be classified into three categories: passive, semi-passive and active tags. Table 5.1 compares these three categories based on the power source, maximum distance range, type of communications and cost [77]. The advantages of RFID technology over other technologies, such as non-contact and non-line-of-sight automatic way of communication, made it a good replacement to

Table 5.1: Comparison between passive, active and semi-passive tags.

Property	Passive	Semi-Passive	Active
1- Power source	RF energy	Battery	Battery
2- Maximum distance	10 M	100 M	1000 M
3- Communications	Only response	Only response	Initiation or response
4- Tag's signal	Very low	Moderate	High
5- Cost	The lowest	Higher than passive	The highest

them in their applications and a very useful tool in many other new applications. The growing tendency to benefit from these advantages arose new requirements, such as security and privacy, which should be suitable to the limited resources in some parts of the system.

5.2 Security and Privacy in RFID Systems

RFID tags are considered as a good example of the low storage and computing capabilities units in Internet of Things (IoT) systems. Therefore, they can be considered as the weakest point in such IoT security systems. To discuss any RFID security system, a complete set of security requirements should be defined and verified on that system.

5.2.1 Security Requirements

The main advantage of the RFID technology is the non-contact and non-line-of-sight automatic way of communication which is not the case with the other technologies. In spite of being the communication way in RFID systems an advantage in the usability side, it may exacerbate privacy threats or pose new security risks. The security achieved physically in the contact or line-of-sight technologies became the responsibility of the communication protocol [78]. Therefore, based on the literature review, following security requirements are considered as the common and minimum requirements to satisfy a high level of privacy and security in any RFID authentication protocol:

1. **Untraceability**: Only the authorized parties are allowed to identify or trace a tag at any time.

2. **Resistance to replay attack**: The messages of any previous valid or invalid authentication processes cannot be used to gain a valid authentication for unauthorized party.
3. **Resistance to Denial-of-Service (DoS) attack**: Modification, forging, blocking, or delaying of message from/to any party of the system are unable to make the tag or the server unreachable later under proper conditions.
4. **Mutual authentication**: Both the server and the tag verify the proper identity of each other in the authentication process.
5. **Tag forgery resistance**: Generating, predicting, or reusing a valid authentication key is infeasible without a valid authentication data.
6. **Server forgery resistance**: Generating, predicting, or reusing a valid server verification key is infeasible without a valid authentication data.
7. **Data recovery**: Any flaw in the synchronization or the status of any of the parties at a moment is recoverable.

5.2.2 Low Cost Operations

RFID tags are renowned for being cheap, small, and made of sustainable materials with limited storage and processing capabilities. Chien [82] suggested categorizing the RFID tags into four categories according to the operations in the tag during the authentication process. The four categories are as follows:

- Ultralight tags: Passive tags which are implemented using only simple operation such as OR, AND, or/and XOR.
- Lightweight tags: Beside the operations in the ultralight tags, lightweight tags use Cyclic Redundancy Check (CRC), or/and PseudoRandom Number Generator (PRNG).
- Simple tags: Beside the operations in the lightweight tags, simple tags use cryptographic one-way hashing functions (HASH).

- High cost: Beside the operations in the simple tags, High cost tags use conventional cryptographic algorithms, such as symmetric or/and asymmetric encryption algorithms.

Therefore, efficiency requirements in this chapter are defined as follows:

1. **Low cost operations in the tag**: Tags belong to one of the first two categories, i.e. ultralight or lightweight, which means including only OR, AND, XOR, CRC, or/and PRNG functions.
2. **Low cost operations in the server**: No brute-force operations on the tags data (or part of it) during the authentication protocol. Otherwise, each authentication process takes unexpected amount of time and iterations.

Along with the seven security requirements, these two efficiency requirements for low cost operations are noted as the 7+2 requirements in the rest of this chapter. Any authentication protocol should satisfy at least these 7+2 requirements to be eligible for such a system.

5.2.3 Related Works

There have been a number of techniques proposed to address some of these security problems. The techniques available in the literature can be grouped under three headings:

(a) Hash-based.

(b) Matrix multiplication-based.

(c) Public key-based.

However, to the best of our knowledge, none of them provide solutions to the complete set of 7+2 requirements. Weis et al. [83] proposed two different hash-based techniques for authentication in RFID systems. Despite of using pseudo-random numbers in one of the techniques to prevent traceability, the server cannot verify the tags. Both techniques detect DoS attacks, but suffer from eavesdropping, spoofing and replay attacks. Yeo and Kim [84] proposed another hash chain-based technique which updates the key of the tag after each query to prevent traceability. However, it does not handle the eavesdropping, spoofing and replay attacks since the attacker can authenticate itself in the server by extracting information from previous tag responses. Dixit et al. [85] modified Yeo and Kim [84] technique to prevent eavesdropping, spoofing and replay attacks by including a random number that is received from the reader in the hashed values. However, the server still needs to hash a group of the keys according to the size of tags grouping and search in the hashed values to identify the communicating tag. Bringer et al. [86] proposed an extension and improvement of Juels and Weis [87] technique. Both techniques use binary inner product to simplify the processes in the tag. However, they do not consider the server authentication and they suffer from the tracking attacks as well.

Karthikeyan and Nesterenko [88] use matrix multiplication to generate the authentication messages between the server and the tag. The key of the tag is updated after each completed authentication process. The attacker can spoof the tag by sending a request to the tag when it is not in the range of any reader. The tag will reply by an authentication message as if the request came from a legitimate reader. If no response is received by the tag, it will return back to listening stage without updating the key. The attacker may either use the authentication message which is not used yet in the server to authenticate himself in the server, or it may send another request to trace the tag since the tag will reply by the same authentication message. If the value sent by the reader to update the key is changed during the transmission, then, the tag become unreachable.

Yi et al. [81] proposed an improved technique which is based on EPC Class-1 Generation-2 standard for tags and uses CRC and PRNG circuits. The technique requires multiple shift and XOR operations to compute CRC which is a costly operation in terms of computation. Moreover, the technique still needs to go through the records of all the tags in the server to identify the communicated tag in each query which can be exhausting process in case of large number of tags.

Public key encryption algorithms are used for authentication in many techniques [89–91]. In spite of the security capabilities of the encryption algorithms, they are inefficient on tags having limited computational resources.

5.3 The HEADA Approach

The aim of the authentication process in the HEADA is to identify and authenticate the tag in the server through insecure channels. Although, the authentication process can be considered as an initial stage for a server to proceed with further communications to trace, read from, or write to the tag, our focus within the scope of this chapter is only the authentication process of the tag in the server. The authentication protocol works as follows: As soon as a tag arrives at a proximity of a reader, it receives a query from that reader and responds with an authentication request. The reader forwards the request to the server to check it through fast and secure communication channel. Server authenticates the tag and sends a verification code to the tag through the reader. The tag verifies the server and sends a verification code to the server. Once the verification code from the tag to the server is received, the server verifies the tag and the protocol terminates. However, it is required to show that no attacker can identify or trace the tag at any time. An attacker is considered to be able to mimic a tag or a reader. It can also simply eavesdrop the communications between a legitimate tag and reader. The attacker could be either a single entity consisting of a fake tag, a fake reader and an eavesdropping module, or multiple entities encapsulating one or more modules.

The HEADA is based on finding a set of sub-keys for each tag; these sub-keys are partitioned into groups and stored in the tag in the deployment course. The tag selects a number of combinations (a combination, from now on, refers to a set of sub-keys composed of a single sub-key from each group) from these groups and adds them to generate a different authentication key in each authentication process. The server uses a reversibility property found in the sub-keys to reconstruct the used combinations from the authentication key to identify and authenticate the tag. Based on these assumptions, to achieve the 7 security requirements defined in Sub-Section 5.2.1, following conditions need to be achieved:

1. **Uniqueness of sub-keys:** For all the tags, sub-keys are unique.

2. **Uniqueness of authentication keys:** The summation of any combination of sub-keys should be unique.

3. **Irreversibility:** Reversibility property should be restricted only to the server using constant time complexity algorithm and key data.

4. **Randomness of the authentication key**: Authentication keys need to be selected randomly without duplication based on a key, known only to the tag and the server, and a simple algorithm.

Therefore, the mentioned four features will be provided through new proposed methodology and an algorithm which are:

(a) A new methodology for finding a huge series of unique numbers using a relatively very small set of integers and a constant time algorithm (by utilizing integer addition). Moreover, only the parties that have a specific key data have to be able to identify the generator of each number using a constant time algorithm and simple binary search. Note that such numbers can be efficiently utilized in many applications such as privacy-preserving identification, temporary password generation, etc.

(b) A constant time algorithm for random selection without duplication from a set of elements.

Then, given methodology and algorithm are applied to derive an RFID anonymous authentication technique that satisfies all the 7+2 requirements.

5.3.1 Key Components of the HEADA

Suppose that there are W different tags, then, the set of all tags will be T, where

$$T = \{t_n : n \in \mathbb{Z},\, 1 \leq n \leq W\}$$

For each tag t_n, there exist a unique ID (id_n) and an authentication key (k_n), (for simplicity, it will be named as key for the rest of this chapter). The sets of all IDs and keys are ID and K (which is shown in column 1 in Figure 5.2), respectively, where

$$ID = \{id_n : n \in \mathbb{Z},\, 1 \leq n \leq W\}$$

and

$$K = \{k_n : n \in \mathbb{Z},\, 1 \leq n \leq W\}$$

Each key k_n consists of M numbers (or, sub-keys). These M sub-keys are partitioned into I groups as shown in column 2 in Figure 5.2. The number of groups should be ≥ 2. The set of groups is G, where

$$G = \{g_i : i \in \mathbb{Z},\ 1 \leq i \leq I\}$$

and

$$M = \sum_{i=1}^{I} g_i$$

g_i is the number of sub-keys in the i'th group (or, the size of the i'th group). G is supposed to be the same for all the keys. Therefore, sub-keys can be referred by the key number (n), the group number (i), and the order of the sub-key in that group as shown in column 3 in Figure 5.2. Thus, k_n can be considered as

$$k_n = \{x_{n,i,j} : i, j \in \mathbb{Z},\ (1 \leq i \leq I) \wedge (1 \leq j \leq g_i)\}$$

where $x_{n,i,j}$ is the j'th sub-key of the i'th group of the key k_n.

Suppose that $d_{n,i}$ is the set of sub-keys of the i'th group of the key k_n, then d_n is the set of all combinations resulting from selecting one key from each group. This can be shown in the relation between columns 3 and 4 in Figure 5.2. D is the set of all combinations of all the tags sub-keys as shown in column 4 in Figure 5.2, where

$$d_n = d_{n,1} \bigtimes d_{n,2} \bigtimes \cdots \bigtimes d_{n,I}$$

where "$\bigtimes$" indicates the Cartesian product, and

$$D = \bigcup_{n=1}^{W} d_n$$

Suppose that s_n is the set of the sums of the sub-keys for each combination of d_n, then the set of the sums of the sub-keys of each combination for all the tags sub-keys is S, which is shown in column 5 in Figure 5.2, where

$$S = \bigcup_{n=1}^{W} s_n$$

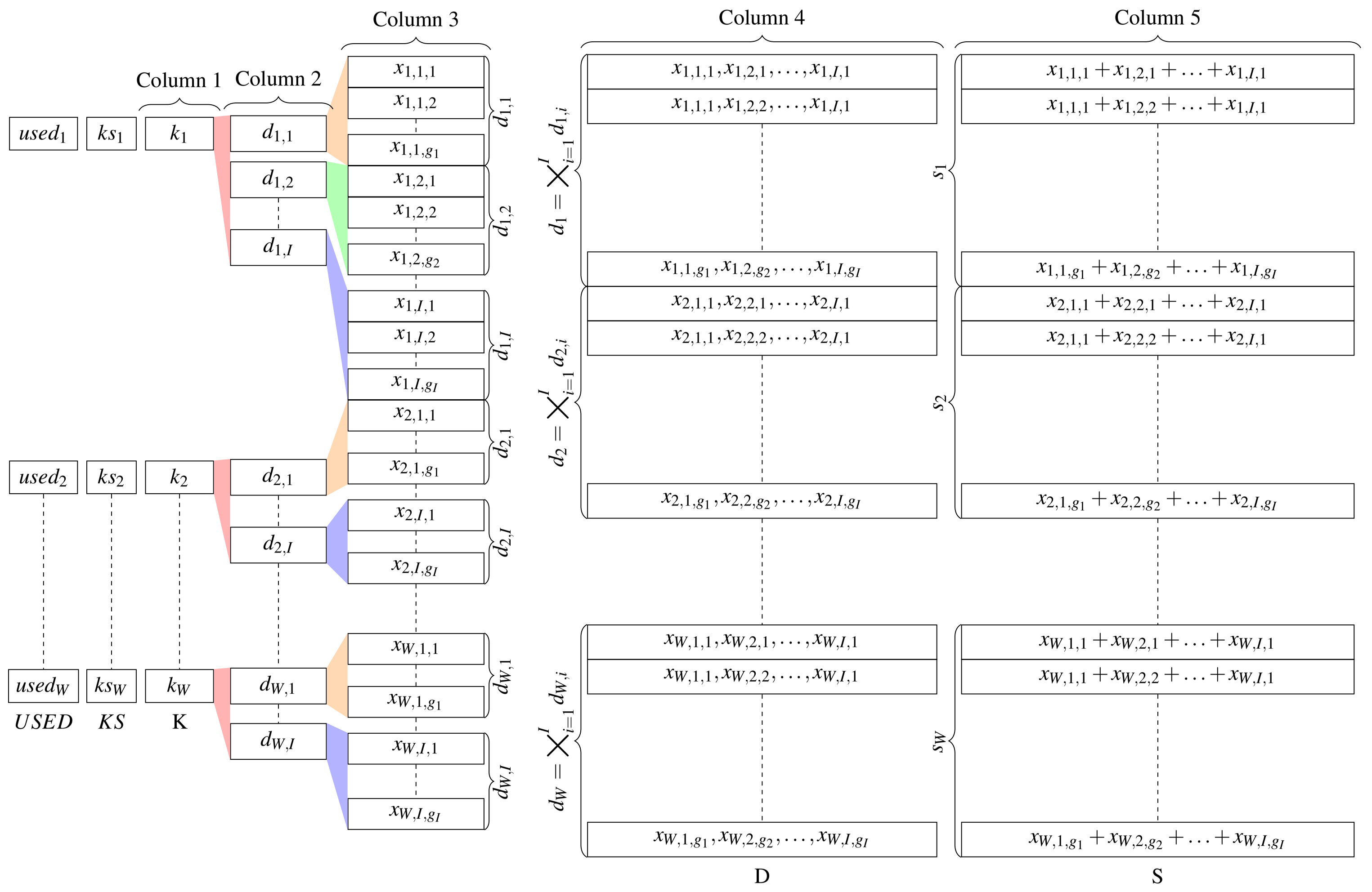
Column 1
Column 2
Column 3
Column 4
Column 5
$used_1$
ks_1
k_1
$d_{1,1}$
$d_{1,2}$
$d_{1,I}$
$used_2$
ks_2
k_2
$d_{2,1}$
$d_{2,I}$
$used_W$
ks_W
k_W
$d_{W,1}$
$d_{W,I}$
$USED$
KS
K
$x_{1,1,1}$
$x_{1,1,2}$
$x_{1,1,g_1}$
$x_{1,2,1}$
$x_{1,2,2}$
$x_{1,2,g_2}$
$x_{1,I,1}$
$x_{1,I,2}$
x_{1,I,g_I}
$x_{2,1,1}$
$x_{2,1,g_1}$
$x_{2,I,1}$
x_{2,I,g_I}
$x_{W,1,1}$
$x_{W,1,g_1}$
$x_{W,I,1}$
x_{W,I,g_I}
$d_1 = \bigtimes_{i=1}^{I} d_{1,i}$
$d_2 = \bigtimes_{i=1}^{I} d_{2,i}$
$d_W = \bigtimes_{i=1}^{I} d_{W,i}$
$x_{1,1,1}, x_{1,2,1}, \ldots, x_{1,I,1}$
$x_{1,1,1}, x_{1,2,2}, \ldots, x_{1,I,1}$
$x_{1,1,g_1}, x_{1,2,g_2}, \ldots, x_{1,I,g_I}$
$x_{2,1,1}, x_{2,2,1}, \ldots, x_{2,I,1}$
$x_{2,1,1}, x_{2,2,2}, \ldots, x_{2,I,1}$
$x_{2,1,g_1}, x_{2,2,g_2}, \ldots, x_{2,I,g_I}$
$x_{W,1,1}, x_{W,2,1}, \ldots, x_{W,I,1}$
$x_{W,1,1}, x_{W,2,2}, \ldots, x_{W,I,1}$
$x_{W,1,g_1}, x_{W,2,g_2}, \ldots, x_{W,I,g_I}$
D
s_1
s_2
s_W
$x_{1,1,1} + x_{1,2,1} + \ldots + x_{1,I,1}$
$x_{1,1,1} + x_{1,2,2} + \ldots + x_{1,I,1}$
$x_{1,1,g_1} + x_{1,2,g_2} + \ldots + x_{1,I,g_I}$
$x_{2,1,1} + x_{2,2,1} + \ldots + x_{2,I,1}$
$x_{2,1,1} + x_{2,2,2} + \ldots + x_{2,I,1}$
$x_{2,1,g_1} + x_{2,2,g_2} + \ldots + x_{2,I,g_I}$
$x_{W,1,1} + x_{W,2,1} + \ldots + x_{W,I,1}$
$x_{W,1,1} + x_{W,2,2} + \ldots + x_{W,I,1}$
$x_{W,1,g_1} + x_{W,2,g_2} + \ldots + x_{W,I,g_I}$
S

Figure 5.2: Key preparations and details for the HEADA.

