Privacy Preserving Search in Large Encrypted Databases

Shahzaib Tahir

This dissertation is submitted for the degree of

Doctor of Philosophy

School of Mathematics, Computer Science and Engineering

City, University of London

November 2018

Dedicated to Papa and Mama

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Shahzaib Tahir

November 2018

Acknowledgements

First and foremost, I would like to thank Allah Almighty for giving me the strength to carry out this research.

My deepest gratitude to my supervisor Prof. Muttukrishnan Rajarajan for his continuous guidance and support during my Ph.D. study. I cannot imagine having a better supervisor and mentor than him. I am also thankful to my co-supervisor Nikos Komninos. A heartfelt thank you to my collaborators Sushmita Ruj and Yogachandran Rahulamathavan for their valuable feedback and thought provoking questions.

I am grateful to City, University of London, UK and National University of Sciences and Technology (NUST), Islamabad, Pakistan for sponsoring my Ph.D. and making it possible for me to study here. I would like to thank Baber Aslam from NUST for his continuous encouragement during my Ph.D.

I would like to thank Ali Sajjad from British Telecommunications for providing support and guidance with the deployment of this work on the BT Cloud offering. I would also like to thank Cornelius Glackin from Intelligent Voice for providing the datasets that have been used in the development and testing of this work. Completion of this Ph.D. would not have been possible without their help. I also thank my fellow lab mates Waqar Asif, Prof. Bruno Bogaz Zarpelão and Anna Hajduk for being sincere colleagues.

I am thankful to the IT and the administrative staff at City, University of London, especially Ms. Paula Green for looking after the needs of the Ph.D. students inside the lab. I am also thankful to the staff at the Finance Directorate and the HR Directorate at NUST for the timely provision of the scholarship every month during my Ph.D.

I am also thankful to InnovateUK and all the people involved within the Cyber Security Academic Startups Programme for providing the funding and giving me the opportunity to network with people from the world's leading institutes, organizations and companies.

I am thankful to all the reviewers who had reviewed my publications and gave their valuable feedback. Their comments helped to significantly improve the quality of my research.

On a personal note, I must also extend my respect to my parents, who supported me emotionally and have been a source of guidance throughout my life. I cannot imagine a life without their love and blessings. I am also thankful to my sister and brother for their guidance, encouragement and prayers. I consider myself the luckiest person in this world to have such a caring, loving and supportive family. I thank you all for showing faith in me. Finally, I must mention my nephew: Zaim and nieces: Eesha and Anabia who's smiles have been a source of joy for me even when I was stressed.

List of Publications

Journals

[J1] **S. Tahir**, S. Ruj, Y. Rahulamathavan, M. Rajarajan, C. Glackin, “A New Secure and Lightweight Searchable Encryption Scheme over Encrypted Cloud Data”, IEEE Transactions on Emerging Topics in Computing, 2017. (Impact Factor: 3.826)

[J2] **S. Tahir**, L. Steponkus, S. Ruj, M. Rajarajan, A. Sajjad, “A Parallelized Disjunctive Query based Searchable Encryption Scheme for Big Data”, Elsevier Future Generation Computer Systems, 2018. (Impact Factor: 3.997)

[J3] **S. Tahir**, S. Ruj, A. Sajjad, M. Rajarajan, “Fuzzy Keywords enabled Ranked Searchable Encryption Scheme for a Public Cloud Environment”, Elsevier Computer Communications, 2018. (Impact Factor: 2.613)

[J4] **S. Tahir**, H. Tahir, A. Sajjad, R. Tahir, M. Rajarajan, “Privacy-Preserving Healthcare Framework using Permissioned Blockchain”, Elsevier Journal of Biomedical Informatics, Under Revision. (Impact Factor: 2.882)

Conferences

[C1] **S. Tahir**, M. Rajarajan, A. Sajjad, "A ranked searchable encryption scheme for encrypted data hosted on the Public Cloud", The 31st IEEE International Conference on Information Networking (ICOIN), Vietnam, 2017.

[C2] **S. Tahir**, S. Ruj, M. Rajarajan, "An Efficient Disjunctive Query Enabled Ranked Searchable Encryption Scheme", The 16th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (IEEE TrustCom), Australia, 2017.

[C3] C. Glackin, G. Chollet, N. Dugan, N. Cannings, J. Wall, **S. Tahir**, I. Ray, M. Rajarajan, "Privacy preserving encrypted phonetic search of speech data", The 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), USA, 2017.

[C4] **S. Tahir**, M. Rajarajan, "Privacy-Preserving Searchable Encryption Framework for Permissioned Blockchain Networks", The IEEE International Conference on Blockchain (Blockchain-2018), Canada, 2018.

[C5] W. Asif, I. Ray, **S. Tahir**, M. Rajarajan, "Privacy-preserving Anonymization with Restricted Search (PARS) on Social Network Data for Criminal Investigations", 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Korea, 2018.

List of Successful Grants

[G1] Innovate UK: Cyber security academic startups programme Year2-Phase 1 (16,000£)

[G2] Innovate UK: Cyber security academic startups programme Year2-Phase 2 (22,000£)

[G3] Innovate UK: Cyber security academic startups programme Year2-Phase 3 (100,000£)

[G4] City, University of London: Proof of Concept (PoC) Fund - Enterprise Show-case (25,000£)

Abstract

The Cloud is an environment designed for the provision of on-demand resource sharing and data access to remotely located clients and devices. Once data is outsourced to the Cloud, clients tend to lose control of their data thus becoming susceptible to data theft. To mitigate/ reduce the chances of data theft, Cloud service providers employ methods like encrypting data prior to outsourcing it to the Cloud. Although this increases security, it also gives rise to the challenge of searching and sifting through the large amounts of encrypted documents present in the Cloud.

This thesis proposes a comprehensive framework that provides Searchable Encryption-as-a-Service (SEaaS) by enabling clients to search for keyword(s) over the encrypted data stored in the Cloud. Searchable Encryption (SE) is a methodology based on recognized cryptographic primitives to enable a client to search over the encrypted Cloud data. This research makes five major contributions to the field of Searchable Encryption:

The first contribution is that the thesis proposes novel index-based SE schemes that increase the query effectiveness while being lightweight. To increase query effectiveness this thesis presents schemes that facilitate single-keyword, parallelized disjunctive-keyword (multi-keyword) and fuzzy-keyword searches.

The second contribution of this research is the incorporation of probabilistic trapdoors in all the proposed schemes. Probabilistic trapdoors enable the client to hide the search pattern even when the same keyword is searched repeatedly. Hence,

this quality allows the client to resist distinguishability attacks and prevents attackers from inferring the search pattern.

The third contribution is the enumeration of a “Privacy-preserving” SE scheme by presenting new definitions for SE; *i.e.*, keyword-trapdoor indistinguishability and trapdoor index indistinguishability. The existing security definitions proposed for SE did not take into account the incorporation of probabilistic trapdoors hence they were not readily applicable to our proposed schemes; hence new definitions have been studied.

The fourth contribution is the validation that the proposed index-based SE schemes are efficient and can be deployed on to the real-world Cloud offering. The proposed schemes have been implemented and proof-of-concept prototypes have been deployed onto the British Telecommunication’s Cloud Server (BTCS). Once deployed onto the BTCS the proof-of-concept prototypes have been tested over a large real-world speech corpus.

The fifth contribution of the thesis is the study of a novel homomorphic SE scheme based on probabilistic trapdoors for the provision of higher level of security and privacy. The proposed scheme is constructed on a Partially Homomorphic Encryption Scheme that is lightweight when compared to existing Fully Homomorphic-based SE schemes. The scheme also provides non-repudiation of the transmitted trapdoor while eliminating the need for a centralized data structure, thereby facilitating scalability across Cross-Cloud platforms.

Table of contents

List of figures

List of tables

Nomenclature

Notations / Variables

$\lvert\mathscr{W}\rvert$	cardinality of the set $\mathscr{W}$
$b \overset{\$}{\leftarrow} \{0,1\}$	sample a random element of {0,1} into b independently
(k_{pub}, k_{pri})	asymmetric key pairs
$*$	positioning of a letter within a keyword
$\cap$	intersection of sets
$\cup$	union of sets
Δ	dictionary of keywords
λ	security parameter
$\mathscr{A}$	a set of polynomial time adversaries $\mathscr{A} = \{A_1, A_2, \cdots\}$
$\mathscr{B}$	polynomial time adversary
$\mathscr{D}$	a set of documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$
$\mathscr{Q}$	polynomial time distinguisher
$\mathscr{W}$	a set of keywords $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$
$\sim$	nearly similar but not identical

$\star$	a mathematical operation such as addition or multiplication
$\subset$	a proper subset
$a \leftarrow b$	a contains the value of b
$H(.)$	cryptographic hash function
$id(D)$	document identifier corresponding to the document D
K	master key
k_s	session key
O	Big O notation representing upper bound complexity
$P(\mathscr{W})$	power set of the set $\mathscr{W}$
$st_{\mathscr{A}}$	state of adversary
T_W	trapdoor generated for the keyword W
Boldfaced letters	represent row vectors

Acronyms / Abbreviations

FI	Fuzzy Index Table
I	Index table
AE	Application Encryption
AES	Advanced Encryption Standard
AES-CBC	Advanced Encryption Standard-Cipher Block Chaining
AES-ECB	Advanced Encryption Standard-Electronic Codebook
BT	British Telecommunication

BTCS	British Telecommunication’s Cloud Service
CAGR	Compound Annual Growth Rate
CPA	Chosen Plaintext Attack
CS	Cloud Server
CSP	Cloud Service Provider
CSPRNG	Cryptographically Secure Pseudo Random Number Generator
DaaS	Database-as-a-Service
FHE	Fully Homomorphic Encryption
FIPS	Federal Information Processing Standard
FRSE	Fuzzy Ranked Searchable Encryption
GCD	Greatest Common Divisor
GRSE	Geometric Range Searchable Encryption
HE	Homomorphic Encryption
HSE	Homomorphic-based Searchable Encryption
IND-CKA	Semantic Security Against Adaptive Chosen Keyword Attack
KDM	Key Distribution Manager
KMS	Key Management Service
LEA	Law Enforcement Agency
M/M	Multi Writer/ Multi Reader
M/S	Multi Writer/ Single Reader

MPSE	Multi-party Searchable Encryption
MRSE	Multi-keyword Ranked Searchable Encryption
MT	Multi-thread
OC	Outcome
OPE	Order Preserving Encryption
PDP	Provable of data possession
PHE	Partially Homomorphic Encryption
PKE	Public Key Encryption
PoC	Proof-of-Concept
POD	Proof of Deletion
RF	Relevance Frequency
RMSE	Ranked Multi-keyword Searchable Encryption
RSA	Ron Rivest, Adi Shamir and Leonard Adleman Algorithm
RSE	Ranked Searchable Encryption
S/M	Single Writer/ Multi Reader
S/S	Single Writer/ Single Reader
SE	Searchable Encryption
SEaaS	Searchable Encryption-as-a-Service
SHA	Secure Hash Algorithm
SO	Search Outcome

SSE	Searchable Symmetric Encryption
ST	Single-thread
SWHE	Somewhat Homomorphic Encryption

Chapter 1

Introduction

Over the past decade, cloud has emerged as a large scale "Storage-as-a-Service" [156][71][112] platform that provides on-demand resource sharing and data access to fulfill the needs of its users. The cloud storage adds business value to the existing small scale enterprises through the enormous benefits it has to offer; including accessibility, reduced costs, disaster recovery, scalability, speed and storage immortality. Generally, small scale enterprises have limited resources and limited expertise which drives them to rely on the cloud for their operations, and benefit from the advantages the cloud offers. However, the cloud also poses certain challenges to the small scale enterprises and their clients in terms of data theft and data misuse. These concerns prevent clients from fully trusting the cloud and storing their sensitive data into it, leaving many industries disadvantaged; such as, the healthcare sector, financial sector, and the Internet of Things (IoT).

Searchable Encryption (SE) is a technology that allows enterprises and individuals to store encrypted data into the cloud while allowing them to search over it. SE gives clients full control over their data stored into the cloud. This ever growing usage of the cloud and the rapid growth in file sharing over the cloud has created a necessity for efforts towards designing SE schemes. A recent survey [1] estimates the unique monthly visitors to the top 6 most popular file sharing websites during

July, 2017. From the Figure 1.1, one may anticipate the amount of data that the clients are outsourcing to the cloud. It is indeed surprising to see that DropBox alone has experienced 35 million clients accessing its cloud servers. SE is imperative for the security and privacy of the clients, so that the data storage and any transactions (search queries) thereafter may be carried out in a secure manner.

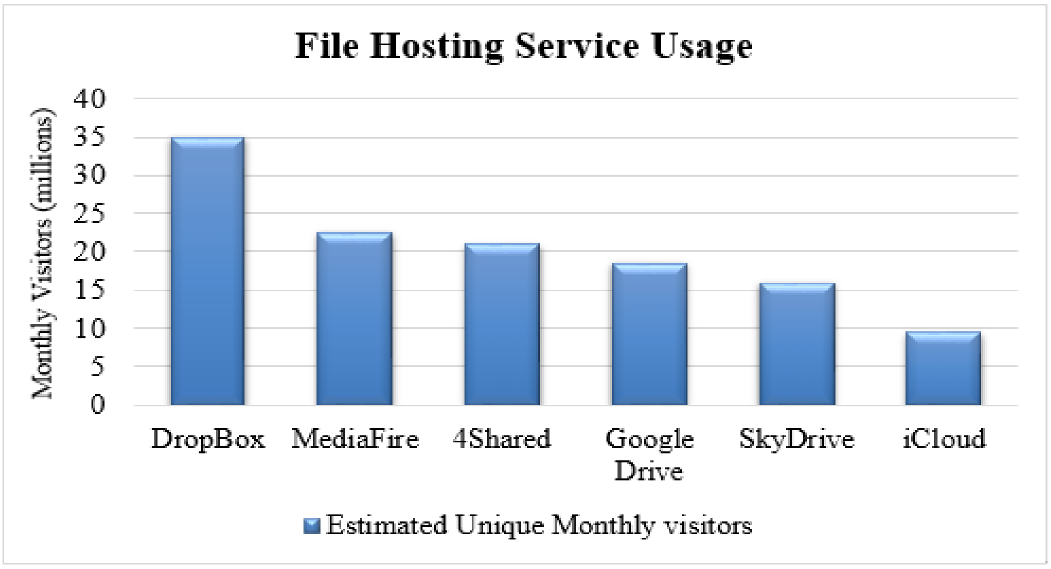

Fig. 1.1 File sharing website usage.

1.1 Research Motivation

According to Forrester, the global public cloud market will be $178B in 2018, up from $146B in 2017, and will continue to grow at a 22% compound annual growth rate (CAGR) [44]. Despite the increased interest and demand, 75% of enterprises have highlighted security concerns related to cloud computing. In addition, 60% of enterprises have highlighted data protection concerns [2]. Furthermore, since 2017, 2.6 billion data records have been compromised worldwide which means every second, 82 data records are stolen [3]. These concerns prevent people from benefiting from the resource sharing the cloud offers and keeps them from outsourcing their private and confidential data to the cloud.

Even though the sharing of resources has many advantages, several confidentiality and privacy concerns prevail. Therefore, the clients may be interested to encrypt the documents prior to outsourcing them to the cloud. Although encryption seems to be a very straightforward solution, the conventional encrypt-then-outsource data model cannot be applied directly, because to search for a keyword within a document conventionally, the client has to download and decrypt all the documents that are outsourced to the cloud and then search for the relevant one. That is certainly not a feasible solution for huge datasets. To address this problem, a SE scheme is required that allows the cloud to search over the encrypted data on the client's behalf. A trapdoor is a search token that is generated to search for keyword(s) over the encrypted data. Only an authorized individual (in possession of the correct credentials) should be able to generate meaningful and correct trapdoors. A SE scheme provides data confidentiality and preserves the privacy of the client and his trapdoors respectively.

1.1.1 Challenges in Searchable Encryption

There are three main challenges associated with SE as discussed in [22] *(a)* security and privacy *(b)* efficiency and *(c)* query effectiveness. Security and privacy is given utmost importance and is aimed at limiting the amount of information that an adversary or a cloud server (CS) can learn from the encrypted documents stored in the cloud or by implementing the SE scheme itself. The enhancement of security and privacy may be termed as a primary objective of the designed SE schemes. However, efficiency and query effectiveness are not of primary consideration. Query effectiveness is the degree to which a user can search for complex queries, for example, a multi-keyword query can be termed more expressive as compared to a single-keyword query. Similarly, a ranked SE scheme is more expressive and effective as compared to an unranked SE scheme. The efficiency of a SE scheme is directly dependent upon the underlying design primitives of the scheme and the types of queries that are allowed (discussed in the Section 2.2). For example, there are primarily two main approaches for performing SE, *i.e.,* by using a specialized encryption scheme such as homomorphic SE [113][60][13][59] that allows to search directly over the encrypted documents or design an index-based SE scheme [46][132][130][144] that maps the encrypted documents to a secure index.

In [110], a comparison of fully homomorphic SE schemes and index-based SE schemes has been conducted, and it is evident that homomorphic encryption is inefficient therefore homomorphic based SE is not preferred for practical purposes. Considering the index-based constructions, until now, several schemes [47][147][46][78] have been designed and proposed that are unable to maintain a balance between the aforementioned challenges *i.e.,* security and privacy, efficiency and query effectiveness. For that reason, they cannot be deployed onto the existing cloud infrastructure. Hence, while designing SE schemes (whether HE-based or index-based), a balance between the level of security and privacy, efficiency, and query effectiveness needs to be maintained.

1.1.2 Security Aims

As mentioned in the previous section, security and privacy are the primary objectives that are to be achieved through this research. It is important to list the security goals and design a scheme accordingly. Security and privacy work in harmony so defining the security goals will lead to highlighting the associated privacy concerns. The previous discussion leads to the following security goals:

- Authorized Search: only an authorized individual (in possession of the correct credentials) should be able to generate meaningful and correct trapdoors.

- Data Confidentiality: the data outsourced to the cloud server should be stored in the encrypted format and the data stored in the index table should also be secure.

1.1.3 Privacy Concerns in Searchable Encryption

To throw light on the privacy concerns, we consider two threat models which are widely used in the existing schemes [146][28][82]:

- Known Cipher-text Model: the CS is only given access to the data that the client outsources, such as, encrypted set of documents, secure index table, and trapdoors. The CS is also allowed to maintain a history of the trapdoors and the search outcome.

- Known Background Model: apart from the information that the CS has in the known cipher-text model, the CS also has some additional information about the datasets. The CS also knows the nature of the documents and the frequency of the encrypted keywords with which they appear within a document.

The privacy of the document's content is achieved by encrypting the documents before outsourcing them to the CS. Therefore, the concern here is to study privacy violations caused as a result of implementing an SE scheme. Hence, the above mentioned threat models lead to the following privacy related concerns:

- Keyword Privacy: apart from the outcome of the search, the CS should not deduce any keyword related information from the secure index and trapdoors. This requires the SE scheme to be designed in such a way that the leakage profile may not indicate leakage of any meaningful information.

- Trapdoor Unlinkability: the CS should not be able to link the trapdoor to the previous queries or index table 'prior' to the search. This requires a probabilistic trapdoor that results in the generation of a different trapdoor for the same set of keywords searched again.

1.1.4 Applications of Searchable Encryption

SE has both generic and specific applications on the cloud. SE can be applied to many domains like healthcare where searching can be performed over encrypted medical records [14]. In [165] authors have explored the use of SE in performing secure search over the genomic signatures placed on the public CS. SE has also been considered in IoT for accomplishing encrypted queries. Extending this concept, in [88], the authors explore the use of SE in performing search while preserving the privacy of connected cars. In [19], SE has been carried out over the email servers to secure the emails from being disclosed to unauthorized individuals. Apart from these domains SE could have a profound impact on e-commerce to hide confidential transactions, business intelligence and data science.

1.2 Thesis Statement

The cloud is being used for storing large amounts of data seeking on-demand availability and remote access. Commonly the data stored on the cloud is unencrypted, moreover once an individual outsources the data into the cloud he/she loses control of the data. Thereafter, the data may be used without prior permission. This makes individuals prone to privacy breaches, resulting in lack of trust in the cloud. Applying encryption prior to outsourcing the data to the cloud can help achieve confidentiality of the data, but this comes with an overhead of downloading all the documents and decrypting them every time an individual wants to search for a keyword(s).

SE allows to perform search over the encrypted documents. Conventional SE schemes [78][47][151] are based on deterministic trapdoors (search query), *i.e.*, for the same keyword searched again the same trapdoor is generated. Therefore, the server is able to learn the search pattern. From this the behaviors of an individual or a group of people can be inferred and could lead to passive attacks (passive attacks are discussed in the Section 3.1.4). This gives rise to the need of SE schemes that are based on probabilistic trapdoors, *i.e.*, for the same keywords searched repeatedly a unique trapdoor should be generated every time. In this way the trapdoor is unlinkable and it prevents distinguishability attacks. This also preserves the privacy of the user by preventing the CS from learning the underlying keyword being searched.

This research presents a comprehensive framework that provides Searchable Encryption-as-a-Service (SEaaS) [135] to the individuals outsourcing their private data into the cloud. The aim of this research is to build the client's trust onto the cloud so that the clients may confidently outsource their confidential data into the cloud. The benefits are two-fold; firstly, by presenting index-based SE schemes that enhance the query effectiveness while trying to attain high levels of security and privacy, and efficiency. Secondly, by proposing a homomorphic-based searchable encrpytion scheme to achieve greater levels of security and privacy while allowing scalability across cross-cloud/nested cloud platforms [15][66]. This research incorporates probabilistic trapdoors in the proposed SE schemes to enhance the privacy-preserving property to prevent deterministic attacks. This research discusses the limitations of the existing security definitions and the existing literature. Following this discussion new security definitions for SE are proposed that highlight the "*privacy preserving*" property of the SE schemes. The security of the framework is analyzed in coherence with the new security definitions. This research studies the feasibility of the presented framework over a real-world cloud platform and real dataset of encrypted documents.

1.3 Thesis Contributions

This thesis explores a set of domains related to the field of SE. This research provides a privacy-preserving SE framework which can be adapted by an existing cloud service Provider (CSP). Conventional SE schemes (discussed in the Chapter 2) have an inherent weakness that they leak the search patterns and the access patterns. The disclosure of the search patterns leads to reduced privacy and successful launch of distinguishability attacks as the adversary may carry out passive attacks. To deal with these inherent problems this thesis makes five contributions to the field of SE. The contributions are aimed towards enhancing the privacy-preserving property offered by a SE framework while enhancing the query effectiveness. The research presented in this thesis has been published in [132], [130], [135], [133], [134], [8], [61] and [129]. The following original contributions are made in this thesis:

i. Index-based SE schemes that enhance the query effectiveness:

This thesis presents novel index-based SE schemes to enhance the query effectiveness. These schemes primarily facilitate single-keyword, parallelized disjunctive-keyword and fuzzy keyword searching. The increase in query effectiveness effects the security and privacy, and efficiency. Hence, this research aims to propose schemes that maintain a balance between the query effectiveness, security and privacy and efficiency by presenting lightweight schemes as compared to the state-of-the-art.

ii. Probabilistic trapdoors for enhancing the privacy of the proposed schemes:

The second, yet most important contribution of this thesis is the provision of probabilistic trapdoors. Therefore, all the schemes are based on probabilistic trapdoors that prevent successful distinguishability attacks. This research shows that the proposed schemes can resist network attacks in which a system may be scanned or observed.

iii. New security definitions "keyword-trapdoor indistinguishability" and "trapdoor-index indistinguishability" for SE:

The third contribution of this thesis is establishing the relationship between privacy-preservation and probabilistic trapdoors. Conventional SE schemes are based on deterministic trapdoors, therefore, the existing security definitions do not take the benefits of having probabilistic trapdoors into consideration. To appreciate this, new security definitions are proposed that focus entirely on indistinguishability. The definitions include keyword-trapdoor indistinguishability and trapdoor-index indistinguishability that lay down the foundation towards proving the privacy-preserving property of the proposed schemes.

iv. Development of proof of concept prototypes, deploying on a public cloud and testing over a real-world dataset:

The fourth paramount contribution is the design and development of the proof-of-concept prototypes. This research justifies the feasibility of the proposed index-based schemes by deploying them to the British Telecommunication's public cloud offering. The proposed schemes are deployed on the British Telecommunications cloud server (BTCS) and tested over a real-world corpus of encrypted documents.

v. A novel privacy-preserving Homomorphic-based SE scheme scalable across cross-cloud platforms:

The fifth contribution of this research is to present a privacy-preserving homomorphic based SE scheme. This eliminates the need of a pre-processed index-table and supports scalability and dispersion of the data across cross-cloud platforms. The database can also be dynamic in this scheme. The scheme utilizes the partial homomorphic property of RSA to perform the search directly over the encrypted documents. The scheme is based on probabilistic trapdoors and provides non-repudiation related to the trapdoors. To demonstrate the effectiveness of the proposed scheme, the proof-of-concept prototype is developed and tested over encrypted documents. Thus the homomorphic-based SE scheme can be used in scenarios where the security is critical as compared to efficiency.

1.4 Thesis Structure

In summary, this thesis presents individual SE schemes that collectively form a SE framework. The thesis breakdown is as follows:

- **Chapter 2: Literature Review** discusses the existing literature on SE. The chapter begins by discussing the design primitives. The existing security definitions and their limitations are analyzed. Then the existing single-keyword, multi-keyword, fuzzy keyword and homomorphic SE schemes are discussed. The literature review discusses the pros and cons of the existing schemes.

- **Chapter 3: Ranked Single Keyword Searchable Encryption Scheme** introduces a novel ranked single keyword SE scheme. The proposed scheme is based on probabilistic trapdoors and resists distinguishability attacks. To highlight the advantage of probabilistic trapdoors, the chapter proposes new security definitions; keyword-trapdoor indistinguishability and trapdoor-index indistinguishability. The algorithmic analysis is presented to discuss the feasibility of the scheme. This also includes asymptotic analysis of the proposed scheme against a few schemes presented in the literature review. The proof-of-concept prototype is implemented and deployed on the British Telecommunications cloud offering and tested over a real-world dataset.

- **Chapter 4: Parallelized Disjunctive Query Searchable Encryption Scheme** extends the scheme presented in the chapter 3 to perform disjunctive keyword search. The proposed security definitions are extended to apply onto the proposed scheme and the formal security proofs are presented. To benefit from multi-core processors, the scheme is deployed onto the BT cloud offering and parallel searching is performed. This performance analysis is accompanied with the algorithmic analysis and storage overhead analysis.

- **Chapter 5: Fuzzy Ranked Searchable Encryption Scheme** is geared towards presenting a scheme that takes human typographical errors or closely related keywords into account. This chapter explores the concept of Shingling, Min hashing, Jaccard Similarity and Euclidean Norm to achieve the search. The scheme is tested over a real-world encrypted corpus and deployed onto the BT cloud offering. The correctness of the scheme is also discussed. The chapter includes a comprehensive security and performance analysis to discuss its feasibility in real-world applications.

- **Chapter 6: Homomorphic-based Searchable Encryption Scheme** presents a scheme that explores the partial homomorphic property of RSA. The scheme for the first time introduces probabilistic trapdoors to the standard RSA, hence preventing from distinguishability attacks. The discussion includes the security definitions, formal security proofs and performance analysis. The proof-of-concept prototype is tested over a real-world speech corpus and the computational complexity is analyzed.

- **Chapter 7: Conclusions and Future Directions** concludes the thesis by discussing the directions that can be explored in the future research.

Chapter 2

Literature Review

SE refers to a methodology that enables clients to perform search over the encrypted data stored in the cloud while preserving the privacy. Privacy preservation is a characteristic that reduces the amount of data leaked to an adversary or a CS when the SE scheme is in effect. The previous chapter gave an overview of the proposed SE framework by discussing the security and privacy concerns related to the cloud storage. The chapter presented two different threat models; known cipher-text model and the known background model. The common privacy goals that became apparent from both the threat models included: keyword privacy and trapdoor unlinkability.

The design of a SE framework is influenced by the domain usability, the underlying use case and the associated cloud infrastructure. Hence, there are a number of design primitives that need to be discussed prior to designing a SE scheme. This chapter gives a detailed overview of the significant and pioneering existing works in the field of SE. As evident from the previous chapter, trapdoor unlinkability and keyword privacy are essential, due to this reason the existing security definitions are not applicable. This chapter also discusses the existing security definitions for SE and their limitations. This chapter provides a foundation for the remaining chapters and this discussion is directed towards designing a framework that would meet the security and privacy goals highlighted in the previous chapter. Before studying the existing literature and discussing the pros and cons of the existing schemes, it is important to understand the design primitives.