In order to achieve anonymous authentication processes using the HEADA, all the numbers in the set S should be unique. The M sub-keys are generated and distributed by the server in a way that guarantees the uniqueness of S values as will be shown in the keys generation process. The server stores the sub-keys in the tag in the deployment stage. The number and sizes of the groups (I and G) are stored in the tag. There is no need for any encryption or hashing circuits to be deployed in tags. Instead of that, an integer addition circuit is implemented in the tag, which is simpler and cheaper than encryption and hashing circuits. This reduces the processing cost as well as the chip cost.

The server also generates a selection key (ks_n) for each tag t_n, where

$$KS = \{ks_n : n \in \mathbb{Z},\ 1 \leq n \leq W\}$$

ks_n is used to determine the order of combination selection in the tag t_n. It is kept secret in both tag and server sides. Generation of the key as well as the selection process using the key will be discussed in Sub-Section 5.3.3.
For each tag, the server also stores a list of used session keys ($used_n$). Therefore,

$$USED = \{used_n : n \in \mathbb{Z},\ 1 \leq n \leq W\}$$

where $USED$ is the set of used combinations lists for the tags in T.

As an example, assume that $W = 3, M = 8, I = 2$ and $G = \{4,4\}$, then the keys of tags t_1, t_2 and t_3 are k_1, k_2 and k_3 respectively, where

$$\begin{aligned}
k_1 &= \{x_{1,1,1}, x_{1,1,2}, x_{1,1,3}, x_{1,1,4}; x_{1,2,1}, x_{1,2,2}, x_{1,2,3}, x_{1,2,4}\},\\
k_2 &= \{x_{2,1,1}, x_{2,1,2}, x_{2,1,3}, x_{2,1,4}; x_{2,2,1}, x_{2,2,2}, x_{2,2,3}, x_{2,2,4}\},\\
k_3 &= \{x_{3,1,1}, x_{3,1,2}, x_{3,1,3}, x_{3,1,4}; x_{3,2,1}, x_{3,2,2}, x_{3,2,3}, x_{3,2,4}\},
\end{aligned}$$

Table 5.2 shows example values of K for $W = 3, M = 8, I = 2$ and $G = \{4,4\}$. Table 5.3 shows K as well as example values of KS and their D, S, ID values. In order to run the HEADA properly, the server keeps K, KS, ID, and a selected homomorphic encryption key (K_H) secret. Each tag t_n keeps k_n, ks_n, and φ (which is discussed in Sub-Section 5.3.3) secret. $M, I, G = \{g_1, \ldots, g_I\}$, m (is discussed in Sub-Section 5.3.4), and W are kept in both the server and the tags, they are also considered as public information, available even to attackers.

Table 5.2: Example of key values for 3 tags ($1 \leq n \leq 3$).

Key	$x_{n,1,1}$	$x_{n,1,2}$	$x_{n,1,3}$	$x_{n,1,4}$	$x_{n,2,1}$	$x_{n,2,2}$	$x_{n,2,3}$	$x_{n,2,4}$
k_0	88	69	92	4	236	118	98	221
k_1	111	176	222	134	226	231	108	216
k_2	157	199	23	46	93	113	211	103

5.3.2 Key Generation

Each key $k_n \in K$ consists of M sub-keys partitioned into I groups. The generation of these sub-keys is done in four steps as shown in Figure 5.3. The details of these steps are as follows:

- **Step 1: Selecting Parameters:** Parameters are selected to satisfy the needed level of security and efficiency as shown in Section 5.4.

- **Step 2: Raw Sub-Keys Generation:** An algorithm that uses the concept of Segment:Offset is used to generate non-overlapped segments. Once segments are produced, each segment is re-partitioned into non-overlapped sub-segments recursively until sufficient number of groups of integers, i.e. $A_1, \ldots, A_I$, can be produced. Given that in (A:B), A indicates the segment and B the offset. Therefore the range of the summations is partitioned into non-overlapped segments as A_1:(A_2:(A_3:($\ldots$:A_I))), where A_i is the set of raw integers that can be used in the ith group. Therefore, $A_1, \ldots, A_I$ can be generated in a recursive way as shown in the algorithm in Figure 5.4. Note that $Q_1, \ldots, Q_I$ are common divisors for the values in $A_1, \ldots, A_I$, respectively, which means that any value $\in A_i$ is a multiplication of Q_i. Moreover, all the values $\in A_{i+1}, \ldots, A_I$ as well as the summations of any combination composed of one number from each set are less than Q_i. For example, applying the algorithm in Figure 5.4 using the example in Sub-Section 5.3.1, where $W = 3, I = 2$, and $G = \{4, 4\}$, gives A_1 and A_2 as follows:

$$A_1 = \{13, 26, 39, 52, 65, 78, 91, 104, 117, 130, 143, 156\}$$
$$A_2 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

Given W, M, and the set $G = \{g_1, \ldots, g_I\}$, then, for any summation value of raw sub-keys combination $s = x_1 + x_2 + \ldots + x_I$, it can be reversed to get the raw

Table 5.3: Example values of K and KS and their D, S and ID values for $W = 3$, $M = 8$, $I = 2$ and $G = \{4, 4\}$.

K	D	S	ID	KS
	$\{88, 236\}$	324		
	$\{88, 118\}$	206		
	$\{88, 98\}$	186		
k_1, $d_{1,1}$: $x_{1,1,1} = 88$	$\{88, 221\}$	309		
$x_{1,1,2} = 69$	$\{69, 236\}$	305		
$x_{1,1,3} = 92$	$\{69, 118\}$	187		
$x_{1,1,4} = 4$	$\{69, 98\}$	167		
k_1, $d_{1,2}$: $x_{1,2,1} = 236$	$\{69, 221\}$	290	1	13
$x_{1,2,2} = 118$	$\{92, 236\}$	328		
$x_{1,2,3} = 98$	$\{92, 118\}$	210		
$x_{1,2,4} = 221$	$\{92, 98\}$	190		
	$\{92, 221\}$	313		
	$\{4, 236\}$	240		
	$\{4, 118\}$	122		
	$\{4, 98\}$	102		
	$\{4, 221\}$	225		
	$\{111, 226\}$	337		
	$\{111, 231\}$	342		
	$\{111, 108\}$	209		
k_2, $d_{2,1}$: $x_{2,1,1} = 111$	$\{111, 216\}$	327		
$x_{2,1,2} = 176$	$\{176, 226\}$	402		
$x_{2,1,3} = 222$	$\{176, 231\}$	407		
$x_{2,1,4} = 134$	$\{176, 108\}$	284		
k_2, $d_{2,2}$: $x_{2,2,1} = 226$	$\{176, 216\}$	392	2	4
$x_{2,2,2} = 231$	$\{222, 226\}$	448		
$x_{2,2,3} = 108$	$\{222, 231\}$	453		
$x_{2,2,4} = 216$	$\{222, 108\}$	330		
	$\{222, 216\}$	438		
	$\{134, 226\}$	360		
	$\{134, 231\}$	365		
	$\{134, 108\}$	242		
	$\{134, 216\}$	350		
	$\{157, 93\}$	250		
	$\{157, 113\}$	270		
	$\{157, 211\}$	368		
k_3, $d_{3,1}$: $x_{3,1,1} = 157$	$\{157, 103\}$	260		
$x_{3,1,2} = 199$	$\{199, 93\}$	292		
$x_{3,1,3} = 23$	$\{199, 113\}$	312		
$x_{3,1,4} = 46$	$\{199, 211\}$	410		
k_3, $d_{3,2}$: $x_{3,2,1} = 93$	$\{199, 103\}$	302	3	7
$x_{3,2,2} = 113$	$\{23, 93\}$	116		
$x_{3,2,3} = 211$	$\{23, 113\}$	136		
$x_{3,2,4} = 103$	$\{23, 211\}$	234		
	$\{23, 103\}$	126		
	$\{46, 93\}$	139		
	$\{46, 113\}$	159		
	$\{46, 211\}$	257		
	$\{46, 103\}$	149		

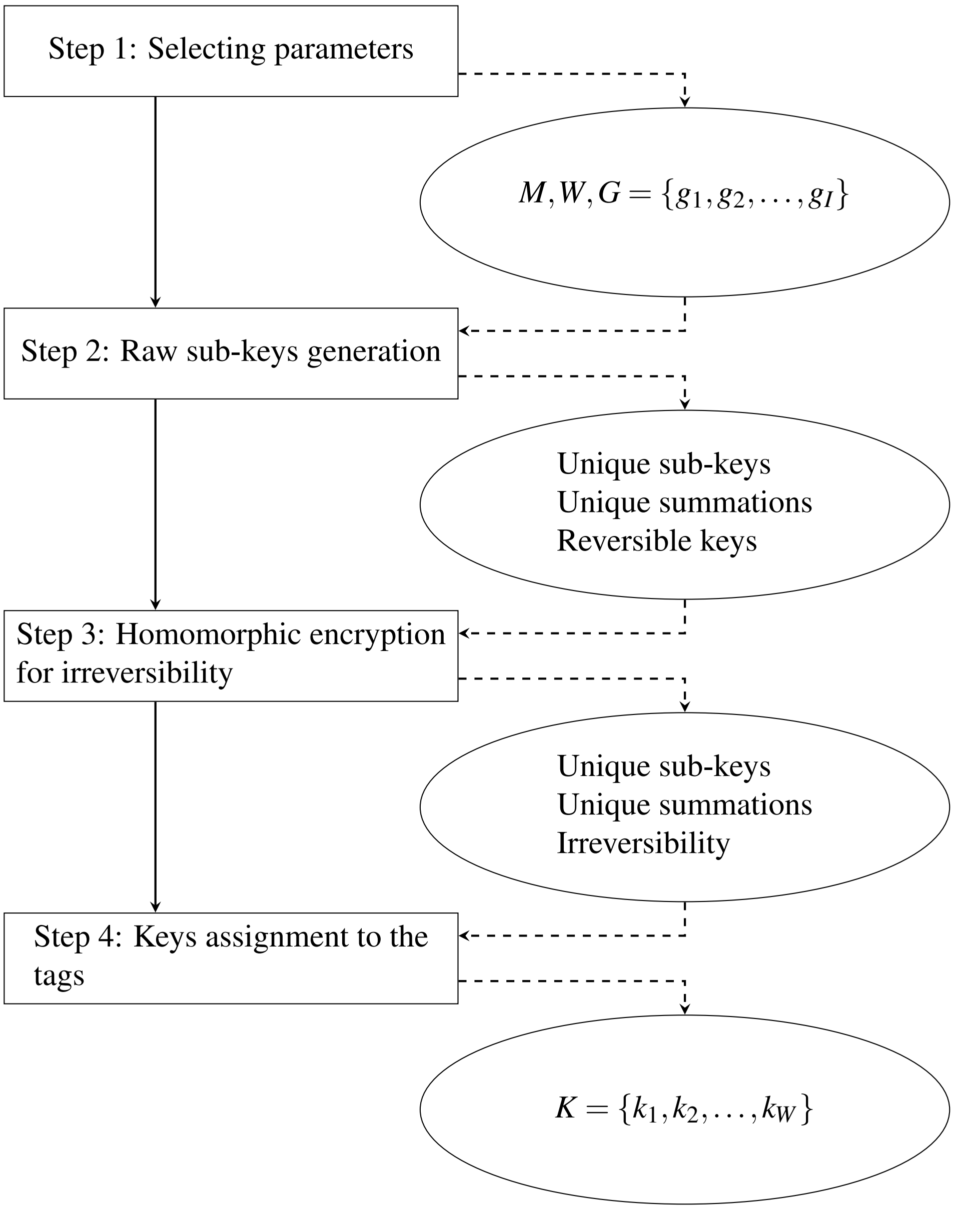
Step 1: Selecting parameters
$M, W, G = \{g_1, g_2, \ldots, g_I\}$
Step 2: Raw sub-keys generation
Unique sub-keys
Unique summations
Reversible keys
Step 3: Homomorphic encryption for irreversibility
Unique sub-keys
Unique summations
Irreversibility
Step 4: Keys assignment to the tags
$K = \{k_1, k_2, \ldots, k_W\}$

Figure 5.3: Steps of key generation

sub-keys in two steps: Initially, the values of $Q_1,\ldots,Q_I$ are computed as follows:

$$Q_i = \begin{cases} Q_{i+1}\big(1+(W*g_{i+1})\big) & \text{for } 1 \le i < I \\ 1 & \text{for } i = I \end{cases} \quad \textbf{(5.1)}$$

Then $x_1,\ldots,x_I$ are computed as follows:

$$x_i = \begin{cases} \left\lfloor \frac{s}{Q_i} \right\rfloor * Q_i & \text{for } i = 1 \\ \left\lfloor \frac{s \bmod Q_1 \ldots \bmod Q_{i-1}}{Q_i} \right\rfloor * Q_i & \text{for } 1 < i \le I \end{cases} \quad \textbf{(5.2)}$$

For example, knowing that $W = 3$, $M = 8$, $G = \{4,4\}$, suppose that the two raw sub-keys 117 and 2 are used, which means $s = 117+2 = 119$, then, Q_1, Q_2, x_1, and x_2 are calculated as follows:

$$Q_2 = 1$$
$$Q_1 = 1*(1+(3*4)) = 13$$
$$x_1 = \left\lfloor \frac{119}{13} \right\rfloor * 13 = 117$$
$$x_2 = \left\lfloor \frac{119 \bmod 13}{1} \right\rfloor * 1 = 2$$

Therefore, raw sub-keys satisfy the uniqueness of sub-keys and summations but do not satisfy irreversibility.

- **Step 3: Homomorphic Encryption for Irreversibility:** To make the reversibility property restricted only to the server, raw sub-keys are encrypted using homomorphic encryption with a key K_H. According to the homomorphic property, the summations of the combinations after encryption give a set of unique values if the original values do the same; this uniqueness is not guaranteed if the encryption algorithm does not have the addition homomorphic property. This can be shown as follws:

  Conclusion 1: Suppose that the function $f(x_1,x_2,\cdots,x_I)$ is a summation function for combination elements that are generated as shown in the algorithms shown in Figure 5.4, then, y is a unique solution of $f(x_1,x_2,\cdots,x_I)$.

  Conclusion 2: As deterministic encryption algorithm is used to encrypt the values, then $x'_1 = g(x_1)$, $x'_2 = g(x_2)$, $\cdots$, $x'_I = g(x_I)$ and $y' = g(y)$ are one to one functions, where g() is the encryption function.

  Conclusion 3: If homomorphic encryption is used, then, $y' = f(x'_1,x'_2,\cdots,x'_I)$.

  Conclusion 4: From conclusions 1, 2, and 3, y' is a unique solution of

```
**Input:** $W$, $I$, and $G = \{g_1, \ldots, g_I\}$
**Output:** $A_1, \ldots, A_I$

  $i \leftarrow 1$
  $\{Q_1, Q_2, \ldots, Q_I\} \leftarrow \{null, null, \ldots, null\}$
  $\{A_1, A_2, \ldots, A_I\} \leftarrow \{null, null, \ldots, null\}$
  $GETA(i)$
  **return** $A_1, \ldots, A_I$

  **function** GETA($i$)
      **if** $i = I$ **then**
          $A_i \leftarrow GETLIST(1, (W \times g_i), 1)$
          $Q_i \leftarrow (W \times g_i) + 1$
          **return** $Q_i$
      **else**
          $Q_i \leftarrow GETA(i+1)$
          $A_i \leftarrow GETLIST(1, (W \times g_i \times Q_i), Q_i)$
          **return** $(W \times g_i \times Q_i) + Q_i$
      **end if**
  **end function**

  **function** GETLIST($a, b, c$)
          **return** all integers $i$, where ($a \leq i \leq b \wedge i$ is multiple of $c$)
  **end function**
```

Figure 5.4: A recursion algorithm to generate the $A_1, \ldots, A_I$ sets used in key generation

$f(x'_1, x'_2, \cdots, x'_I)$.