2.1 Triangle of Searchable Encryption

Over the past decade extensive research has been done on designing novel SE schemes. Our analysis shows that the feasibility of the existing schemes is judged on three perimeters [22]; *i.e.,* security and privacy, efficiency and query effectiveness. Security and privacy refer to measure the amount of information that the encrypted documents, trapdoors or index tables leak to the cloud server or adversary. The security and privacy are analyzed against a set of security models and definitions. As per the security definitions presented in Chapter 3, a SE scheme is secure and privacy preserving if it is based on probabilistic trapdoors. Probabilistic trapdoors help to mitigate the risk of search pattern leakage. A limitation of the existing SE schemes is highlighted in [78];

"A limitation of all known SSE constructions is that the tokens they generate are deterministic, in the sense that the same token will always be generated for the same keyword. This means that searches leak statistical information about the user's search pattern. Currently, it is not known how to design efficient SSE schemes with probabilistic trapdoors."

Efficiency is measured by performing the computational and algorithmic complexity analysis of the SE schemes. Query Effectiveness refers to the usability of the system and the types of queries that can be performed over the encrypted documents. Considering these perimeters as vertices of a triangle an ideal SE scheme should be able to preserve the equilateral property (all sides and angles are equal) of the triangle shown in the figure 2.1, which is very difficult to achieve. It should be realized that enhancing the query effectiveness may reduce the security and efficiency.

For example, index-based SE schemes can achieve ranked searching at the cost of leakage of statistics. Similarly, it is also possible that increased levels of security and privacy may decrease the efficiency or query effectiveness of the system. For example, homomorphic-based SE schemes enhance the security and privacy at the cost of increased computations and reduced efficiency. Therefore, these perimeters are directly related and dependent upon the design primitives presented in the next section and the selection of these primitives influence the security and privacy, efficiency and query effectiveness. This aspect is highlighted in more detail in the Section 2.4 while analyzing the existing schemes.

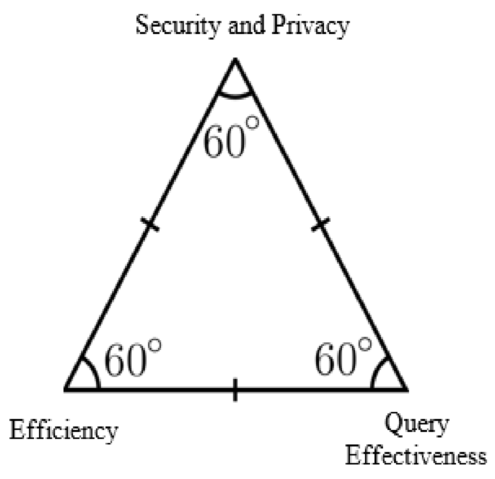

Fig. 2.1 Triangle of Searchable Encryption

2.2 Design Primitives

A SE framework is highly influenced by the domain usability [10] and the underlying cloud architecture. Domain usability refers to the area/domain where the scheme is to be deployed. For example, in the healthcare domain there may be many stakeholders involved that leads to the requirement of asymmetric keys. Similarly, there are certain requirements that arise by relying on a particular cloud server such as the use of an application encryption server (AE). Therefore, the following variations are studied by relating them to the underlying architecture and the associated challenges are identified. This discussion will lead us to formally outline our system model for the individual proposed schemes.

2.2.1 Symmetric versus Asymmetric Primitives

These primitives are related to the security parameters of the system and are dependent upon the underlying use case where the scheme is aimed to be deployed. These primitives identify whether the cryptosystem will be using a single key or multiple keys. A writer is a person who is the author/owner of the encrypted documents stored in the cloud, whereas, a reader is a client who is searching for the keywords over those documents. It is worth mentioning that a writer can also be a reader. Therefore, if the SE scheme is single writer/single reader (S/S) [123][64] then by using the symmetric primitives such as AES [90] based on a master key, the client can generate trapdoors and search over the cipher texts respectively. If the architecture is multi-writer/single-reader (M/S) [19][138], single-writer/multi-reader (S/M) [159][47] or multi-writer/multi-reader (M/M) [12][73] then asymmetric primitives are used. This may require a collection of multiple keys generated through public key encryption (PKE) [27][81] schemes because multiple users are taking part in the search and several keys are to be used and kept secure. This also requires a mechanism to generate, store, distribute and revoke the keys across the network. Hence, the key management is the responsibility of a key distribution manager (KDM) [124] or an application encryption(AE) Server [118] may be required.

2.2.2 Forward Index versus Inverted Index

The purpose of using an index is to speed up the search process and make the scheme efficient. As shown in Figure 2.2(a), a Forward index [79] also called a bloom filter [139][17], forms a searchable index of keywords against each document. Until 2006 the schemes developed were based on the use of the forward index. In 2006, Curtmola *et al.* [46] introduced a new concept based on the use of an inverted index that formed an index of documents corresponding to the keywords for developing the SE schemes [33]. Figure 2.2(b) represents an inverted index table where RF are the "relevance frequencies". The selection of index table is totally dependent on the underlying scheme but it is observed that searchable inverted index provides efficient searching as compared to the forward index [22] because the forward index requires a data structure corresponding to every document, whereas an inverted index is formed collectively while considering individual keywords. Therefore, the searching is more efficient while using an inverted index, however the forward index provides scalability.

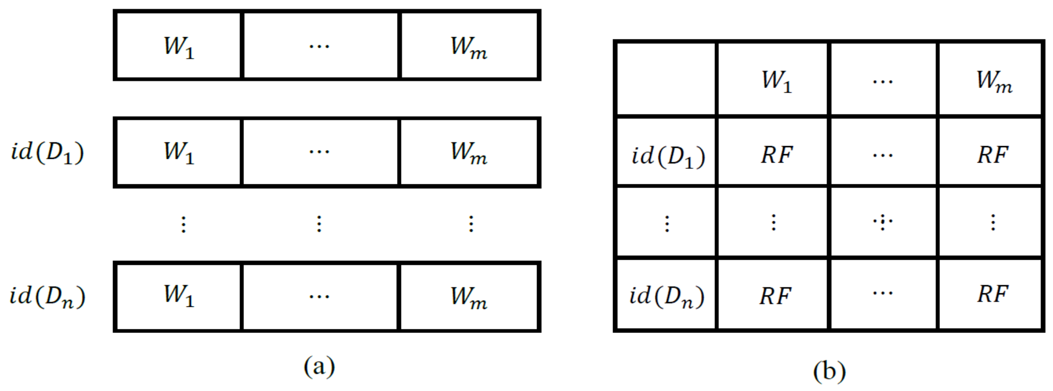

Fig. 2.2 Forward Index versus Inverted Index
Note: $W_1, \cdots, W_m$ represent the keywords, $id(D_1), \cdots, id(D_n)$ represent the document identifiers and RF represent the relevance frequencies.

2.2.3 Single Query versus Multiple Query

Single 1uery is referred to single keyword search [123][33][32][130], whereas, multiple queries refers to schemes that allow multiple-keyword searching [56][30] and complex queries such as fuzzy searching [142][83][49] or string searching [38][34][113]. Therefore, the query effectiveness or expressiveness is based on the type of query a client can perform. From the previously developed schemes it is noted that an increase in the query expressiveness affects the privacy and efficiency of the scheme. Hence the choice between the single query and multiple queries is to be made while maintaining a balance between the challenges already highlighted in the Section 2.1.

2.2.4 Ranked Searching versus Non-Ranked Searching

Ranked searching [164][72][167][130][132][51] facilitates the search by identifying the frequency of occurrence of a keyword within a set of documents and giving the user the liberty to select the most relevant documents from a collection. Whereas, non-ranked searching [157][153][58] returns all the documents to the user containing particular keyword(s). Ranked searching is mainly used for single keyword search because the server may find several files satisfying the query, whereas in complex queries, the server might be able to identify a few files meeting the search query. It is also observed that ranked searching is more resource consuming as compared to unranked searching [132]. Ranking is discussed in more detail in the Chapter 3.

2.2.5 Index-based SE versus Homomorphic-based SE

Index-based SE (forward index or inverted index) requires the preprocessing of the data to populate a data structure and generate an index table to carry out the searches in the future. On the contrary, a homomorphic-based SE scheme does not require a centralized data structure and eliminates the need of preprocessing of the data. A comparison of homomorphic and index-based SE schemes is presented in [110]. There are three types of homomorphic encryption schemes that depend upon the number of operations allowed on the encrypted text: *(1)* Partially Homomorphic Encryption (PHE) [94][119][48] allows one operation to be carried out unlimited number of times, *(2)* Somewhat Homomorphic Encryption (SWHE)

[13][59][158] allows a few operations to be carried out limited number of times, *(3)* Fully Homomorphic Encryption (FHE) [60][140][24] allows unlimited operations to be carried out unlimited number of times. Therefore, the computation complexity of homomorphic encryption schemes depend upon the mathematical operations that the scheme supports [4].

2.3 Existing Security Definitions

The problem of searching over encrypted data has received attention for more than a decade now. Back in 2000, Song *et al.* in [123] were the first to come up with a practical way of searching symmetrically over encrypted data. Till then there was no formal definition regarding security for SE. Since 2000 several definitions and constructions related to SE have been presented. In 2003, Goh [64] for the first time came up with the security definitions of SE called Semantic Security Against Adaptive Chosen Keyword Attack (IND-CKA). In the same paper, the author proposed a SE scheme that satisfied the proposed definition. There were some assumptions related to the definitions, *i.e.,* the number of keywords (size of the documents) within the document should be same in order to achieve indistinguishability and if the index is indistinguishable, the trapdoors need not to be kept secure. Since their definitions were focused towards secure indices and not probabilistic trapdoors, their definitions could not be generalized.

In [32], authors came up with an extension of IND-CKA that aimed to counter the assumption of same sized documents. They supported their definition by presenting a secure index construction called z-index which was based on bloom filters. As highlighted in [46], the definition was not secure and would be fulfilled by any insecure SE scheme. Later Goh [64] introduced extended definitions IND1/2-CKA and now the documents did not need to be of the same size, and the trapdoor was again not kept secure. Curtmola *et al.* in [46][47] claimed that all the previous definitions did not provide adequate security and proposed two new definitions Adaptive/Non-Adaptive Indistinguishability Security for SSE. Both of the newly proposed definitions have their weaknesses and don't provide an adequate level of indistinguishability. We discuss the limitation of their slightly stronger definition, *i.e.,* Adaptive Indistinguishability below.

2.3.1 Limitations of Previous Definitions

As mentioned earlier, Curtmola's definitions are widely accepted and used. They introduce four terms in [47][77] incurred as a result of a search query *i.e.,* History, Access Pattern, Search Pattern and Trace. The history defines a tuple containing the document collection and the keywords. Access patterns represents the outcome, *i.e.,* the documents that contain a particular keyword. The search pattern tells if the same keyword is being searched every time. The trace of a history consists of the exact information that we are willing to leak about a history after the search has been performed. Their security definition is defined as nothing is leaked beyond the access pattern and the search pattern while the trapdoor is deterministic. Their definition of Indistinguishability refers to the indistinguishable index table generated based on pseudo-random functions.

We remark that Curtmola's work clearly provides the desired level of security when the trapdoor is deterministic but their construction (SSE-2) lacks in maintaining privacy associated to the trapdoor and hence it is prone to distinguishability attacks. Their construction generates the same trapdoor (deterministic) every time the same keyword is queried. As a result the search pattern reveals the trapdoors that correspond to the same underlying keywords resulting in privacy concerns (*cf.* Section 4.2 of [47]). The deterministic trapdoor reveals the corresponding history tuple "prior" to the search. The drawbacks of having deterministic trapdoors are discussed in Section 3.1.4.

Hence, we term their definitions a primary “baseline” for any SE scheme but improved definitions are required for enhancing the security and highlighting the advantage of a probabilistic trapdoor under those improved definitions.

Therefore, based on the improved security definitions a secure construction is required that primarily provides a secure index table and ensures trapdoor indistinguishability that results in increased security and privacy of the entire system.

Now, we can state the privacy concerns associated to the existing SE schemes by extending the concerns highlighted in Section 1.1.3. So a SE scheme is privacy preserving if it has the following attributes:

- The trapdoor should not reveal any information about the keyword (unencrypted) that is being queried and should maintain the privacy of search.
- The trapdoor should be probabilistic and should not disclose the corresponding underlying encrypted keywords or document identifiers “prior” to the search.
- The outcome of the trapdoor should not uncover any information about the encrypted document that is returned as a result of the query to the user.

The next section analyzes the security and privacy of the existing SE schemes by relating them to the privacy concerns highlighted above and also discussed in the Section 1.1.3.

2.4 Related Work

This section highlights the significant works already carried out in the field of SE. This section takes a modular approach by dividing the schemes into different categories based on the query effectiveness and discussing their pros and cons thereafter.

2.4.1 Single-keyword SE schemes

Wang *et al.* in [148][147] for the first time introduced the concept of ranked keyword searching over encrypted data. The authors have proposed two schemes for single keyword search over encrypted text. Their scheme is an extension of [46] (already discussed in the previous section) and they have added secure ranking to it. Both the schemes facilitate the server to perform ranked keyword searching on user's behalf. In both the schemes, the user will generate the same trapdoor while searching for a particular file. Therefore, the schemes lack in providing resistance against distinguishability attacks. There is an advantage of their later scheme as it provides dynamic inverted index, *i.e.,* whenever a new file is added to the server, the re-ranking is not required but this comes at an increased computational cost. Furthermore, the later scheme helps to keep the ranking score encrypted that will help to avoid leakage of frequency of occurrence of a particular keyword to the server. However, in [84] the authors have launched a successful differential attack on the aforementioned scheme. The authors have demonstrated that the scheme still leaks the relevance scores to the adversary from which the encrypted keywords can be inferred by using the estimated distributions. Therefore, their scheme lacks in providing resistance against distinguishability attacks and hence leaks information.

Kamara *et al.* in [78] have proposed a dynamic searchable symmetric encryption scheme. Their work can be termed as an extension of their previous scheme that they had proposed in [46]. Their scheme facilitates the addition, deletion or modification of a document. The change is brought to the server at run time and comes with minimal modification and recompilation of the inverted index. For the deletion of the file they use an additional data structure that contains the pointers to the file being deleted. For the modification they use homomorphic encryption to encrypt the pointer so that based on the homomorphic encryption properties the server can modify the file. Though this can be termed as a breakthrough in the field of SE, there is a drawback of this scheme, *i.e.,* the generated trapdoor is deterministic and the same trapdoor is generated for the same word every time, hence, it cannot resist distinguishability attacks. Furthermore, they have also analyzed that their scheme leaks even more information as compared to their previous construction [46], *i.e.*, the CS is able to identify the exact frequency of the keywords and the their presence within a document. As a result the adversary can launch more sophisticated attacks. Therefore their construction cannot be termed as an ultimate solution.

Kamara and Papamanthou in [77] present a parallelizable and dynamic SE scheme. Similar to an inverted index table, authors introduce keyword red-black tree. The leaf nodes represent the document, whereas the internal nodes are the vectors associated to a keyword. The vector helps identify the number of documents containing the keyword. The tree traversal is highly parallelizable and supports dynamic databases. The authors benefit from the multi-core processors to achieve parallel processing. However, the authors say that;

"*A more serious limitation of known SSE constructions (including ours) is that the tokens they generate are deterministic, in the sense that the same token will always be generated for the same keyword.*"

Therefore, their scheme does not resist distinguishability attacks.

Wang *et al.* in [143] have proposed a range search scheme on encrypted spatial data. Their scheme, *i.e.,* Geometric Range Searchable Encryption (GRSE) supports searchable symmetric encryption by mapping the datasets to a set of points lying within a geometric shape. Their design is indeed remarkable as it is not dependent upon a particular geometric shape and supports axis-parallel rectangles, circles, non-axis-parallel rectangles and triangles. However, in this scheme all the data records within a dataset will be returned as the result and the user may have to download every file containing that particular keyword, hence, it will result in extra network traffic. Furthermore, with the increase in the outsourced data, the size of the bloom filter is increased that will result in the slowing down of the searching. They have also proposed an extension of their probabilistic GRSE by using trees to increase the efficiency of searching. However, as we have mentioned earlier, this searching comes with a trade-off of privacy as the tree may reveal the path pattern. So this scheme does not provide the desired level of security and privacy and reveals too much information.

Tang in [137] has proposed a multi-party searchable encryption (MPSE) scheme that is an extension of [109] and based on symmetric primitives. In their scheme they introduce a 'Follow' algorithm that allocates a token to the owner/writer to be distributed among the readers (user) of the index table. This token authorizes the reader to perform the search on the index table. This scheme facilitates the dynamic users but does not allow dynamic databases. The authors assume that there is a secure channel between the user and CS to transmit the trapdoors. The secure channel hides the leakage of the trapdoor during transmission but since the trapdoor is based on one-way hash function, the server itself can learn the search pattern and the access pattern as the same trapdoor is generated for the same keyword searched again. In other words the trapdoor is distinguishable. Their scheme uses forward index, *i.e.,* an index for each file due to which the ranking cannot be done.

In [114], authors introduce a secure searchable public-key encryption (PEKS) scheme. The authors for the first time introduce the concept of "trapdoor indistinguishability" by designing probabilistic trapdoors. The scheme is based on bilinear pairings to achieve the matching when the trapdoor is probabilistic. The probabilistic trapdoors thwart keyword-guessing attacks but their construction cannot achieve ranking as the authors are not forming an index table. This reduces the search time and adds to the network latency.

2.4.2 Multi-keyword SE schemes

In [20], authors introduce public-key systems that support conjunctive, subset and range queries over encrypted data. Their proposed scheme uses bilinear maps and may be termed better than the trivial constructions but it is prone to distinguishability attacks as the trapdoors are probabilistic. Similarly, in [109], authors use bilinear maps for performing search but their scheme also lacks in providing indistinguishability.

Moataz and Shikfa in [91] present a novel SE scheme that allows boolean queries over the encrypted documents. The authors use the method of orthogonalization of the keyword field according to the Gram-Schmidt process [39]. In the scheme, a label is associated to every individual document that contains information about the keywords that the document contain. Since there is no relationship between the individual labels, ranked searching cannot be achieved that results in an increase in the network latency, furthermore, slowing down the search process. However, this does allow scalability by providing dynamic databases.

Wang *et al.* in [144] introduce the concept of probabilistic trapdoors over the inverted index table. It is claimed that the scheme is efficient as compared to the existing schemes as they are able to eliminate the need of pairing operations and use multiplications and exponentiations instead. The proposed scheme uses the Pailier homomorphic algorithm which is based on the quadratic residuosity problem [101] to provide semantic security [120]. Where an encryption scheme is semantically secure if no polynomially bound adversary can extract any partial information about the plaintext from a given ciphertext [11]. The scheme is efficient as compared to the pairing operations, however, the exponentiations still add to the computation overhead and is more resource intensive as compared to the schemes presented in this thesis.

The concept of "coordinate matching" while performing multi-keyword search is introduced in [28] by Cao *et al.*. The proposed scheme allows "as many matches as possible" and enables ranked searching. To generate trapdoors they make use of a vector similar to a bloom filter, where each entry of the bloom filter represents the presence or absence of a keyword. It is also noted that the authors introduce dummy keywords to the documents to hide the leakage associated to the index table. It is observed that this helps to reduce the leakage but the dummy keywords also reduce the accuracy of the results because the dummy keywords also add/increase the rate of false positives. This also affects the accuracy associated with the ranking functionality. The authors introduce two schemes to perform multi-keyword ranked Searchable Encryption; MRSE_I and MRSE_II where the main difference between MRSE_I and MRSE_II is that MRSE_II is more privacy preserving.

In [82], authors introduce a conjunctive query-enabled ranked SE scheme. Conjunctive queries refer to the searching where the keywords to be searched are inter-related, on the contrary, a disjunctive query is a search where the searched keywords are independent and not inter-related. The authors aim to facilitate "conjunctive keywords" but the search is performed similar to disjunctive keywords. The relevance score generation formula is similar to [73]. The authors use inverted index table for the blind storage. Similar to [28], the authors use a vector/bloom filter to represent a trapdoor, where each element represents the presence or absence of a keyword. They also introduce dummy integers to the vector to increase the privacy of the trapdoor and reduce the leakage. However, the authors have not measured the accuracy of the proposed scheme and the effect that the dummy keywords have on the precision. In [125], authors propose a multi-keyword SE encryption scheme and use homomorphic encryption for the index generation and the trapdoor generation. Due to the inefficiency of homomorphic encryption the scheme cannot be deployed onto a real world cloud offering. The authors have implemented their scheme using multi-threading.

2.4.3 Fuzzy-keyword SE schemes

In [83], authors for the first time introduce a privacy preserving construction that enables fuzzy keyword searching. The proposed scheme takes the typographical errors into account and performs fuzzy search. It requires to construct a fuzzy keyword set and take all the possible erroneous keywords into consideration. Therefore, for the keyword CASTLE the substitution operation on the first character of the keyword produces the following set $\{$AASTLE. BASTLE, DASTLE, $\cdots$, YASTLE, ZASTLE$\}$. This process is called 'wildcard-based fuzzy set' construction. This may not be feasible for large datasets containing hundred thousand keywords because the size of the index may grow exponentially. Since this task is performed by the data owner having low memory and computational resources, it may lead to the entire memory being consumed and wastage of resources. Shekokar *et al.* in [121] extend the scheme presented in [83] and implement it to form a proof-of-concept prototype. In [166], authors prove that the scheme presented in [83] is insecure and they prove it through an adversarial model. The authors successfully demonstrate that the adversary is able to distinguish between the search results. Few other constructions based on the wildcard methodology include [150][149].

Wang *et al.* in [146] for the first time introduce the concept of utilizing Locality Sensitive Hashing (LSH) into SE. Their scheme definitely brings a new perspective to designing fuzzy SE schemes because a predefined dictionary spanning over the entire possible set of keywords (correct and erroneous) as discussed in [83] was not required. The authors present two schemes that are able to resist attacks in the known ciphertext model and the known background model respectively. The schemes require per-document bloom filter or vector to perform the fuzzy search which is not feasible in real deployment as it may consume too much storage in the cloud. Furthermore, due to the use of independent bloom filters, the ranking of the documents cannot be achieved.

Wang, Yu and Zhao in [76] present a novel dynamic ranked fuzzy SE scheme. They also follow the wildcard-based approach for performing keyword search. The authors have included a feedback pointer that helps to predict the future searches. The authors claim that;

"Due to the system threat model regarded the cloud server 'honest-but curious', feedback scheme may give rise to the leakage of user's private retrieval information. However, the scheme is suitable and applicable under the hybrid cloud circumstances, which achieves the conduction of feedback operations in private cloud, while public cloud accomplishes the rest cipher-text retrieval procedures".

Although, the scheme may be applicable to their scenario, it contradicts the privacy-guarantees that the existing SE schemes offer [146][78][83][132] and hence their construction leaks too much information. Furthermore, the authors have not given details of the trapdoor formation mechanism.

In [57] authors for the first time introduce uni-gram vectors for the fuzzy keyword search. They claim to have improved the accuracy as compared to the scheme proposed in [146]. They also propose a stemming algorithm that can query the keywords with the same root, and the ranking of the results is based on it. Although their scheme is novel and enhances the query effectiveness, the scheme does not resist distinguishability attacks as the trapdoors are deterministic. Therefore, it cannot be termed as the ultimate fuzzy SE scheme.

2.4.4 Homomorphic-based SE schemes

The first FHE scheme was presented in [60] and up until now many HE schemes have been presented and analyzed [87][4][92][103] . However, fewer schemes have been used for the purpose of SE due to the computational resources required.

Chase and Shen in [34][35] present a partially homomorphic-based SE scheme. Their scheme enables substring-search over the PHE data. Their scheme uses suffix trees as a data structure to store the encrypted data. The authors claim that they are able to achieve asymptotic efficiency which is comparable with the unencrypted suffix trees. However, since the scheme uses a deterministic encryption algorithm, the search pattern is leaked to the cloud [145]. Therefore, their scheme does not resist distinguishability attacks. Chen *et al.* in [38] implement the scheme presented in [34][35]. The authors demonstrate that the scheme does not give the correct results in certain cases where the substring may be occurring many times. They introduce leaf array to the suffice tree to represent the ending of the path. However, the underlying problem of distinguishability remains unaddressed.

Papadimitriou *et al.* in [102] propose Seabed, a system that allows efficient analytics over encrypted data. Seabed is based on additively symmetric homomorphic SE scheme that is three orders of magnitude faster than Paillier [101]. The authors propose two constructions, one is based on the randomized encryption such as AES, whereas, the second construction requires deterministic or OPE to form join operations. Even though the authors have presented an efficient construction by using symmetric encryption, their scheme does not prevent frequency analysis attacks. The attacks over OPE are discussed in [68][96].

Keita *et al.* in [52] introduce the concept of mis-operation in homomorphic-based SE. Mis-operation refers to the modification that takes place when the cloud performs operations over the entire data (whether the data is being searched for or not). The authors introduce a homomorphic encryption scheme that is mis-operation resistant by including an evaluation phase to the decryption process. The authors follow the security model similar to [19] and do not introduce probabilistic trapdoors. Furthermore, the authors claim that their work has the linear search complexity of PEKS [19].

Authors in [113] present a string identification scheme that is based on [19]. The scheme is based on deterministic trapdoors and proved to be adaptively secure under the security definitions presented in [46], however, it does not resist distinguishability attacks. A few other homomorphic-based SE schemes have been proposed that are based on deterministic trapdoors [163].

In [36], authors further study different methods of searching and sorting over the FHE-based encrypted data and discuss their security implications. Their study mainly includes two methodologies that are divided into further subcategories; linear encrypted search and sorting on encrypted data.

We also refer readers to [126][70][136][22][108][85] that present a survey on the existing SE schemes.

2.5 Summary

In this chapter the design primitives were presented that helped to give an overview of the different domains involved in the field of SE. Through the literature it was suggested the design primitives can be represented in the form of a triangle where the vertices of the triangle represent efficiency, security and privacy, and query effectiveness. Based on the query effectiveness, the existing works were broken down into different categories and the schemes were discussed separately. Before heading towards discussing the existing schemes, the existing security definitions were presented and their limitations were discussed. The discussion highlighted the pros and cons of the existing pioneering and state-of-the-art SE schemes.

It was identified that most of the existing schemes were based on deterministic trapdoors, therefore, they are prone to passive attacks (further discussed in Chapter 3). It was also established that many schemes were based on data structures that could not facilitate ranked searching. Furthermore, the schemes were not efficient enough to be deployed onto a real-world cloud offering. The asymptotic comparative analysis can be found in the proceeding chapters where the complexities of the existing schemes are compared against the proposed schemes.

Now that the limitations of the existing schemes have been highlighted, the chapters hereafter present novel schemes that are focused towards enhancing the security and privacy of the encrypted data stored on the cloud. Chapter 3 presents a novel ranked single keyword SE scheme. Chapter 4 presents a novel parallelized disjunctive query based ranked SE scheme. Similarly, Chapter 5 and 6 present novel fuzzy-based ranked SE scheme and a homomorphic-based SE scheme respectively. Unlike the schemes discussed in this chapter, the proposed schemes are based on probabilistic trapdoors that provide higher levels of security and privacy guarantees. The schemes are also implemented and the proof-of-concept prototype is deployed on the British Telecommunications public cloud offering and tested over a real dataset of encrypted documents. Further details can be found in the following chapters.

Chapter 3

Ranked Single Keyword Searchable Encryption Scheme

Cloud is an environment that provides the utility of on-demand resource sharing and data access to the clients and their devices remotely. Apart from the core categories of cloud services, *i.e.,* SaaS, PaaS, IaaS, nowadays, Database-as-a-Service (DaaS) [69][45][16] enables people to store their files on the cloud. This DaaS helps in achieving availability of the documents but there are some interrelated concerns associated to DaaS that are security, trust, expectations, regulations and performance issues [154]. The above concerns are interdependent and should be addressed simultaneously. Encryption is probably the best solution that comes to one's mind while talking about security. However, in the context of DaaS, searching over the encrypted text or SE is a difficult and resource consuming task. This requires a SE scheme to be developed that would facilitate performing textual searches over encrypted data. Such a scheme would help maintain privacy of the outsourced documents while enabling the search over the encrypted documents.

This chapter is the first step towards the development of a framework that performs ranked single-keyword SE. The following contributions to the field of SE are made in this research:

- In this chapter, a novel Ranked Searchable Encryption (RSE) scheme is designed and presented. The proposed RSE scheme is completely based on a probabilistic encryption algorithm to generate probabilistic trapdoors. The probabilistic trapdoors resist distinguishability attacks and mitigate the risk of successful passive attacks.
- The second contribution is that we enumerate the properties of a "*privacy-preserving*" RSE scheme in the context of probabilistic trapdoors by formally defining keyword-trapdoor indistinguishability and trapdoor-index indistinguishability.
- We design and implement a proof of concept prototype and deploy it onto a real cloud environment. We then test our scheme with a real dataset of files containing 120,000 keywords and more than 100,000 documents to analyze the performance of our scheme.

3.1 Problem Formulation

SE allows to secure the data outsourced into the cloud and helps to preserve the privacy of the search queries. A threat model helps to explain potential point of threats and understand the entities from whom the data is to be kept secure. Our approach is similar to [141] as it highlights the assumptions related to the system being developed and the capabilities/ limitations of the adversary. The privacy concerns introduced in the section 1.1.3 can be deduced directly from the threat model presented below. This section also highlights the importance of RSE by explaining the system model followed by the design goals. The threat model remains the same throughout the following chapters however the system model is modified as the query effectiveness or underlying use cases are remodeled.

3.1.1 Threat Model and Assumptions

A SE framework mainly involves two entities: a cloud server (CS) and a data owner. The data owner encrypts and outsources the documents to the CS. If there are multiple data owners involved, it is assumed that they are fully trusted, *i.e.*, the data owners are no threat to the system. The main threat lies with the CS and while performing the security analysis of the schemes it is assumed that the CS acts as an adversary that intends to launch successful attacks. The characteristics of an adversary are given below:

Trusted-but-curious or honest-but-curious server

In the proposed SE schemes it is assumed that the CS is a trusted-but-curious or honest-but-curious server [104][41][23]. Being trusted/honest means that the CS acts in a known and designated manner, but CS is also willing and curious to get a hold of full or partial information about the documents uploaded and held within it. The CS can only launch passive attacks [105] to analyze the data or monitor the network traffic aiming to uncover any possible data or information related to the encrypted documents stored in the CS. In this research it is also assumed that the CS does not launch any active attacks that may lead to denial of service or modification of the data.

Polynomial time adversary

The adversary may perform a polynomially bounded number of encryptions or other operations *i.e.* the adversary is not allowed infinite steps to make a guess, instead the adversary is limit to polynomial number of time steps to make the output.

Adaptive adversary

The adversary (primarily known as the CS) is allowed to maintain a history of all the past searches performed over the encrypted data. Therefore, the adversary knows the trapdoors, the corresponding search patterns and the access patterns. To break the system during the security analysis, the adversary is given access to the history and it allowed to choose the keyword adaptively by analyzing the past history.

Standard Model

A model of computation in which the adversary is limit by only the time and computational resources available and the system is not assumed to be ideal (replaced by a random oracle). The schemes prove to provide high levels of security in the standard model as the schemes are based on hard problems or complex problems that cannot be solved in polynomial time, such as the integer factorization problem.

3.1.2 The System Model

We consider a single writer/single reader (S/S) architecture and use the client-server infrastructure by visualizing a scenario in which there are two parties, Bob (client) and a CS. Bob intends to upload all of his documents $\mathscr{D} = \{D_1, D_2, \ldots, D_n\}$ to the CS to enable remote access. The CS performs the searching of relevant documents on behalf of Bob. In this scheme, the CS is trusted-but-curious. Bob identifies a set of keywords $\mathscr{W} = \{W_1, W_2, \ldots, W_m\}$ from the set of documents $\mathscr{D}$, and generates a relevance score based on the frequency of occurrence of the keywords within the set of documents. These relevance scores help in performing the ranked search. Ranked searching facilitates the search by giving the user the liberty of selecting the most relevant documents from a collection, by identifying the frequency of the occurrence of a keyword within a set of documents. Ranked searching is mainly used for keyword search because the server may find several documents satisfying the query, whereas, in complex queries, the server might be able to identify a few documents in response to the search query.

The relevance frequency (RF) helps to rank the documents. The RF formula [73] presented here is widely accepted and already used in SE [49, 147, 130, 132]. So, given a keyword W, and a document D, the relevance frequency (RF), is calculated as:

$$RF(W,D) = \sum_{U=1}^{|\mathscr{W}|} \frac{1}{|D|} \cdot (1 + \ln f_{(D,W_U)}) \cdot \ln(1 + \frac{n}{f_{W_U}}) \tag{3.1}$$

where $|D|$ denotes the length of the document obtained by counting the keywords appearing in the document D; $f_{(D,U)}$ denotes number of times a keyword U appears within a particular document D; f_U denotes the number of documents in the dataset that contain the keyword U, and n denotes the total number of documents in the dataset.

Apart from this ranking function other scoring functions such as Apache Lucene [55] or Juru [29], after modification may be used. All the scoring functions vary in computational time and quality of outcome. The Apache Lucene scoring function developed by the Apache foundation is based on crawlers and mainly used for the search engine optimization. The Juru ranking mechanism does not take into consideration the entire dataset, instead considers the frequency of occurrence of a keyword within the same document. In [43] a comparison of Lucene and Juru has been performed. This research uses the RF formula (equation 3.1) that is widely used in the literature and helpful for the doing a direct comparison with the state-of-the-art in the following chapters.

Now Bob generates an index table I, and outsources I along with the encrypted documents $\mathscr{D}$ to the CS.

If Bob wants to search for a document containing a specific keyword, he simply generates a probabilistic trapdoor T and sends it to CS. CS uses the trapdoor T to search the index table I and returns a set of relevant documents in a ranked order.

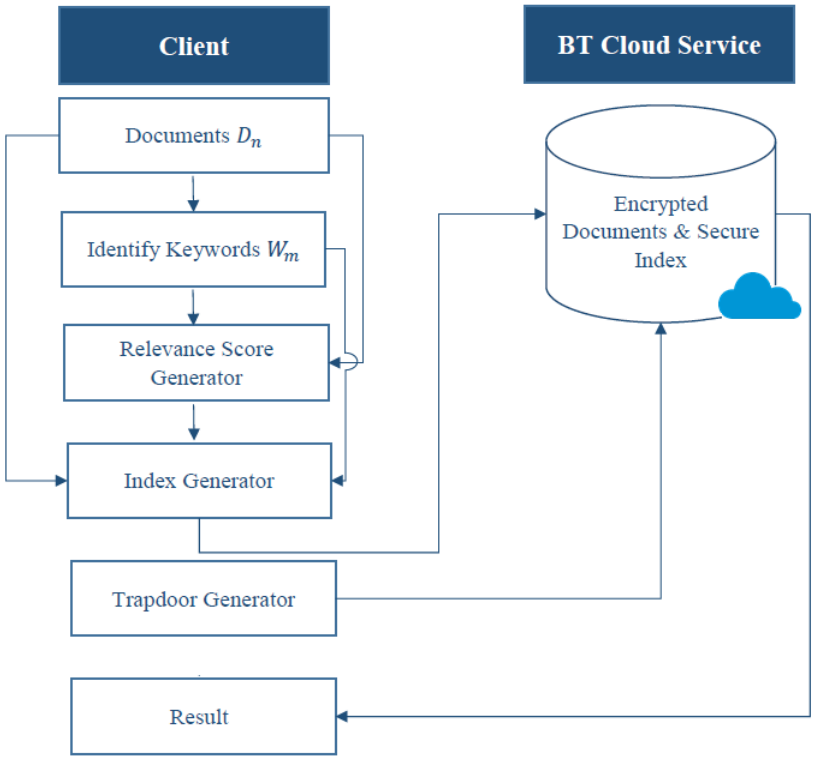

Fig. 3.1 System Architecture Diagram for RSE

Figure 3.1 shows the flow of events in the RSE scheme where a client is interacting with the CS. It can be seen that mainly all the tasks are performed on the client's side, whereas the searching is done on the CS.

3.1.3 Probabilistic Encryption

In order to highlight the advantage of indeterministic/ probabilistic trapdoors, we revisit the definition of probabilistic encryption also termed as randomized encryption [116].

Probabilistic Encryption: A probabilistic encryption system is a quadruple (M, K, C, Π), where M is the message space, K is the key space, C is the ciphertext space and Π represents a relation $\Pi \subseteq M \times K \times C$ such that:

- for each key $k \in K$ and each ciphertext $c \in C$, there is at most one $x \in M$ such that $(x, k, c) \in \Pi$.

- for each message $x \in M$ and each key $k \in K$ there is at least one ciphertext $c \in C$ such that $(x,k,c) \in \Pi$.

The encryption process works as follows:

A bit sequence $r \in R$ is chosen randomly. The value $\Psi(r,x,k)$ would be computed, where Ψ is a deterministic function such that $\Psi : R \times M \times K \mapsto C$ and $\Pi = \cup_{r \in R, k \in K, x \in M}(x,k,\Psi(r,x,k))$.

If the trapdoor generation process is based on probabilistic encryption, then the resulting trapdoors will also be probabilistic. The probabilistic trapdoors are indistinguishable and resist distinguishability attacks. We explain this with the help of a scenario highlighting passive attacks below.

3.1.4 Scenario related to Passive Attacks

Probabilistic trapdoors mean that for the same keyword being searched repeatedly, a unique search token (trapdoor) may be generated every time. One may appreciate the advantage of having a probabilistic trapdoor by considering an adversary $\mathscr{A}$, capable of launching a passive attack. A passive attack over the network enables the adversary to monitor, read and capture data exchanges. There are different types of passive attacks including sniffing, interception, traffic analysis etc [5] [155]. Suppose the adversary is able to sniff the transmitted trapdoor and uncover the keyword from it, in such a case if the trapdoor is not probabilistic, this will reveal all the future searches to the adversary $\mathscr{A}$, and over a longer period of time the adversary $\mathscr{A}$ may uncover the entire set of keywords. On the contrary, if the trapdoors are probabilistic, then even if the adversary $\mathscr{A}$ is able to uncover the keyword from the trapdoor, it may be effective only for that particular trapdoor and the adversary may not be able to predict future trapdoors. Therefore, probabilistic trapdoors help to thwart keyword-guessing attacks [114] and reduces the risks of successful brute-force attacks [100]. As discussed in Chapter 2, the existing schemes are based on deterministic encryption or OPE, the attacks on deterministic encryption schemes and Order Preserving Encryption (OPE) are discussed in [96]. Sections 3.2 and 3.4 further explain the advantages of using probabilistic trapdoors, and how the existing schemes may be insecure against a passive adversary.

3.1.5 Design Goals

The proposed construction should bear the following security and performance goals:

- *Trapdoor Unlinkability:* The primary goal is that the trapdoor should be probabilistic, therefore, for the same keyword searched again a new trapdoor should be generated. This helps resist distinguishability attacks.
- *Privacy Guarantee:* The search pattern should be kept hidden, whereas, the access pattern may be disclosed. It should be secure in the known cipher-text model. In other words, apart from the outcome of the search, the CS (referred to a polynomial time adversary) should not deduce any keyword related information from the secure index and trapdoors even if an adaptive query is made.
- *Lightweight:* The scheme should be secure and should offer lightweight computations as compared to the state-of-the-art.

We now formally define our proposed RSE scheme that facilitates the search over encrypted documents in ranked order. The following definition presents the algorithms and the phases that our scheme comprises of:

Definition (Ranked Searchable Encryption (RSE) Scheme) A RSE comprises of five polynomial time algorithms Π=(KeyGen, Build_Index, Build_Trap, Search_ Outcome, Dec) such that:

$(K, k_s) \leftarrow KeyGen(\lambda)$: is a probabilistic key generation algorithm run by the client. The algorithm takes a security parameter λ as input and returns a master key K and a session key k_s.

$(I) \leftarrow Build\_Index(K, \mathscr{D})$: is a deterministic algorithm run by the client to generate an index table I. The algorithm takes as input a master key K and a collection of documents $\mathscr{D}$ to be outsourced to the CS. The algorithm returns a secure index table I.

$T_W \leftarrow Build\_Trap(K, k_s, W, num)$: is a probabilistic algorithm run by the client. The algorithm as input requires the master key K, a session key k_s, a keyword W and the number (num) of documents D required. The algorithm returns a trapdoor T_W.

$X \leftarrow Search\_Outcome(k_s, I, T_W)$: is a deterministic algorithm run by the CS. The algorithm takes the session key k_s, index table I and the trapdoor T_W as the input and returns X, a set of desired encrypted document identifiers $Enc_K(id(D_i))$ containing the keyword W in ranked order.

$D_i \leftarrow Dec(K, X)$: is a deterministic algorithm run by the client. The algorithm takes the client's master key K and encrypted set of document identifiers $Enc_K(id(D_i))$ as input to decrypt and recover the document id's.

Correctness: A RSE scheme is correct if for the security parameter λ, the master key K and the session key k_s generated by $KeyGen(\lambda)$, for (I) output by $Build\_Index(K, \mathscr{D})$, the search using the trapdoor T_W always returns the correct set of encrypted document identifiers $E_K(id(D_i))$ in ranked order. A RSE scheme is correct if the following are true:

- If $W \in D_i$, then the following should hold with an overwhelming probability:

$$Search\_Outcome(k_s, I, T_W) = \mathscr{D} \cap Dec(K, X) = D_i, where 1 \leq i \leq n \quad (3.2)$$

- If $W \notin D_i$, then the following should hold with an overwhelming probability:

$$Search\_Outcome(k_s, I, T_W) = \mathscr{D} \cap Dec(K, X) = \emptyset \tag{3.3}$$

Soundness: A RSE scheme is sound if for the security parameter λ, the master key K and the session key k_s generated by $KeyGen(\lambda)$, for (I) output by $Build\_Index(K, \mathscr{D})$, the search using the trapdoor T_W always returns sound results, *i.e.,* the result should not contain any false positives. A RSE scheme is sound if the following are true:

- If $W \in D_i$, then the following should hold with an overwhelming probability

$$Search\_Outcome(k_s, I, T_W) = 1 \tag{3.4}$$

- If $W \notin D_i$, then the following should hold with an overwhelming probability

$$Search\_Outcome(k_s, I, T_W) = 0 \tag{3.5}$$

3.2 Security Definitions

This section proposes new definitions for indistinguishability in terms of ranked searchable encryption. An ideal SE scheme should fulfill these definitions to ensure privacy. Section 3.4 proves that the proposed RSE scheme complies with the following definitions.

3.2.1 Keyword-Trapdoor Indistinguishability for RSE

Keyword-Trapdoor Indistinguishability refers to the act of performing search over encrypted text in such a way that the redundancy in the statistics of the (plaintext) keywords should be dissipated into the associated trapdoor. Therefore, for the same keyword appearing twice, the trapdoor should not be distinguishable even if the history (keyword, trapdoor) is generated adaptively. To guess the keyword or the document's content, the attacker has to intercept a tremendous amount of data to uncover the underlying plaintext in polynomial time.

Description

The challenger begins by generating an index table I against a document collection $\mathscr{D}$. The adversary $\mathscr{A}$ selects a keyword W and sends it to the challenger. The challenger generates a trapdoor and sends it back to the adversary $\mathscr{A}$. This continues until the adversary $\mathscr{A}$ has submitted polynomial-many keywords. Now the challenger tosses a fair coin b, the adversary $\mathscr{A}$ has to submit two keywords (W_0, W_1) to the challenger and receives a trapdoor corresponding to the keyword W_b. The adversary $\mathscr{A}$ has to guess and output the bit b. If the guess is made with a probability of greater than 1/2 then the adversary wins when the security parameter λ is negligible *i.e.*, λ is sufficiently small that it can be ignored in the interests of expediency.

Definition 3.1 *Let* RSE= (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) *be a Ranked Searchable Encryption scheme over a dictionary* $\mathscr{W}$*,* λ *be the security parameter,* $\mathscr{D}$ *be a set of documents and* $\mathscr{A} = (\mathscr{A}_0, \mathscr{A}_1, \cdots, \mathscr{A}_{m+1})$ *be adversaries such that* $m \in \mathbb{N}$*. Consider the following probabilistic experiment* $Trap\_Index_{RSE,\mathscr{A}}(\lambda)$:

```
$Trap\_Index_{RSE,\mathscr{A}}(\lambda)$
    $(K, k_s) \leftarrow KeyGen(\lambda)$
    $(I) \leftarrow Build\_Index(K, \mathscr{D})$
    $for\, 1 \leq i \leq m$
        $(st_{\mathscr{A}}, W_i) \leftarrow \mathscr{A}_i(st_{\mathscr{A}}, T_{W_1}, \cdots, T_{W_i})$
        $T_{W_i} \leftarrow Build\_Trap_K(W_i)$
    $b \overset{\$}{\leftarrow} \{0,1\}$
```

$$(st_{\mathscr{A}}, W_0, W_1) \leftarrow \mathscr{A}_0(\lambda)$$
$$(T_{W_b}) \leftarrow Build\_Trap(K, k_s, W_b, num)$$
$$b' \leftarrow \mathscr{A}_{m+1}(st_{\mathscr{A}}, T_{W_b})$$
$$(T_{W'}) \leftarrow Build\_Trap_K(W_j); j \in \mathbb{N}$$
$$if\, b' = b, output\, 1$$
$$otherwise\, output\, 0$$

where $st_{\mathscr{A}}$ represents a string that captures $\mathscr{A}$'s state. The keyword-trapdoor indistinguishability holds if for the polynomial time adversaries $(\mathscr{A}_0, \mathscr{A}_1, \cdots, \mathscr{A}_{m+1})$,

$$Pr[Key\_Trap_{RSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \tag{3.6}$$

where probability is over the choice of b.

3.2.2 Trapdoor-Index Indistinguishability for RSE

Trapdoor-Index Indistinguishability relates to the complexity offered by a SE scheme. The keywords, trapdoor and index table should be complex, and involved in such a way that the trapdoor should not reveal the corresponding index table entries prior to the search, and should not be distinguishable. Therefore, for the same keyword appearing twice, the trapdoor should not be distinguishable even if the history (keyword, trapdoor, index) is generated adaptively. Furthermore, a change of one bit/character of the keyword, should completely change the Trapdoor and Index Table or vice versa.

Description

The challenger begins by generating an index table against a data collection $\mathscr{D}$. The challenger sends the set of keywords $\mathscr{W}$, the trapdoors generated for all the keywords $\mathscr{W}$, along with the associated index table entries $I[0][W_m]$ to the adversary, while maintaining the order in which they occur. Now the challenger tosses a fair coin b, the adversary has to submit two keywords (W_0, W_1) to the challenger and receives a trapdoor corresponding to the keyword W_b. The adversary has to now decide the corresponding index value and is challenged to output the bit b. If the adversary is able to make a guess with a probability greater than 1/2 then the adversary has succeeded and the scheme lacks in providing trapdoor-index indistinguishability. The security parameter λ is negligible *i.e.*, λ is sufficiently small that it can be ignored to highlight the security of the scheme.

Definition 3.2 *Let* RSE= (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) *be a Ranked Searchable Encryption scheme over a dictionary* $\mathscr{W}$*,* λ *be the security parameter,* $\mathscr{D}$ *be a set of documents and* $\mathscr{A} = (\mathscr{A}_0, \mathscr{A}_1)$*. Consider the following probabilistic experiment* $Trap\_Index_{RSE,\mathscr{A}}(\lambda)$:

```
$Trap\_Index_{RSE,\mathscr{A}}(\lambda)$
    $(K, k_s) \leftarrow KeyGen(\lambda)$
    $(I) \leftarrow Build\_Index(K, \mathscr{D})$
    $for\, 1 \leq i \leq m$; where $m \in \mathbb{N}$
        $let\, I' = I[0][i]$
        $let\, W = (W_1, \cdots, W_i)$
        $T_{W_i} \leftarrow Build\_Trap(K, k_s, W_i, num)$
    $b \overset{\$}{\leftarrow} \{0,1\}$
    $(st_{\mathscr{A}}, W_0, W_1) \leftarrow \mathscr{A}_0(st_{\mathscr{A}}, \lambda, W_m, I', T_{W_m})$
    $(T_{W_b}) \leftarrow Build\_Trap(K, k_s, W_b, num)$
    $b' \leftarrow \mathscr{A}_1(st_{\mathscr{A}}, I_{W_b})$
    $(T_{W'}) \leftarrow Build\_Trap_K(W_j); j \in \mathbb{N}$
    $if\, b' = b, output\, 1$
    $otherwise\, output\, 0$
```

where $st_{\mathscr{A}}$ represents a string that captures $\mathscr{A}$'s state. The trapdoor-index indistinguishability holds if for the polynomial time adversaries $(\mathscr{A}_0, \mathscr{A}_1)$,

$$Pr[Trap\_Index_{RSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \tag{3.7}$$

where probability is over the choice of b.

Theorem 3.1: *The proposed RSE scheme provides Keyword-Trapdoor Indistinguishability and Trapdoor-Index Indistinguishability if the Inverted Index table* (I) *is secure and the trapdoors are probabilistic.*

3.3 Proposed RSE scheme

As discussed in Section 3.1.5, the proposed RSE scheme comprises of five phases. This section presents and discusses each of the phases.

3.3.1 Scheme Construction

- **Phase 1-KeyGen** (λ)**:** Given a security parameter λ, generate a master key K and session key k_s; such that $K, k_s \leftarrow \{0,1\}^{\lambda}$.

- **Phase 2-Build_Index** $(K, \mathscr{D})$**:**
  - Initialized dynamic 2D array A.
  - Scan $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$ and build $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$, a set of unique and distinct keywords occurring in $\mathscr{D}$.
  - Initialize Prime number p of the size $\lambda + 1$ bits.
  - For $1 \leq t \leq m$

    - let $a \leftarrow H_K(W_t)$

```
            - compute $a^{-1}$ and store it in $A[1][t]$.
            - compute $E_K(id(D_n))$, store it in $A[t][1]$
            - calculate the $RF$ for each $W_m$ occurring in $\mathscr{D}$ using the equation
        3.1 and store the value at the respective locations within $A$.
    – Mask($RF$)
            - for $1 \leq M \leq number\ of\ columns\ in\ A$
                - $A[n+1][M+1] = A[n+1][M+1] \times random\ values$
        The index table is generated (represented as $A$) and stored in the CS.
```

- **Phase 3-Build_Trap** (K, k_s, W, num)**:**

```
    – let $a \leftarrow (H_K(W))$
    – let $b \leftarrow Enc_K(W)$
    – let $c \leftarrow a \cdot b$
    – let $d \leftarrow H_{k_s}(b)$
    – Set Trapdoor $T_W \leftarrow (d, c, num)$
```

- **Phase 4- Search_Outcome** (k_s, I, T_W)**:** Identify the documents $D_i \in \mathscr{D}$ as the outcome of the search as follows:

```
    – Initialize dynamic Array $X$.
    – For $1 \leq l \leq size\ of\ I$:
            - if $(d == H_{k_s}(c \cdot a^{-1}))$:
                - for $1 \leq N \leq num$:
                    - find highest $RF$, return $Enc_K(id(D_i))$;
    – $X[\,] \leftarrow Enc_K(id(D_i))$;
```

Return X to the client.

- **Phase 5-Dec** (K,X)**:** Given X; a set of encrypted document identifiers, decrypt X using the master key K to uncover the outcome of the search.

3.3.2 Description

We now briefly discuss each of the phases involved in the RSE scheme that have been presented in the previous section.

KeyGen Phase

The KeyGen algorithm helps the client to generate the keys. The algorithm takes as input a security parameter λ. The client generates a master key K; where, $K \in \{0,1\}^{\lambda}$, a session key k_s; where, $k_s \in \{0,1\}^{\lambda}$. The master key K is kept secret with the client whereas the session key k_s is shared with the server prior to the Build_Index phase.

Build_Index Phase

The client generates an index table I represented by a dynamic 2D array A. The client uses a cryptographic Hash function:

$$H : \{0,1\}^{\lambda} \times W \rightarrow \mathbb{Z}_p$$

The keyed Hash function H uses the master key K to generate a Hash of the keywords. The array A holds three attributes; the first row of the array consists of values that are generated by calculating the inverse of the Hash of keywords, the first column of the array A consists of the encrypted document identifiers $Enc_K(id(D_n))$ of all the outsourced documents, whereas, the remaining entries of the array are the relevance frequencies of the keywords $\mathscr{W}$ among the documents $\mathscr{D}$. The relevance frequencies are calculated according to equation 3.1. Each column represents the relevance frequencies associated to a particular keyword W. We multiply each column (excluding the first row and first column of the array A) with a random number obtained from a CSPRNG, represented by $Mask(RF)$. This way the relevance frequencies are masked while maintaining proportion between the relevance scores of the keyword W occurring in different documents. This helps to prevent frequency analysis attack and disclosure of document size while maintaining correct ranking of documents.

Build_Trap Phase

The client generates a trapdoor to search for documents containing a particular keyword. The client using the master key K generates the hash $H(\cdot)$ of the keyword, represented by a. Again, with a probabilistic symmetric encryption algorithm, the client encrypts the keyword, represented by b. Now c is computed by multiplying a with b. The client uses a cryptographic keyed Hash function:

$$H : \{0,1\}^{\lambda} \times W \rightarrow \mathbb{Z}_p$$

The keyed Hash function H uses the master key K to generate a; the hash of the keyword, and uses session key k_s to generate d; the $H_{k_s}(b)$. The trapdoor consists of d, c and the desired number of documents represented by *num*. The trapdoor is transmitted to the CS.

Search_Outcome

The CS now performs the search based on the received trapdoor. The server has d, c and *num*. The CS tries to find an index table's column entry for which the following condition holds true $d == H_{k_s}(c \cdot a^{-1})$. On a positive hit, the CS returns client the encrypted document identifiers in ranked order, based on the documents having the highest relevance frequencies. The total number of documents returned will be equal to *num*.

Dec Phase

The client after receiving the ranked encrypted document identifiers, decrypts them to uncover the document identifiers containing the searched keyword.

Remark 1: The index table I needs to be regenerated whenever the database is modified, but this can be avoided if we remove ranking, because the re-ranking is to be performed only whenever a modification is made to the outsourced database.

Remark 2: By multiplying the relevance score with random numbers, we mask the actual frequency of the keywords and avoid the frequency analysis attack, while performing effective and efficient ranked searching. This also helps to prevent the disclosure of the size of the documents and maintaining privacy. To further enhance the security of the index, one may also use Order Preserving Hashing (OPH) [152] or Order Preserving Encryption (OPE) [18].

3.3.3 Correctness and Soundness

This section proves the correctness and soundness (defined in Section 3.1.5) of the proposed RSE scheme.

Let (K, k_s) represent the output of the KeyGen phase, where, the master key is $K \in \{0,1\}^\lambda$ and the session key $k_s \in \{0,1\}^\lambda$. Given $W, W' \in \mathscr{W}$, it is straight forward to verify that the following are true:

- Given $T_W = Build\_Trap(K, k_s, W, num)$, the following equality holds with an overwhelming probability:

$$T_W = \begin{Bmatrix} H_{k_s}((Enc_K(W)) \cdot H_K(W)), \\ ((H_K(W)) \cdot (Enc_K(W))), num \end{Bmatrix} \tag{3.8}$$

- Given $T_W = Build\_Trap(K, k_s, W', num)$, and $W' \neq W$, the following inequality holds with an overwhelming probability:

$$T_W \neq \begin{Bmatrix} H_{k_s}((Enc_K(W')) \cdot H_K(W')), \\ ((H_K(W')) \cdot (Enc_K(W'))), num \end{Bmatrix} \tag{3.9}$$

  In fact, this inequality can hold only if $H_K(W) = H_K(W')$ which is having a negligible probability.

This leads to the conclusion that a unique trapdoor is mapped to a distinct keyword. Since the index table contains encrypted document identifiers $Enc_K(id(\mathscr{D}))$ for every document that maps to the keywords, therefore, as a result, the outcome of the Search_Outcome phase corresponds to the value outlined in the correctness and soundness definitions mentioned in Section 3.1.1 and equations 3.2, 3.3, 3.4, 3.5 respectively. Hence, the proposed RSE scheme is correct and sound.

3.4 Security Analysis

All of the previously known SE constructions leak some information because they were based on deterministic trapdoors [78][77]. In [74] authors have studied the search pattern disclosure of the previously known SE schemes. The proposed scheme is based on a probabilistic trapdoor, so before mapping the proposed scheme against the security definitions stated in Section 3.2, we formally highlight any information that the proposed scheme leaks.