Note that traditional encryption algorithms do not necessarily satisfy the third conclusion. Therefore, the homomorphic property keeps the first two conditions satisfied. It also nonlinearly maps the raw sub-keys into an irreversible ones. For example, encrypting the values in A_1 and A_2 shown above using the Algebra Homomorphic Encryption Scheme (AHEE) [49] and the AHEE homomorphic key set $\{p = 241, q = 197, x = 23, g = 13, r = 54, k = 68\}$ gives

$$A'_1 = \{88, 176, 23, 111, 199, 46, 134, 222, 69, 157, 4, 92\}$$
$$A'_2 = \{118, 236, 113, 231, 108, 226, 103, 221, 98, 216, 93, 211\}$$

Suppose that the encrypted values are used, instead of the raw sub-keys in the same reversibility example shown above. Note that the encryptions of 117 and 2 are 69 and 236, respectively, and $s = 69 + 236 = 305$. Applying the same reversibility process gives:

$$x_1 = \left\lfloor \frac{305}{13} \right\rfloor * 13 = 299$$
$$x_2 = \left\lfloor \frac{305 \bmod 13}{1} \right\rfloor * 1 = 6$$

Decrypting 299 and 6 using the same key K_H gives 25 and 94 which are not found in any set of raw sub-keys. Therefore, Encrypting the raw sub-keys satisfies the third condition: Irreversibility.

- **Step 4: Keys Assignment to the Tags:** For each key k_n, the server selects randomly, without duplication, numbers from $A'_1, \ldots, A'_I$ sets and place them in the related groups of k_n until all the sub-keys are selected.

5.3.3 Sub-keys Combination Selection

Sub-keys combination selection technique should satisfy the following conditions:

1. Each combination is selected only once.
2. It has to be easy for the parties that have the selection key to find the order of the combinations.
3. 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.

Note that for any two natural numbers α and β where $1 < \alpha < \beta$ and $gcd(\alpha, \beta) = 1$, the function

$$\theta(i) = (i + \alpha) \mod \beta \qquad \text{for } 0 \leq i \leq \beta$$

is one-to-one function and $\theta(i) \neq i$. The selection algorithm uses the function $\theta(i)$ to select the combinations in different order based on a selection key (ks_n) and maximum (max) values which are α and β, respectively, in the function $\theta(i)$. Given W, I and $G = \{g_1, \ldots, g_I\}$, to find the max value, the minimum number of binary bits that represent all the elements in each group are added in ω. Then,

$$max = (2^{\omega}) - 1 \tag{5.3}$$

To select a selection key for a tag t_n, the server selects randomly ks_n where $0 < ks_n < max$ and $gcd(ks_n, max) = 1$. Each tag also stores a number that indicates the last selected combination, which is denoted as φ. In each selection process, φ is updated as

$$\varphi = (\varphi + ks_n) \mod max \tag{5.4}$$

then, it is used to select the next combination. The selection of the sub-keys of the next combination is done by initially splitting the binary bits of φ into I binary groups where the length of ith group is equal to the minimum number of bits that represent g_i. The selection operation is completed by using the divided bits as indexes for the sub-keys in the groups $g_1, \ldots, g_I$. Note that φ has to be set to zero in the deployment stage.

The sizes of the groups are selected to be an exponential of 2 in the HEADA, such as 4 where 3 is the optimum value as shown in Sub-Section 5.4.1. In this case, the mod operation can be done by simple integer addition with ignoring any overflow that may result from this addition process, which simplifies the implementation of the algorithm in the tags. Therefore, based on Table 5.3, the first combination that is selected in tag t_1 is $\{4, 118\}$. It is selected as follows:

$$\begin{aligned}
Current\ state &: \varphi = 0 \\
New\ state &: \varphi = (0 + 13) \bmod 15 = 13 \\
Splitting\ the\ binary\ bits &: 13 \rightarrow 1101 \rightarrow \{11, 01\} \\
Indexing &: x_{1,1,4} = 4,\ x_{1,2,2} = 118
\end{aligned}$$

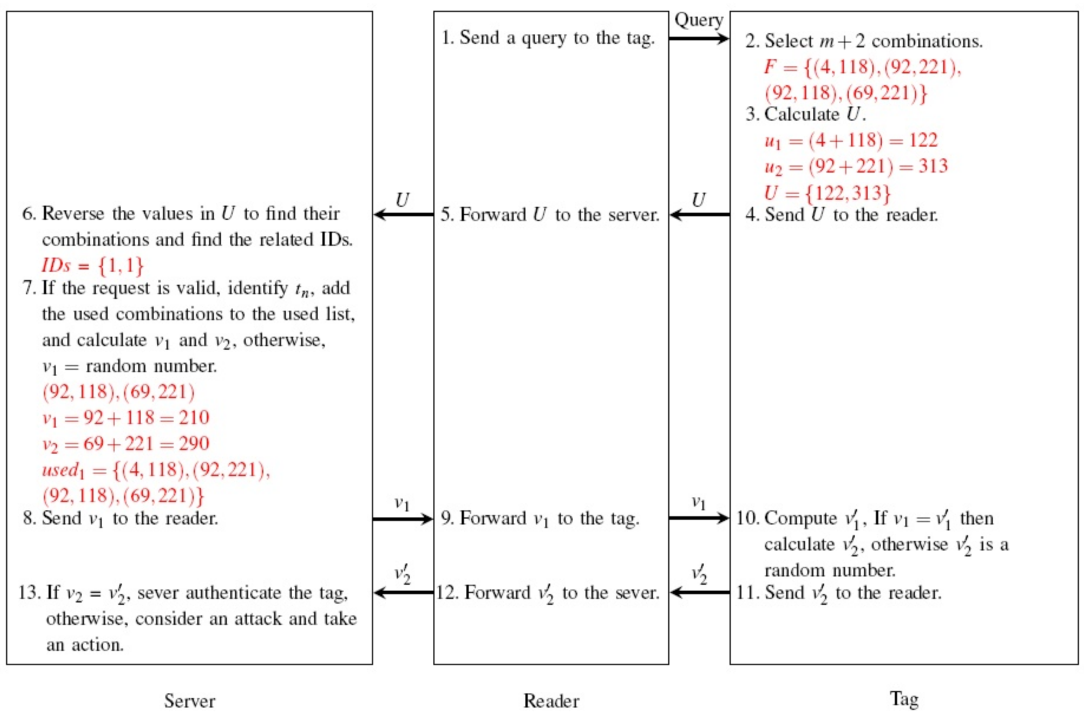
1. Send a query to the tag.
Query
2. Select $m+2$ combinations.
$F=\{(4,118),(92,221),(92,118),(69,221)\}$
3. Calculate U.
$u_1=(4+118)=122$
$u_2=(92+221)=313$
$U=\{122,313\}$
4. Send U to the reader.
U
5. Forward U to the server.
U
6. Reverse the values in U to find their combinations and find the related IDs.
$IDs=\{1,1\}$
7. If the request is valid, identify t_n, add the used combinations to the used list, and calculate v_1 and v_2, otherwise, $v_1=$ random number.
$(92,118),(69,221)$
$v_1=92+118=210$
$v_2=69+221=290$
$used_1=\{(4,118),(92,221),(92,118),(69,221)\}$
8. Send v_1 to the reader.
v_1
9. Forward v_1 to the tag.
v_1
10. Compute v_1', If $v_1=v_1'$ then calculate v_2', otherwise v_2' is a random number.
11. Send v_2' to the reader.
v_2'
12. Forward v_2' to the sever.
v_2'
13. If $v_2=v_2'$, sever authenticate the tag, otherwise, consider an attack and take an action.
Server
Reader
Tag

Figure 5.5: Sequence of operations and communications between the tag, the reader and the server

The server checks the successiveness of the received combinations by creating an m-length linked list that contains all the received combinations. The *next* pointer of each node (except the last node) points to the node that has the combination expected next to the one in the current node based on the same selection algorithm and key ks_n of the identified tag t_n. If the linked list is defined without disconnections or loops for m different combinations, they are considered successive, otherwise, they are considered disconnected.

5.3.4 Authentication Protocol

Figure 5.5 shows the authentication processes with an example values based on the values in Table 5.3 with $n=1$ and $m=2$. Following are the details of the processes to authenticate a tag t_n in the server:

1. The reader starts by a query to the tag t_n.

2. The tag t_n selects a set of $m+2$ sequential combinations of sub-keys (F) from d_n using its selection key ks_n, so, $F=\{f_1,\ldots,f_{m+2}\}$ where f_i indicates a single sequential combination.

3. t_n calculates the summations of the sub-keys for the first m selected combinations and store them in U, therefore, $U = \{u_1, \ldots, u_m\}$ where u_i is the sum of the related combination, i.e. f_i.

4. t_n sends U to the reader.

5. The reader forwards U to the server.

6. The server: a) Decrypts the numbers in U using K_H, b) Reverses the decrypted values to retrieve the selected combinations, c) Searches the keys list (K) for the retrieved combinations and returns the related IDs.

7. The server checks whether: a) There is no ID' that appears m times, or b) The combinations which are used to generate U are not sequential, or c) At least one of the combinations which are used to generate U is used before.

   If any of the above checks is correct, the server generates a random number for (v_1). Otherwise, the server considers ID' as an ID of a candidate tag subject to verification. The server: a) Selects two combinations next to the ones used in U and calculate their summations (v_1 then v_2), b) Add the combinations selected for U as well as the ones selected for v_1 and v_2 to the related used list, i.e. $used_n$.

8. The server sends v_1 to the reader.

9. The reader forwards v_1 to t_n.

10. t_n calculates v'_1 in the same way as in the server using the $m+1$th selected combination and compares it to v_1 value received from the reader. If $v'_1 = v_1$, then the tag authenticates the server and generates v'_2 using the $m+2$th selected combination. Otherwise, a random number is stored in v'_2.

11. t_n sends v'_2 to the reader.

12. The reader forwards v'_2 to the server.

13. The server compares v_2 to v'_2. If $v_2 = v'_2$, then the tag is authenticated in the server. Otherwise, the server considers the process as an attack and takes an action.

Therefore, the authentication is done and both the server and the tag verified each other. To explain these steps formally, suppose the following notations [92]:

- $P \models X$: P believes X, or P would be entitled to believe X. In particular, P may act as though X is true.

- $P \lhd X$: P sees X. Someone has sent a message containing X to P, who can read and repeat X (possibly after doing some decryption).

- $P \sim X$: P once said X. P at some time sent a message including the statement X. It is not known whether the message was sent long ago or during the current run of the protocol, but it is known that P believed X when he sent the message.

- $P \Rightarrow X$: P has *jurisdiction* over X. P is an authority on X and should be trusted on this matter. This construct is used when a party has delegated authority over some statement. For example, encryption keys need to be generated with some care, and in some protocols certain servers are trusted to do this properly. This may be expressed by the assumption that the parties believe that the server has jurisdiction over statements about the quality of keys.

- $\sharp(X)$:The formula X is fresh, that is, X has not been sent in a message at any time before the current run of the protocol.

- $P \xleftrightarrow{K} Q$: P and Q may use the shared key K to communicate. The key K is good, in that it will never be discovered by any party except P or Q, or a party trusted by either P or Q.

- $\{X\}_K$: This represents the formula X is encrypted under the key K using fully homomorphic encryption algorithm.

- $\Gamma_1(X,Y)$: This represents a function of X an Y. In our protocol, X represents the user key and Y represents a session key. The result of the function is a new unique session key.

- $\Gamma_1^{-1}(X,Y)$: This represents the inverse of the function $\Gamma_1(X,Y)$.

- $\Gamma_2(X,Y)$: This represents a function of X an Y. In our protocol, X represents the user key and Y represents a session key. The result of the function is a new unique nonce.

- $\Gamma_3(X,Y)$: This represents a function of X an Y. In our protocol, X represents the user key and Y represents a session key. The result of the function is a new unique nonce.

Suppose that the server and the user are noted as S and A_i, respectively, therefore, the used assumptions are listed below:

- $S \models S \xleftrightarrow{K_{A_i}} A_i$
- $A_i \models S \xleftrightarrow{K_{A_i}} A_i$
- $S \lhd UsedList$
- $S \lhd K_S$
- $S \models (A_i \Rightarrow (A_i \sim \{\Gamma_1(K_{A_i}, U_{SA_i})\}_{K_S}))$
- $S \Rightarrow \Gamma_1^{-1}(K_{A_i}, U_{SA_i})$
- $A_i \models S \Rightarrow \Gamma_1^{-1}(K_{A_i}, U_{SA_i})$
- $S \Rightarrow \{\Gamma_2(K_{A_i}, U_{SA_i})\}_{K_S}$
- $A_i \Rightarrow \{\Gamma_2(K_{A_i}, U_{SA_i})\}_{K_S}$
- $S \models A_i \Rightarrow \{\Gamma_2(K_{A_i}, U_{SA_i})\}_{K_S}$
- $A_i \models S \Rightarrow \{\Gamma_2(K_{A_i}, U_{SA_i})\}_{K_S}$
- $S \Rightarrow \{\Gamma_3(K_{A_i}, U_{SA_i})\}_{K_S}$
- $A_i \Rightarrow \{\Gamma_3(K_{A_i}, U_{SA_i})\}_{K_S}$
- $S \models A_i \Rightarrow \{\Gamma_3(K_{A_i}, U_{SA_i})\}_{K_S}$
- $A_i \models S \Rightarrow \{\Gamma_3(K_{A_i}, U_{SA_i})\}_{K_S}$

Then the protocol steps are formalized as follows:
Steps 1, 2, and 3:

$$A_i \models \{\Gamma_1(K_{A_i}, U_{SA_i})\}_{K_S}$$

Steps 4, 5, 6, and 7:

$$S \lhd \{\mathit{\Gamma}_1(K_{A_i}, U_{SA_i})\}_{K_S}$$

$$\frac{S \lhd \{\mathit{\Gamma}_1(K_{A_i}, U_{SA_i})\}_{K_S} \quad , \qquad S \lhd K_S}{S \lhd \mathit{\Gamma}_1(K_{A_i}, U_{SA_i})}$$

$$\frac{S \lhd \mathit{\Gamma}_1(K_{A_i}, U_{SA_i}) \quad , \qquad S \Rightarrow \mathit{\Gamma}_1^{-1}(K_{A_i}, U_{SA_i})}{S \lhd K_{A_i}}$$

$$\frac{S \lhd \mathit{\Gamma}_1(K_{A_i}, U_{SA_i}) \quad , \qquad S \Rightarrow \mathit{\Gamma}_1^{-1}(K_{A_i}, U_{SA_i})}{S \lhd U_{SA_i}}$$

$$\frac{S \lhd K_{A_i} \quad , \qquad S \models S \xleftrightarrow{K_{A_i}} A_i}{S \models (A_i \sim U_{SA_i})}$$

$$\frac{S \models (A_i \sim U_{SA_i}) \quad , \qquad S \lhd UsedList}{S \models \sharp(U_{SA_i})}$$

Steps 8 and 9:

$$\frac{S \Rightarrow \{\mathit{\Gamma}_2(K_{A_i}, U_{SA_i})\}_{K_S}}{A_i \lhd \{\mathit{\Gamma}_2(K_{A_i}, U_{SA_i})\}_{K_S}}$$

Steps 10, 11 and 12:

$$\frac{A_i \lhd \{\mathit{\Gamma}_2(K_{A_i}, U_{SA_i})\}_{K_S} \quad , \qquad A_i \models S \Rightarrow \{\mathit{\Gamma}_2(K_{A_i}, U_{SA_i})\}_{K_S}}{A_i \models (S \sim \{\mathit{\Gamma}_2(K_{A_i}, U_{SA_i})\}_{K_S})}$$

$$\frac{A_i \Rightarrow (S \sim \{\mathit{\Gamma}_2(K_{A_i}, U_{SA_i})\}_{K_S}) \quad , \qquad A_i \Rightarrow \{\mathit{\Gamma}_2(K_{A_i}, U_{SA_i})\}_{K_S}}{A_i \models \sharp(\{\mathit{\Gamma}_2(K_{A_i}, U_{SA_i})\}_{K_S})}$$

$$\frac{A_i \Rightarrow \{\mathit{\Gamma}_3(K_{A_i}, U_{SA_i})\}_{K_S}}{S \lhd \{\mathit{\Gamma}_3(K_{A_i}, U_{SA_i})\}_{K_S}}$$

Step 13:

$$\frac{S \lhd \{\mathit{\Gamma}_3(K_{A_i}, U_{SA_i})\}_{K_S} \quad , \qquad S \models A_i \Rightarrow \{\mathit{\Gamma}_3(K_{A_i}, U_{SA_i})\}_{K_S}}{S \models (A_i \sim \{\mathit{\Gamma}_3(K_{A_i}, U_{SA_i})\}_{K_S})}$$

$$\frac{S \Rightarrow (A_i \sim \{\mathit{\Gamma}_3(K_{A_i}, U_{SA_i})\}_{K_S}) \quad , \qquad S \Rightarrow \{\mathit{\Gamma}_3(K_{A_i}, U_{SA_i})\}_{K_S}}{S \models \sharp(\{\mathit{\Gamma}_3(K_{A_i}, U_{SA_i})\}_{K_S})}$$

5.4 Analysis of HEADA

There are many variables that affect security, time complexity and memory complexity of the HEADA, such as number of sub-keys (M), number of groups (I), sizes of the groups (G), number of temporary keys (m), number of tags (W) and verification values (v_1 and v_2). This section will discuss these variables as well as the operations in both server and tag sides.

5.4.1 Number of Sub-keys (M), and Number and Sizes of the Groups (I and G)

M, I and G determine the number of different combinations that can be generated from a key k_n. Since G is the same for all the keys, then, the number of different combinations that can be generated from a key k_n ($size(s_n)$) will be the same for all the tags. This number will be referred as s in the rest of this chapter, where

$$s = \prod_{i=1}^{I} g_i$$

For any M value, the minimum value of I that maximizes s should be selected to minimize the number of addition operations in each query in the tag side. Therefore, the optimum value of I, i.e. I', with respect to M has to be computed.

Suppose that $g_i = g$ for $1 \le i \le I$, the maximum number of different combinations that can be generated from a key k_n, i.e. s, becomes:

$$s = g^{\frac{M}{g}}$$

To maximize s, the first derivative of the above equation is taken and made equal to zero to find the optimum value of g, so

$$\begin{aligned} \tfrac{ds}{dg} &= (g^{\frac{M}{g}} \times \tfrac{-M}{g^2} \times \ln(g)) + (g^{\frac{M}{g}-1} \times \tfrac{M}{g} \times 1) \\ &= -(Mg^{\frac{M}{g}-2})(\ln(g) - 1) = 0 \end{aligned}$$

Solving the equation for g gives us $g_2 = e$ where e is the Euler's number which equals to 2.718281828. Therefore, the optimum value of g would be the nearest integer value to e which is 3.

5.4.2 The Use of Verification Values (v_1 and v_2)

Suppose that an attacker sent a fake query to a tag t_n, and t_n replied with a valid session key U. The attacker will not be able to generate the verification code v_1 because he does not have k_n and ks_n. Therefore, tag will not authenticate him. On the other side, if he send this session key to the server, the server will identify the tag, and reply with v_1. Again, the attacker will not be authenticated in the server because he does not have k_n and ks_n to generate v_2.

When a query comes to the tag, it directly selects m combinations as well as two combinations used for v_1 and v_2. These extra two combinations should be selected regardless of being the query is completed or not. Suppose that the combinations are not selected unless the query is completed, then the attacker can do the following steps to authenticate himself in the server:

1. Send a first query to a tag, the tag will reply with U.

2. Send a second query to the same tag, the tag will reply with U'.

3. Send U to the server.

4. Reply with the second number in U' as v_2.

By this way, the attacker will impersonate the identity of the tag and the server will authenticate him. This is not the case if the combinations are not used again as in the HEADA.

5.4.3 Number of Temporary Keys (m)

The security of the system depends mainly on the hardness of finding many unknown variables with lower number of equations. If an attacker has the U set of a query on tag t_n, then, he will need to solve the problem of finding $I \times (m+2)$ unknowns using only $m+2$ equations (considering v_1 and v_2) to find the $I \times (m+2)$ sub-keys used to generate U, v_1 and v_2. Therefore, increasing m will increase the security of the system. On the other hand, if each combination of sub-keys will be used only one time in the session keys, then, increasing m will decrease the total number of different session keys of tag t_n. The total number of different session keys of tag t_n is $\frac{s}{m+2}$.

Supposing that m is 1, an attacker can impersonate a reader identity and send a random number to the server. The server finds a match for that number with an actual record for a tag and authenticates the tag, which should not happen. This is why m should be ≥ 2. In this case it is hard for an attacker to find two (or more according to m) successive numbers that belongs to the same tag.

5.4.4 Number of Tags (W)

The server can generate pools of sub-keys for a number of keys greater than the actual number of tags. When a new tag is generated, the sub-keys of each group of the key of that tag are selected randomly from the related pools of sub-keys. When a tag is destroyed, the set of sub-keys can be returned to the pools to be used later in different sets.

5.4.5 Security of the System Based on the 7 Security Requirements

The HEADA is verified to meet the 7+2 requirements as follows:

Tag forgery resistance: Without knowing the security and selection keys, it is infeasible for an attacker to find $m+2$ sequential summations ($u_1, \ldots, u_m$, v'_1, and v'_2) related to the same tag. The probability of being $m+2$ randomly selected numbers sequential summations related to the same tag is P_{forge}, where

$$P_{forge} = \frac{(m+2)!}{(W*s)^{m+1}} \tag{5.5}$$

For example, suppose that $W = 10^8$, $M = 160$, $I = 40$, $g_i = 4$ for $1 \leq i \leq I$, and $m = 5$, then $P_{forge} < 1.62 \times 10^{-189}$, which is considered negligible. On the other hand, without the selection key, an attacker is unable to predict the selected combinations since the selection is nonlinear. To show that, the correlation between linearly selected combinations and the combinations selected using the selection algorithm and the selection keys are compared. For a key k_n, the average correlation value is calculated by partitioning the selected combinations into four equal parts. The correlation between each two parts is calculated separately and the average of absolute correlation values is taken as the average correlation value of k_n. Note that s_n is partitioned into four parts according to the size of the sub-key groups which cause the patterns to be repeated recursively number of times equal to this size if they are selected in linear order. The correlation value for the linear ordered s_n values is always 1. Table 5.4 shows the average correlation values of 5 tags for $M = 12, 16, 20, \text{and } 40$, where $g_i = 4$ for $1 \leq i \leq I$. It shows that the selection algorithm removes any patterns in s_n

Table 5.4: Average correlation value for randomly selected s_n values, where $W = 5$, $G = \{4, \ldots, 4\}$, and $M = 12, 16, 20$ and 40

Key	$M = 12$	$M = 16$	$M = 20$	$M = 40$
k_1	0.174	0.081	0.029	0.006
k_2	0.117	0.068	0.031	0.005
k_3	0.241	0.084	0.011	0.001
k_4	0.096	0.032	0.008	0.008
k_5	0.204	0.065	0.022	0.003

where the average correlation values are getting close to zero as M increases, which prevents any patterns detection attacks.

Untraceability: The tag uses different authentications key in each query. It was shown in the discussion of the *Tag forgery resistance* requirement that there is no detectable relation between any two or more authentication keys, or authentication keys and specific tags, that can be used to trace or identify a tag.

Resistance to DoS attack: The tag does not use any transmitted data to update its key in each session. Also, it does not need to be synchronized with the server. Therefore, the attacker cannot make the tag unreachable by changing any transmitted data or by desynchronizing the tag and the server. Moreover, it is infeasible to use all the authentication keys that can be generated by a tag to make it exhausted. For example, suppose that $M = 160$, $m = 5$, $I = 40$ and $g_i = 4$ for $1 \leq i \leq I$, then the number of authentication keys that can be generated by a tag is $4^{40}/(5+2) \simeq 1.7 \times 10^{23}$. Knowing that a century consists of 3.1536×10^9 seconds, an attacker needs to make more than $(1.7 \times 10^{23})/(3.1536 \times 10^9) \simeq 5.3 \times 10^{13}$ queries per second for one century to use all the possible authentication keys, which is infeasible.

Server forgery resistance: v_1 is used to authenticate the server. It is shared only between the server and the related tag by sharing the authentication and selection keys. However, v_1 should be sequential to the related numbers in U. It was shown in the discussion of the *tag forgery resistance* requirement that it is infeasible for an attacker to find v_1 for U without the security and the selection keys.

Resistance to replay attack: Any used session key cannot be used again since the server marks the used temporary keys. In case the attacker has got a session key using a fake query on the tag, he will not be able to authenticate himself in the server because he does not have the tag-to-server verification code (v_2). Moreover, any attempt to reuse a valid authentication will be detected by the server.

Data recovery: If an authentication process did not complete, the server and tag continue in the next authentication process without any problem since there is no need for synchronization between them.

Mutual authentication: v_1 is used to authenticate the server. U and v_2 are used to authenticate the tag. Therefore, both the server and the tag are authenticated.

Low cost operations in the tag: In each authentication process, the tag executes a constant number of integer addition operations for combinations selection and temporary key calculation which has $O(1)$ complexity. Moreover, comparing the execution time of one SHA-250 hash function to 5×40 integer additions (which is the case in the HEADA example) shows that the execution time of the HEADA in the tag is approximately 8% of the SHA-250 hash function execution time in the same processor and under the same conditions. Moreover, the tag in the HEADA needs small memory to store the key. For example, if $M = 160$, $m = 5$, $W = 10^8$, $I = 40$, and $g_i = 4$ for $1 \leq i \leq I$, then, the memory size required to store the key in the tag in the HEADA is 64 Bits $\times$ 160 Sub-keys $=$ 1.25 KB which is more than feasible compared to the memory size required to store the same number of the generated authentication keys which is more than 104 Bits $\times$ $\left(4^{40}/(5+2)\right)$ Authentication keys $\simeq 2 \times 10^{12}$ TB.

Low cost operations in the server: The server decrypts the values received from the reader (U) and uses reversing algorithm to obtain the sub-keys. Both of these operations use constant cost algorithms. The server uses simple search for the sub-keys which in $O\big(log\ (W * M)\big)$ which is more efficient than brute-force operations.

Table 5.5 compares the HEADA to some similar tag anonymous authentication techniques according to the 7+2 requirements.

5.5 Summary

In this chapter, a new lightweight anonymous authentication technique (HEADA) is proposed. An unconventional use of homomorphic encryption is proposed to provide low-cost security and privacy in the HEADA. The homomorphic encryption is used solely in the generation of keys during deployment stage. The methodology is verified

Table 5.5: Comparison between different tag authentication techniques

Requirements	Chien [93]	P.Lopez [94]	Karthikeyan [95]	Chen [96]	Qingling [97]	Choi [98]	Sun [99]	Yi [100]	Weis [101]	Yeo [102]	Dixit [103]	Bringer [104]	Juels [105]	Akgun [50]	HEADA
1- Untraceability	·	√	√	√	√	·	√	√	∇	√	√	·	·	∇	√
2- Resistance to replay attack	√	√	·	·	√	√	·	√	·	·	√	∇	∇	√	√
3- Resistance to DoS attack	∇	·	·	√	√	√	√	∇	√	·	·	·	·	∇	√
4- Mutual Authentication	√	√	√	√	√	√	√	√	·	·	√	·	·	√	√
5- Tag Forgery Resistance	·	√	√	√	·	√	√	√	·	·	√	·	·	√	√
6- Server Forgery Resistance	·	√	√	√	√	·	·	√	·	∇	∇	√	√	√	√
7- Data Recovery	√	·	·	√	√	√	√	√	√	·	·	√	√	·	√
8- Operations in the Server	·	√	√	·	·	√	·	·	·	·	·	√	√	·	√
9- Operations in the Tag	√	√	√	√	√	√	√	√	·	·	·	√	√	·	√

√= provided ∇= partially provided ·= not provided on the RFID systems which composed of thre parties: an authentication server, readers, and tags (or users). Users in the HEADA are shown able to generate a huge number of anonymous authentication keys which are identifiable and verifiable only by the authentication server. These authentication keys can be generated by the users using a small number of sub-keys. These sub-keys can be stored easily in the users units. In the next chapter, HEADA is modified and applied on the users quires to prevent any replay attacks. It is an essential part of the complete privacy preserving data retrieval system.

6. HIGHLY SECURED DOCUMENT RETRIEVAL IN DATA CLOUDS

The retrieval technique proposed in Chapter 4 was shown more efficient compared to binary keyword-based search systems. However, it is still unable to hide the query and documents patterns. It is also vulnerable to replay attacks. In this chapter, anonymous authentication as well as multi-server setting is used to extend the technique to satisfy the 9+1 requirements defined in Chapter 1.

6.1 Introduction

The technique in Chapter 4 starts by encrypting the documents and their IDs, separately, using a symmetric or asymmetric encryption algorithm and a key K_s by the data owner. He also creates the TF-IDF table and normalize the values to hide any frequency in the data. Finally, he encrypts the normalized values separately using a homomorphic encryption algorithm and a key K_h to generate the searchable index S. The index S is sent to the cloud, while K_h and K_s are sent to the user (the trapdoors). To generate a query, the user creates the TF vector of the query document. Then, it normalizes the values and encrypts them separately using the same homomorphic encryption algorithm and the key K_h used by the data owner. In the first round, the user sends this query to the cloud which in turn calculates the similarity between the query and each document in the dataset using S. The cloud sends the similarity vector to the user which decrypts its values and orders them. In the second round, the user sends the IDs of the documents that have the highest similarity values to the cloud. The cloud sends back the encrypted documents. The user decrypts them using K_s. The conditions 2,3, and 6 are broken as follows:

Query pattern: The normalization and encryption of the values of a TF-IDF vector hide any data frequency in that vector. However, the same document generates the same query if it is queried different times. Moreover, if two documents have a similar TF values but different distributions of these values, they will give two queries that have the same values in different distributions.

Documents pattern: The documents are requested in a clear format from the cloud in the second round. Therefore, the cloud can relate the documents requested by a user in the second round to the query sent by the same user in the first round. It can also relate the requested documents to each other.
Replay attacks: Any valid query can be reused by any party. Although unauthorized parties can't compromise the contents of the query, similarity vector, and retrieved documents, they still able to use these valid queries in denial-of-service attacks.

Therefore, finding a technique that satisfies the 9+1 conditions is the contribution in this chapter. The technique benefits the achievements of the technique proposed in Chapter4 [106]. It is extended to overcome its deficiencies to reach a complete ranked multi-keyword secure data retrieval system over cloud system.

6.2 The Technique

Beside the data owner, cloud (which called searching server in this chapter), and user, which are similar to the ones shown in Figure 1.1, the proposed technique model includes an authentication server, ranking server, private server, and L document servers. Searching server, authentication server, ranking server, private server, and document servers are assumed to be "honest-but-curious" and do not collaborate with each other, which is consistent with previous works. The same assumptions regarding the data owner, cloud, and user reported in Section 6.1 for the model in Figure 1.1 are used here. Also, the needed authorizations and communication security between the system parties are assumed to be appropriately done. Moreover, hiding the communications paths maybe essential in such systems, however, it is considered out of the scope of this thesis. Following notations are used in the explanation and discussion of the proposed technique:

- N: The number of documents owned by the data owner.
- M: The number of unique keywords in the entire documents.
- L: The number of data servers.
- K_h: Homomorphic encryption key.
- K_s: Key for documents and IDs encryption.

- K_a: Key for authentication keys generation.
- k_a: Single authentication key.
- K_c: Key for combination selection.
- $D = d_1, d_2, \ldots, d_N$: The set of documents.
- $ID = id_1, id_2, \ldots, id_N$: The set of IDs of the documents.
- $E[D] = E[d_1], E[d_2], \ldots, E[d_N]$: The set of encrypted documents using K_s.
- $E[ID] = E[id_1], E[id_2], \ldots, E[id_N]$: The set of encrypted IDs using K_s.
- $TF - IDF = [x_{n,m} | 1 \leq n \leq N$ and $1 \leq m \leq M]$: TF-IDF table.
- $\Gamma(TF - IDF) = [x'_{n,m} | 1 \leq n \leq N$ and $1 \leq m \leq M]$: Normalized TF-IDF table.
- $S = [s_{n,m} | 1 \leq n \leq N$ and $1 \leq m \leq M]$: The encrypted searchable index.
- QTF: Term Frequency vector of the query document.
- $\Gamma(QTF) = [f_1, f_2, \ldots, f_M]$: Normalized TF vector of the query document.
- ρ: Random number ≥ 1.
- $Q = [q_1, q_2, \ldots, q_M]$: Query vector.
- r: The number of documents to be retrieved.
- $CS = [cs_1, cs_2, \ldots, cs_N]$: Cosine similarity vector between Q and S.
- κ: Randomization factor ≥ 0.