3.4.1 Leakage Profiles

The leakage profiles analyze any possible leakage of information significant or insignificant, encrypted or unencrypted based on a set of assumptions. We analyze all the three artifacts that are obtained from the five polynomial time algorithms explained previously, *i.e.,* index table I, trapdoor T_W and the outcome of a search. While defining the leakage we assume that the attack is launched by an adversary $\mathscr{A}$ in a standard model so we do not restrict the adversary by replacing our scheme with any weak construction. The leakage focuses on the information that is revealed within polynomial time. Our security analysis yields the following results:

Leakage $L_{3.1}$

Description: The leakage $L_{3.1}$ is associated to the index table I. It is assumed that I is revealed to all the stakeholders, *i.e.,* the client, the CS and the adversary $\mathscr{A}$. This leakage is defined as:

$$L_{3.1}(I) = \begin{Bmatrix} ((H_K(W_m)))^{-1}, Enc_K(id(D_n)), \\ Mask(RF),\ (H_K(W_m)) \\ Enc_K(\mathscr{W}) \in Enc_K(\mathscr{D}) \vee Enc_K(\mathscr{W}) \notin Enc_K(\mathscr{D}) \end{Bmatrix} \tag{3.10}$$

Leakage $L_{3.2}$

The leakage $L_{3.2}$ is associated to the Trapdoor T_W generated for a particular keyword W to be searched. It is assumed that T_W is generated by the client and revealed to all the stakeholders, *i.e.,* the CS and the adversary $\mathscr{A}$.

$$L_{3.2}(T_W) = \begin{Bmatrix} a \leftarrow H_K(W_i) \cdot Enc_K(W_i), \\ b \leftarrow H_{k_s}(Enc_K(W_i)), num \end{Bmatrix} \tag{3.11}$$

Leakage $L_{3.3}$

The leakage $L_{3.3}$ is associated to search outcome (SO) of the Trapdoor generated for a particular keyword T_W.The search outcome is revealed to all the stakeholders, *i.e.,* the client, CS and the adversary $\mathscr{A}$. This leakage is defined as:

$$L_{3.3}(SO) = \left\{ \text{OC}(W), Enc_K(id(D_i))_{\forall T_W \in \mathscr{D}} \right\} \tag{3.12}$$

where OC represents the outcome corresponding to the searched keyword.

Discussion on Leakage: As the trapdoor is based on a probabilistic encryption algorithm and a keyed hash function, therefore, we can say that the leakage associated with the trapdoor is meaningless and we do not need to be concerned about it. In other words, suppose if an adversary is accidentally given access to the trapdoor generation oracle then all the future searches are still secure. We explain this with the help of the leakage profile of the scheme presented in [77]. Their scheme gives away the search pattern and access pattern (*cf.* Section 2 [77]). The search pattern identifies whether the same keyword is being searched again. Since our proposed scheme is based on probabilistic trapdoors, we are able to avoid the leakage associated with the search pattern. However, similar to Kamara *el al.*'s scheme [77], our scheme also leaks the access pattern. The access pattern reveals the documents that are accessed as a result of a successful search.

As mentioned earlier, the relevance frequency can be masked using a random number, or made highly secure by using Order-Preserving Hashing. However, both of the techniques may reveal the presence or absence of an encrypted keyword within a document. Although this leakage does not affect the property of trapdoor unlinkability and indistinguishability, this is the only leakage related to the relevance frequencies. Therefore, it is evident that $L_{3.1}$ (equation 3.10) and $L_{3.3}$ (equation 3.12) might lead to the security and privacy concerns, but we will prove that these leakages do not reveal any information related to the data outsourced. Another point to be noted here is that these leakages and assumptions are interrelated and interdependent, hence, to maintain security all the assumptions should be strictly met.

3.4.2 Formal Security Proofs

Lemma 3.1. *The Ranked Searchable Encryption Scheme (RSE) presented above is "privacy-preserving" as it is* $(L_{3.1}, L_{3.2}, L_{3.3})$*-secure and according to Definition* 3.1, 3.2*, where* $L_{3.1}$ *is associated with the index table I and leaks the encrypted document identifiers, masked relevance frequencies, inverse of hash of keyword and presence/absence of an encrypted keyword within an encrypted document. Whereas,* $L_{3.2}$ *leaks* a, b *and the number of required documents and* $L_{3.3}$ *leaks the outcome of a trapdoor and the encrypted document identifiers.*

Proof: We start the proof of this lemma by establishing evidence that the proposed RSE scheme is in accordance with Theorem 3.1. This is achieved by simulating the keyword-trapdoor indistinguishability and trapdoor-index indistinguishability definitions for the proposed RSE scheme. The proof requires an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. We demonstrate if the adversary $\mathscr{A}$ is successfully able to distinguish between outcome of the algorithms and between the keywords, trapdoors and associated index tables, it will result in compromising the privacy-preserving property of RSE.

We take a game-based approach similar to [143] in which the security proof is divided into phases, *i.e.*, setup, challenge and the outcome phase.

Keyword-Trapdoor Indistinguishability in RSE:

Let RSE be a SE scheme. Suppose there are at most m keywords $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$ and n documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$, where $n, m \in \mathbb{N}$ (set of natural numbers) associated to an index table. The game is played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. The game is divided into three phases as follows:

- **Setup Phase:** The adversary $\mathscr{A}$ sends a keyword to the challenger $\mathscr{C}$. The challenger $\mathscr{C}$ returns a trapdoor to $\mathscr{A}$. This continues between the adversary $\mathscr{A}$ and the challenger $\mathscr{C}$ until trapdoors for all the keywords have not been generated and sent to the adversary. As a result the adversary forms a history of trapdoors and the corresponding keywords.

- **Challenge Phase:** The adversary $\mathscr{A}$ selects two keywords $W'_0, W'_1 \in \mathscr{W}$ and sends them to the challenger $\mathscr{C}$. The selection of the keywords can be done; such that the adversary $\mathscr{A}$ intends to search for unique keywords, *i.e.*, $W'_0 \neq W'_1$; The challenger $\mathscr{C}$ in response tosses a fair coin $b \leftarrow \{0,1\}$ and generates a trapdoor corresponding to the value of b, *i.e.,* $T'_{W'_b}$. After the challenge has been completed, setup phase is run again. We allow the adversary to search for the same keywords again if interested.

- **Outcome Phase:** The adversary $\mathscr{A}$ is given the generated Trapdoor $T'_{W'_b}$. $\mathscr{A}$ will now have to guess and output $b' \in \{0,1\}$ and if $b = b'$ then the adversary wins. In other words the adversary $\mathscr{A}$ has to output the keyword W'_b corresponding to $T'_{W'_b}$ to the challenger $\mathscr{C}$ in polynomial time. If the adversary $\mathscr{A}$ correctly guesses the trapdoor corresponding to the keyword then it has won otherwise RSE provides keyword-trapdoor indistinguishability and the challenger $\mathscr{C}$ wins.

Since the trapdoors are probabilistic and unique, therefore the probability that the adversary $\mathscr{A}$ wins is 1/2. That is the adversary can make a successful guess with the probability of 1/2 which is according to the Definition 3.1 and equation 3.6. Therefore, the challenger wins and the RSE scheme provides keyword-trapdoor indistinguishability.

Trapdoor-Index Indistinguishability in RSE:

Let RSE be a SE scheme. Suppose there are at most m keywords $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$ and n documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$, where $m, n \in \mathbb{N}$ (set of natural numbers) are associated to an index table. The game is played between an adversary $\mathscr{A}$ and a Challenger $\mathscr{C}$. The game is divided into three phases as follows:

- **Setup Phase:** The challenger $\mathscr{C}$ generates an index table I corresponding to the set of documents. The challenger $\mathscr{C}$ generates and sends the trapdoors for all keywords $\mathscr{W}$, the index table entries corresponding to the trapdoor and the keywords to the adversary $\mathscr{A}$.

- **Challenge Phase:** The adversary $\mathscr{A}$ is allowed to select two keywords $W'_0, W'_1 \in \mathscr{W}$ and send them to the challenger $\mathscr{C}$. The selection of the keywords can be done in such a way that the adversary $\mathscr{A}$ intends to search for distinct keywords $W'_1 \neq W'_2$. The challenger $\mathscr{C}$ in response tosses a fair coin $b \leftarrow \{0,1\}$ and generates a trapdoor corresponding to the value of b, *i.e.*, $T'_{W'_b}$. After the challenge has been completed, the adversary $\mathscr{A}$ is given access to the previously generated history that was sent in setup phase.

- **Outcome Phase:** $\mathscr{A}$ is given the generated trapdoor $T'_{W'_b}$. Adversary $\mathscr{A}$ will now have to guess and return the index table entry corresponding to the trapdoor $T'_{W'_b}$ and W'_b in polynomial time. The adversary $\mathscr{A}$ wins if the guess is correct otherwise RSE provides trapdoor-index table indistinguishability, and the challenger $\mathscr{C}$ wins.

It is observed that although the adversary has a history of the past searches, it cannot guess the index table entry because a unique trapdoor is generated every time. Therefore, the probability that the adversary $\mathscr{A}$ wins is 1/2 which is in-line with the Definition 3.2 and the equation 3.7 respectively.

Proof to the Theorem 3.1 leads to the following corollary:

Corollary 3.1: *Keyword-Trapdoor Indistinguishability and Trapdoor-Index Indistinguishability results in a Privacy Preserving Ranked Searchable Encryption Scheme.*

Proof: Let RSE=(KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) be a Ranked Searchable Encryption scheme. We make the following claim that leads to the proof of this corollary.

Claim: If RSE is Keyword-Trapdoor Indistinguishable, then it is Trapdoor-Index Indistinguishable.

Firstly, we assume that there exists a polynomial time adversary $\mathscr{A}$ that succeeds in the experiment $Key\_Trap_{RSE,\mathscr{A}}(\lambda)$ with non-negligible probability over 1⁄2 (high probability), then there exists a polynomial time adversary $\mathscr{B}$ and a polynomial time distinguisher $\mathscr{Q}$ that distinguishes between the output of the experiment $Trap\_Index_{RSE,\mathscr{A}}(\lambda)$ with non-negligible probability over 1⁄2. Let adversary $\mathscr{B}$ sample $b \xleftarrow{\$} \{0,1\}$; compute $(st_{\mathscr{A}}, W_0, W_1) \leftarrow A_0(\lambda)$. The distinguisher $\mathscr{Q}$ is given access to a history consisting of trapdoors and corresponding keywords. The adversary proceeds as follows:

1. It parses $(st_{\mathscr{A}}, W_i) \leftarrow \mathscr{A}_i(st_{\mathscr{A}}, T_{W_2}, \cdots, T_{W_{i-1}})$ where $2 \le i \le m$; $m \in \mathbb{N}$
2. It computes $b' \leftarrow A_{i+1}(st_{\mathscr{A}}, T_{W_b})$
3. It outputs 1 if $b' = b$, and 0 otherwise.

Since A_{i+1} are polynomial time adversary, hence, $\mathscr{B}$ and $\mathscr{Q}$ are also polynomial time adversaries. Now, we have to guess the probability of $\mathscr{Q}$'s success. $\mathscr{Q}$ will output 1 if and only if $A_{i+1}(st_{\mathscr{A}}, T_{W_b})$ succeeds in correctly guessing b. It is to be noted that the Build_Trap phase is dependent upon trusted atomic primitives and uses a probabilistic encryption algorithm therefore the outcome is independent of b. Therefore, A_{i+1} will guess b with the probability utmost $1/2$ which is according to the Definitions 3.1. Therefore, our initial assumption of such an adversary who can succeed in the experiment $Key\_Trap_{RSE,\mathscr{A}}(\lambda)$ with a non-negligible probability over $1/2$ is wrong. Hence the distinguisher $\mathscr{Q}$ that distinguishes between the output of the experiment $Trap\_Index_{RSE,\mathscr{A}}(\lambda)$ with non-negligible probability over $1/2$ does not exist and it is according to our Definition 3.2. Hence our claim (stated above) is correct.

Now, we prove that an RSE is *"Privacy Preserving"*. As discussed earlier, the entire scheme is dependent upon a probabilistic trapdoor and provides Keyword-Trapdoor and Trapdoor-Index indistinguishability. According to definition 3.1, since a probabilistic trapdoor maps to an index location while maintaining privacy, the privacy of the corresponding document identifiers is also preserved. Due to the probabilistic trapdoor, the indistinguishability and privacy between the entities involved in the RSE is maintained on the whole that results in privacy preservation.

Now we need to prove the security of our scheme against the leakages $L_{3.1}, L_{3.2}$ and $L_{3.3}$ (represented by equations 3.10, 3.11 and 3.12 respectively). We argue that the leakages $L_{3.1}, L_{3.2}$ and $L_{3.3}$ are meaningless and do not affect our scheme. The security of our proposed scheme is dependent upon trusted atomic primitives, therefore, we claim that our scheme adds to the security of these primitives and does not weaken the security provided by the atomic primitives. We refer to the algorithm explained in Section 3.3. The KeyGen phase generates two keys $(K, k_s) \leftarrow KeyGen(\lambda)$. The Setup phase generates an index table $(I) \leftarrow Setup(K, \mathscr{D})$ corresponding to the set of documents. The Build_Trap(K, k_s, W, num) generates a trapdoor T_W corresponding to the keyword W to be searched and Search_Outcome(k_s, I, T_W) represents the outcome of the search. Since our scheme uses indeterminisitic/ probabilistic encryption for the trapdoor generation, the generated trapdoor T is also indeterministic and unique for the same keyword searched repeatedly. It is hard for an adversary to map the trapdoor to the keyword or form a relationship between the keyword, trapdoor and index table prior to the search. This also holds true for an adversary maintaining a history of the search and outcome. Hence it satisfies the security Definitions $3.1, 3.2$.

It can be seen that the three leakages are either encrypted, masked or hashed values. Based on the assumption of the master key (K) being secret, the hash cannot be regenerated by an adversary. To be more precise, the hash is hard to invert given the image of an input. Furthermore, we use a probabilistic encryption algorithm for the encryption due to which no meaningful information can be obtained in polynomial time. Therefore, the trapdoor leads to the integer factorization problem which is a hard problem. Therefore our scheme is $(L_{3.1}, L_{3.2}, L_{3.3})$-secure against adaptive/non-adaptive indistinguishability attacks and provides keyword-trapdoor indistinguishability and trapdoor-index indistinguishability.

3.5 Performance Metrics

This section focuses on the algorithmic performance evaluation of the proposed RSE scheme. This is performed two fold; firstly the asymptotic analysis is performed, then the storage overhead is analyzed to highlight the memory consumption of the proposed RSE scheme.

3.5.1 Algorithmic Analysis

The algorithmic analysis presented here is based on the complexity analysis of the target schemes. This analysis is based on upper bound analysis of the set of keywords $(\mathscr{W})$ and set of documents $(\mathscr{D})$. In the asymptotic analysis, the complexities of a set of keywords $(\mathscr{W})$ is denoted by m, whereas the complexity of the set of documents $\mathscr{D}$ is denoted by n. The complexity for hashing is denoted by h and the encryption is denoted by e. As discussed previously, each scheme mainly comprises of 5 phases, *i.e.,* KeyGen, Build_Index, Build_Trap Search_Outcome and Dec phase. KeyGen and Dec phases are fairly identical to the other existing schemes. This is why we skip the comparative analysis of these phases and move onto the Build_Index phase. We extend the analysis of the remaining phases for all the schemes. We analyze our scheme while considering ranking and no-ranking. This way the readers can easily relate and evaluate the efficiency of our scheme compared to existing schemes under discussion.

Table 3.1 Algorithmic Comparative Analysis RSE vs. Existing Schemes

Schemes	Build_Index	Build_Trap	Search_Outcome
[148][147]	$O(mn+3n)$	$O(2hm)$	$O(mn)$
[78]	$O(mn+m)$	$O(m)$	$O(3n)$
[143] & [137]	$O(mn+n)$	$O(m)$	$O(mn)$
Proposed RSE (ranked)	$O(mn+m)$	$O(2h+e)$	$O(mn)$
Proposed RSE (unranked)	$O(mn)$	$O(2h+e)$	$O(m+1)$

Note: For the direct comparison the complexities of h and e are assumed to be 1.

From the complexity analysis of our scheme, it is evident that the Build_Index phase requires $O(mn+m)$. The Build_Trap phase is bound by $O(h+e)$. The Search_Outcome phase is bound by $O(mn)$. We would like to highlight that if we remove the ranking functionality from our scheme then the efficiency of the Build_Index phase increases to $O(mn)$. Whereas, the efficiency of the Search_ Outcome phase can be increased to $O(n+1)$.

Table 3.1 shows the algorithmic comparative analysis of the proposed RSE scheme against the schemes presented in the Section 2.4.1. From the table, it is evident that our scheme is efficient as compared to the existing schemes.

3.5.2 Storage Overhead

As discussed in Section 3.3, the client stores a master key K, a session key k_s and a prime number p. Having the security parameter λ, the prime number p is of the size $\lambda+1$ bits, whereas, the keys k_s and K are of the size 128 bit. Having $\lambda = 160$ bits (obtained from the output of the hash), we get $p = 161$ bits. Hence the client stores $(2 \cdot 128 + 161)/8$ bytes. Thus the client requires 52.125 bytes in terms of storage overhead.

Referring to the CS, the CS preserves the encrypted documents and the secure index table. The storage of the encrypted documents can be represented as $n \cdot D_{avg}$, where n represents the total number of documents and D_{avg} is the average size of the documents. For the secure index, the storage overhead is $8(mn)$ bytes, where

m represents the total number of keywords and n is the total number of documents. Hence the total storage overhead at the CS would be $8(mn) + n \cdot D_{avg}$. In [82], the storage overhead of existing ranked SE schemes is presented. It may be observed that the proposed RSE scheme also outperforms existing schemes in terms of storage overhead.

3.6 Computational Analysis

This section analyses the performance of the proposed RSE scheme by deploying it onto British Telecommunication's cloud service (BTCS) and testing it over a real-world corpus of encrypted documents. Before proceeding towards discussing the computational costs, a few preliminary details about the BTCS architecture, dataset and the specification of the system used for the deployment are given.

3.6.1 Application Encryption Service

The Application Encryption (AE) service is available as a part of the BTCS that implements and offers cryptographic services for its clients. These services include core cryptographic operations like encryption/ decryption based on symmetric ciphers and cryptographic-hash based integrity checking, as well as supporting operations like key management, key storage and key retrieval etc, through a Key Management Service (KMS). The KMS component, in addition to the storage and management of the cryptographic keys, also enforces policy-based access control over the client's keys. The AE service can also be hosted inside the client's premises for their complete control and trust in the cryptographic operations, or the clients can even construct their own version of the AE service from scratch as it is based on open standards and technologies. However, in our deployment the AE service offered by British

Telecommunication (BT) is FIPS-140-1 [111] certified and is treated as a trusted third party.

AE service provides the following features to the clients:

- Centralized control and management of application-layer encryption services through XACML policies.
- NIST [54][97] standard implementation of Advanced Encryption Standard (AES-256) [90], RSA [115], Secure Hash Algorithm (SHA-256) [50] and other cryptographic primitives.
- Provides a library that implements the OASIS PKCS#11 APIs [65], which the clients can integrate in their applications.

Figure 3.2 illustrates a layered architecture that highlights the pre-processing required for the index generation. AE service provides the keys, key management and AES based cryptographic operations to the client. BTCS contains the index table and encrypted set of documents.

Figure 3.3 illustrates the flow of events that take place when the proposed RSE scheme is deployed onto the BTCS using the AE service. The cryptographic primitives are taken from the AE server and the standard SE algorithm is followed.

3.6.2 Dataset Description

The Switchboard-1 Telephone Speech Corpus (LDC97S62) [63] was originally collected by Texas Instruments in 1990-1, under DARPA sponsorship. The first release of the corpus was published by NIST and distributed by the LDC in 1992-3. The Switchboard-1 speech database [62] is a corpus of spontaneous conversations which addresses the growing need for large multi-speaker databases of telephone bandwidth speech. The corpus contains 2430 conversations averaging 6 minutes in length; in other words, over 240 hours of recorded speech, and about 3 million words

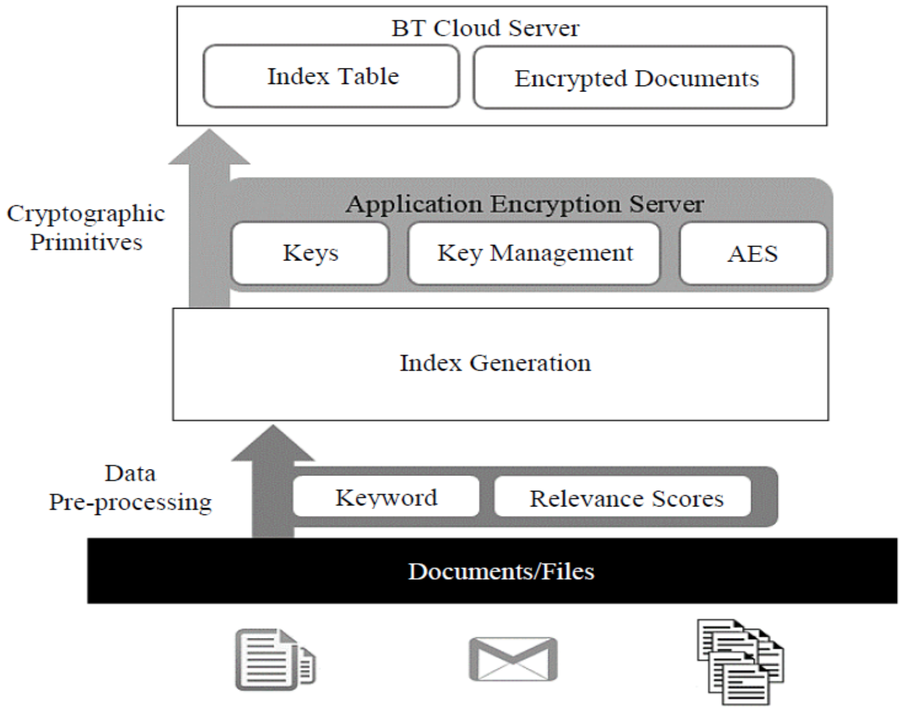

Fig. 3.2 Layered Architecture Inspired by the BTCS and AE Server.

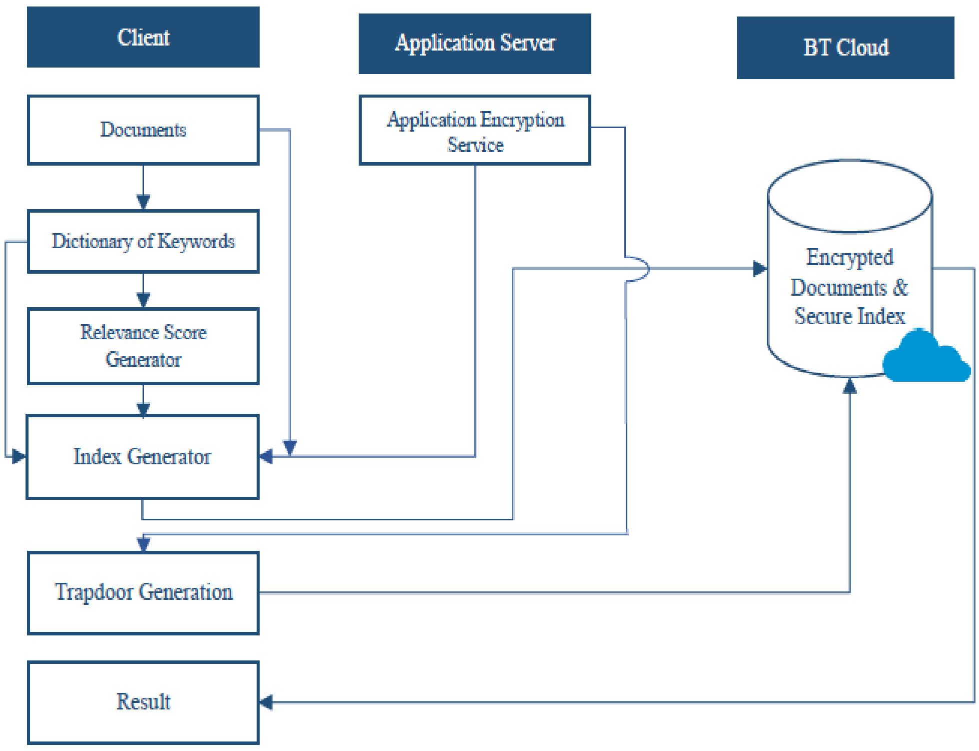

Fig. 3.3 Flow of Events of the RSE Deployed on the BTCS.

of text, spoken by over 500 speakers of both genders from every major dialect of American English. The dataset comprises of 120,000 distinct keywords and 115,998 documents. A time-aligned word for word transcription accompanies each recording. As such it constitutes a realistic dataset of telephone speech, and for this reason the Switchboard-1 transcriptions were used to illustrate the functionality of the SE presented in this research.

3.6.3 System Specification

To demonstrate the feasibility of the proposed RSE scheme, we have implemented our algorithms in Java and present the results in the form of graphs using MATLAB2016. The implementation helps us analyse the time that each phase of the algorithm takes. We have deployed the server side on a public cloud platform (BTCS). The client is able to interact with the AE service. Hence, the results include the network latency occurring as a result of the communication with BTCS and AE service.

- **Client side**

  The workstation used for the client side demonstration runs with an Intel Core i5 CPU running at 3.00 GHz and 8 GB of RAM.

- **Server side**

  The resources allocated at the BTCS include a Dual Core Intel (R) Xeon (R) CPU E5-2660 v3 running at 2.60 GHz and 8GB of RAM.

3.6.4 Implementation Details

The implementation helps us analyze the time that each phase of the algorithm takes while gradually scaling the input (documents or keywords). In order to highlight the cost of encryptions, we have implemented the testbed such that; firstly the client and server side implementation is done on the same machine, and then it is deployed on the BTCS. Hence, initially the analysis does not take into consideration the cost incurred while transferring the documents, index tables or trapdoor over the network, to the CS. Then we deploy the scheme on the BTCS and measure the network latency.

Phases 1-3, 5 of the RSE scheme proposed in Section 3.3 are implemented on the client's system. Phase 4 of the RSE proposed in Section 3.3 is implemented on the BTCS. Therefore, all the searching is performed at the BTCS.

The implementation uses all the algorithms presented in Section 3.3. We achieve confidentiality by implementing 128-bit AES-CBC and the keyed cryptographic hash function used is SHA-128. The dataset used is of the size 2.6GB and it contains 115,998 documents in total.

3.6.5 Computation Overhead

To determine the computation overhead of the proposed RSE scheme, we analyze the performance of individual phases separately. The individual phases are represented in the form of graphs. The graphs are analyzed via the curve fitting/ linear interpolation [86][106] that helps to identify the trend and highlight the R-squared value. The R-squared value [26] is a statistical measure of how close the data are to the fitted regression line. In the analysis of the phases, since KeyGen and Dec phase are fairly identical to that of other schemes, we therefore skip the performance analysis of these phases and shift our focus onto the remaining phases starting from the Build_Index phase.

Build_Index Phase

The Build_Index phase comprises of index generation. After the index table is generated it is transmitted to the CS. We analyze the computation time for the index table generation by running the code on a total of 120,000 distinct keywords identified and extracted from a dataset of 100,000 documents.

Our scheme facilitates both ranked and un-ranked searches depending upon the required functionality and area of application. The proposed scheme provides ranking that comes with an increase in the number of computation performed, resulting in an increase in the computational time. Therefore, we execute ranked and un-ranked index generation separately.

Figure 3.4 shows a graphical representation of the computational time for the index generation (ranked vs unranked) in minutes (min). The solid line represents the time required for the ranked index generation. We execute this phase for a total of 100,000 documents, starting from 10,000 documents and gradually scaling the number of documents to 100,000. For 10,000 documents, the ranked index generation takes a total of approximately 1.3 minutes and that increases to 16.23 minutes for 100,000 documents. The spikes are observed due to the variations in the number of keywords within a document. Due to which the search space increases and more computation is required to calculate the relevance frequencies. For fewer keywords or documents having constant size, these spikes may not appear. The linear trend is highlighted by applying curve fitting/ linear interpolation that shows $y = 0.0002x - 0.1127$ and $R^2 = 0.9443$ for ranked searching.