Figure 6.1 shows the model of the proposed technique. It can be divided into six processes: data outsourcing, query generation, query authentication, similarity vector calculation, similarity vector ranking, and documents retrieval. The rest of this section discusses the implementation of these processes. The security of the proposed technique is discussed in Section 6.3.

6.2.1 Data Outsourcing

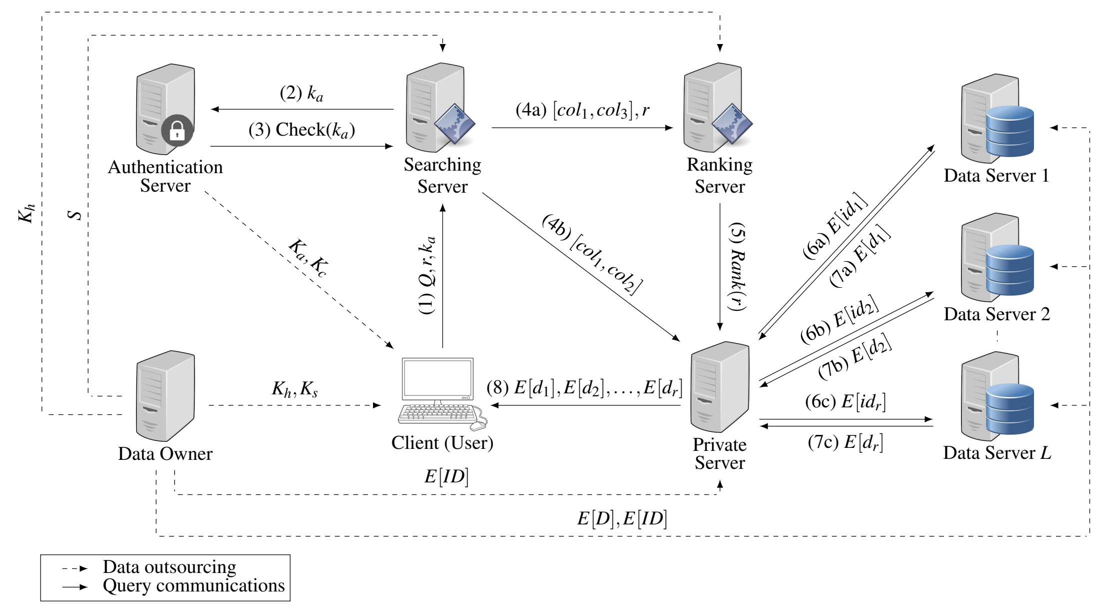

Figure 6.1: The architecture of the proposed technique.

The data owner generates the TF-IDF ($TF - IDF$) table of the set of documents D. The searchable index S is generated by normalizing $TF - IDF$ values and encrypting them by a homomorphic encryption algorithm and key K_h as described in Chapter 4. Moreover, the documents (D) and their IDs (ID) are encrypted separately by symmetric or an asymmetric encryption algorithm and key K_s to generate $E[D]$ and $E[ID]$, respectively. Thereafter, S is sent to the searching server, K_h and K_s are sent to the user, K_h is sent to the ranking server, $E[ID]$ is sent to the private server, while $E[D]$ and $E[ID]$ are sent to the document servers.

6.2.2 Query Generation

The user calculates QTF of the query document. The values of QTF are normalized as described in Chapter 4 to generate $\Gamma(QTF)$. An extra step before encrypting these normalized values is to multiply each normalized value by a number ρ, which is a random number greater than zero generated for each single query, to generate $\Gamma_\rho(QTF)$. Therefore,

$$\Gamma_\rho(QTF) = [(f_1 \times \rho), (f_2 \times \rho), \ldots, (f_M \times \rho)]$$

Multiplication by ρ is used to hide any query pattern as will be shown in Section 6.3. Finally, the values of $\Gamma_\rho(QTF)$ are encrypted by the same homomorphic encryption algorithm used by the data owner and the key K_h to generate Q.

The user sends the query which consists of Q, the number of documents to be retrieved (r), and the authentication key k_a (will be discussed in subsection 6.2.3) to the searching server.

6.2.3 Query Authentication

Utilizing the anonymous query authentication in the proposed technique may provide many properties such as prevention of replay attack, allowing only authorized users to create a valid query, services pricing and billing, privileges granting, etc. However, only the added security properties are discussed in this thesis, where the others are out of its scope.

In Chapter 5, an anonymous RFID authentication technique called HEADA is proposed. The RFID system consists of three parties: servers, readers, and tags. The servers have the data needed for authentication. The readers are responsible of transferring data between the tags and the servers. The tags are entities, with very limited resources, attached to objects to get information about them. Each tag has a different key K_a which is composed of grouped sub-keys. These sub-key groups are used to generate authentication keys. Each authentication key is composed of $\Delta + 2$ integer numbers. Each of these integer numbers is calculated by adding one sub-key from each group of the sub-key groups. Combinations of sub-keys are selected in an order which is known only for the tag and the server based on a key K_c. The first Δ integer numbers (k_a) are used for tag identification and authentication. The other two integer numbers (Θ_1 and Θ_2) are used for server verification. Therefore, only the authorized RFID tags can generate valid authentication keys which are identifiable and acceptable by the server. Moreover, the keys are changing in each authentication process to prevent any unauthorized party (including the readers) from tracking a tag. Although the technique was originally designed to authenticate RFID tags, it is more than suitable to be applied in the proposed technique based on the following properties:

1. The technique was shown secure against key forgery and key exposure attacks.

2. The technique was shown secure against replay attacks.

3. Generation of an authentication key in the tag is very simple.

4. Identification of a tag in the server is done by a simple search in the tag list.

5. Only the authentication server is able to identify the user.
6. No need for key synchronization between the tag and the server.
7. After key deployment, the tag does not need any data to start an authentication process.
8. Users in cloud systems have relatively huge resources compared to tags in RFID systems, which gives them higher capabilities for security.

In the proposed technique, the user plays the tag rule, the authentication server plays the server rule, while the searching server plays the reader rule. Moreover, Θ_1 and Θ_2 are not needed in the technique proposed in this chapter, which means that the user generates only k_a. Therefore, k_a is called the authentication key from now on.

The authentication server generates K_a and K_c and sends them to the user. In each query, the user generates k_a (which is different for each query) and sends it to the searching server as part of the query. The searching server forwards only k_a to the authentication server. The authentication server checks the validity of k_a. If k_a is valid, the authentication server sends $Random\_number||Accept\_Msg$ to the searching server, else, it sends $Random\_number||Reject\_Msg$. The searching server checks whether the message coming from the authentication server ends with $Accept\_Msg$ to proceed, else, the query is ignored.

6.2.4 Similarity Vector Calculation

In order to find the similarity vector between the query and the documents, the searching server uses the cosine similarity measure. Cosine similarity measure calculates the cosine value between two vectors [107]. Therefore, CS is calculated as follows:

$$CS = [cs_n | 1 \leq n \leq N]$$

where,

$$cs_n = \frac{\sum_{m=1}^{M} (q_m \times s_{n,m})}{\sqrt{\sum_{m=1}^{M} (q_m)^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} \tag{6.1}$$

However, the multiplication of normalized TF values by ρ in query generation (shown in subsection 6.2.2) has no effect on the final similarity value. This can be shown for any $n|1 \leq n \leq N$ as follows:

$$cs_n = \frac{\sum_{m=1}^{M} (\rho \times \frac{q_m}{\rho} \times s_{n,m})}{\sqrt{\sum_{m=1}^{M} (\rho \times \frac{q_m}{\rho})^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} = \frac{\rho \times \sum_{m=1}^{M} (\frac{q_m}{\rho} \times s_{n,m})}{\sqrt{\rho^2 \sum_{m=1}^{M} (\frac{q_m}{\rho})^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} \tag{6.2}$$

$$= \frac{\rho \times \sum_{m=1}^{M} (\frac{q_m}{\rho} \times s_{n,m})}{\sqrt{\rho^2} \times \sqrt{\sum_{m=1}^{M} (\frac{q_m}{\rho})^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} = \frac{\sum_{m=1}^{M} (\frac{q_m}{\rho} \times s_{n,m})}{\sqrt{\sum_{m=1}^{M} (\frac{q_m}{\rho})^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} \tag{6.3}$$

$$= \frac{\sum_{m=1}^{M} (f_m \times s_{n,m})}{\sqrt{\sum_{m=1}^{M} (f_m)^2} \times \sqrt{\sum_{m=1}^{M} (s_{n,m})^2}} \tag{6.4}$$

The searching server creates a three N-length columns. The first column (col_1) consists of the numbers between 1 and N distributed randomly in the rows. The second column (col_2) consists of the encrypted IDs ($E[ID]$). The third column (col_3) consists of the cosine similarity values (CS) in the same order of $E[ID]$. The searching server orders the table [col_1, col_2, col_3] according to col_1. Finally, it sends the table [col_1, col_3] and r to the ranking server, while the table [col_1, col_2] is sent to the private server.

6.2.5 Similarity Vector Ranking

The ranking server uses K_h to decrypt the values of col_3 received from the searching server in col'_3. The table [col_1, col'_3] is ordered in descending order according to col'_3. The highest r rows of the ordered [col_1, col'_3] table are stored in a new table called $Rank(r)$. The ranking servers sends $Rank(r)$ to the private server. The private server matches the values of col_1 column of table $Rank(r)$ to the values of col_1 column of the [col_1, col_2] table received from the searching server and retrieves the encrypted IDs of the documents from the column col_2. The retrieved encrypted IDs (ε) are the encrypted IDs of the documents selected to be retrieved for the query Q.

6.2.6 Documents Retrieval

Considering that there are L data servers, assume that the private server received υ sets of documents $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_\upsilon$ to be retrieved for υ different queries $Q_1, Q_2, \dots, Q_\upsilon$, respectively. The private server selects randomly $\kappa \times \sum_{i=1}^{\upsilon} |\varepsilon_i|$ documents from $E[ID]$. These random documents together with the υ sets of documents are inserted randomly in a queue. For each document in the queue, the private server selects a data server randomly to retrieve that document. Once a requested document is retrieved from a data server, it is forwarded to the user who was sent the query related to that document, else, it is ignored. The value of κ can be changed according to υ as will be discussed in Section 6.3.

6.3 Analysis of the Technique

This section discusses the achievement of the 9 security conditions listed in Section 6.1 by the proposed technique. The tenth condition (high efficiency of data retrieval) was discussed in the beginning of this chapter as a part of the advantages of using the technique proposed in Chapter 4 [106] as a base of the proposed technique.

1. **No Index Pattern:** Whatever the values of $TF - IDF$ are, normalization described in Chapter 4 guarantees that all the values of $\Gamma(TF - IDF)$, and therefore all the values of S, are unique as shown in the example in Section 6.2. Uniqueness of S values means that for any two documents α and β where $\alpha \neq \beta$ and $features(\alpha) = features(\beta)$, $index(\alpha) \neq index(\beta)$, which achieves the first condition.

2. **No Query Pattern:** As shown in Section 6.2, for each set of similar values in QTF, a new set of unique values is generated by the normalization technique, these values are distributed randomly in place of the original values. Therefore, the similarity values of different generated queries of a single document are different. However, these queries can be recognized since they have the same values but in different distributions. For this reason, multiplication by ρ, mentioned in subsection 6.2.2, is used. Multiplying the values of $\Gamma(QTF)$ by ρ hides the distribution of the values. Suppose that $H_m = h_1, h_2, \dots, h_s$ is the histogram of the TF values of a document d_m, then the number of different queries that can be generated from the document d_m is MQ, where

$$MQ = \prod_{i=1}^{s} h_i! \qquad \text{for } h_i > 1$$

Table 6.1: Average MQ values of different datasets.

Dataset	Number of Documents	Number of Unique Keywords	Average MQ
1- webdata [66]	314	15756	2033×10^{59270}
2- mini_newsgroups [70]	400	16360	1115×10^{61691}
3- classic [69]	800	6291	3143×10^{21158}

Table 6.1 shows the average MQ values for three different datasets. It can be seen that the probability of generating two similar queries for the same document is less than $\frac{1}{10^{20000}}$, which is negligible. Therefore, random distribution of the normalized values together with hiding this distribution achieve the second condition. Moreover, using anonymous authentication technique shown in Chapter 5 makes the searching server unable to identify the user, which is another advantage of using it among the other authentication techniques.

3. **No Documents Pattern:** In [106], the similarity vector is sent to the user to decrypt it and select the documents with the highest similarity values. This may cause an extra overhead to the user. It also reveals the IDs of the documents related to the query as well as to each other. Then, the ranking server is used to decrypt and order the values, which means that it has the key K_h. To prevent the ranking server from generating queries, query authentication is used as explained in subsection 6.2.3. As the ranking server does not have the key K_a, it is unable to generate queries that are acceptable by the authentication server. Accordingly they are also ignored by the searching server.

   The searching server calculates the similarity values of the encrypted indexes without revealing their actual values. However, sending the encrypted similarity vector together with the related document IDs to the ranking server makes it able to reveal the similarity between the query and the exact documents. So, the searching server sends a temporary random IDs (col_1), instead of the original IDs, to the ranking server. In this way, the ranking server is simply decrypting and ordering random numbers before sending them to the private server. On the other hand, the private server uses the information received from both the searching server and the ranking server to find the related documents.

Referring to subsection 6.2.6, to retrieve υ sets of documents $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_\upsilon$ coming from υ different queries, the private server selects randomly $\kappa \times \sum_{i=1}^{\upsilon} |\varepsilon_i|$ documents from $E[ID]$. These random documents together with the υ sets of documents are inserted randomly in a queue. For each document in the queue, the private server selects a data server randomly from L data servers to retrieve that document. Assume that the queue is static, which means that no online insertion into the queue. Then, for a data server l, the probability of being two requested documents related to the same query is $\leq P_r$, where:

$$P_r = \frac{1}{\upsilon(max^2 - max)} \times \frac{1}{\left((\kappa+1)\sum_{i=1}^{\upsilon} \varepsilon_i\right)} \times \frac{1}{\left((\kappa+1)\sum_{i=1}^{\upsilon} \varepsilon_i\right) - 1} \times \frac{1}{L} \quad \textbf{(6.5)}$$

According to Equation 6.5, increasing L decreases P_r. However, finding large number of data servers which are "honest-but-curious"and do not collaborate with each other is not easy. Therefore, for specific values of L and υ, increasing κ decreases P_r. The private server can dynamically change κ to keep a maximum value of P_r.

4. **No Index Frequency:** Normalization of $TF - IDF$ values removes any frequency of them. It was shown by example in Chapter 4 that even if the documents are exactly similar they will have different feature vectors after normalization.

5. **No Query Frequency:** Similar to index normalization, normalization of QTF values removes any frequency of them.

6. **No Replay Attack:** Each single authentication key k_a is used only once until the authentication keys are updated [108]. Therefore, if a valid query is resent, it will be rejected by the authentication server and ignored by the searching server.

7. **Query Privacy:** The query values are encrypted using K_h. Although the ranking server has the key K_h, it is unable to create query since it does not have an authentication key. Therefore, any fake query generated by the ranking server is rejected by the authentication server and ignored by the searching server. which achieves the seventh property.

8. **Index Privacy:** The index values are encrypted using K_h. Although the searching server has the encrypted index, it is unable to disclose its values since it does not have the key K_h. Although the ranking server has the key K_h, it also unable to

disclose the index contents since it receives only the similarity vector in a random order. Moreover, if the cloud added a fake index, it is unable to get any important information since the similarity vector is encrypted. Therefore, the eighth condition is achieved.

9. **Documents Privacy:** The documents are encrypted using K_s which is known only to the data owner and users. Therefore, unauthorized parties are unable to disclose the contents of the documents. which achieves the ninth property.

Table 6.2 compares the proposed technique to the techniques reported in the literature regarding the 9 security conditions as well as the ranking property. Among the discussed techniques, it can be seen that the only technique that achieves the 9 security is the proposed technique. Moreover Table 6.3 shows the execution time of the new proposed stages in the technique. The experiments are applied on Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz (4 CPUs) with 4GB RAM and Windows 7 64 bit. Note that the index generation and query generation processes include the generation of the TF-IDF of the index and the TF of the query, respectively as well as their normalization processes. These results showed that the technique is practical and applicable on the current data retrieval systems. The proposed technique provides also a multi-keyword ranked search based on the similarity of the normalized TF-IDF values.