The dotted line represents the computational time for the unranked index generation. In this case the index table only shows the presence or absence of a keyword within a document. Therefore, the results are not ranked. For 10,000 documents the index generation takes only 0.43 minutes that gradually increases to 5.75 minutes over 100,000 documents. Therefore, it is evident that ranked index generation is more resource consuming as compared to unranked index generation. The linear trend is also depicted for the unranked index generation by applying curve fitting/ linear interpolation that signifies $y = 6E - 05x - 0.488$ and $R^2 = 0.901$.

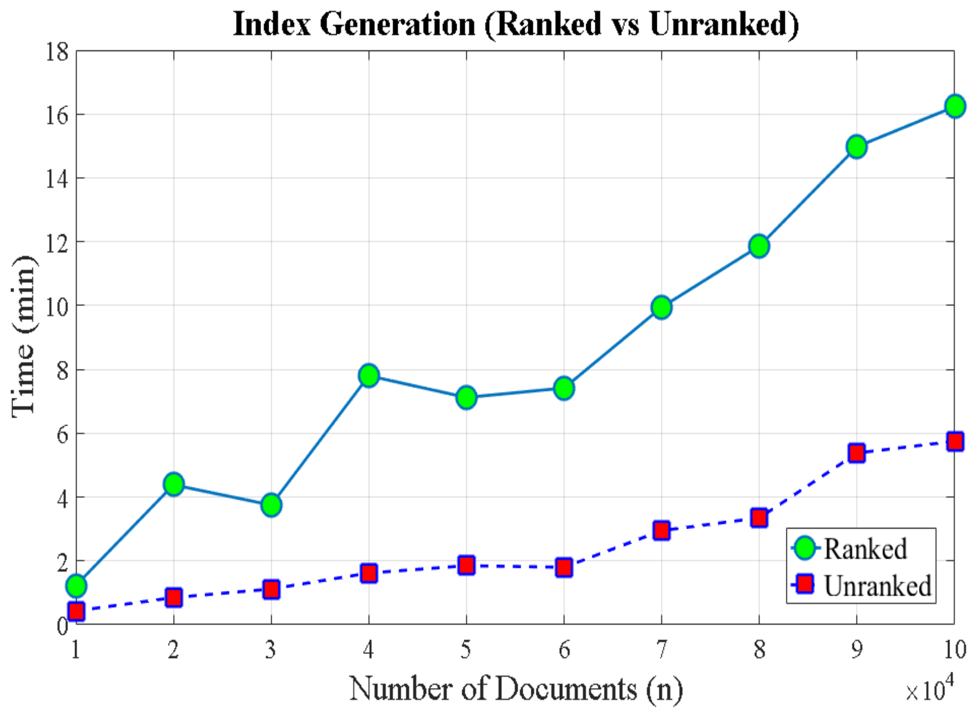

Fig. 3.4 RSE: Computational Time for Index Generation

Build_Trap Phase

As discussed earlier, the trapdoor acts as a search query and is generated by the client for a particular keyword. The generated trapdoor is transmitted to the server and it facilitates the search of the relevant documents. The trapdoor generation is not affected by the ranked or unranked searching so the computational time remains constant. The Build_Trap phase is executed for the keyword “about” and the trapdoor generation takes a constant time of a mere 0.016 seconds. Therefore, the Build_Trap phase is optimized and efficient.

Search_Outcome Phase

Once the encrypted documents along with the index table are uploaded into the CS and the trapdoor has been generated and transmitted to the CS, the next step is the searching of the relevant documents. Figure 3.5, represents the graph generated on executing the Search_Outcome phase against the trapdoor generated for the keyword "about". The number of documents are represented along the x-axis and the time in seconds is along the y-axis. The searching takes a total of mere 3.42 seconds against 100,000 documents and shows a fairly linear growth. The outcome of the search is ranked. The labels on the nodes represent the number of documents that are returned against the trapdoor, containing the searched keyword. For example, out of the total 100,000 documents in the dataset, 98,144 documents contain the keyword "about". It is observed that the searching process is highly efficient and shows a linear trend when the dataset contains 100,000 documents. This trend is observed by applying curve fitting/ linear interpolation that shows $y = 4E - 05x - 0.47$ and $R^2 = 0.9614$.

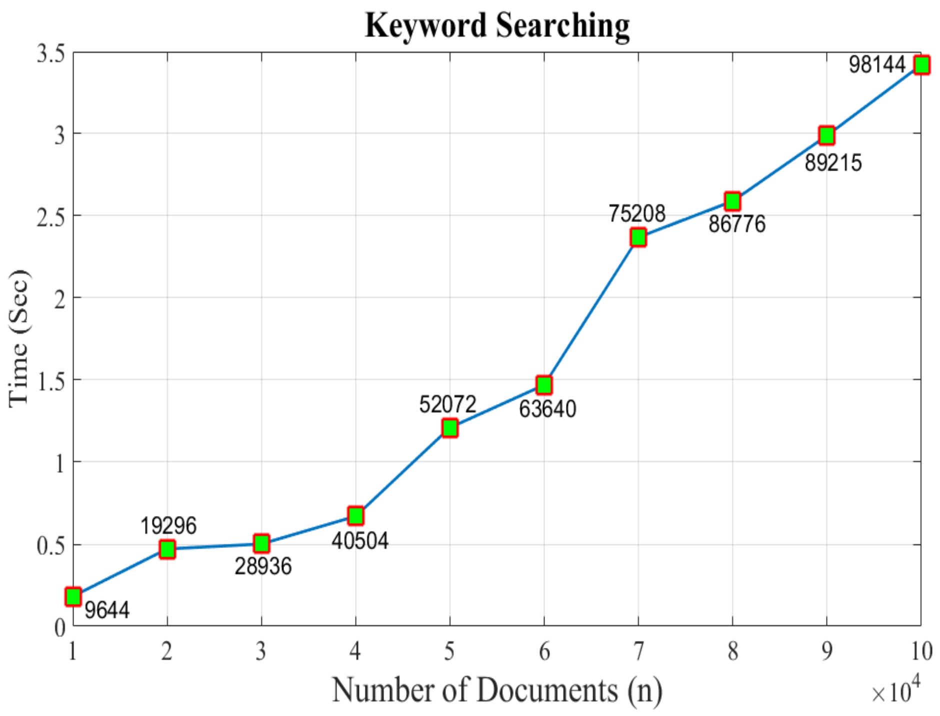

Fig. 3.5 RSE: Computational Time for Searching for the Keyword "about"

To demonstrate the feasibility of the proposed RSE scheme, the scheme is deployed onto the BTCS. As mentioned earlier, the index table and the trapdoor is generated by the client and the searching is done on the BTCS. Figure 3.6 represents the time that the BTCS takes while searching without considering the network latency. Along the x-axis we have the number of documents that are scaled to 2000, whereas along the y-axis we have the time in seconds. While searching for the keyword “about” our RSE scheme takes 0.061 seconds to perform the search for the documents in the ranked order. The curve fitting/ linear interpolation shows a linear trend of $y = 2E-05x + 0.015$ and $R^2 = 0.9661$ when the search is applied over the encrypted documents stored on the BTCS.

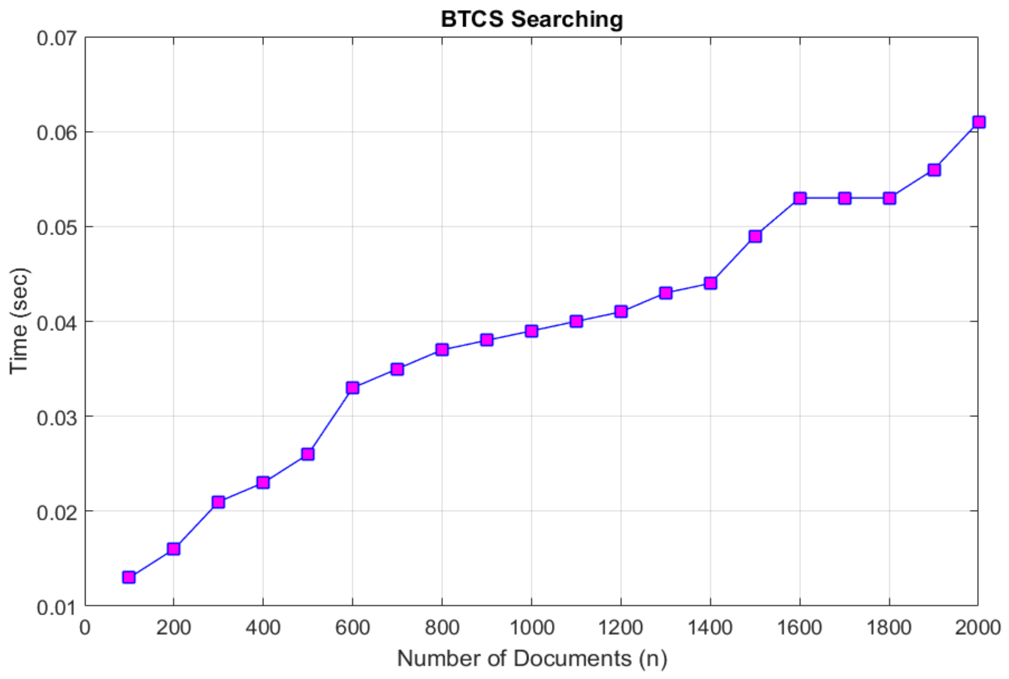

Fig. 3.6 RSE: Computational Time for Searching for the Keyword “about" on the BTCS without Network Latency

In Figure 3.7, we perform the same searching as done in the Figure 3.6 but this time we include the network latency that occurs while communicating with the BTCS. Now, every node represents the number of documents that are returned against the trapdoor generated for the keyword “about”. To search for the keyword “about” over 2000 documents, 1943 documents are returned in ranked order and the search takes 11.4 minutes inclusive of the network latency. The number of documents are represented along the x-axis and the time in minutes along y-axis. Since this graph represents a real-world cloud deployment, the reduction in network latency is depicted in the graph for 1700 documents. This fluctuation observed is not because of the proposed RSE scheme rather it is entirely because of the network

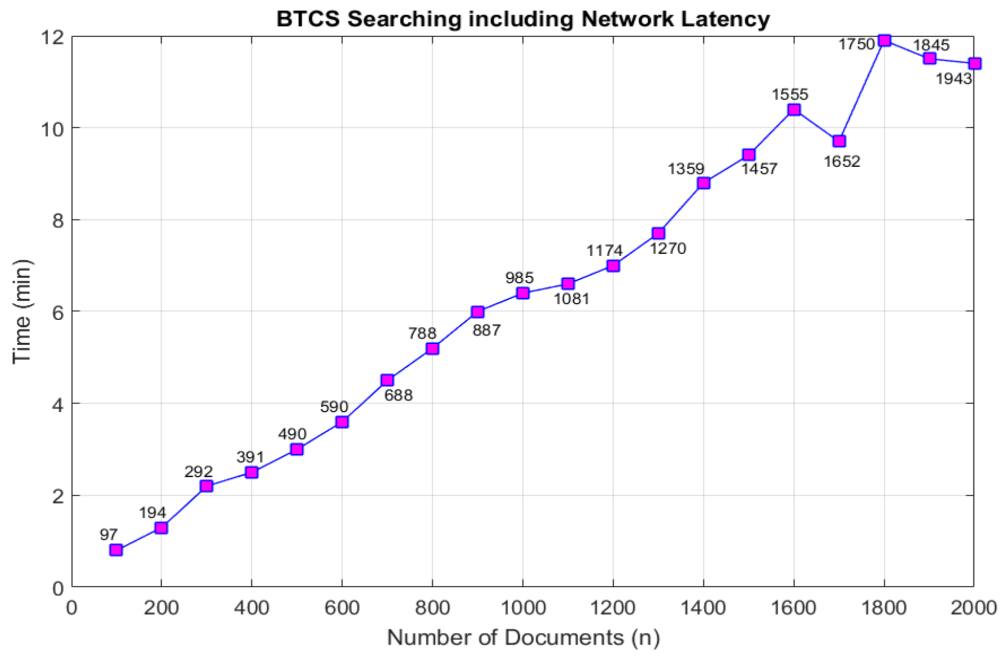

Fig. 3.7 RSE: Computational Time for Searching for the Keyword “about" on the BTCS including Network Latency

latency. On analyzing the graph via curve fitting/ linear interpolation, the graph signifies $y = 0.006x + 0.2321$ and $R^2 = 0.9872$ depicting a linear trend.

3.7 Summary

This chapter has readdressed the problem of supporting keyword search on encrypted data outsourced to the cloud. A novel ranked SE scheme has been presented that exploits the properties of modulo inverse to generate a probabilistic trapdoor. The proposed RSE scheme aims to maintain a balance between security, efficiency and query expressiveness.

In order to perform the security analysis of the proposed RSE scheme, this chapter introduced the concept of indistinguishability and proved the security of the scheme by giving formal proofs to the new security definitions and designing games in the standard model. Since the RSE scheme leads to the integer factorization problem, it enhances the security and reduces the risk of a successful attack and provides security in the standard model. From the security analysis of the construction it is realized that the scheme provides greater security under these proposed definitions as compared to previous schemes. In order to prove the efficiency of the proposed RSE scheme, this chapter presented an asymptotic analysis of existing schemes against the proposed RSE scheme. The results yield that the presented scheme is lightweight and outperforms existing schemes in terms of required computational resources.

In this research, a proof-of-concept prototype is designed and implemented, and tested over a real dataset of encrypted files. The analysis of the result yields that the proposed scheme shows a linear growth with the increase in the input. Therefore, the proposed RSE scheme is optimized and can be deployed onto a real cloud offering. Based on the results the proposed RSE scheme can be termed to be extremely lightweight and resource optimized as compared to the state-of-the-art.

Although this chapter addresses the primary issue of performing keyword search, it is observed that single-keyword searching may not be useful in scenarios where large amounts of data appear as result of a search. The next chapter extends the RSE scheme proposed in this chapter and increases the query effectiveness by supporting multi-keyword disjunctive searching instead of allowing only single keyword searching. To enhance the efficiency, the use of multi-core processors and multi-threads are explored in the next chapter.

Chapter 4

Parallelized Disjunctive Query based Searchable Encryption Scheme

Searching for a keyword over the encrypted data stored in the cloud is desirable and it requires to design sophisticated SE schemes. The single keyword SE scheme presented in the previous chapter is helpful for the databases that mainly comprise of distinct/ unique data tuples. Therefore, the search results could easily be narrowed down to a few. In scenarios where a single keyword search is not helpful in reducing the outcome, multi-keyword searching may be required.

Multi-keyword searching allows a client to search for keywords that collectively reduce the search space as only a few keywords qualify against the search query. There are two types of multi-keyword queries; conjunctive and disjunctive. Conjunctive queries refer to the searching where the keywords are inter-related, on the contrary, a disjunctive query is a search where the keywords are independent and not inter-related. In case of a disjunctive query, one may benefit from the advancements made in the field of multi-core and multi-threading to enhance the search process.

This chapter extends the SE scheme presented in the previous chapter to perform parallelized disjunctive searching. The following contributions to the field of SE are made through this research:

- A novel Ranked Multi-keyword Searchable Encryption (RMSE) scheme is proposed. The proposed scheme is a lightweight (as compared to the existing schemes) and optimized ranked multi-keyword SE scheme that supports disjunctive queries.

- The RMSE scheme is based on probabilistic trapdoors that resist distinguishability attacks. To appreciate the advantage gained from the probabilistic trapdoors, security definitions are proposed and the security of the RMSE scheme is validated.

- The performance is measured two-fold; firstly by implementing the proof of concept prototype and deploying it onto the BTCS. Secondly, tuning the scheme to enable multi-threading to achieve parallelization in a multi-core setup. Both variants are tested over an encrypted telephone speech corpus (real dataset).

4.1 Problem Formulation

This section sheds light on the significance of SE by discussing the system model and the design goals.

4.1.1 The System Model

As illustrated in Figure 4.1, the Ranked Multi-Keyword Searchable Encryption (RMSE) scheme mainly comprises of two entities: Bob (client/ data owner) and the cloud server (CS). Bob is in possession of a corpus containing n documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$. Bob wants to encrypt and outsource his documents to the CS while being able to search for keyword(s) over the encrypted corpus. Instead of single keyword search, Bob is interested in performing disjunctive multi-keyword search over the encrypted corpus while preserving the privacy of the documents and the search. Bob identifies a set of unique keywords $\mathscr{W} = \{W_1, W_2, \ldots, W_m\}$ from the documents $\mathscr{D}$ and forms a dictionary. There is a high probability that Bob's trapdoor (search query) may be disjunctively mapped to multiple documents, therefore, he wishes to retrieve documents based on some ranking mechanism. The ranking comes with an increase in the computational time as discussed in the Section 4.6. The proposed construction also uses Equation 3.1 for ranking the documents.

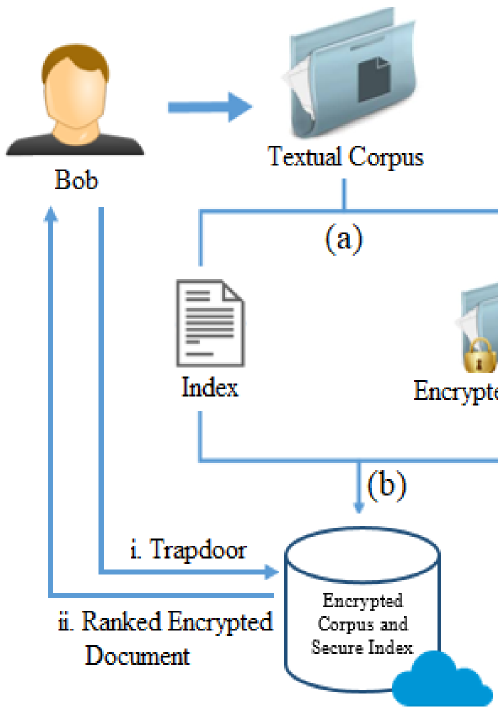
Bob
Textual Corpus
(a)
Index
Encrypted Corpus
(b)
i. Trapdoor
ii. Ranked Encrypted Document
Encrypted Corpus and Secure Index

Fig. 4.1 The System Architecture for the RMSE scheme.

Using the master key K, Bob generates a secure ranked index table (I) and encrypts the corpus (represented as step-a). The index table along with the encrypted documents are outsourced to the CS (step-b). The CS is assumed to be “trusted-but-curious" or “honest-but-curious” (*i.e.*, semi-honest), in other words the CS provides reliable services but it is also interested in learning private information that can be extracted from the outsourced documents, index table or trapdoor. In order to perform a disjunctive keyword search, Bob using his private key generates a valid probabilistic trapdoor and sends it to the CS. The CS splits the trapdoor over different threads that will search in parallel over the secure index table (I) on Bob's behalf and reduce the obtained result to return the encrypted document identifiers in the ranked order, as shown in Figure 4.2.

The benefits of multi-threading are already highlighted in [122]. Multi-threading coupled with SE offers the following advantages:

- Effective resource sharing; especially the resources that will be idle otherwise.
- Accelerated efficiency and reduced search time.
- Utilization of advances made in the multi-core and multi-processor architectures.
- Increase in the responsiveness of the system by performing parallel searches.

From here on, Bob will be represented as a client throughout the remaining of the chapter.

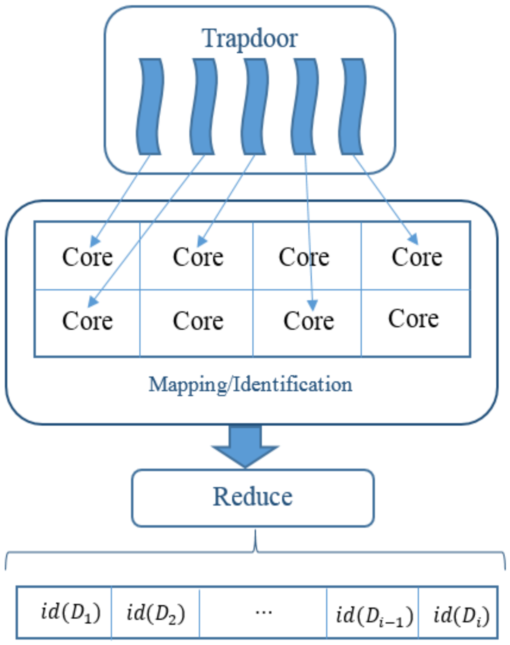

Fig. 4.2 Parallel Searching using the MapReduce Framework.

4.1.2 Design Goals

The proposed construction bears the following security and performance goals:

- *Trapdoor Unlinkability:* The primary goal is that the trapdoor should be probabilistic, therefore, for the same keyword searched again a new trapdoor should be generated. This helps resist distinguishability attacks.
- *Privacy Guarantee:* Apart from the outcome of the search, the CS should not deduce any keyword related information from the secure index and trapdoors. Therefore, the scheme should provide security in the known cipher-text model, *i.e.*, it should provide security against an adaptive polynomial time adversary.
- *Lightweight and parallelizable:* The scheme should be lightweight by default, *i.e.*, based on efficient primitives as compared to the state-of-the-art. To accelerate the performance of the scheme in the disjunctive keyword scenario, the proposed scheme should support parallel search.

Now that the goals have been mentioned, we formally define our RMSE Scheme.

Definition (Ranked Multi-Keyword Searchable Encryption Scheme (RMSE)): The proposed RMSE comprises of five polynomial time algorithms $\Pi =$ (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) such that:

$(K, k_s) \leftarrow$ KeyGen(λ): is a probabilistic key generation algorithm that takes a security parameter λ as the input. It outputs a master key K and a session key k_s. This algorithm is run by the client.

$(I) \leftarrow$ Build_Index$(K, \mathscr{D})$: is a deterministic algorithm that takes the master key K and collection of documents $\mathscr{D}$ as the input. The algorithm returns a secure index I. This algorithm is run by the client.

$T_w \leftarrow$ Build_Trap(K, k_s, w, num): is a probabilistic algorithm that takes the master key K, a session key k_s, set of disjunctive keywords $w \subset \mathscr{W}$, the number (num) of documents $\mathscr{D}$ required as the input. The algorithm returns a trapdoor T_w. The algorithm is run by the client.

$X \leftarrow$ Search_Outcome(k_s, I, T_w): is a deterministic algorithm run by the CS. The algorithm takes the session key k_s, index table I and the trapdoor (T_w) as the input and returns X, a set of desired number of encrypted document identifiers $Enc_K(id(D_i))$ containing the set of disjunctive keywords w in ranked order.

$D_i \leftarrow Dec(K,X)$: is a deterministic algorithm. The algorithm requires the client's master key K and encrypted set of document identifiers $Enc_K(id(D_i))$ to decrypt and recover the document id's. This algorithm is executed by the client.

4.2 Security Definitions

This section extends the security definitions presented in the Section 3.2 to be applied to the proposed RMSE scheme.

4.2.1 Keyword-Trapdoor Indistinguishability for RMSE

Keyword-Trapdoor Indistinguishability is the ability of a SE scheme to hide and dissipate the redundancy in the statistics of the keywords into the trapdoor. This helps to achieve adaptive security, *i.e.*, for the same keywords being searched repeatedly a new and unique trapdoor will be generated. Hence, even if the adversary is maintaining a history of keywords and associated trapdoors, it cannot guess the future searches. Therefore, to guess the keywords or the content of the documents a large amount of data needs to be intercepted in polynomial time.

Description

The challenger generates an index table for the document collection $\mathscr{D}$. The adversary maintains a history containing the set of disjunctive keywords w_i, where $for\, 1 \leq i \leq M; M = |P(\mathscr{W}) - \phi|$ and the trapdoors generated for all the keywords w_i. The adversary submits two keywords (w_0, w_1) to the challenger and receives a trapdoor corresponding to the keyword w_b, where b represents the outcome of tossing a fair coin. The adversary is challenged to output the bit b. If the adversary is able to make a guess with a probability greater than 1/2 then the adversary has succeeded and the scheme lacks in providing keyword-trapdoor indistinguishability. The security parameter λ is negligible *i.e.*, λ is sufficiently small that it can be ignored to highlight the security of the scheme.

Definition 4.1 *Let* RMSE=(KeyGen, Build_ Index, Build_Trap, Search_Outcome, Dec) *be a Ranked Multi-keyword Searchable Encryption Scheme over a dictionary of keywords* $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$, *set of documents* $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$, λ *be the security parameter,* $P(\mathscr{W}) - \phi$ *represent a power set containing* $M = 2^m - 1$ *possible disjunctive members and* $A_{j; 0 \leq j \leq M+1}$ *be a non-uniform adversary. Consider the following probabilistic experiment* $Key\_Trap_{RMSE,\mathscr{A}}(\lambda)$:

```
$Key\_Trap_{RMSE,\mathscr{A}}(\lambda)$
    $(K, k_s) \leftarrow KeyGen(\lambda)$
    $(I) \leftarrow Build\_Index(K, \mathscr{D})$
    $for\, 1 \leq i \leq M; M = |P(\mathscr{W}) - \phi|$
        $(st_{\mathscr{A}}, w_i) \leftarrow \mathscr{A}_i(st_{\mathscr{A}}, T_{w_1}, \cdots, T_{w_i})$
        $T_{w_i} \leftarrow Build\_Trap_K(w_i)$
    $(st_{\mathscr{A}}, w_0, w_1) \leftarrow \mathscr{A}_0(\lambda); (w_0, w_1) \in P(\mathscr{W}) - \phi$
    $b \xleftarrow{\$} \{0, 1\}$
    $(T_{w_b}) \leftarrow Build\_Trap(K, k_s, w_b, num)$
    $b' \leftarrow \mathscr{A}_{M+1}(st_{\mathscr{A}}, T_{w_b})$
```

$$T'_w \leftarrow Build\_Trap_{k_s}(w_l); l \in M$$
$$if\, b' = b, output\, 1$$
$$otherwise\, output\, 0$$

where $st_{\mathscr{A}}$ represents a string that captures $\mathscr{A}$'s state. The keyword-trapdoor indistinguishability holds for all the polynomial time adversaries $(\mathscr{A}_0, \mathscr{A}_1, \cdots, \mathscr{A}_{M+1})$ such that $N = poly(\lambda)$,

$$Pr[Key\_Trap_{RMSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \tag{4.1}$$

4.2.2 Trapdoor-Index Indistinguishability for RMSE

Trapdoor-Index Indistinguishability refers to the complexity offered by a SE scheme that keeps an adversary from identifying the index table entries and documents corresponding to the search query prior to the search. This should hold true even if the adversary maintains a history of keywords, trapdoor and outcome of the searches. Such a property can only be achieved if the trapdoors are probabilistic. Therefore, to guess the index table entry corresponding to a trapdoor and associated keywords, a large amount of data needs to be intercepted in polynomial time.

Description

The challenger generates an index table against a data collection $\mathscr{D}$. The challenger sends adversary the set of disjunctive keywords w_i, where $(w_0, w_1, \cdots, w_i) \in P(\mathscr{W}) - \phi$, the trapdoors generated for all the keywords w_i, along with the associated index table entries $I[0][w_i]$, while maintaining the order in which they occur. Now the adversary submits two disjunctive keywords (w_0, w_1) to the challenger and receives a trapdoor corresponding to the keyword w_b. The adversary has to now decide the corresponding index value and is challenged to output the bit b. If the adversary is able to make a guess with a probability greater than 1/2 then the adversary has succeeded and the scheme lacks in providing trapdoor-index indistinguishability. The security parameter λ is negligible *i.e.*, λ is sufficiently small that it can be ignored to highlight the security of the scheme.

Definition 4.2 *Let* RMSE=(KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) *be a Ranked Multi-keyword Searchable Encryption Scheme over a dictionary of keywords* $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$*, set of documents* $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$*,* λ *be the security parameter,* $P(\mathscr{W}) - \phi$ *represent a power set containing* $M = 2^m - 1$ *possible disjunctive members and* $\mathscr{A} = (A_0, A_1)$ *be a non-uniform adversary. Consider the following probabilistic experiment* $Key\_Trap_{RMSE,\mathscr{A}}(\lambda)$*:*

```
$Trap\_Index_{RMSE,\mathscr{A}}(\lambda)$
    $(K, k_s) \leftarrow KeyGen(\lambda)$
    $(I) \leftarrow Build\_Index(K, \mathscr{D})$
    $for\, 1 \leq i \leq M$
        $let\, I' = I[0][i] = H_K^{-1}(\mathscr{W})$
        $T_{w_i} \leftarrow Build\_Trap(K, k_s, w_i, num)$
        where $(w_0, w_1, \cdots, w_i) \in P(\mathscr{W}) - \phi$
    $b \xleftarrow{\$} \{0, 1\}$
    $(st_{\mathscr{A}}, w_0, w_1) \leftarrow \mathscr{A}_0(st_{\mathscr{A}}, \lambda, w_M, I', T_{w_M});$
    where $(w_0, w_1) \in P(\mathscr{W}) - \phi$
    $(T_{w_b}) \leftarrow Build\_Trap(K, k_s, w_b, num)$
    $b' \leftarrow \mathscr{A}_1(st_{\mathscr{A}}, I_{w_b})$
    $if\, b' = b, output\, 1$
    $otherwise\, output\, 0$
```

where $st_{\mathscr{A}}$ represents a string that captures $\mathscr{A}$'s state. The trapdoor-index indistinguishability holds if for the polynomial time adversaries $(\mathscr{A}_0, \mathscr{A}_1)$,

$$Pr[Trap\_Index_{RMSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \tag{4.2}$$

where probability is over the choice of b.

4.3 Proposed RMSE Scheme

This section presents the RMSE scheme along with the description that helps to explain the scheme.

4.3.1 Scheme Construction

- **Phase 1-KeyGen** (λ)**:** Given a security parameter λ, the client generates the cryptographic keys $K, k_s \leftarrow \{0,1\}^{\lambda}$; where K, k_s are the master key and session key respectively.

- **Phase 2-Build_Index** $(K, \mathscr{D})$**:** Given a set of documents $\mathscr{D}$, dictionary of keywords $\mathscr{W}$, a master key K, hash functions $H(\cdot)$, prime number p of the size $\lambda+1$ bits, random number $R \leftarrow CSPRNG(1^{\lambda})$, the index table (I) is generated by the client and sent to the Server. The index table has the dimensions $(n+1)\times(m+1)$, where, m represents the total number of keywords and n represents the total number of documents. The index table is generated as follows:

  - For $1 \leq t \leq m$:

    - let $a \leftarrow (H_K(W_t))$

    - Compute $I[1][t] = a^{-1}$;

  - For $1 \leq u \leq n$:

    - Compute $I[u][1] = E_K(id(D_u))$;

    - Calculate the RF for W_t occurring in D_u using Equation 3.1 and store as $I[u][t]$;

  - $Mask(RF)$:

```
            - For $1 \leq j \leq m$:
                - Choose $R$ in $Z_p$;
                - $I[n+1][j+1] = I[n+1][j+1] \times R$;
```

- **Phase 3-Build_Trap** (K, k_s, w, num)**:** Given the master key (K), the session key (k_s), a Hash function $H(\cdot)$, desired number of documents (num), the trapdoor T_w is generated by the client as follows:

```
  – let $b \leftarrow Enc_K(w)$
  – For $1 \leq u \leq |w|$, where $w \subset \mathscr{W}$:
            - let $a \leftarrow (H_K(w_u))$;
            - let $c \leftarrow a \times b$;
            - $B[u] = c$;
  – let $d \leftarrow H_{k_s}(b)$;
  – $T_w \leftarrow (d, B, num)$
```

- **Phase 4-Search_Outcome** (k_s, I, T_w)**:** Given a trapdoor T_w transmitted by the client, a session key k_s, a hash function $H(\cdot)$ (same as Build_Trap phase) and the index table I, the search is done by the server and the ranked encrypted documents are returned to the client. The search is done as follows:

```
  – Split trapdoor into $q = |w|$ parts, assign to an individual thread.
  – For each thread do the following:
            - For $1 \leq l \leq size of I$:
                - if $(d == H_{k_s}(B[q] \times a^{-1}))$:
                - $Y[q] = l$
                - Sort the RFs in descending order.
            - $X_w[\,] \leftarrow Enc_K(id(D_i))$
```

  - Add all the X_w into another array Y.
  - Input the arrays Y into the MapReduce framework to obtain the documents containing the keywords. Store the obtained results in an array X.

- **Phase 5-Dec** (K,X)**:** Given the master key (K) and a set X of encrypted document identifiers stored in ranked order, the decryption is achieved using the master key (K).

4.3.2 Description

A brief description of the client side and the server side tasks are given below:

KeyGen Phase

The client triggers the KeyGen algorithm. Having the security parameter λ, the client generates two cryptographic keys, *i.e.*, a master key K and a shared key k_s. The master key is kept secret whereas the shared key is shared with the CS.

Build_Index Phase

This phase is run by the client. The client initializes a prime number p of the order $\lambda + 1$ bits. Using the master key, the client computes the hashes of all the keywords in the dictionary as $H_K(\mathscr{W})$ and stores their inverses in the first row of the index table I. The encrypted document identifiers $E_K(id(\mathscr{D}))$ are placed along the first column of the index table I. Using the Equation 3.1, the client calculates the relevance frequencies that represent the frequency of the occurrence of a keyword within a document and the entire dataset thereafter. The RFs are calculated and placed at the respective location within the index table I. A frequency analysis attack is the study of the frequency of the keywords within a ciphertext or encrypted document. In order to enhance the security of the index table and to prevent frequency analysis attacks the RFs are masked in such a way that the correlation between the RFs remains entire but deters the possibility of an attack. Upon the successful generation of the index table I, it is outsourced to the CS.

Build_Trap Phase

In order to search for a keyword, the client runs the Build_Trap phase. This phase requires a probabilistic encryption algorithm such as AES-CBC to produce probabilistic trapdoors. The algorithm also makes use of the hash function used in the Build_Index phase by taking the master key and the keyword as the input. It is to be noted that the inverse is already present in the index table. So even though the trapdoor is probabilistic but it can easily be mapped to the entry where $a \times b \times a^{-1} = b$. Now the server only needs to compute the hash using the shared key k_s. The Build_Trap algorithm is designed in such a way that it can facilitate single keyword search or multi-keyword searching.The trapdoor is generated and sent to the CS.

Search_Outcome Phase

This algorithm is run by the CS. Depending upon the number of keywords for which the trapdoor T_w is generated, the CS splits the trapdoor into $|w|$ parts among different threads. Since the search can be performed in parallel, the encrypted document identifiers are identified for each thread. Then the results from the individual threads are fed as the input to another thread that reduces to identify encrypted document identifiers in ranked order. The results are shared with the client.

Dec Phase

The client using his master key K decrypts the results to uncover the underlying document identifiers.

4.4 Security Analysis

This section analyzes the security of the proposed RMSE scheme in relation to the security definitions presented in the Section 4.2. Firstly the leakage profiles are discussed to highlight any leakage that the RMSE scheme has and then the formal security proofs are presented.

4.4.1 Leakage Profiles

In order to validate the security of a scheme, it is important to analyze the leakage profile of the proposed RMSE scheme. This helps to examine the affects of the scheme on the security definitions by demonstrating whether the scheme is in line with the security definitions proposed in the Section 4.2. This analysis includes all the artifacts that evolve during the lifetime of the system and we analyze them individually. The leakages $L_{4.1}, L_{4.2}, L_{4.3}$ are associated with the Index Table (I), Trapdoor (T_w) and the Search Outcome (SO) respectively. The leakages are explained below:

Leakage $L_{4.1}$

This leakage is linked with the Index Table I and highlights the information revealed by the index table. The index table is generated by the data owner (client) and outsourced to the CS. This leakage is defined as:

$$L_{4.1}(I) = \left\{ \begin{array}{l} ((H_K(W_m)))^{-1}, Enc_K(id(D_n)), \\ Mask(RF), \, (H_K(W_m)) \\ Enc_K(\mathscr{W}) \in Enc_K(\mathscr{D}) \vee Enc_K(\mathscr{W}) \notin Enc_K(\mathscr{D}) \end{array} \right\} \tag{4.3}$$

Leakage $L_{4.2}$

This leakage is related to the trapdoor generated for a set of keywords (w) and represented by T_w. This leakage is defined as follows:

$$L_{4.2}(T_w) = \{(i = |w|, B[i] \leftarrow Enc_K(W_i)(H_K(W_i))), (d \leftarrow H_{k_s}(W_i)), num\} \quad (4.4)$$

Leakage $L_{4.3}$

This leakage incurs due to the outcome of the search when the CS searches for the disjunctive keywords (w) against which trapdoor is formed. The leakage is defined as follows:

$$L_{4.3}(SO) = \{OC(w), Enc_K(id(D_i))_{\forall T_{w_i} \in D_i}\} \quad (4.5)$$

where OC represents the outcome of the search against the keyword set.

4.4.2 Formal Security Proofs

In order to validate the security of the proposed RMSE scheme, we present a theorem that confirms whether the proposed scheme provides Keyword-Trapdoor and Trapdoor-Index indistinguishability. This leads to a lemma that helps validate the conformance of the proposed RMSE scheme to the privacy preserving property by taking the leakage profile into account.

Theorem 4.1: *RMSE provides Keyword-Trapdoor and Trapdoor-Index Indistinguishability.*

Proof: We now prove that the scheme presented in the Section 4.3.1 provides indistinguishability. The proof is two fold; firstly we prove that the scheme provides Keyword-Trapdoor Indistinguishability and then we prove that it is Trapdoor-Index indistinguishable. We present the game based proofs as follows:

Keyword-Trapdoor Indistinguishability

The term Keyword-Trapdoor Indistinguishability states that an adversary should not be able to distinguish between two trapdoors generated for different keywords. Therefore, a unique trapdoor should be generated for the same keywords being searched again. The game proceeds as follows:

Let RMSE be a ranked multi-keyword SE scheme. Suppose an index table (I) is generated over a dictionary of keywords $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$ extracted from a set of documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$ where $m, n \in \mathbb{N}$. The game is played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$ and comprises of three phases:

- **Setup Phase:** The adversary $\mathscr{A}$ sends a keyword to the challenger $\mathscr{C}$. The challenger runs Build_Trap algorithm and generates a trapdoor corresponding to the disjunctive set of keywords $w \in P(\mathscr{W}) - \phi$ and sends it to the adversary $\mathscr{A}$. This may continue until the adversary has not queried all the possible keyword subsets of $\mathscr{W}$ and received the associated trapdoors. Hence now the adversary has formed a dictionary against all the previous search queries.

- **Challenge Phase:** The adversary $\mathscr{A}$ outputs two set of disjunctive keywords w'_1, w'_2 such that $w'_1 \neq w'_2$. The challenger $\mathscr{C}$ tosses a fair coin $b \leftarrow \{0, 1\}$ and runs Build_Trap phase to generate a trapdoor $T'_{w'_b}$. The trapdoor $T'_{w'_b}$ is sent to the adversary $\mathscr{A}$. The adversary $\mathscr{A}$ is allowed to run the setup phase again and query the same keywords again if interested.

- **Outcome Phase:** The adversary $\mathscr{A}$ returns a guess $b' \in \{0, 1\}$ of b

Adversary's advantage is the measure of how successfully an adversary can attack the cryptosystem. Since the RMSE scheme is based on probabilistic trapdoors, the advantage of the adversary $\mathscr{A}$ in winning the game is defined as:

$$KTAdv_{\mathscr{A}} = |Pr[b' = b] - 1/2| \tag{4.6}$$

The above equation is simplified to give an adversarial advantage of 1/2, since the trapdoors are probabilistic the $Pr[b' = b] = 0$. Therefore the adversarial advantage of the keyword-trapdoor indistinguishability for the RMSE scheme is 1/2. This is according to the definition 4.1 and the equation 4.1 respectively.

Trapdoor-Index Indistinguishability

The term Trapdoor-Index Indistinguishability states that an adversary should not be able to distinguish between two index table entries corresponding to the trapdoors prior to the search. Therefore, a unique trapdoor should be generated for the same keywords being searched again in such a way that the corresponding index entries should not be revealed. The game proceeds as follows:

Let RMSE be a ranked multi-keyword SE scheme. Suppose an index table (I) is generated over a dictionary of keywords $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$ extracted from a set of documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$ where $m, n \in \mathbb{N}$. The game is played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. The game proceeds as follows:

- **Setup Phase:** The adversary $\mathscr{A}$ sends a disjunctive keyword $w \in P(\mathscr{W}) - \phi$ to the challenger $\mathscr{C}$. The challenger $\mathscr{C}$ runs Build_Trap algorithm and generates a trapdoor corresponding to the keyword. The challenger also identifies the index table entries corresponding to the trapdoor and sends the keywords w, trapdoor and index table entries to the adversary $\mathscr{A}$. This may continue until the adversary has not queried all the possible keywords $w \in P(\mathscr{W}) - \phi$ and received the associated trapdoors and index table $I_{\mathscr{W}}$ entries. Hence now the adversary has formed a dictionary against all the previous search queries and associated outcomes.

- **Challenge Phase:** The adversary $\mathscr{A}$ outputs two disjunctive keywords w_1', w_2' such that $w_1' \neq w_2'$ and $w_1', w_2' \in P(\mathscr{W}) - \phi$. The challenger $\mathscr{C}$ tosses a fair coin $b \leftarrow \{0, 1\}$ and runs Build_Trap phase to generate a trapdoor $T'_{w_b'}$. The trapdoor $T'_{w'_b}$ is sent to the adversary $\mathscr{A}$. The adversary $\mathscr{A}$ is allowed to run the setup phase again and query the same keywords again if interested.

- **Outcome Phase:** The adversary $\mathscr{A}$ returns a guess of the index entry $I'_{w'_{b'}}$ such that $b' \in \{0,1\}$ of b.

In the RMSE scheme, the advantage of the adversary $\mathscr{A}$ in winning the game is defined as:

$$TIAdv_{\mathscr{A}} = |Pr[\beta' = \beta] - 1/2| \tag{4.7}$$

The simplification of the above equation leads to an adversarial advantage of 1/2, since the trapdoors are probabilistic the $Pr[b' = b] = 0$. Therefore the adversarial advantage of the trapdoor-index indistinguishability for the RMSE scheme is 1/2. This is according to the definition 4.2 and the equation 4.2 respectively.

Hence, from the equations 4.6 and 4.7 it is inferred that the proposed RMSE scheme provides Keyword-Trapdoor and Trapdoor-Index Indistinguishability.

Lemma 4.1: *The proposed RMSE is a Privacy Preserving SE scheme as it provides* $(L_{4.1}, L_{4.2}, L_{4.3})-$*security and is according to the definitions 4.1 and 4.2.*

Proof Sketch: Theorem 4.1 already proves that the proposed scheme conforms to the Definitions 4.1, 4.2 and provides keyword-trapdoor and trapdoor-index indistinguishability. We now have to examine the impacts of the leakages onto the definition's conformity. Referring to the leakages $(L_{4.1}, L_{4.2}, L_{4.3})$ defined above, similar to the RSE scheme proposed in the Chapter 3, it is observed that most of the leakages are either encrypted or hashed and do not lead to any privacy or security concerns. We also make use of a masking function (represented as $Mask(RF)$ in the Build_Index phase, section 4.3.1) that helps to enhance the security and privacy of the scheme as the masking function helps to deter frequency analysis attacks. The masking function hides the frequencies of the occurrence of keywords while being able to correctly rank the documents. Our proposed masking function can also be replaced with Order Preserving Encryption (OPE). The proposed trapdoors are also probabilistic that help to prevent the distinguishability attacks if eavesdropping takes place. Therefore, the proposed RMSE is privacy preserving.

Lemma 4.2: *The proposed RMSE provides correctness and soundness*

It may be observed that the RMSE scheme is an extension of the RSE scheme presented in the Chapter 3. Similar to the RSE scheme, the proposed RMSE scheme exploits the property of modular inverse to generate probabilistic trapdoors, to perform the search and lead to the exact matching of the disjunctive keywords. Therefore, the proof of this lemma flows directly from the correctness and soundness proof of RSE scheme presented in the Section 3.3.3.

4.5 Performance Metrics

This section discusses the algorithmic performance of the RMSE scheme by performing the asymptotic analysis followed by the storage overhead analysis.

4.5.1 Algorithmic Analysis

The asymptotic analysis of the proposed scheme is performed. This algorithmic analysis includes the Build_Index, Build_Trapdoor and Search_Outcome phases. The analysis is the upper bound complexity analysis. The complexities associated to the set of keywords $\mathscr{W}$ and documents $\mathscr{D}$ is denoted by m and n respectively. Similar to the complexity of the RSE scheme (Chapter 3) presented in the Section 3.5.1, the Build_Index phase of the proposed RMSE scheme takes $O(mn+m)$ for ranked index generation and $O(mn)$ for unranked index generation.

The complexity related to the trapdoor generation is $O(m(2h+e))$ where h and e represent the complexities associated to the hash and encryption. Having c cores,

Table 4.1 Algorithmic Comparative Analysis RMSE vs. Existing Schemes

Schemes	Build_Index	Build_Trap	Search_Outcome
[28]:MRSE_I	$O(md^2)$	$O(d^2)$	$O(md)$
[28]:MRSE_II	$O(md^2+U^2)$	$O(d^2)$	$O(md+U)$
[82]	$O(md^2+d^2)$	$O(d^2)$	$O(d^{'}ds^{'})$
[144]	$O(mLE+m^3M)$	$O(m^2M+E)$	$O((m2+1)M+mE)$
Proposed RMSE (ranked)	$O(mn+m)$	$O(m(2h+e))$	$c/O(m^2n+n)$
Proposed RMSE (unranked)	$O(mn)$	$O(m(2h+e))$	$c/O(m^2+n)$

Note: L is a constant decided by the dataset and E denotes an exponentiation function. U denotes the dummy keywords added to deter frequency analysis attacks, M represents the multiplications and d denotes the number of fields for each record.

the Search_Outcome phase is bound by $c/(O(m^2n+n))$. We would like to highlight that if we remove the ranking functionality the efficiency of the proposed RMSE scheme can be enhanced by $O(mn+n)$. In terms of unranked parallel search the complexity will be $c/O(mn+n)$

Table 4.1 shows the algorithmic comparative analysis of the proposed RMSE scheme against the existing schemes presented in the Section 2.4.2. To achieve direct comparison it is assumed that the complexities of the encryption and hashing are $O(1)$. From the table, it is evident that our scheme is efficient as compared to the other existing schemes.

4.5.2 Storage Overhead

Storage overhead is an important metrics that helps to analyze the memory acquired by the proposed RMSE scheme. The memory consumption is highly dependent upon the underlying data structure. Referring to the Section 4.3.1, it is observed that two keys (*i.e.*, K and k_s) and a prime number p are stored at the client side. Having the security parameter λ, the keys are of the size 128 bit. Since the prime number p is of the size $\lambda+1$ bits, having $\lambda = 160$ bits (obtained from the output of the hash), we get $p = 161$ bits. Hence the total storage at the client side is $161+128\times 2/8 = 52.125$ bytes.

Now we analyze the storage overhead of the CS. It is observed that the CS has to store the session key k_s, the index table I and the encrypted documents $\mathscr{D}$. The session key k_s is similar to the client's, *i.e.*, 128 bits. Given n documents and m keywords, the storage overhead of the index table I is $8(m \times n)$. The storage incurred as a result of the stored encrypted documents is $n \times D_{avg}$, where D_{avg} represents the average size of the document. Hence the storage required at the CS is $128/8 + 8(m \times n) + n \times D_{avg}$ bytes.

As already discussed in the Section 4.1, multi-threading is dependent upon the number of queried keywords. Each thread has to search over the entire index table I so the space complexity in the worst-case scenario would be $O(m \times n)$, where m and n represent the number of keywords and the number of documents respectively.

4.6 Computational Analysis

This section discusses the computational time when the proposed RMSE scheme is implemented and the proof-of-concept prototype is deployed onto the BTCS. The details of the Application Encryption (AE) server can be found in the Section 3.6.1.

Figures 4.3 and 4.4 represent the activity diagrams that illustrate the flow of events that take place during the life cycle of the RMSE scheme when deployed onto the BTCS. It is observed that the index generation is done by the client and the cryptographic primitives are obtained from the AE server to encrypt the documents. Upon the successful encryption of the documents, they are stored in the BTCS along with the secure inverted index table.

Whenever a client wants to search for a keyword, the trapdoors are generated using the client's credentials and then the trapdoor is sent to the BTCS. The trapdoor is fed to the Map Reduce framework and the search is performed over the secure index table. The ranked result is sent back to the client and the related documents

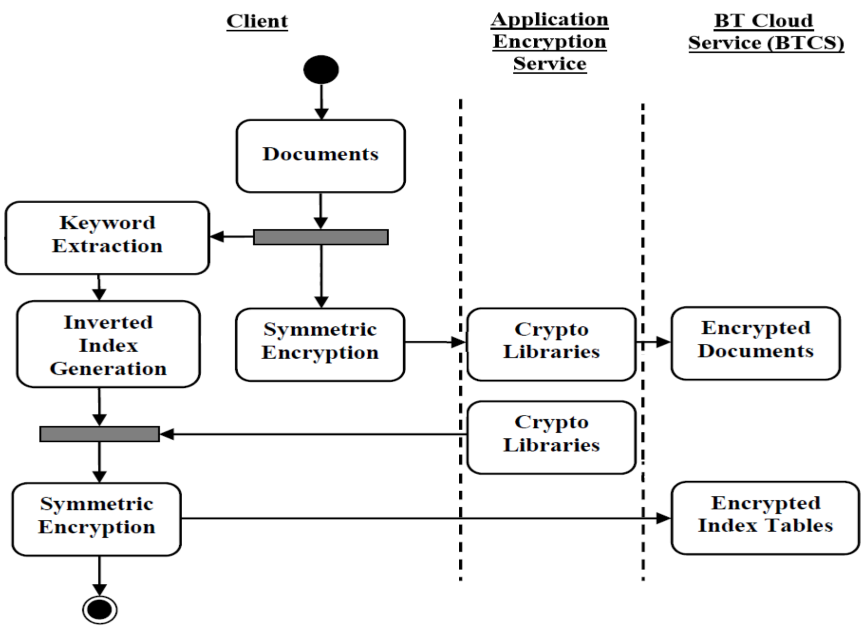

Fig. 4.3 RMSE Activity Diagram: Setup

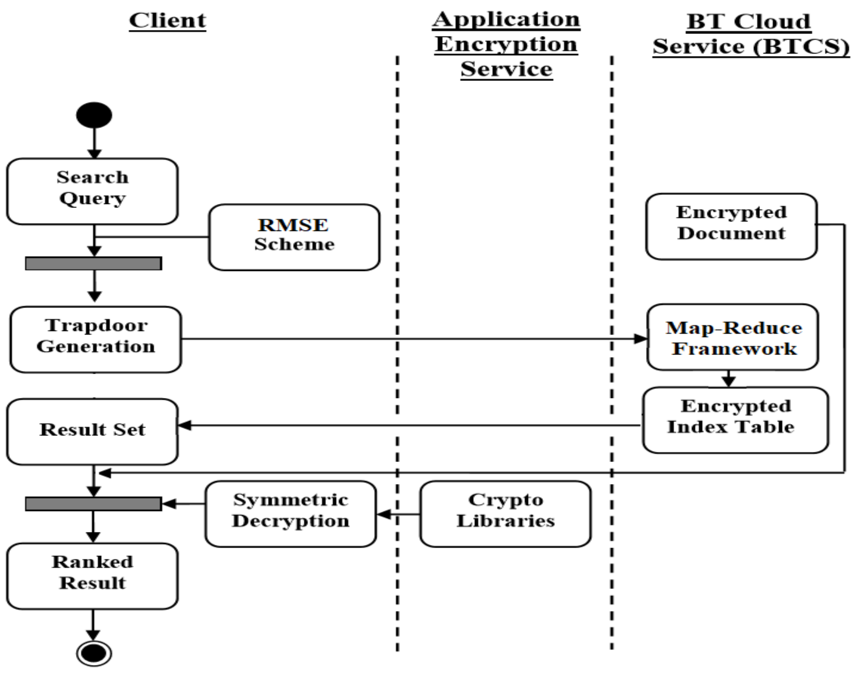

Fig. 4.4 RMSE Activity Diagram: Searching

are downloaded from the BTCS. The obtained encrypted document identifiers are then decrypted using the symmetric key.

4.6.1 System Specification and implementation details

The proposed RMSE scheme is implemented in Java and the results have been generated in Matlab R2016a. The client-server architecture has been implemented on separate machines. The client-side is our workstation and server-side is the BTCS. The communication takes place through sockets.

- Client side: The confidentiality is achieved by implementing 128-bit AES-CBC and the keyed cryptographic hash function used is SHA-256. The specification of the workstation is 2.7 GHz Intel Core i5 processor, 8GB RAM, running at 1867 MHz DDR3.

- Server side: The searching is performed at the BTCS. The resources allocated at the BTCS include a Dual Core Intel (R) Xeon (R) CPU E5-2660 v3 running at 2.60 GHz and 8GB of RAM.

The details related to the dataset used in the implementation and testing of the proposed RMSE scheme can be found in the Section 3.6.2.

4.6.2 Computation Overhead

To analyze the performance of the proposed RMSE scheme in a multi-threaded environment, we analyze each of the phases (already discussed in Section 4.3.1) separately. Since the KeyGen phase and the Dec phase are identical to other schemes, we do not evaluate them. The performance estimation for the remaining phases is discussed below:

Build_Index Phase

Figure 4.5 graphically represents the computational time of the proposed RMSE scheme for the index generation. With the addition of ranking to the scheme the query effectiveness increases but as a result the performance also decreases. Therefore, we measure the computational time for the ranked and unranked index generation separately. We start by generating the index for 100 documents and gradually scale them to 2,000 documents while keeping the number of keywords fixed to 120,000. We experience several peak values in the graphs due to the inconsistency among the documents, *i.e.*, due to the different size of documents. The number of documents are represented along the x-axis whereas the time in seconds is along the y-axis. The curve fitting/ linear interpolation when applied to the graph tends to show a linear trend of $y = 0.01x - 1.4898$ and $R^2 = 0.9294$ for ranked searching and $y = 0.0047x + 0.5343$ and $R^2 = 0.8106$ for unranked searching. It is also observed that although in general our construction shows a linear growth, however, unranked index generation is efficient as compared to the ranked index generation. The ranked scheme for 2,000 documents takes 20.3 seconds for the index generation, whereas, the unranked index generation takes 11 seconds. Hence, in scenarios where ranking is not required the proposed RMSE construction, modified as unranked, can ensure much better results. It is to be noted that the index generation is a one time process and does not need to be generated for every query.

Build_Trap Phase

Figure 4.6 illustrates the computational time that the proposed RMSE scheme requires for the trapdoor generation. We start with 1 keyword and scale it to 5 keywords. The time in seconds is represented along the y-axis, whereas, the number of keywords are along the x-axis. The rise in the computational time is experienced while generating a trapdoor for two keywords. The plaintext input (keywords) exceeds the 128bit AES block size, and therefore this rise depicts the padding of the blocks for the AES calculations. A slight drop in the computation time is observed once the computations are complete and the CPU resources are free. It can be seen that the proposed RMSE scheme on applying graph normalization tends towards a constant

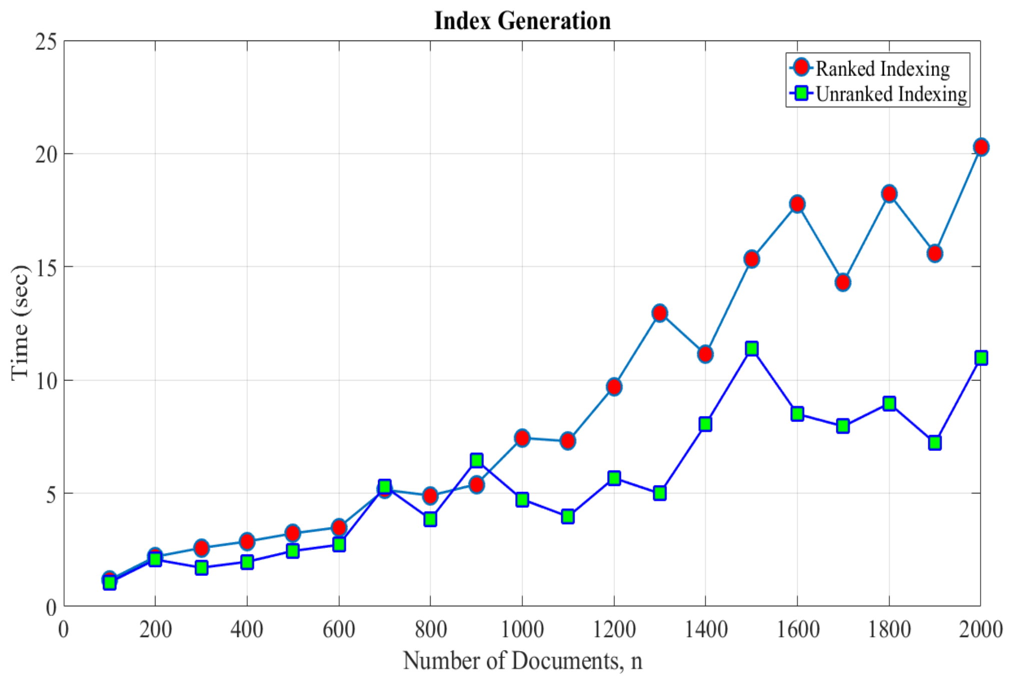

Fig. 4.5 RMSE: Computational Time for the Index Generation.

trend with the increase in the number of disjunctive keywords. This is also evident in the readings depicted in the graph. Hence for generating a trapdoor containing 5 keywords, we require 5.21 seconds.