6.4 Summary

In this chapter, a technique that utilizes anonymous query authentication together with a multi-server setting is proposed. The technique provides an efficient ranking-based data retrieval by using the Cosine Similarity of the TF-IDF vectors. It also satisfies all the 9 security requirements, which are unsatisfied completely in the techniques reported in the literature. The given results in this chapter shows that the technique is able to satisfy all the 9+1 requirements defined at the beginning of this thesis. In other words, the technique is able to provide a complete ranked privacy-preserving data storage and retrieval on data clouds based on users queries, which is the aim of this thesis.

Table 6.2: Comparison between different privacy-preserving data retrieval techniques

Condition	Boneh et al. [19]	Liu et al. [20]	Li et al. [21]	ChinnaSamy and Sujatha [22]	Kuzu et al. [23]	Tseng et al. [24]	Li et al. [25]	Wang et al. [26]	Chuah and Hu [27]	Wang et al. [28]	Cao et al. [29]	Wang et al. [30]	Sun et al. [31]	Orencik and Savas [32]	Chen et al. [33]	Dawoud and Altilar [106]	The proposed Technique
1- No Index Pattern	×	×	×	×	√	×	×	×	×	√	×	×	×	×	×	√	√
2- No Query Pattern	×	×	×	×	×	×	×	×	×	√	×	×	×	√	×	×	√
3- No Documents Pattern	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	√
4- No Index Frequency	×	×	×	×	√	×	×	×	×	√	√	×	×	√	√	√	√
5- No Query Frequency	×	×	×	×	√	√	×	×	×	√	√	×	×	√	√	√	√
6- No Replay Attack	×	×	×	×	×	×	×	×	×	×	×	×	×	√	×	×	√
7- Query Privacy	×	×	√	×	√	√	√	√	√	√	√	√	×	√	√	√	√
8- Index Privacy	√	√	√	×	√	√	√	√	√	×	√	×	×	√	√	√	√
9- Documents Privacy	√	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√	√
10- Ranked	×	×	×	×	√	×	×	×	×	√	√	√	√	√	√	√	√

√= Achieved, ×= Not achieved.

Table 6.3: Execution time of different parts of the system (in milliseconds).

Dataset	uw-can-data	mini-20newsgroups	mini-classicdocs
Index Generation	71643	68675	22412
Index Encryption	26214	34663	26863
Query Generation	204	163	31
Query Encryption	156	108	132

7. CONCLUSIONS AND FUTURE DIRECTIONS

The thesis started by defining the security and implementation (9+1) requirements for a highly secure model of privacy-preserving search on data clouds. These requirements are defined based on encrypting both the index and the queries; however, it was shown that these requirements cannot be satisfied by only encrypting them. The diversity of cloud models as well as the different ownerships of them makes the encrypted data retrieval systems vulnerable to many privacy revealing attacks.

The explanation of the proposed model is divided into three stages: normalizing the index and query, a lightweight anonymous authentication, and using the anonymous authentication technique together with a multi-server setting to satisfy all the 9+1 requirements. The first stage considers the frequency analysis attacks keeping in mind conserving the data retrieval efficiency. Therefore, a technique that normalizes the TF-IDF tables as well as the TF vector of the queries is proposed. This technique hides any highly frequented values in the tables as well as any other relation between documents. The technique was applied on three different datasets. Results show that the technique improves the retrieval efficiency even with small values. However, the second stage is still unable to hide the query and documents patterns. The technique also is vulnerable to replay attacks.

In the second stage, another 7 security and 2 implementation requirements are defined for a convenient anonymous authentication technique. Moreover, a new lightweight anonymous authentication technique (HEADA) is proposed. The technique was applied and tested on the RFID systems. These requirements are noted as the 7+2 requirements. It was shown that some of the techniques in the literature used simple operations in the tag or the server or both, but they did not achieve all the security requirements. On the other hand, some techniques achieved many security requirements using high complexity operations in the tag and the server. None of the discussed techniques achieved all the 7+2 security and efficiency requirements which gives the proposed technique the advantage over them. It was shown that the HEADA not only achieves the 7 security requirements, but also improves the way of working in both the server and the tag by using simple search, instead of brute-force search, in the server as well as using integer addition operation in both the server and the tag in the authentication process. These properties of the HEADA make it suitable to be applied efficiently in the mobile and cloud computing systems. Therefore, the third stage shows how anonymous authentication and multi-server setting is used to extend the technique to satisfy the 9+1. It was shown that none of the techniques reported in the literature is able to satisfy all of these 9 conditions. Moreover, some of them use similarity measures which lose important information of the features. The proposed technique is shown able to satisfy these 9 security conditions. It utilizes a multi-server setting to separate the leaked information. However, it is shown that none of the servers is able to infer any information from the data that pass through it. The technique uses anonymous authentication of the queries to prevent any unauthorized party from generating a query as well as preventing the replay attacks. It also uses the cosine similarity measure to calculate the similarity between the TF of the query and the TF-IDF vectors of the documents to rank them according to their similarity to the query. This similarity measure is shown effective and applicable in the proposed technique.

It was shown that the model is built on three stages, however, each stage can be used, adapted, or improved separately. One of the current researches is applying the HEADA in the biosensor systems for the health monitoring. Another open research, is adapting the model to support fuzzy-keywords retrieval property.

REFERENCES

[1] **Hausman, K.**, **S.Cook and Sampaio, T.** (2013). *Cloud Essentials: CompTIA Authorized Courseware for Exam CLO-001*, 978-1-118-40873-5, Willy.

[2] **InfoWatch Analytical Center** (2017). GLOBAL DATA LEAKAGE REPORT 2016, **Technical Report**.

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

[4] **RightScale** (2016). State of the Cloud Report, **Technical Report**, Santa Barbara, CA 93101 USA, `http://assets.rightscale.com/uploads/pdfs/RightScale-2016-State-of-the-Cloud-Report.pdf?mkt_tok=3RkMMJWWfF9wsRonuqvAd%2B%2FhmjTEU5z17%2BokW662gIkz2EFye%2BLIHETpodcMRMFgN6%2BTFAwTG5toziV8R7fBL81u3c8QXRjq`.

[5] **Foster, I.T.**, **Zhao, Y.**, **Raicu, I. and Lu, S.** (2009). Cloud Computing and Grid Computing 360-Degree Compared, *CoRR*, ***abs/0901.0131***, `http://arxiv.org/abs/0901.0131`.

[6] **Buyya, R.** (2009). Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility, *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pp.1–1.

[7] **Amazon**, (2015), Amazon Elastic Compute Cloud (Amazon EC2), `http://aws.amazon.com/ec2/`, accessed: 2015-01-07.

[8] **IBM**, (2015), IBM Cloud Computing, `http://www.ibm.com/cloud-computing`, accessed: 2015-01-07.

[9] **Microsoft**, (2015), Azure Services Platform, `http://azure.microsoft.com`, accessed: 2015-01-07.

[10] **Google**, (2015), Google Cloud Platform, `https://cloud.google.com/`, accessed: 2015-01-07.

[11] **Pastaki Rad, M.**, **Sajedi Badashian, A.**, **Meydanipour, G.**, **Ashurzad Delcheh, M.**, **Alipour, M. and Afzali, H.**, (2009). A Survey of Cloud Platforms and Their Future, **O. Gervasi**, **D. Taniar**, **B. Murgante**, **A. Laganà**, **Y. Mun and M. Gavrilova**, editors, Computational Science and Its Applications – ICCSA 2009, volume5592 of *Lecture Notes in Computer Science*,

Springer Berlin Heidelberg, pp.788–796, http://dx.doi.org/10.1007/978-3-642-02454-2_61.

[12] **Dikaiakos, M.D.**, **Katsaros, D.**, **Mehra, P.**, **Pallis, G. and Vakali, A.** (2009). Cloud Computing: Distributed Internet Computing for IT and Scientific Research, *IEEE Internet Computing*, ***13***(5), 10–13.

[13] **Armbrust, M.**, **Fox, A.**, **Griffith, R.**, **Joseph, A.D.**, **Katz, R.**, **Konwinski, A.**, **Lee, G.**, **Patterson, D.A.**, **Rabkin, A.**, **Stoica, I. and Zaharia, M.**, (2009). Above the Clouds: A Berkeley View of Cloud Computing, http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[14] **Jansen, W. and Grance, T.** (2011). SP 800-144. Guidelines on Security and Privacy in Public Cloud Computing, **Technical Report**, Gaithersburg, MD, United States.

[15] **Shimbre, N. and Deshpande, P.** (2015). Enhancing Distributed Data Storage Security for Cloud Computing Using TPA and AES Algorithm, *2015 International Conference on Computing Communication Control and Automation*, pp.35–39.

[16] **Tan, P.**, **Steinbach, M. and Kumar, V.** *Introduction to Data Mining*.

[17] **El-Khair, I.A.**, (2009). TF*IDF, Springer US, Boston, MA, pp.3085–3086, https://doi.org/10.1007/978-0-387-39940-9_956.

[18] **Song, D.X.**, **Wagner, D. and Perrig, A.** (2000). Practical techniques for searches on encrypted data, *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, pp.44–55.

[19] **Boneh, D.**, **Di Crescenzo, G.**, **Ostrovsky, R. and Persiano, G.**, (2004). Public Key Encryption with Keyword Search, **C. Cachin and J. Camenisch**, editors, Advances in Cryptology - EUROCRYPT 2004, volume3027 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.506–522, http://dx.doi.org/10.1007/978-3-540-24676-3_30.

[20] **Liu, Q.**, **Wang, G. and Wu, J.** (2009). An Efficient Privacy Preserving Keyword Search Scheme in Cloud Computing, *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 2, pp.715–720.

[21] **Li, M.**, **Yu, S.**, **Cao, N. and Lou, W.** (2011). Authorized Private Keyword Search over Encrypted Data in Cloud Computing, *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pp.383–392.

[22] **ChinnaSamy, R. and Sujatha, S.** (2012). An efficient semantic secure keyword based search scheme in cloud storage services, *Recent Trends In Information Technology (ICRTIT), 2012 International Conference on*, pp.488–491.

[23] **Kuzu, M.**, **Islam, M.S. and Kantarcioglu, M.** (2012). Efficient Similarity Search over Encrypted Data, *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, IEEE Computer Society, Washington, DC, USA, pp.1156–1167, `http://dx.doi.org/10.1109/ICDE.2012.23`.

[24] **Tseng, F.K.**, **Liu, Y.H. and Chen, R.J.** (2012). Toward Authenticated and Complete Query Results from Cloud Storages, *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pp.1204–1209.

[25] **Li, J.**, **Wang, Q.**, **Wang, C.**, **Cao, N.**, **Ren, K. and Lou, W.** (2010). Fuzzy Keyword Search over Encrypted Data in Cloud Computing, *INFOCOM, 2010 Proceedings IEEE*, pp.1–5.

[26] **Wang, C.**, **Ren, K.**, **Yu, S. and Urs, K.** (2012). Achieving usable and privacy-assured similarity search over outsourced cloud data, *INFOCOM, 2012 Proceedings IEEE*, pp.451–459.

[27] **Chuah, M. and Hu, W.** (2011). Privacy-Aware BedTree Based Solution for Fuzzy Multi-keyword Search over Encrypted Data, *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, pp.273–281.

[28] **Wang, C.**, **Cao, N.**, **Li, J.**, **Ren, K. and Lou, W.** (2010). Secure Ranked Keyword Search over Encrypted Cloud Data, *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pp.253–262.

[29] **Cao, N.**, **Wang, C.**, **Li, M.**, **Ren, K. and Lou, W.** (2011). Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data, *INFOCOM, 2011 Proceedings IEEE*, pp.829–837.

[30] **Wang, C.**, **Cao, N.**, **Ren, K. and Lou, W.** (2012). Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data, *IEEE Transactions on Parallel and Distributed Systems*, ***23***(8), 1467–1479.

[31] **Sun, W.**, **Wang, B.**, **Cao, N.**, **Li, M.**, **Lou, W.**, **Hou, Y.T. and Li, H.** (2013). Privacy-preserving Multi-keyword Text Search in the Cloud Supporting Similarity-based Ranking, *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, ACM, New York, NY, USA, pp.71–82, `http://doi.acm.org/10.1145/2484313.2484322`.

[32] **Orencik, C. and Savas, E.** (2014). An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking, *Distributed and Parallel Databases*, ***32***(1), 119–160, `http://dx.doi.org/10.1007/s10619-013-7123-9`.

[33] **Chen, L.**, **Sun, X.**, **Xia, Z. and Liu, Q.** (2014). An Efficient and Privacy-Preserving Semantic Multi-Keyword Ranked Search over Encrypted Cloud Data, *International Journal of Security and Its Applications*, ***8***(2), 323–332.

[34] **Wei, Y. and Blake, M.** (2010). Service-Oriented Computing and Cloud Computing: Challenges and Opportunities, *Internet Computing, IEEE*, ***14***(6), 72–75.

[35] **Garg, S.K.**, **Yeo, C.S. and Buyya, R.**, (2011). Green cloud framework for improving carbon efficiency of clouds, Euro-Par 2011 Parallel Processing, Springer Berlin Heidelberg, pp.491–502.

[36] **Takouna, I.**, **Dawoud, W. and Meinel, C.** (2011). Dynamic configuration of virtual machine for power-proportional resource provisioning, *Proc. of the ACM Int. Workshop on Green Comp. Middleware*, 1–6.

[37] **Cho, H.**, **Park, J.**, **Gil, J.M.**, **Jeong, Y.S. and Park, J.H.** (2015). An Optimal Path Computation Architecture for the Cloud-Network on Software-Defined Networking, *Sustainability*, ***7***(5), 5413, `http://www.mdpi.com/2071-1050/7/5/5413`.

[38] **Mell, P. and Grance, T.** (2011). SP 800-145. The NIST Definition of Cloud Computing, **Technical Report**, Gaithersburg, MD, United States.

[39] **He, S.**, **Guo, L.**, **Ghanem, M. and Guo, Y.** (2012). Improving Resource Utilisation in the Cloud Environment Using Multivariate Probabilistic Models, *2012 IEEE Fifth International Conference on Cloud Computing*, pp.574–581.

[40] **Mao, M. and Humphrey, M.** (2012). A Performance Study on the VM Startup Time in the Cloud, *2012 IEEE Fifth International Conference on Cloud Computing*, pp.423–430.

[41] **Rivest, R.L.**, **Shamir, A. and Adleman, L.** (1978). A Method for Obtaining Digital Signatures and Public-key Cryptosystems, *Commun. ACM*, ***21***(2), 120–126, `http://doi.acm.org/10.1145/359340.359342`.

[42] **Gamal, T.E.** (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *Proceedings of CRYPTO 84 on Advances in Cryptology*, Springer-Verlag New York, Inc., New York, NY, USA, pp.10–18, `http://dl.acm.org/citation.cfm?id=19478.19480`.

[43] **Goldwasser, S. and Micali, S.** (1982). Probabilistic Encryption &Amp; How to Play Mental Poker Keeping Secret All Partial Information, *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, ACM, New York, NY, USA, pp.365–377, `http://doi.acm.org/10.1145/800070.802212`.

[44] **Clarkson, J.B.** (1994). Dense Probabilistic Encryption, *In Proceedings of the Workshop on Selected Areas of Cryptography*, pp.120–128.

[45] **Paillier, P. and Pointcheval, D.**, (1999). Efficient Public-Key Cryptosystems Provably Secure Against Active Adversaries, **K.Y. Lam**, **E. Okamoto and C. Xing**, editors, Advances in Cryptology - ASIACRYPT?99, volume1716 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.165–179, `http://dx.doi.org/10.1007/978-3-540-48000-6_14`.

[46] **Okamoto, T. and Uchiyama, S.**, (1998). A new public-key cryptosystem as secure as factoring, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.308–318, `http://dx.doi.org/10.1007/BFb0054135`.

[47] **Naccache, D. and Stern, J.** (1998). A New Public Key Cryptosystem Based on Higher Residues, *Proceedings of the 5th ACM Conference on Computer and Communications Security*, CCS '98, ACM, New York, NY, USA, pp.59–66, `http://doi.acm.org/10.1145/288090.288106`.

[48] **Gentry, C.** (2009). Fully Homomorphic Encryption Using Ideal Lattices, *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, ACM, New York, NY, USA, pp.169–178, `http://doi.acm.org/10.1145/1536414.1536440`.