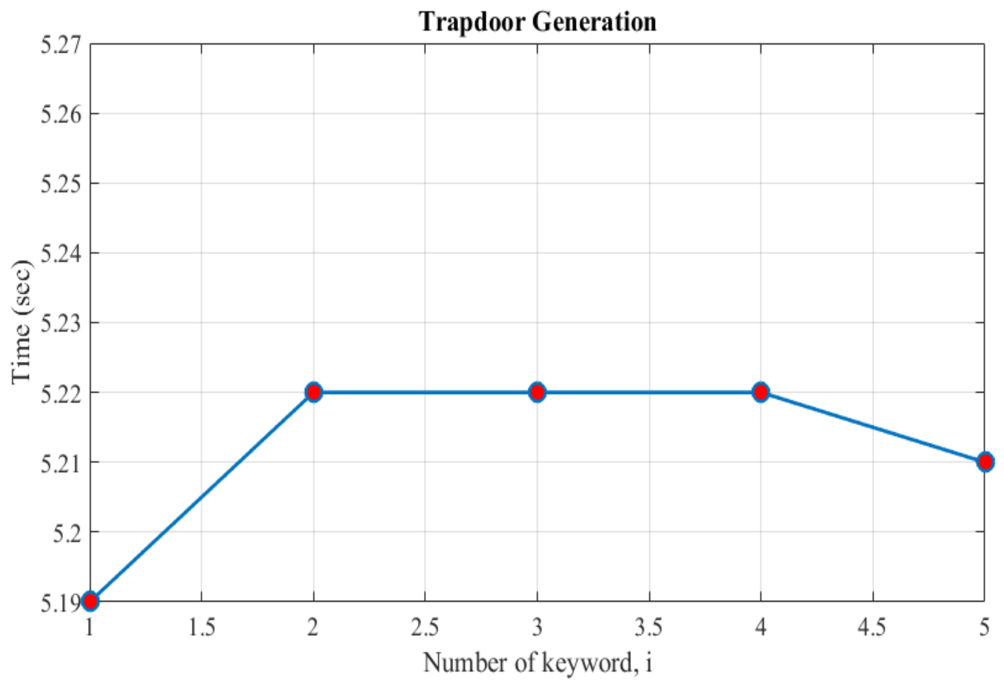

Fig. 4.6 RMSE: Computational Time for the Trapdoor Generation.

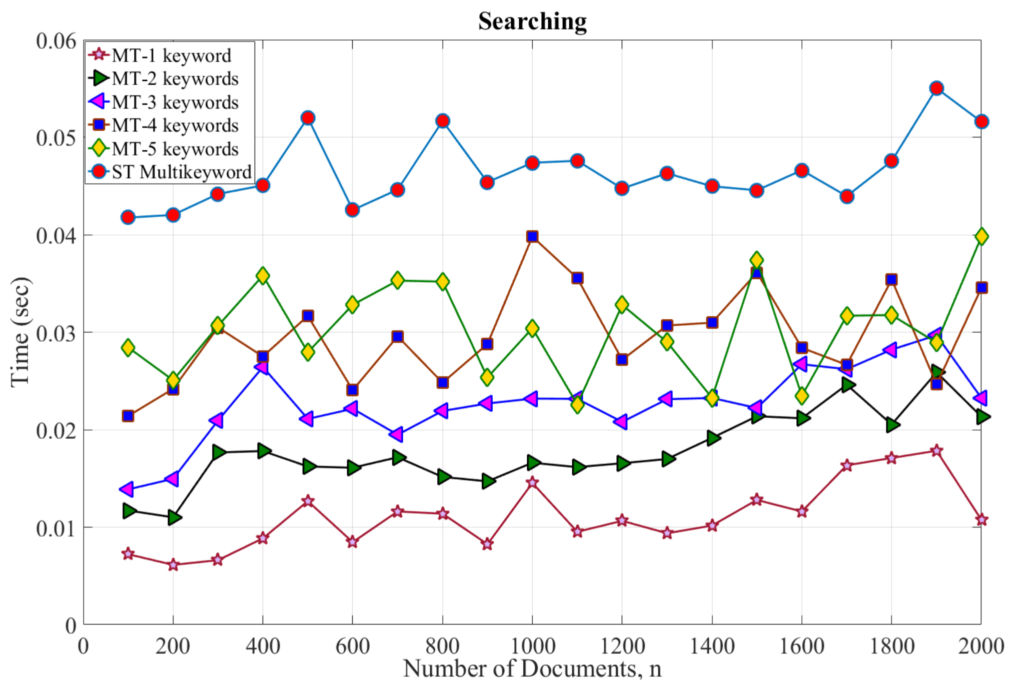

Fig. 4.7 RMSE: Computational Time for Multi-threaded Search.

Search_Outcome Phase

Search_Outcome phase refers to the computational time required for performing disjunctive search and obtaining ranked documents. We delegate the search across multiple threads. The threads search for the individual keywords and reduce the result from all the threads to an outcome. The total number of threads used are equal to the number of keywords in our search query.

Figure 4.7 shows the computational time while searching for disjunctive keywords. In the graph, ST represents single-thread and MT represents multi-threads. Firstly we search for two keywords “about time" using a single thread. Later on we perform search over multiple threads. It can be observed that the efficiency increases by extending the search over multiple threads as compared to the single-threaded search. The increase in the efficiency is due to the parallel and concurrent execution of the search algorithm across multiple threads. Since the designed system supports multi-threading architecture, the computational time of the algorithm can be interpreted as a function of the number of threads. For multiple threads the keywords

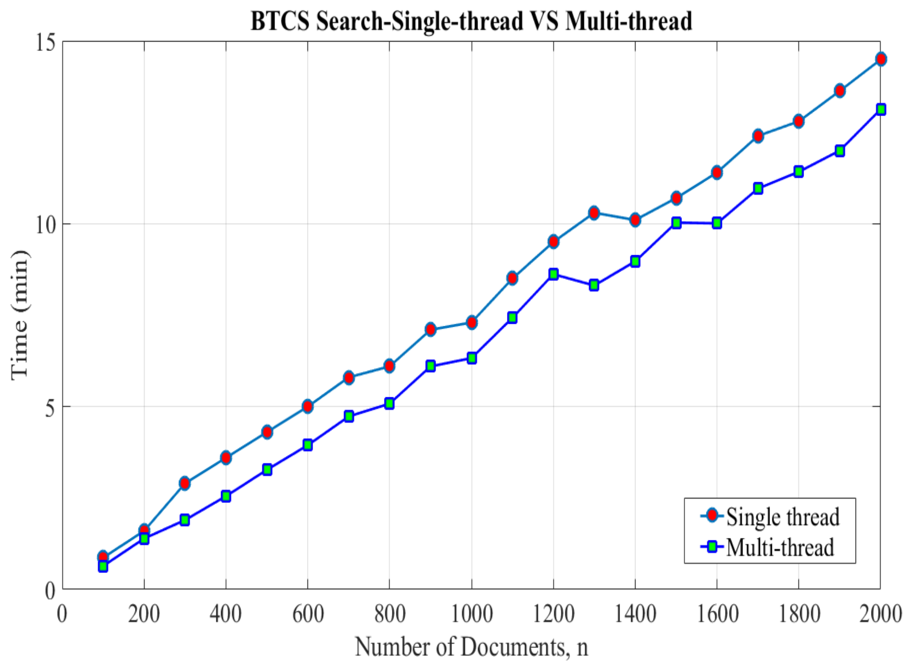

Fig. 4.8 RMSE: Computational Time for Search on the BTCS Including the Network Latency.

used are {about, time, and, is, or}. Each thread processes a particular keyword to be searched across the database. The number of documents are represented along the x-axis and the time in seconds is along the y-axis. We observe the peak values across the results because of the varying size of the documents, *i.e.*, with the increase in the size of the documents the number of keywords increases, resulting in an increase in the search space. For 5 keywords the multi-threads takes a maximum of 0.04 seconds, whereas, for 2 keywords the single threaded system takes 0.052 seconds. All the searches have been performed over 2,000 documents.

It is evident that by using multiple threads the performance increases significantly. We now deploy the RMSE scheme onto the BTCS. To do a comparison, we deploy the RMSE (Multi-threading) scheme and RMSE (Single-threading) scheme. The communication between the client and the BTCS takes place through sockets, therefore, the results also include the network latency incurred during the communication. The network latency also includes the communication with the AE Service.

Figure 4.8 shows a comparison among the single-thread RMSE scheme and the multi-thread RMSE scheme on the BTCS. We again validate our claim of the multi-thread RMSE scheme being more efficient than the single-thread RMSE scheme. The number of documents are presented along the x-axis and the time in minutes is along the y-axis. For the query “about time" the single-thread search takes 13.49 minutes, whereas, the multi-thread search takes 13.1 minute. With the increase in the number of queried keywords in the multi-threading setting, we believe that we will not see an exponential growth in the computational time of the scheme when deployed onto the BTCS. The curve fitting/ linear interpolation highlights a linear trend of $y = 0.007x - 0.04$ with $R^2 = 0.9577$ for multi-threading and $y = 0.0067x + 1.0342$ with $R^2 = 0.9945$ for single-threading. Therefore, by extending the proposed scheme over multiple threads we are able to enhance the efficiency of the proposed RMSE scheme.

4.7 Summary

SE is an approach that allows to search for keyword(s) over the encrypted set of documents. With the reliance onto the cloud, users are outsourcing enormous amounts of data to the cloud. Since the data continues to increase this gives rise to the term “Big data". Prior SE schemes fail to maintain efficiency when extended over the Big Data because they primarily perform sequential search over the data. Searching for a single-keyword may not be adequate in certain scenarios where the documents are closely related. Hence, complex queries may be required to achieve effective search. This leads to a compelling case to have a parallel search mechanism that allows to search for multiple keywords over large amounts of data using parallel processing.

This chapter presented a disjunctive keyword-based SE scheme that could perform parallelized searching by benefiting from the multiple cores and processors.

The scheme supported the map-reduce framework and was implemented over multiple threads. The extension may have led to security and privacy concerns, but formal security analysis yields that the scheme still maintains the property of probabilistic trapdoors. The performance of the proposed RMSE scheme was measured two-fold; firstly the asymptotic analysis was performed, followed by the storage overhead analysis. It was observed that the proposed RMSE scheme outperformed the existing schemes. To perform the computational analysis, the proposed scheme was developed and the PoC prototype was deployed onto the BTCS. The implementation was then tested over a real-dataset and the performance gain between the primary scheme and the multi-threaded scheme was measured. It was observed that the efficiency of the disjunctive keyword searching could be enhanced further by taking advantage of the multi-core processors and multi-threading respectively.

The schemes proposed until now do not take the human typographical errors or closely related keywords into consideration. Therefore, these schemes perform exact matching and may return a null result in case the queried keyword is misspelt. To address this problem, the next chapter presents a fuzzy-based SE scheme that takes human typographical errors into account and allows similarity search instead of exact matching.

Chapter 5

Fuzzy Ranked Searchable Encryption Scheme

Cloud storage is a service that enables clients to maintain, manage and backup their data onto the cloud. To achieve confidentiality of sensitive data stored in the cloud, the data is encrypted prior to being outsourced to the cloud. This gives rise to the problem of searching and sifting through the large amounts of encrypted data. Although, searchable encryption makes the searching over the encrypted data possible, from time to time it may be observed that while searching for a keyword the search result is null. This may be due to a typographical error in the search query or alternative spellings (such as the American and British English spelling differences). The schemes discussed in the previous chapters do not take the human typographical errors or spelling differences into account and rely on the exact matching of the trapdoor to a keyword already existing in the dictionary.

Fuzzy logic [93][161] is an approach that instead of the usual ’true or false’ or boolean logic considers the degree of truth. The use of fuzzy logic in SE allows us to take the human typographical errors and spelling variations into account. This can benefit law enforcement agencies (LEAs) to perform search when the spellings of an individual’s name may not be known.

The following contributions to the field of SE are made through this research:

- This chapter presents a novel Fuzzy Ranked Searchable Encryption (FRSE) scheme. The proposed scheme is privacy preserving by design and provides privacy in the known cipher-text model. The proposed FRSE scheme is based on probabilistic trapdoors that resist distinguishability attacks and preserve the privacy of the outsourced documents and search queries.
- The proposed FRSE scheme is designed in such a way that it is pluggable with any existing single-keyword or multi-keyword SE schemes.
- Searchable Encryption-as-a-Service (SEaaS) is achieved in terms of fuzzy searching for the existing cloud architectures by designing and developing a proof-of-concept prototype and testing it over an encrypted real-world corpus of speech transcriptions outsourced to the BTCS.

This chapter discusses the FRSE architecture and the important design primitives that are used to achieve fuzzy searching. The security definitions are tuned according to the FRSE scheme and the formal security proofs are presented. This chapter also presents the algorithmic analysis and the computational complexity analysis of the proposed FRSE scheme.

5.1 Problem Formulation

A Fuzzy Ranked Searchable Encryption (FRSE) scheme is designed so that it can be deployed on the cloud server (CS). The CS is assumed to be an honest-but-curious component of the system. The entire architecture is explained with the help of the following scenario:

5.1.1 System Model

Bob outsources his encrypted documents $\mathscr{D}=\{D_1, D_2, \cdots, D_n\}$ to the CS. He wants to perform fuzzy search over the encrypted set of documents. Typically, a SE algorithm does not take the user's typographical errors into account. Therefore, for the wrongly spelt keyword (e.g., “aboot" instead of “about"), the search outcome will return null. On the contrary, a fuzzy SE algorithm will still successfully infer a meaningful keyword from the misspelt keyword. So the search result will include the documents containing the keywords “about”, “abort” etc.

To perform efficient search, Bob will extract a dictionary of keywords $\mathscr{W} =\{W_1, W_2, \cdots, W_m\}$ from $\mathscr{D}$ and form an inverted index table and fuzzy index table respectively. The inverted index table (I) makes use of relevance score generator for the ranking of the documents, whereas, the fuzzy index table (FI) enables the fuzzy matching of the keywords. The formation of the secure index table (I) is based on the use of cryptographic primitives. Upon the successful generation of the index tables (I, FI), Bob outsources the index tables along with the encrypted set of documents to the CS. This is a one time process.

As mentioned above, the CS is assumed to be “honest-but-curious” or “trusted-but-curious”. Here honesty means that the CS performs all its operations properly and correctly, but it is also curious to learn any information that it can infer from the outsourced documents it is storing on behalf of the client, or from the search queries it receives from the client. In future, to search for a keyword over the encrypted set of documents, Bob uses his private key to generate a probabilistic trapdoor and send it to the CS. The probabilistic trapdoor is unique even for the same keyword being searched repeatedly, hence, it resists distinguishability attacks. Now the CS using the trapdoor performs the search over the index tables and returns the results, *i.e.*, the encrypted document identifiers in a ranked order. Figure 5.1 illustrates the flow of events that take place during the entire life cycle of the FRSE scheme.

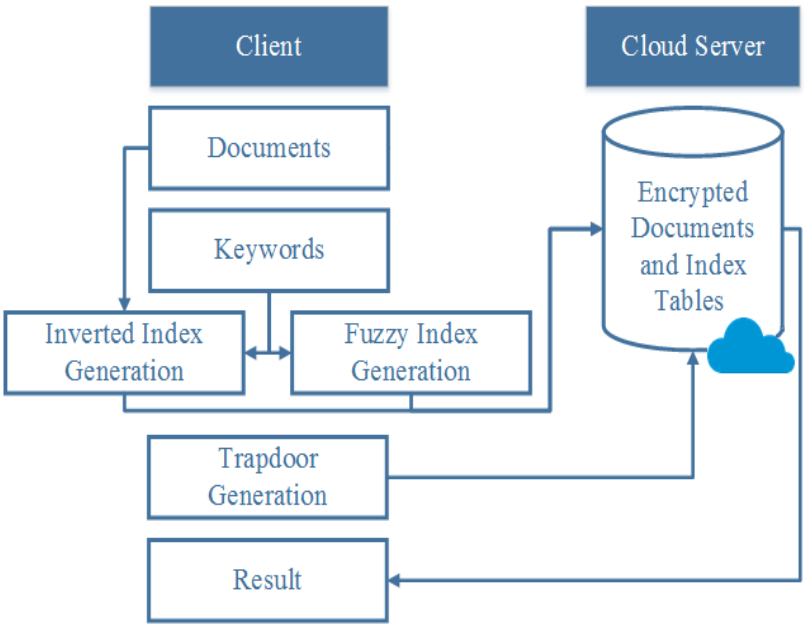

Fig. 5.1 The System Architecture for FRSE Scheme

Definition (Fuzzy Ranked Searchable Encryption Scheme (FRSE)): The proposed FRSE scheme comprises of five polynomial time algorithms $\Pi =$ (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) such that:

$(K, k_s, r) \leftarrow \text{KeyGen}(\lambda)$: is a probabilistic key generation algorithm that takes a security parameter λ as the input. It outputs a master key K, a session key k_s and a random number r. This algorithm is run by the client.

$(I, FI) \leftarrow \text{Build\_Index}(\mathbf{S}', \mathscr{D})$: is a probabilistic algorithm that takes a randomly permuted shingle vector $\mathbf{S}'$ and collection of documents $\mathscr{D}$ as the input. The algorithm returns an inverted index table I and a fuzzy index table FI. This algorithm is run by the client.

$T_W \leftarrow \text{Build\_Trap}(\mathbf{S}', k_s, W, num)$: is a probabilistic algorithm that takes a randomly permuted shingle vector $\mathbf{S}'$, session key k_s, keyword(s) W, number (num) of required documents as the input. The algorithm returns a trapdoor, T_W. The algorithm is run by the client.

$X \leftarrow$ Search_Outcome(k_s, I, FI, T_W): is a deterministic algorithm run by the CS. The algorithm takes the session key (k_s), inverted index table (I), fuzzy index table (FI), the trapdoor (T_W) as the input and returns (X), a set of required encrypted document identifiers $Enc_K(id(D_i))$ in ranked order.

$D_i \leftarrow Dec(K, X)$: is a deterministic algorithm. The algorithm requires the client's master key K and encrypted set of document identifiers $Enc_K(id(D_i))$ as input to decrypt and recover the document ids. This algorithm is executed by the client.

It is important to examine the correctness that a scheme has to offer. We revisit the definition on correctness proposed in [132] and extend it according to the FRSE scheme.

Correctness: An FRSE scheme is correct if for the security parameter λ, the master key K, the session key k_s and the random number r generated by the KeyGen(λ), for (I, FI) output by Build_Index($\mathbf{S}', \mathscr{D}$), the search against the trapdoor T_W always returns the correct set of ranked encrypted document identifiers $Enc_K(id(D_i))$ to the client. A FRSE scheme is correct if the following hold true:

- If $W \in D_i$ then the following should hold with an overwhelming probability:

$$Search\_Outcome(k_s, I, FI, T_W) = \mathscr{D} \cap Dec(K, X) = D_i; \tag{5.1}$$

$$\text{where } 1 \leq i \leq n$$

- If $W' \notin D_i$ then the following should hold with an overwhelming probability:

$$Search\_Outcome(k_s, I, FI, T_{W'}) = \mathscr{D} \cap Dec(K, X) = D_i; \tag{5.2}$$

$$\text{where } 1 \leq i \leq n,\ W' {\sim} W,\ W \in D_i$$

5.2 Preliminaries

This section presents a brief overview of the design primitives that lead to the conceptualization of the proposed FRSE scheme.

5.2.1 Design Primitives

The design of the proposed scheme is primarily based on several important primitives *i.e.* Relevance score, Shingling [25][9], Min Hashing [107], Jaccard Similarity [162] and Euclidean Norm [31]. A brief introduction is given below:

Relevance Frequency

The relevance frequency (RF) helps to rank the documents. This scheme uses the equation 3.1 for the ranking of the documents (further details on RF generation can be found in the Section 3.1.1).

Inverted Index Table

An inverted index table I is a matrix of the order $(n+1) \times (m+1)$. All the entries are initially set to zero. Given a set $\mathscr{D}$ of documents $\{D_1, D_2, \ldots, D_n\}$ and a set $\mathscr{W}$ of keywords $\{W_1, W_2, \ldots, W_m\}$, the entry located at the position $(1, j+1)_{1 \leq j \leq m}$ contains the corresponding masked keyword identifier $mask(id(\mathscr{W}))$. Moreover, the entry located at $(i+1, 1)_{1 \leq i \leq n}$ contains the corresponding encrypted document identifier $Enc_K(id(\mathscr{D}))$. The remaining entries of the index table I represent the corresponding RFs generated by the equation 3.1.

Shingled Keywords

Given a set $\mathscr{W}$ of keywords $\{W_1, W_2, \ldots, W_m\}$, let l be a constant that represents the sequence of l characters to appear within the keywords $\mathscr{W}$. The choice of l is made such that the probability of any shingle appearing within a keyword should be low.

The keyword is firstly transformed into a shingle set S consisting of contiguous l characters appeared in the keyword. The shingle set is converted to a vector $\mathbf{S}$ that represents the presence or absence of a particular shingle. Hence for each shingled keyword, a vector $\mathbf{S}$ of length γ^l-bit is required, where γ represents the characters, *i.e.*, the entries of the vector $\mathbf{S}$ represent the occurrence of shingles within a keyword.

Min Hashing

Min Hashing [107] in SE is defined as; given q number of random hash functions (*i.e.* random permutations), represented as $f_q : \mathbf{S} \rightarrow R$, Min Hashing reduces the shingle vector $\mathbf{S}$ and assigns a real number R to form a signature vector $(\mathbf{SV})$. Let $\mathbf{S}_a$ and $\mathbf{S}_b$ represent two shingle vectors for two different keywords. The random hash function permutations should satisfy the condition: $f_q(\mathbf{S}_a) \neq f_q(\mathbf{S}_b)$. Hence the permutations are independent. The Min Hash value of any vector is the number of first row, in the permuted order, in which the vector has a 1. As a result a signature vector (**SV**) is formed.

Min hashing and Jaccard similarity

With reference to [75], Min Hashing and Jaccard similarity *(JS)* are closely related as follows:

- Given two shingled vectors $\mathbf{S}_a$ and $\mathbf{S}_b$. The probability of $JS(f_q(\mathbf{S}_a), f_q(\mathbf{S}_b))$ is equal to the $JS(S_a, S_b)$, where vectors $\mathbf{S}_a$ and $\mathbf{S}_b$ have been converted to a set.

The Jaccard similarity between two sets X and Y is calculated by:

$$JS(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{5.3}$$

Corollary 5.1: *The Jaccard Similarity between two sets X and Y is 0 if and only if $X \cap Y = \phi$.*

Euclidean Norm

Given two vectors $\mathbf{A} = [a_1, a_2, \cdots, a_i]$ and $\mathbf{B} = [b_1, b_2, \cdots, b_i]$, the Euclidean Norm d represents the distance between the vectors $(\mathbf{A}, \mathbf{B})$. The distance between the two i-dimensional vectors **A** and **B** is represented as:

$$d(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{g=1}^{i} (a_g - b_g)^2} \tag{5.4}$$

5.2.2 Scheme Conceptualization

Prior to presenting the proposed scheme, this section introduces the crux that help to conceptualize the scheme. This section is explained keeping in view the public cloud infrastructure. Before going into the explanation a toy example is presented in Figure 5.2 to illustrate the significant FRSE phases.

- **Step 1** is to transform a keyword into a shingle vector.
- **Step 2** is to permute a random vector from the shingle vector.
- **Step 3** is to form a Fuzzy Index table comprising of the signature vectors after applying q permutations for the Min Hashing.
- **Step 4** is to calculate the RFs and form an inverted index by masking the RFs.
- **Step 5** is to generate a trapdoor, form a shingle vector and a randomized vector. Then using the q permutations calculate the Min Hashes and compute the Euclidean Norm between the two vectors.
- **Step 6** is to compute the Jaccard Similarity between the Fuzzy index entries and the trapdoor.
- **Step 7** is to obtain documents in ranked order.

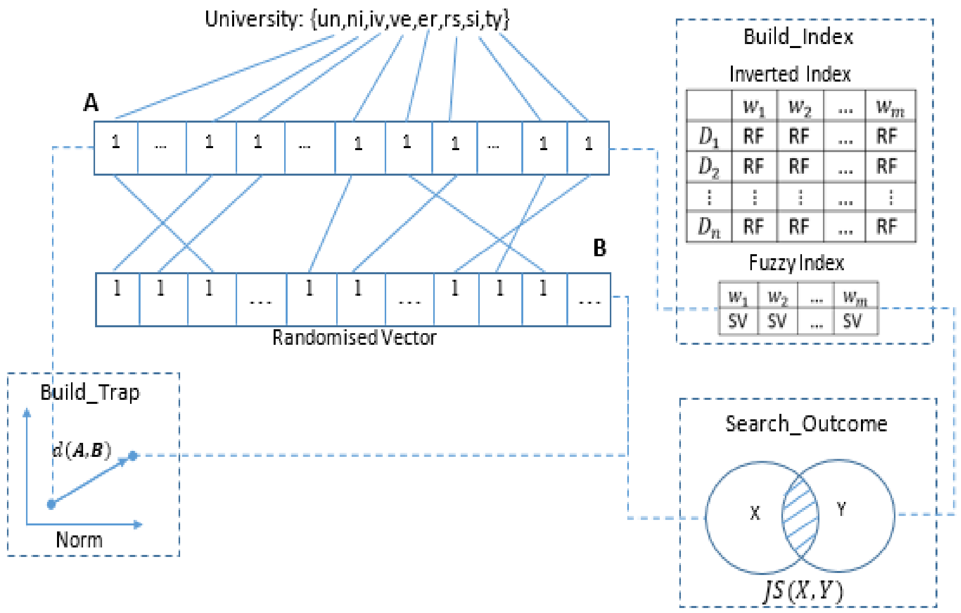

Fig. 5.2 FRSE: Toy Example

Inverted Index Table Generation and Document Encryption

The scheme identifies a dictionary of keywords $\mathscr{W}$ from the document set $\mathscr{D}$. The inverted index is formed such that the corresponding entries are populated using the RF generation mechanism (equation 3.1). On the successful generation of the index table I, the documents are encrypted using the client's master key K.

Fuzzy Index Generation

The fuzzy index table (FI) generation requires the client to shingle the keywords and compute the signature vectors $(\mathbf{SV})$. Each keyword can be represented in the form of shingles. The keywords containing uppercase letters, lowercase letters, numbers, duplicate letters and special characters are also considered. Therefore, each keyword can be represented in the form of a vector having the length γ^l-bits, where γ represents characters including letters, numbers, special characters etc and l represents the sequence of characters. This is explained with the help of an example, suppose the keyword is "university". The shingled set representation of "university" is $\{$un,ni,iv,ve,er,rs,si,it,ty$\}$. The next phase is to represent the shingled set in the form of a vector $\mathbf{S}$ $\{aa, ab, ac, \cdots, zz\}$, such that; if a shingle appears within a keyword then set the corresponding entry in $\mathbf{S}$ as 1 otherwise 0. This yields the length of the shingle vector to be $26^2 = 676$-bits. The default shingle vector may reveal the keyword to the CS, so $\mathbf{S}$ needs to be randomized such that the total number of permutations are 676!. As mentioned in [7], the permutations are done as:

Algorithm *(PermS)*: Let $\mathbf{S} = [e_1, e_2, \cdots, e_y]$ denote a vector containing y entries. The random permutation of the indexes of the vector $\mathbf{S}_y$ to the vector $\mathbf{S}'_y$ is represented by $V : \mathbf{S}_y \to \mathbf{S}'_y$. The vector $\mathbf{S}'$ is generated such that:

```
For $k = 1, 2, \cdots, y$, do:
    Select at random $\mathbf{S}_y$ such that $\mathbf{S}_y \notin \mathbf{S}'_k$.
    Set $\mathbf{S}'_k$=$\mathbf{S}_y$.
```

Therefore, the first element is picked at random from all the y elements contained in $\mathbf{S}$, the second element is picked from $y-1$ elements and so on. There are $y!$ different ways to form a randomly permuted vector $\mathbf{S}'$.