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

[50] **Smart, N. and Vercauteren, F.**, (2010). Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes, **P. Nguyen and D. Pointcheval**, editors, Public Key Cryptography ? PKC 2010, volume6056 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.420–443, `http://dx.doi.org/10.1007/978-3-642-13013-7_25`.

[51] **van Dijk, M.**, **Gentry, C.**, **Halevi, S. and Vaikuntanathan, V.**, (2010). Fully Homomorphic Encryption over the Integers, **H. Gilbert**, editor, Advances in Cryptology ? EUROCRYPT 2010, volume6110 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.24–43, `http://dx.doi.org/10.1007/978-3-642-13190-5_2`.

[52] **Brakerski, Z. and Vaikuntanathan, V.**, (2011). Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages, **P. Rogaway**, editor, Advances in Cryptology ? CRYPTO 2011, volume6841 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.505–524, `http://dx.doi.org/10.1007/978-3-642-22792-9_29`.

[53] **Coron, J.S.**, **Mandal, A.**, **Naccache, D. and Tibouchi, M.**, (2011). Fully Homomorphic Encryption over the Integers with Shorter Public Keys, **P. Rogaway**, editor, Advances in Cryptology ? CRYPTO 2011, volume6841 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.487–504, `http://dx.doi.org/10.1007/978-3-642-22792-9_28`.

[54] **Armknecht, F. and Strufe, T.** (2011). An efficient distributed privacy-preserving recommendation system, *The 10th IFIP Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2011, Favignana Island, Sicily, Italy, 12-15 June, 2011*, pp.65–70, `http://dx.doi.org/10.1109/Med-Hoc-Net.2011.5970495`.

[55] **Jeckmans, A.**, **Peter, A. and Hartel, P.**, (2013). Efficient Privacy-Enhanced Familiarity-Based Recommender System, Springer Berlin Heidelberg,

Berlin, Heidelberg, pp.400–417, `http://dx.doi.org/10.1007/978-3-642-40203-6_23`.

[56] **Lauter, K.E.** (2012). Practical Applications of Homomorphic Encryption, *Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop*, CCSW '12, ACM, New York, NY, USA, pp.57–58, `http://doi.acm.org/10.1145/2381913.2381924`.

[57] **Naehrig, M.**, **Lauter, K. and Vaikuntanathan, V.** (2011). Can Homomorphic Encryption Be Practical?, *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, ACM, New York, NY, USA, pp.113–124, `http://doi.acm.org/10.1145/2046660.2046682`.

[58] **Bösch, C.**, **Peter, A.**, **Hartel, P. and Jonker, W.** (2014). SOFIR: Securely Outsourced Forensic Image Recognition, *39th IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014*, IEEE, Los Alamitos, CA, USA, `http://doc.utwente.nl/90722/`.

[59] **Brakerski, Z. and Vaikuntanathan, V.** (2011). Efficient Fully Homomorphic Encryption from (Standard) LWE, *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, IEEE Computer Society, Washington, DC, USA, pp.97–106, `http://dx.doi.org/10.1109/FOCS.2011.12`.

[60] **Yang, Z.**, **Zhong, S. and Wright, R.N.** (2005). Privacy-Preserving Classification of Customer Data without Loss of Accuracy, *In SIAM SDM*, pp.21–23, `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.9978`.

[61] **Sander, T. and Tschudin, C.F.** (1998). Protecting Mobile Agents Against Malicious Hosts, *Mobile Agents and Security*, Springer-Verlag, London, UK, UK, pp.44–60, `http://dl.acm.org/citation.cfm?id=648051.746191`.

[62] **Dawoud, M. and Altilar, D.**, (2014). Privacy-Preserving Search in Data Clouds Using Normalized Homomorphic Encryption, Euro-Par 2014: Parallel Processing Workshops, volume8806 of *Lecture Notes in Computer Science*, Springer International Publishing, pp.62–72, `http://dx.doi.org/10.1007/978-3-319-14313-2_6`.

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

[64] **Pfitzmann, B. and Waidner, M.**, (1997). Anonymous Fingerprinting, **W. Fumy**, editor, Advances in Cryptology ? EUROCRYPT ?97, volume1233 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.88–102, `http://dx.doi.org/10.1007/3-540-69053-0_8`.

[65] **Adelsbach, A.**, **Katzenbeisser, S. and Ahmad-Reza** (2002). Cryptography Meets Watermarking: Detecting Watermarks with Minimal or Zero Knowledge Disclosure, *In: Proceedings of the European Signal Processing Conference (EUSIPCO 2002). 2002. http://www.dbai. tuwien.ac.at/ staff/katzenb/download/eusipco02.ps.gz.*

[66] **Hammouda, K. and Kamel, M.**, (2013), Web Mining Data - UW-CAN-DATASET, `http://pami.uwaterloo.ca/~hammouda/webdata`.

[67] **Rajaraman, A. and Ullman, J.D.** (2011). *Data Mining: Mining of Massive Datasets*, 978-1107015357, Cambridge University Press.

[68] **Salton, G. and Buckley, C.** (1988). Term-weighting approaches in automatic text retrieval, *Information Processing and Management*, pp.513–523.

[69] **Volkan, T.**, (2012), Data Mining Research - Classic3 and Classic4 DataSets, `http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets`.

[70] **Lang, K.** (1995). Newsweeder: Learning to filter netnews, *Proceedings of the Twelfth International Conference on Machine Learning*, pp.331–339.

[71] **Fukuda, I.**, **Morishita, S. and Asama, H.** (2008). Personal identification in dynamic images using UHF band RFID system for service provision, *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp.492–497.

[72] **Wisanmongkol, J.**, **Sanpechuda, T. and Ketprom, U.** (2008). Automatic vehicle identification with sensor-integrated RFID system, *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 2, pp.757–760.

[73] **Qinghua, Z.**, **Guoquan, C.**, **Zhuan, W.**, **Jun, W. and Dawei, Y.** (2009). Development of RFID application system in cargo inbound and outbound, *TENCON 2009 - 2009 IEEE Region 10 Conference*, pp.1–6.

[74] **Ahuja, S. and Potti, P.** (2010). An Introduction to RFID Technology, *Communications and Network*, ***2***(3), 183–186.

[75] **Shih, D.H.**, **Chin-Yi, L. and Lin, B.** (2005). RFID Tags: Privacy and Security Aspects, *Int. J. Mob. Commun.*, ***3***(3), 214–230, `http://dx.doi.org/10.1504/IJMC.2005.006581`.

[76] **Want, R.** (2006). An Introduction to RFID Technology, *IEEE Pervasive Computing*, ***5***(1), 25–33.

[77] **Weis, S.A.**, (2007). RFID (Radio-Frequency Identification), John Wiley & Sons, Inc., pp.974–984, `http://dx.doi.org/10.1002/9781118256107.ch63`.

[78] **Sarma, S.E.**, **Weis, S.A. and Engels, D.W.**, (2003). RFID Systems and Security and Privacy Implications, Cryptographic Hardware and Embedded Systems - CHES 2002, volume2523 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.454–469, `http://dx.doi.org/10.1007/3-540-36400-5_33`.

[79] **Sun, H.M. and Ting, W.C.** (2009). A Gen2-Based RFID Authentication Protocol for Security and Privacy, *IEEE Transactions on Mobile Computing*, ***8***(8), 1052–1062.

[80] **Lee, H. and Kim, J.** (2006). Privacy threats and issues in mobile RFID, *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pp.5 pp.–.

[81] **Yi, X.**, **Wang, L.**, **Mao, D. and Zhan, Y.** (2012). An Gen2 Based Security Authentication Protocol for RFID System, *Physics Procedia*, ***24, Part B***(0), 1385 – 1391, `http://www.sciencedirect.com/science/article/pii/S1875389212002490`, international Conference on Applied Physics and Industrial Engineering 2012.

[82] **Chien, H.Y.** (2007). SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity, *Dependable and Secure Computing, IEEE Transactions on*, ***4***(4), 337–340.

[83] **Weis, S.A.**, **Sarma, S.E.**, **Rivest, R.L. and Engels, D.W.**, (2004). Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems, **D. Hutter**, **G. Müller**, **W. Stephan and M. Ullmann**, editors, Security in Pervasive Computing, volume2802 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.201–212, `http://dx.doi.org/10.1007/978-3-540-39881-3_18`.

[84] **Yeo, S.S. and Kim, S.**, (2005). Scalable and Flexible Privacy Protection Scheme for RFID Systems, **R. Molva**, **G. Tsudik and D. Westhoff**, editors, Security and Privacy in Ad-hoc and Sensor Networks, volume3813 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.153–163, `http://dx.doi.org/10.1007/11601494_13`.

[85] **Dixit, V.**, **Verma, H.K. and Singh, A.K.**, (2011), Enhanced Hash chain based scheme for security and privacy in RFID systems.

[86] **Bringer, J.**, **Chabanne, H. and Dottax, E.** (2006). HB++: a Lightweight Authentication Protocol Secure against Some Attacks, *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2006. SecPerU 2006. Second International Workshop on*, pp.28–33.

[87] **Juels, A. and Weis, S.A.**, (2005). Authenticating Pervasive Devices with Human Protocols, **V. Shoup**, editor, Advances in Cryptology ? CRYPTO 2005, volume3621 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.293–308, `http://dx.doi.org/10.1007/11535218_18`.

[88] **Karthikeyan, S. and Nesterenko, M.** (2005). RFID Security Without Extensive Cryptography, *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '05, ACM, New York, NY, USA, pp.63–67, `http://doi.acm.org/10.1145/1102219.1102229`.

[89] **Arbit, A.**, **Livne, Y.**, **Oren, Y. and Wool, A.** (2015). Implementing public-key cryptography on passive RFID tags is practical, *International Journal of Information Security*, ***14***(1), 85–99, `http://dx.doi.org/10.1007/s10207-014-0236-y`.

[90] **Canard, S.**, **Ferreira, L. and Robshaw, M.**, (2013). Improved (and Practical) Public-Key Authentication for UHF RFID Tags, **S. Mangard**, editor, Smart Card Research and Advanced Applications, volume7771 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.46–61, `http://dx.doi.org/10.1007/978-3-642-37288-9_4`.

[91] **Batina, L.**, **Guajardo, J.**, **Kerins, T.**, **Mentens, N.**, **Tuyls, P. and Verbauwhede, I.** (2007). Public-Key Cryptography for RFID-Tags, *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pp.217–222.

[92] **Burrows, M.**, **Abadi, M. and Needham, R.M.** (1989). A Logic of Authentication, *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, ***426***(1871), 233–271, `http://rspa.royalsocietypublishing.org/content/426/1871/233`, `http://rspa.royalsocietypublishing.org/content/426/1871/233.full.pdf`.

[93] **Chien, H.Y. and Chen, C.H.** (2007). Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 standards, *Computer Standards & Interfaces*, ***29***(2), 254–259, `ttp://www.sciencedirect.com/science/article/pii/S092054890600064X"`.

[94] **Peris-Lopez, P.**, **Castro, J.C.H.**, **Estevez-Tapiador, J.M. and Ribagorda, A.** (2009). An Ultra Light Authentication Protocol Resistant to Passive Attacks under the Gen-2 Specification, *J. Inf. Sci. Eng.*, ***25***(1), 33–57, `http://www.iis.sinica.edu.tw/page/jise/2009/200901_03.html`.

[95] **Karthikeyan, S. and Nesterenko, M.** (2005). RFID Security Without Extensive Cryptography, *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '05, ACM, New York, NY, USA, pp.63–67, `http://doi.acm.org/10.1145/1102219.1102229`.

[96] **Chen, C.L. and Deng, Y.Y.** (2009). Conformation of EPC Class 1 Generation 2 standards RFID system with mutual authentication and privacy protection, *Engineering Applications of Artificial Intelligence*, ***22***(8), 1284 – 1291, `http://www.sciencedirect.com/science/article/pii/S0952197608001814`.

[97] **Qingling, C.**, **Yiju, Z. and Yonghua, W.** (2008). A Minimalist Mutual Authentication Protocol for RFID System & BAN Logic Analysis, *Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on*, volume 2, pp.449–453.

[98] **Choi, E.Y.**, **Lee, D.H. and Lim, J.I.** (2009). Anti-cloning protocol suitable to EPCglobal Class-1 Generation-2 RFID systems, *Computer Standards & Interfaces*, ***31***(6), 1124 – 1130.

[99] **Sun, H.M. and Ting, W.C.** (2009). A Gen2-Based RFID Authentication Protocol for Security and Privacy, *IEEE Transactions on Mobile Computing*, ***8***(8), 1052–1062.

[100] **Yi, X.**, **Wang, L.**, **Mao, D. and Zhan, Y.** (2012). An Gen2 Based Security Authentication Protocol for RFID System, *Physics Procedia*, ***24, Part B***(0), 1385 – 1391, `http://www.sciencedirect.com/science/article/pii/S1875389212002490`, international Conference on Applied Physics and Industrial Engineering 2012.

[101] **Weis, S.A.**, **Sarma, S.E.**, **Rivest, R.L. and Engels, D.W.**, (2004). Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems, **D. Hutter**, **G. Müller**, **W. Stephan and M. Ullmann**, editors, Security in Pervasive Computing, volume2802 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.201–212, `http://dx.doi.org/10.1007/978-3-540-39881-3_18`.

[102] **Yeo, S.S. and Kim, S.**, (2005). Scalable and Flexible Privacy Protection Scheme for RFID Systems, **R. Molva**, **G. Tsudik and D. Westhoff**, editors, Security and Privacy in Ad-hoc and Sensor Networks, volume3813 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.153–163, `http://dx.doi.org/10.1007/11601494_13`.

[103] **Dixit, V.**, **Verma, H.K. and Singh, A.K.**, (2011), Enhanced Hash chain based scheme for security and privacy in RFID systems.

[104] **Bringer, J.**, **Chabanne, H. and Dottax, E.** (2006). HB++: a Lightweight Authentication Protocol Secure against Some Attacks, *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2006. SecPerU 2006. Second International Workshop on*, pp.28–33.

[105] **Juels, A. and Weis, S.A.**, (2005). Authenticating Pervasive Devices with Human Protocols, **V. Shoup**, editor, Advances in Cryptology - CRYPTO 2005, volume3621 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.293–308, `http://dx.doi.org/10.1007/11535218_18`.

[106] **Dawoud, M. and Altilar, D.**, (2014). Privacy-Preserving Search in Data Clouds Using Normalized Homomorphic Encryption, **L. Lopes**, **J. ?ilinskas**, **A. Costan**, **R. Cascella**, **G. Kecskemeti**, **E. Jeannot**, **M. Cannataro**, **L. Ricci**, **S. Benkner**, **S. Petit**, **V. Scarano**, **J. Gracia**, **S. Hunold**, **S. Scott**, **S. Lankes**, **C. Lengauer**, **J. Carretero**, **J. Breitbart and M. Alexander**, editors, Euro-Par 2014: Parallel Processing Workshops,

volume8806 of *Lecture Notes in Computer Science*, Springer International Publishing, pp.62–72, `http://dx.doi.org/10.1007/978-3-319-14313-2_6`.

[107] **Salton, G. and Buckley, C.** (1988). Term-weighting Approaches in Automatic Text Retrieval, *Inf. Process. Manage.*, ***24***(5), 513–523, `http://dx.doi.org/10.1016/0306-4573(88)90021-0`.

[108] **Dawoud, M. and Altilar, D.T.** (2015). HEADA: A Low Cost RFID Authentication Technique using Homomorphic Encryption for Key Generation, *International Journal of Information Security*, ***10***(4), 213–222, `http://dx.doi.org/10.1007/s10207-011-0139-0`.

APPENDICES

APPENDIX A.1

Dataset preparation is done as follows:

1. html documents are parsed using htmlparser-1.6 to extract the data from them.
2. Stopwords are removed using three different lists of stopwords: Long list, Short list and Google list.
3. Porter stemmer is used to stem the keywords.
4. The datasets are classified using k-means classification with cosine similarity distance.

A.1.1 uw-can-data Dataset

The original dataset consists of 314 HTML web pages from various web sites at the University of Waterloo, and some Canadian websites [66]. These documents are grouped into 10 groups as shown in Table A.1.

A.1.2 mini-20newsgroups

The original dataset (called 20_newsgroups) [70] consists of a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups, the number of documents is minimized to 400 documents with the same number of classes by selecting 20 documents from each group into a minimized dataset called mini-20newsgroups. The details of the minimized dataset are shown in Table A.2.

A.1.3 mini-classicdocs

The original dataset (called classicdocs) [69] consists of 7097 documents grouped into 4 groups: *cacm* with 3204 documents, *cisi* with 1460 documents, *cran* with 1400 documents, and *med* with 1033 documents. The dataset is minimized by selecting 200 documents from each group and re-classify them into 10 classes as shown in Table A.3. Note that the selected documents are renamed by the numbers 1-200, 201-400, 401-600, and 601-800 for the classes cacm, cisi, cran, and med respectively.

Table A.1: uw-can-data dataset.

Cluster Name	Number of Documents	List of Documents
black-bear-attack	30	000299; 200172417514; 4EnUp; attack; attacks; bbear; bear facts; bear-attack-in-smokies; bear-diamond2-01; bear; BearEncounters; BearhuntingA; bears-great-smoky; bears; BearTips; bear_attacks; bear_safety; Bkbear6.28.00; blackbear; blackbearfacts; black_bears; cabc; grizzly; HikeGuide; How_Dangerous_are_Black_Bears; incident; itywbr; oldnews; trav_bears; walk-in-woods
campuse-network	33	atm; ATreq; directions; external; funding; guarmin; history; homeIP; homeIP1st; index-residence; index-uwng; index; internet-account; IPreq; kwarea; local; logic; netadmin; netdescr; netfirew; netmgmt; netpract; netUpgrade; opportunities; rfi; rfp; rfp199706; rn-excess; schedule; service; strategy; tech; vlans
canada-transportation-roads	22	alberta; backgr; bibliography; c2823; ctar-recommendations; cta_rec; Default; directory; great-lakes; hampton; IISTPS; kingston; newfoundland; pgrpe; position_paper_april01; tac; transport; transportation_act; transportation_frost_com; trans_e; tu.gov.ab; whatsnew
career-services	52	abroad.add.resources; abroad.credits; abroad; actionplan; career-life; careerobjectives; chronological; communityservice; CRC_General; credits; decision; emp-contact; employercontact; entrepreneurism; functional; handson; header; index; informationinterview; informationsearch; interests; intern; introduction; jobshadow; jobworkinterview; jobworksearch; knowledge; letters; lifelonglearning; manual-home; modifiedchrono; occup-res; order; personalcareer; personality; personalobjectives; publications; reevaluation; resumes; SCA; self-assess; skills; steps; success; summer_register; tableofcontents; trends; values; vitae; Volunteering; work; workoffers
co-op	55	10_1; 10_2; 10_3; 10_4; 10_5; 6_1; 6_2; 6_3; 6_4; 7_1; 7_10; 7_11; 7_12; 7_2; 7_3; 7_4; 7_5; 7_6; 7_7; 7_8; 7_9; 8_1; 8_2; 8_3; 8_4; 8_5; 8_6; 8_7; 9_1; 9_10; 9_11; 9_2; 9_3; 9_4; 9_5; 9_6; 9_7; 9_8; 9_9; advisors; affiliations; appendixa; appendixb; emp_services; help; index-about; index-access; index-employees; index-students; index; index; info_sessions; staff; suppforfirstyearstudents; upcoming_events
health-services	23	acne; appointments; CHI; chicken; chlam; different; emergencies; events; FED; flu; hours; index; insurance1; lactose; links; services; skin; staff; support; tb; topics; verification; warts
river-fishing	23	alaska-fishing-guide-1; alaska-fishing-trip; blueskiesfishing; booking; combahee-trip; dir; dkgld; fish; fishing; fishingstoneyriverlodge; fishraider; gtjemez; guided9; indianriver; lake-tahoe-Fishing; northwestfishingguides; premium_fishing; report; rippingales; rogueklamath; salmon-fishing; trip; wolfriverangler
river-rafting	29	800classvi; alpineadventures; gauley-river-rafting; knownworldguides; nantahal; newriver-rafting; pwibowo; raft-colorado; raft; raftarizona; rafting; raftnepal; raftwet; richmondraft; riveradventures; riverrafting; riverrider; riverriders; riverwild; sierramac; travelsource-rafting; trinityriverrafting; turtleriver; welcome; westernriver; whitewater.safpar; whitewatervoyages; wildrivers.com; zrafting
snowboarding-skiing	24	8-184; Canada; cmhski; courses; default; guide-canada; helicanada; htl_snb; intro; january99; maplesquare-skiing; mountainzone; overlandersports; report; skicentral; skiing-snowboard; skiing-western-canada; skiing; SkiingSnowboardinginCanada; skileb; skischool; snoweb; thealps; tightwadtours
winter-canada	23	01-11-26-back; 6008e; almanac; climate; climate_e; driving-cond-winter; eastern_canada; icestorm98_winter_weather_warnings_e; kanclimate-e; newfoundland-winter; northeastern-winter; nwclimat; page32_e; quiz.en; snowsnowsnow; storms_index; storms_index2; turfgras; warning.en; winter; winterpage; wintstrm; ww_e

Table A.2: Minimized 20_newsgroups dataset (mini-20newsgroups).

Cluster Name	Number of Documents	List of Documents
alt.atheism	20	53670; 53753; 53759; 53760; 54137; 54140; 54144; 54160; 54170; 54171; 54201; 54215; 54222; 54234; 54237; 54244; 54250; 54251; 54254; 54485
comp.graphics	20	38983; 38998; 39000; 39008; 39013; 39017; 39022; 39027; 39048; 39049; 39072; 39078; 39615; 39620; 39621; 39659; 39663; 39664; 39668; 39675
comp.os.ms-windows.misc	20	10160; 10167; 10188; 10692; 10742; 10781; 10790; 10791; 10806; 10812; 10814; 10830; 10835; 10838; 10843; 10848; 10849; 10850; 10857; 10942
comp.sys.ibm.pc.hardware	20	61019; 61022; 61026; 61039; 61044; 61046; 61060; 61076; 61090; 61094; 61098; 61120; 61130; 61153; 61154; 61158; 61164; 61168; 61173; 61175
comp.sys.mac.hardware	20	52190; 52214; 52223; 52231; 52234; 52238; 52246; 52248; 52264; 52269; 52270; 52276; 52284; 52296; 52300; 52312; 52335; 52342; 52403; 52404
comp.windows.x	20	67542; 67572; 67973; 67981; 67983; 67995; 68002; 68012; 68019; 68047; 68110; 68137; 68174; 68185; 68228; 68232; 68237; 68239; 68243; 68311
misc.forsale	20	76589; 76592; 76594; 76607; 76649; 76704; 76782; 76785; 76795; 76814; 76831; 76847; 76851; 76879; 76880; 76927; 76936; 76937; 76940; 76945
rec.autos	20	103440; 103445; 103497; 103503; 103510; 103663; 103667; 103680; 103689; 103698; 103704; 103714; 103723; 103728; 103734; 103740; 103758; 103771; 103777; 103806
rec.motorcycles	20	105133; 105135; 105140; 105150; 105153; 105154; 105159; 105205; 105207; 105217; 105220; 105223; 105238; 105243; 105249; 105252; 105254; 105257; 105661; 105662
rec.sport.baseball	20	104942; 104957; 104963; 104966; 104972; 104973; 104977; 104984; 105034; 105048; 105070; 105075; 105093; 105102; 105104; 105111; 105114; 105122; 105123; 105163
rec.sport.hockey	20	54304; 54365; 54478; 54507; 54512; 54548; 54549; 54709; 54714; 54715; 54721; 54724; 54737; 54739; 54745; 54754; 54765; 54771; 54776; 54780
sci.crypt	20	15914; 15919; 15924; 15953; 15955; 15962; 15967; 15996; 16013; 16026; 16029; 16036; 16039; 16043; 16068; 16085; 16088; 16117; 16121; 16135
sci.electronics	20	54157; 54160; 54165; 54175; 54176; 54212; 54224; 54244; 54248; 54255; 54265; 54302; 54305; 54306; 54310; 54325; 54337; 54353; 54489; 54490
sci.med	20	59424; 59434; 59435; 59456; 59458; 59460; 59477; 59478; 59480; 59497; 59559; 59575; 59579; 59583; 59584; 59624; 59627; 59632; 59633; 59652
sci.space	20	61401; 61404; 61431; 61440; 61450; 61455; 61459; 61461; 61484; 61505; 61532; 61534; 61546; 61558; 62319; 62398; 62408; 62428; 62477; 62480
soc.religion.christian	20	21621; 21648; 21658; 21663; 21672; 21696; 21698; 21699; 21702; 21708; 21709; 21754; 21761; 21773; 21777; 21784; 21788; 21799; 21800; 21804
talk.politics.guns	20	55036; 55060; 55063; 55068; 55073; 55080; 55106; 55115; 55116; 55123; 55231; 55239; 55249; 55260; 55264; 55278; 55468; 55470; 55484; 55489
talk.politics.mideast	20	76548; 76557; 77177; 77203; 77212; 77218; 77232; 77235; 77250; 77272; 77275; 77288; 77305; 77322; 77332; 77383; 77387; 77392; 77813; 77815
talk.politics.misc	20	178870; 178887; 178906; 178907; 178924; 178927; 178939; 178945; 178960; 178965; 178993; 178994; 178997; 178998; 179018; 179066; 179067; 179070; 179095; 179097
talk.religion.misc	20	84278; 84290; 84293; 84302; 84309; 84316; 84317; 84324; 84345; 84350; 84351; 84355; 84356; 84359; 84398; 84400; 84413; 84436; 84510; 84567

Table A.3: Minimized classicdocs dataset (mini-classicdocs).

Cluster Name	Number of Documents	List of Documents
Class1	45	160; 190; 194; 201; 203; 206; 207; 210; 211; 213; 215; 216; 219; 221; 222; 223; 230; 235; 255; 274; 286; 289; 290; 291; 314; 340; 341; 352; 361; 366; 376; 377; 380; 381; 382; 383; 384; 385; 386; 387; 388; 391; 392; 395; 399
Class2	146	136; 20; 351; 401; 402; 403; 406; 407; 408; 409; 410; 415; 416; 417; 418; 420; 421; 422; 423; 424; 425; 426; 427; 432; 433; 434; 435; 436; 437; 438; 439; 44; 440; 442; 443; 444; 447; 449; 452; 453; 454; 455; 456; 457; 458; 460; 461; 462; 463; 464; 468; 469; 470; 471; 472; 473; 475; 478; 479; 480; 483; 484; 485; 486; 487; 488; 492; 493; 495; 496; 497; 502; 503; 504; 505; 506; 508; 509; 511; 513; 514; 515; 516; 517; 518; 520; 522; 523; 524; 525; 527; 528; 530; 531; 532; 533; 534; 537; 538; 539; 540; 542; 544; 545; 546; 547; 548; 549; 55; 551; 554; 555; 556; 559; 560; 564; 565; 566; 568; 569; 570; 572; 573; 574; 575; 576; 577; 578; 579; 581; 582; 583; 585; 586; 587; 588; 589; 590; 591; 592; 595; 596; 597; 598; 599; 761
Class3	70	552; 615; 616; 617; 618; 622; 623; 624; 625; 626; 627; 628; 629; 630; 631; 632; 633; 634; 635; 639; 648; 652; 655; 659; 661; 662; 665; 674; 676; 681; 685; 686; 691; 694; 696; 697; 698; 699; 700; 701; 702; 703; 704; 705; 706; 709; 710; 711; 712; 713; 714; 715; 716; 717; 718; 720; 721; 738; 739; 740; 747; 748; 749; 750; 756; 788; 790; 791; 792; 799
Class4	69	10; 100; 101; 108; 11; 110; 111; 112; 113; 117; 119; 124; 125; 13; 130; 135; 137; 14; 140; 141; 148; 153; 154; 16; 161; 162; 163; 164; 166; 169; 173; 176; 178; 179; 180; 181; 183; 185; 186; 19; 191; 192; 195; 2; 25; 26; 30; 32; 35; 37; 42; 47; 50; 52; 60; 61; 63; 64; 65; 72; 73; 76; 8; 80; 86; 88; 90; 91; 99
Class5	75	115; 168; 200; 202; 208; 214; 217; 220; 228; 231; 232; 236; 238; 239; 240; 245; 246; 247; 259; 262; 275; 284; 287; 288; 292; 293; 294; 295; 296; 297; 298; 299; 300; 301; 302; 303; 304; 305; 306; 307; 308; 309; 310; 311; 312; 317; 321; 329; 330; 331; 332; 335; 336; 337; 338; 342; 346; 347; 353; 354; 359; 360; 362; 365; 368; 369; 370; 371; 372; 397; 398; 445; 48; 7; 77
Class6	86	1; 102; 107; 118; 12; 120; 122; 123; 126; 127; 128; 129; 131; 132; 133; 134; 138; 139; 142; 144; 145; 146; 147; 149; 150; 151; 156; 157; 158; 159; 165; 167; 170; 171; 172; 174; 177; 18; 184; 187; 188; 193; 197; 198; 199; 204; 21; 22; 244; 28; 29; 3; 31; 323; 33; 34; 343; 36; 39; 390; 40; 41; 43; 45; 46; 49; 491; 5; 526; 56; 57; 6; 62; 66; 67; 68; 70; 71; 78; 79; 82; 89; 9; 92; 95; 96
Class7	85	0; 104; 105; 114; 116; 121; 15; 155; 175; 182; 218; 23; 27; 38; 4; 400; 404; 405; 411; 412; 413; 414; 419; 428; 429; 430; 431; 441; 446; 448; 450; 451; 459; 465; 466; 467; 474; 476; 477; 481; 489; 490; 494; 498; 499; 500; 501; 507; 51; 510; 512; 519; 521; 53; 535; 536; 54; 541; 543; 550; 553; 557; 558; 561; 562; 563; 567; 571; 580; 584; 593; 594; 687; 69; 726; 727; 728; 74; 760; 81; 84; 87; 93; 94; 98
Class8	52	106; 610; 614; 620; 621; 646; 651; 653; 654; 657; 658; 660; 666; 668; 669; 670; 672; 677; 690; 692; 693; 695; 707; 722; 729; 736; 742; 743; 744; 746; 751; 753; 754; 755; 759; 762; 765; 771; 772; 774; 775; 776; 777; 778; 784; 789; 793; 794; 795; 796; 797; 798
Class9	98	103; 109; 143; 152; 17; 189; 196; 205; 209; 212; 224; 225; 226; 227; 229; 233; 234; 237; 24; 241; 242; 243; 248; 249; 250; 251; 252; 253; 254; 256; 257; 258; 260; 261; 263; 264; 265; 266; 267; 268; 269; 270; 271; 272; 273; 276; 277; 278; 279; 280; 281; 282; 283; 285; 313; 315; 316; 318; 319; 320; 322; 324; 325; 326; 327; 328; 333; 334; 339; 344; 345; 348; 349; 350; 355; 356; 357; 358; 363; 364; 367; 373; 374; 375; 378; 379; 389; 393; 394; 396; 529; 58; 59; 619; 708; 83; 85; 97
Class10	74	482; 600; 601; 602; 603; 604; 605; 606; 607; 608; 609; 611; 612; 613; 636; 637; 638; 640; 641; 642; 643; 644; 645; 647; 649; 650; 656; 663; 664; 667; 671; 673; 675; 678; 679; 680; 682; 683; 684; 688; 689; 719; 723; 724; 725; 730; 731; 732; 733; 734; 735; 737; 741; 745; 75; 752; 757; 758; 763; 764; 766; 767; 768; 769; 770; 773; 779; 780; 781; 782; 783; 785; 786; 787

APPENDIX A.2

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

1. **Google list stopwords:** 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.

2. **Short list 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.

3. **Long list 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.

CURRICULUM VITAE

Name Surname: Mohanad Dawoud

Place and Date of Birth: Palestine, 27.09.1984

Adress: Istanbul Technical University, Istanbul, 34469, TURKEY

E-Mail: dawoud@itu.edu.tr

B.Sc.: 2001-2006, B.Sc. in Computer Engineering, Islamic University of Gaza, Gaza-Palestine. Project: Online Video Compression and Transmission.

M.Sc.: 2006-2008, M.Sc. in Computer Engineering, Arab Academy for Science, Technology & Maritime Transport, Alexandria-Egypt. Thesis: Effects of Features Transformation and Organization on Image Retrieval.

PUBLICATIONS/PRESENTATIONS ON THE THESIS

- **Dawoud M.**, Altilar D.T., 2016: HEADA: A low cost RFID authentication technique using homomorphic encryption for key generation. *Security and Communication Networks*, August 2016.
- **Dawoud M.**, Altilar D.T., 2016: Privacy-Preserving Data Retrieval Using Anonymous Query Authentication In Data Cloud Services. *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, April 23-25, 2016 Rome, Italy.
- **Dawoud M.**, Altilar D.T., 2014: Privacy-Preserving Search in Data Clouds Using Normalized Homomorphic Encryption. *Euro-Par 2014 International Workshops*, August 25-26, 2014 Porto, Portugal.