The advantage behind randomizing the vector $\mathbf{S}$ is that the keywords cannot be guessed in polynomial time. This is highlighted in more detail in the security proofs presented in the Section 5.5. After the selection of a particular instance of the random permutation, the entire dictionary of the keywords is represented in the form of individual randomly permuted shingled vectors. As the vectors are mostly sparse, the technique of Min Hashing is applied to all the vectors so that the search space may be reduced and a signature vector may be formed (already discussed in the Section 5.2).

Trapdoor Generation

Let W denote the keyword to be searched for. Given a vector $\mathbf{T}$, the keyword is shingled to populate the randomly permuted vector $\mathbf{S}'$ accordingly such that $\mathbf{T} = \mathbf{S}'$. Since $\mathbf{T}$ is mostly sparse, calculating the Jaccard similarity JS and Euclidean norm of

T against all the signature vectors in the Fuzzy Index FI is a resource intensive task. Therefore, Min Hashing is applied to **T**. Up until now, the trapdoor was deterministic and to make the trapdoor probabilistic a random vector $\mathbf{T}'$ is selected, the selection is done as follows:

Algorithm (*ProbT*): Let $\mathbf{T} = [t_1, t_2, \cdots, t_y]$ denote a vector containing y elements. Suppose each element represents a Min Hashed value associated to a keyword W. Initialize a vector $\mathbf{T}'$. A random permutation $Q : \mathbf{T} \to \mathbf{T}'$ is generated as:

```
For $j = 1, 2, \cdots, y$, do:
    Let $U$ be a uniform random variable in the range $[1, \gamma']$
    and $U \notin \mathbf{T}, \mathbf{T}'$.
    Set $\mathbf{T}'_j = U$.
```

Therefore, the vectors do not have any elements in common.

Upon the successful generation of random vector $\mathbf{T}'$, the Euclidean Norm $d = d(\mathbf{T}, \mathbf{T}')$ is computed using Equation 5.4. The trapdoor $Enc_{k_s}(d, \mathbf{T}')$ is transmitted to the CS. The next step is to perform search and identify the documents that contain the queried keyword.

Searching

Upon receiving the trapdoor, the CS using the session key k_s decrypts the trapdoor to uncover the underlying content. The search is based on the corollary 5.1. Since JS is applicable to sets only, so the CS calculates the $JS(\mathbf{T}', \mathbf{SV}_W)$ (using equation 5.3) that requires the representation of vectors into sets. The conversion is done as follows:

Algorithm (*ExtendedVector*): Let $\mathbf{T} = [t_1, t_2, \cdots, t_n]$ denote a vector containing n entries. Initialize an empty set L. The ExtendedVector $E : \mathbf{T} \to L$ is generated as:

```
For $j = 1, 2, \cdots, n$, do:
    Add $t_j$ to $L$ such that $t_j \notin L$ .
```

It may be observed that the ExtendedVector algorithm transforms a vector to a set. Although, the properties of a set and a vector vary, this algorithm is only triggered once the vector properties are no more required and hence only distinct elements are required in the set.

The trapdoor is formed in such a way that the CS looks for the entries where the JS is 0. On identifying the relevant entries, the CS computes the Euclidean Norm $d(\mathbf{T}', \mathbf{SV}_W)$, where $\mathbf{SV}_W \in FI$ and $\mathbf{T}' \in$ Trapdoor. If the searched keyword is contained in the dictionary, calculated Euclidean Norm will be equal to d (computed in the trapdoor generation phase). Otherwise, if the keyword is not contained in the dictionary, the most relevant keyword will be having a Euclidean Norm $\approx d$ varying by ε. ε is the variance from the existing keywords and represents the threshold that can be controlled by the client.

Upon the identification of the corresponding masked keyword identifier $mask-(id(W))$, the CS refers to the Inverted Index Table, I. After identifying the relevant column, the CS returns ranked encrypted document identifiers to the client.

Note: The sole purpose of using session key k_s between the client and the CS is to avoid any passive attacks that may be carried out by an outsider. Session key may not be required if the channel is secure.

5.3 Security Definitions

This section revisits the existing security definitions already presented in the Chapter 3,4 related to probabilistic trapdoors. The definitions are extended to fit the proposed construction.

5.3.1 Keyword-Trapdoor Indistinguishability for FRSE

Keyword-Trapdoor indistinguishability allows the adversary to select a keyword adaptively based on the history; more precisely, the adversary is given tuples $(\mathscr{W}, T_{\mathscr{W}})$. Since the adversary now has all the possible keywords and associated trapdoors, it has to submit two distinct keywords (W_o, W_1). The challenger tosses a fair coin b and sends the trapdoor corresponding to the keyword W_b to the adversary. This process continues until the adversary has submitted polynomially-many queries and is then challenged to output the bit b. If the guess is made with a probability of greater than 1/2 then the adversary wins when the security parameter λ is negligible *i.e.*, λ is sufficiently small that it can be ignored.

Definition 5.1 *Let* FRSE= (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) *be a fuzzy-based ranked searchable encryption scheme over a dictionary* Δ, λ *be the security parameter, and* $\mathscr{A}_{j;1\leq j\leq m+1}$ *be non-uniform adversaries. Consider the following probabilistic experiment* $Key\_Trap_{FRSE,\mathscr{A}}(\lambda)$:

```
$Key\_Trap_{FRSE,\mathscr{A}}(\lambda)$
    $(K, k_s, r) \leftarrow KeyGen(\lambda)$
    $(I, FI) \leftarrow Build\_Index(\mathbf{S}', \mathscr{D})$
    $for\, 1 \leq i \leq m$
        $(st_{\mathscr{A}}, W_i) \leftarrow \mathscr{A}_i(st_{\mathscr{A}}, T_{W_1}, \cdots, T_{W_i})$
        $T_{W_i} \leftarrow Build\_Trap_{k_s}(W_i)$
    $(st_{\mathscr{A}}, W_0, W_1) \leftarrow \mathscr{A}_0(\lambda)$
    $b \xleftarrow{\$} \{0,1\}$
    $(T_{W_b}) \leftarrow Build\_Trap(\mathbf{S}', k_s, W_b, num)$
    $b' \leftarrow \mathscr{A}_{m+1}(st_{\mathscr{A}}, T_{W_b})$
    $T'_W \leftarrow Build\_Trap_{k_s}(W_o); o \leq m$
    $if\, b' = b, output\, 1$
    $otherwise\, output\, 0$
```

where $st_{\mathscr{A}}$ represents a string that captures $\mathscr{A}$'s state. The keyword-trapdoor indistinguishability holds for all the polynomial time adversaries $(\mathscr{A}_0, \mathscr{A}_1, \cdots, \mathscr{A}_{m+1})$ such that $N = poly(\lambda)$,

$$Pr[Key\_Trap_{FRSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \tag{5.5}$$

where probability is over the choice of b.

5.3.2 Trapdoor-Index Indistinguishability for FRSE

Trapdoor-Index indistinguishability refers to the complexity offered by a fuzzy ranked searchable encryption scheme. More precisely, the adversary is given tuples $(\mathscr{W}, T_{\mathscr{W}}), FI$. Since the adversary now has all the possible keywords, associated trapdoors and fuzzy index table entries, he submits two distinct keywords (W_o, W_1). The challenger now tosses a fair coin b and submits the trapdoor, fuzzy index table entries corresponding to the keyword W_b to the adversary. This process continues until the adversary has submitted polynomially-many queries and is then challenged to output the bit b. f the adversary is able to make a guess with a probability greater than 1/2 then the adversary succeeds and the scheme lacks in providing trapdoor-index indistinguishability. The security parameter λ is negligible *i.e.*, λ is sufficiently small that it can be ignored to highlight the security of the scheme.

Definition 5.2 *Let* FRSE= (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) *be a fuzzy ranked searchable encryption scheme over a dictionary* Δ, λ *be the security parameter, and* $\mathscr{A} = (\mathscr{A}_0, \mathscr{A}_1)$ *be non-uniform adversaries. Consider the following probabilistic experiment* $Trap\_Index_{FRSE,\mathscr{A}}(\lambda)$:

```
$Trap\_Index_{FRSE,\mathscr{A}}(\lambda)$
    $(K, k_s) \leftarrow KeyGen(\lambda)$
    $(I, FI) \leftarrow Build\_Index(\mathbf{S}', \mathscr{D})$
    $for\, 1 \leq i \leq m$
```

$$
\begin{aligned}
&\quad let\, I' = FI[0][i] = Enc_K(W_i) \\
&\quad T_{W_i} \leftarrow Build\_Trap_{k_s}(W_i) \\
&b \xleftarrow{\$} \{0,1\} \\
&(st_{\mathscr{A}}, W_0, W_1) \leftarrow \mathscr{A}_0(\lambda, I', T_{W_m}, W_m) \\
&(T_{W_b}) \leftarrow Build\_Trap(\mathbf{S}', k_s, W_b, num) \\
&b' \leftarrow \mathscr{A}_1(st_{\mathscr{A}}, FI_{W_b}) \\
&if\, b' = b, output\, 1 \\
&otherwise\, output\, 0
\end{aligned}
$$

where $st_{\mathscr{A}}$ represents a string that captures $\mathscr{A}$'s state. The trapdoor-index indistinguishability holds if for the polynomial time adversaries $(\mathscr{A}_0, \mathscr{A}_1)$,

$$
Pr[Trap\_Index_{FRSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \tag{5.6}
$$

where probability is over the choice of b.

In SE, Privacy Preservation refers to getting valid search results without letting the CS learn the underlying data. This also includes anything that can be inferred from the trapdoors, index tables and the encrypted documents outsourced to the cloud. This leads to the following corollary:

Theorem 5.1: *The proposed FRSE scheme provides Keyword-Trapdoor Indistinguishability and Trapdoor-Index Indistinguishability if the Inverted Index table* (I) *and Fuzzy index table* (FI) *are secure and the trapdoors are probabilistic.*

5.4 Proposed FRSE scheme

This section formally presents the detailed description of the proposed scheme and discusses its correctness and soundness.

5.4.1 Scheme Construction

As mentioned earlier, the proposed scheme comprises of the following five polynomial time algorithms:

- **Phase 1-KeyGen** (λ)**:** Given a security parameter λ, generate the keys $K, k_s \leftarrow \{0,1\}^\lambda$ and a random number $r \leftarrow CSPRNG(1^\lambda)$.
- **Phase 2-Build_Index** $(\mathbf{S}', \mathscr{D})$**:**
  - Inverted Index (I) Generation: Construct a matrix (I) of the order $(n+1) \times (m+1)$ as follows:
    1. Extract a keyword set $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$ from a set of documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$.
    2. Set entries at $(i+1, 1) = Enc_K(id(D_i))$; $1 \leq i \leq n$.
    3. Set entries at $(1, j+1) = Enc_K(W_j)$; $1 \leq j \leq m$.
    4. Calculate the RF using equation 3.1 and populate the remaining entries $(i+1, j+1) = RF(W_j, D_i)$.
    5. Mask the RFs as:

       - For $a = 1, 2, \cdots, m$

         - $(n, b) = (n, b) \times r$

       Note: To further enhance the security, Order Preserving Encryption (OPE) may be used instead.
    6. Output the matrix (I) that represents the inverted index table I.
  - Fuzzy Index (FI) Generation: Construct a matrix FI of the order $(Q+1) \times (m+1)$ where $Q \in q$ represents the random permutations used for Min Hashing represented as $f_q : \mathbf{S} \rightarrow R$ and m are the total number keywords. Construct FI as follows:
    1. Form a shingle vector $\mathbf{S}$ of the order γ^2 and randomize it to form $\mathbf{S}'$ using the algorithm *PermS* (presented previously in the Section 5.2).

2. For each keyword W_m form a shingle set and permute according to $\mathbf{S}'$.
3. For each $\mathbf{S}'$ compute the Min Hashes and form the corresponding Signature Vectors **SV** having Q-bit length.
4. Set FI at $(1,i) = Enc_K(W_i); 1 \leq i \leq m$
5. Set FI at $(i,j) = \mathbf{SV}[Q]; 1 \leq j \leq q$.

- **Phase 3-Build_Trap** $(\mathbf{S}', k_s, W, num)$**:** Generate a probabilistic trapdoor vector $\mathbf{T}'$ as follows:
  - Represent the search keyword(s) in the form of shingle set(s).
  - Permute the same random vector $\mathbf{S}'$ according to the shingle set.
  - Using the same Q permutations form a signature vector(s) represented as **T**.
  - According to the algorithm (*ProbT*) (presented previously in the Section 5.2) randomize the vector **T** to obtain $\mathbf{T}'$.
  - Compute the Euclidean Norm $d(\mathbf{T}, \mathbf{T}')$ using Equation 5.4.

  The Trapdoor $T_W = (d, \mathbf{T}', num)$; where *num* represents the total number of required documents. Compute $Enc_{k_s}(T_W)$ and send it to the CS.

- **Phase 4- Search_Outcome** $(k_s, I, FI, Enc_{k_s}(T_W))$**:** Identify the documents $D_{num} \in \mathscr{D}$ as the outcome of the search as follows:
  - Using the session key k_s decrypt $Enc_{k_s}(T_W)$.
  - Using algorithm (*ExtendedVector*) (presented in the Section 5.2) Convert $\mathbf{T}'$ to a set L.
  - For each signature vector $\mathbf{SV}_m$ stored in FI, convert $\mathbf{SV}_m$ to a set L' using the algorithm (*ExtendedVector*). Calculate the Jaccard Similarity

$JS(L,L')$ using Equation 5.3.

- If JS is 0 do:

  - Calculate the Euclidean Norm $d'(\mathbf{T}',\mathbf{SV}_m)$

– The required keyword will have $d \approx d'$ varying by ε (threshold).

– Identify the corresponding masked keyword identifier $mask(id(W))$ in I and the encrypted document identifiers $X = Enc_K(D_{num})$ in ranked order.

Return X to the client.

- **Phase 5-Dec** (K,X)**:** Given X a set of encrypted document identifiers, decrypt X using the master key K to uncover the outcome of the search.

5.4.2 Correctness of FRSE Scheme

The correctness of the result is primarily based on the correctness of the Euclidean Norm. To prove the correctness we revisit the theorem proposed in [151, 83]:

Theorem 2: *The intersection of the fuzzy index table entries FI_W and FI_{W_i} for the keywords W and W_i is not null iff Euclidean Norm $dis(W,W_i) \leq d$.*

Proof: We prove that $FI_W \cap FI_{W_i} \neq \phi$ iff $dis(W,W_i) \leq d$. This leads to finding the elements in $FI_W \cap FI_{W_i}$. If the Euclidean Norm $dis(W,W_i) \leq d$ then $W \sim W_i$. Let W be represented by a shingle vector $\mathbf{S}$ formed from the shingle set $S = \{(\alpha|\beta)(\alpha|\beta),\cdots,(\alpha|\beta) : \alpha,\beta \in set\,of\,characters\}$ and W_i be represented by a shingle vector $\mathbf{S}$ obtained from the shingle set $S = \{(\alpha^*|\beta^*)(\alpha^*|\beta^*),\cdots,(\alpha^*|\beta^*) : \alpha^*,\beta^* \in set\,of\,characters\}$. After $dis(W,W_i)$ edit operations, W can be changed to W_i. Let $W^* = \{(\alpha^+|\beta^+)(\alpha^+|\beta^+),\cdots,(\alpha^+|\beta^+) : \alpha^+,\beta^+ \in set\,of\,characters\}$, where $W^*[i] = W[j]$ on applying any operation at this position. On applying $dis(W,W_i)$ edit operations on the same positions containing $*$ at W^*, W^* can be transformed into W_i. Therefore, W^* is an element in both $FI_W \cap FI_{W_i}$.

Now we prove that $dis(W,W_i) \leq d$ if $FI_W \cap FI_{W_i} \neq \phi$. As proved earlier, let W^* represent the common elements in $FI_W \cap FI_{W_i}$. The proof is divided into two cases:
Case-1: Suppose $dis = 0$, *i.e.*, no edit operation is required to transform W_i into W and as a result $W = W_i = W^*$. So Euclidean Norm $dis(W,W_i) = 0 \leq d$.
Case-2: Suppose $dis > 0$, *i.e.*, for $*$ in W^*, $*$ edit operations can be performed to transform W^* to W_i and W. d is variable that depends upon the allowed edit operation. The proposed FRSE scheme achieves *ranked* fuzzy keywords search therefore we allow more than one edit operation. For the clarity of the proof, we consider an edit operation of one shingle, therefore, W^* can be transformed to W by at most 1 edit operation, thus the Euclidean Norm $dis(W^*,W) \leq 1$. Since $W^* \in FI_{W_i} \cap FI_W$, the total number of edit operations $*$ is not greater than d. Hence the Euclidean Norm $dis(W,W_i) \leq d$.

5.5 Security Analysis

Firstly we discuss the leakage of the proposed FRSE scheme, then we present the formal game-based security proofs of the proposed scheme.

5.5.1 Leakage Profiles

It is not possible to present a SE construction that does not leak information to the adversary. The proposed scheme limits the leakage by generating probabilistic trapdoors. Although the proposed scheme leaks very less information as compared to prior existing schemes, it is important to analyze the leakage profiles. The leakage profile is formed over the evolved artifacts; including the inverted index table I, fuzzy index table FI, trapdoor T_W generated for a particular keyword and the outcome of the search. The leakages are as follows:

Leakage $L_{5.1}$

This leakage highlights the information revealed by the index table I. The index table I is generated by the client and outsourced to the CS. This leakage is defined as:

$$L_{5.1}(I) = \left\{\begin{array}{l} \textit{Total number of keywords,} \\ \textit{Total number of documents} \\ Enc_K(\mathscr{W}) \in Enc_K(\mathscr{D}) \vee Enc_K(\mathscr{W}) \notin Enc_K(\mathscr{D}) \end{array}\right\} \tag{5.7}$$

Leakage $L_{5.2}$

This leakage highlights the information revealed by the fuzzy index table FI. FI is generated by the client and outsourced to the CS. This leakage is defined as:

$$L_{5.2}(FI) = \left\{\textit{Total number of keywords}, \mathbf{SV}_{\mathscr{W}}[Q]\right\} \tag{5.8}$$

Leakage $L_{5.3}$

This leakage is associated to the information revealed by the trapdoor T_W generated for a particular keyword W. The probabilistic trapdoor T is generated by the client and sent to the CS. Using the trapdoor, the CS searches on the client's behalf. The leakage is defined as:

$$L_{5.3}(T_W) = \left\{\begin{array}{l} ProbabilisticTrapdoor\mathbf{T}', \\ EuclideanNorm(d(\mathbf{T},\mathbf{T}')), \\ \textit{Number of required documents (num)} \end{array}\right\} \tag{5.9}$$

Leakage $L_{5.4}$

This leakage is bound to the search outcome (SO) of the trapdoor generated for a particular keyword represented as T_W. This leakage is induced as a result of the search carried out by the CS. This leakage is defined as:

$$L_{5.4}(SO) = \left\{ \mathrm{OC}(W),\, Enc_K(id(D_i))_{\forall T_W \in \mathscr{D}} \right\} \tag{5.10}$$

where OC represents the relevant outcome corresponding to the searched keyword.

Discussion on Leakage: In [96] the possible attacks are studied on the SE schemes that require a relational database and based on Order Preserving Encryption (OPE). It may be observed that the proposed scheme does not require a relational database, therefore, the masking function can be strengthened by using OPE.

Referring to the leakage associated to the inverted index table (I), it may be observed that the index may only leak the presence or absence of an encrypted keyword within a document, the total number of keywords and the total number of documents. The leakage related to fuzzy index table (FI) leaks the total number of keywords that form a fuzzy index table (FI) but the keywords itself can never be uncovered. The trapdoor is unlinkable as it is probabilistic, therefore, it does not reveal the access pattern prior to the search. The outcome of the search is only revealed. We now discuss the impact of these leakages on the security of the system as done in [132].

5.5.2 Formal Security Proofs

Lemma 1. *The Fuzzy Ranked Searchable Encryption Scheme (FRSE) presented in the Section 5.4 is "privacy preserving" according to Theorem 5.1, as it is* $L_{5.1}, L_{5.2}, L_{5.3}, L_{5.4}$*-secure (represented as equations 5.75.85.95.10) and according to Definition (5.1,5.2) where* $L_{5.1}$ *is associated with the inverted index table* (I) *and leaks the total number of keywords, total number of documents and the presence/absence of an encrypted keyword within an encrypted document.* $L_{5.2}$ *is related to the fuzzy index table* (FI) *and leaks the total number of keywords, signature vectors* $(\boldsymbol{SV})$*.* $L_{5.3}$ *leaks a probabilistic vector* $\boldsymbol{T}'$*, Euclidean norm and number of required documents.* $L_{5.4}$ *leaks the outcome of a trapdoor and the encrypted file* *identifiers containing the searched keywords.*

Proof: We start the proof by simulating the keyword-trapdoor indistinguishability and trapdoor-index indistinguishability definitions for the proposed FRSE scheme. The proof requires an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. We demonstrate that if the adversary $\mathscr{A}$ is successfully able to distinguish between outcome of the algorithms and between the keywords, trapdoors and associated index tables, it will result in compromising the privacy-preserving property of FRSE.

We take a game-based approach similar to [143] in which the security proof is divided into phases *i.e.* setup, challenge and the outcome phase.

Keyword-Trapdoor Indistinguishability in FRSE:

- **Setup Phase:** The challenger runs the KeyGen(λ) phase to generate a master key K, a session key k_s and a random number r. The master key K is kept private, whereas, the session key k_s is shared with the adversary $\mathscr{A}$. The challenger runs the Build_Index($\mathbf{S}', \mathscr{D}$) phase to generate the inverted index table (I) and the fuzzy index table (FI). The adversary submits each keyword $W \in \mathscr{W}$ where $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$, to the challenger. The challenger runs the Build_Trap($\mathbf{S}', k_s, W, num$) phase to generate the corresponding trapdoors. The trapdoor generation every time requires to compute a randomized shingle vector $\mathbf{S}'$ and random vectors $(\mathbf{S}', \mathbf{T}, \mathbf{T}')$. The randomized vector $\mathbf{S}'$ is of the order γ^l. The Euclidean Norm (d) is calculated between the random vectors $(\mathbf{T}, \mathbf{T}')$. The trapdoors $T_W = (d, \mathbf{T}', num)$ are generated for every keyword and sent to the adversary one-by-one. This leads to the leakage $L_{5.3}$.

Note: The adversary can send any value of num where $1 \leq num \leq n$ as a result the leakage $L_{5.1}$ is induced. It may also be observed that distinct random vectors $(\mathbf{S}', \mathbf{T}, \mathbf{T}')$ are generated at every instance, therefore, every time a new Euclidean

Norm d is generated. Hence, a probabilistic trapdoor is generated even for the same keywords queried again making the leakage $L_{5.3}$ harmless.

- **Challenge Phase:** The adversary $\mathscr{A}$ adaptively issues two distinct keywords W_0, W_1 to the challenger where $(W_0, W_1) \in \mathscr{W}$. The challenger tosses a fair coin $b \in \{0, 1\}$ and generates a trapdoor T_{W_b}. The trapdoor is generated in the same way as done during the setup phase and is shared with the adversary.

  The adversary $\mathscr{A}$ continues to issue a number of keywords to the challenger again to receive the corresponding trapdoors. The adversary $\mathscr{A}$ is also allowed to send the same keyword as in the challenge phase.

- **Outcome Phase:** The adversary $\mathscr{A}$ takes a guess b' of b.

Now that we have successfully simulated the game representing the keyword-trapdoor indistinguishability for the proposed FRSE scheme. If the adversary is successfully able to guess b, then it is able to distinguish between the generated trapdoor *i.e.* T_{W_0} and T_{W_1}. If we generalize this then the adversary can actually distinguish between all the generated n trapdoors in the past (history) and the ones that will appear in the future, which is impossible. Thus the advantage can be interpreted as:

$$Pr[Key\_Trap_{FRSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda)$$

which is according to the definition 5.1 and equation 5.5.

Trapdoor-Index Indistinguishability in FRSE:

- **Setup Phase:** The challenger runs the KeyGen(λ) phase to generate a master key (K), a session key (k_s) and a random number (r). The master key (K) is kept private, whereas, the session key (k_s) is shared with the adversary $\mathscr{A}$. The challenger runs the Build_Index$(\mathbf{S}', \mathscr{D})$ phase to generate the inverted index table (I) and the fuzzy index table (FI). The inverted index table (I) stores the information related to the presence of a keyword within the document along with the relevance scores, which are masked. The fuzzy index table (FI) is formed with the help of a randomly permuted shingle vector $\mathbf{S}'$ of the order γ^l. Calculate Min Hash of $\mathbf{S}'$ to generate a Signature Vector $(\mathbf{SV})$ of the order Q-bit. The index tables (I) and (FI) are shared with the adversary. The adversary submits each keyword $W \in \mathscr{W}$ where $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$, to the challenger. The challenger runs the Build_Trap$(\mathbf{S}', k_s, W, num)$ phase to generate the corresponding trapdoors. Every time the Build_Trap computes a shingle vector and random vectors $(\mathbf{S}', \mathbf{T}, \mathbf{T}')$. The euclidean norm (d) is calculated between the random vectors $(\mathbf{T}, \mathbf{T}')$. The challenger also performs the search over the fuzzy index table (FI) using the generated trapdoor. Therefore, the trapdoors $T_W = (d, T', num)$ are generated for every keyword and sent to the adversary along with the FI_W one-by-one.

- **Challenge Phase:** The adversary $\mathscr{A}$ adaptively issues two distinct keywords W_0, W_1 to the challenger where $(W_0, W_1) \in \mathscr{W}$. The challenger tosses a fair coin $b \in \{0, 1\}$ and generates a trapdoor T_{W_b}. The trapdoor is generated in the same way as done during the setup phase and is shared with $\mathscr{A}$.

  The adversary $\mathscr{A}$ continues to issue a number of keywords to the challenger again to receive the corresponding trapdoor and fuzzy index table entries (FI). The adversary $\mathscr{A}$ is also allowed to send the same keyword as in the challenge phase.

- **Outcome Phase:** The adversary $\mathscr{A}$ takes a guess b' of b and sends FI_{W_b}.

Now that we have successfully simulated the game representing the trapdoor-index indistinguishability for the proposed FRSE scheme. If the adversary is successfully able to guess b, then it is able to distinguish between the generated trapdoors, *i.e.*, T_{W_0} and T_{W_1}. If the trapdoors are distinguishable then the fuzzy index table entries are also distinguishable prior to the search. Hence, leading to the revealing of the access pattern and the search pattern prior to the search, which is impossible. Thus the advantage can be interpreted as:

$$Pr[Trap\_Index_{FRSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda)$$

which is according to the definition 5.2 and equation 5.6 respectively.

Remark: It may be observed that the inverted index table is masked and encrypted using trusted atomic primitives, hence, the index table (I) only leads to the leakage $L_{5.1}$ already defined above. The fuzzy index table (FI) is again based on a random permuted vector that is of the order γ^l. The total number of possible permutations are $\gamma^l!$ that cannot be solved in polynomial time making the leakage $L_{5.2}$ meaningless. The trapdoors are probabilistic, preventing the adversary from launching a brute force attack on the generated trapdoors. So the leakages $L_{5.1}, L_{5.2}, L_{5.3}$ do not effect the security of the proposed FRSE scheme.

The leakage $L_{5.4}$ is the search outcome, that is the encrypted set of document identifiers giving out no information other than the encrypted outcome of the search. Therefore, considering the conformance to the definitions (5.1,5.2), theorem 5.1, the leakages $L_{5.1}, L_{5.2}, L_{5.3}, L_{5.4}$ and the associated proofs, the proposed FRSE scheme is Privacy Preserving.

5.6 Performance Metrics

The feasibility analysis of the proposed FRSE scheme is performed in this section that mainly focuses on the algorithmic analysis and the storage overhead analysis.

5.6.1 Algorithmic Analysis

In order to analyze the computational complexity, it is important to perform the asymptotic analysis of the proposed scheme against similar existing schemes. The

Table 5.1 Algorithmic Comparative Analysis FRSE vs. Existing Schemes

Schemes	Build_Index	Build_Trap	Search_Outcome
[83]	$O(mS)$	$O(mS+1)$	$O(m^2)$
[146] and [57]	$O(mn+Qm)$	$O(Qm+1)$	$O(2m^2)$
Proposed FRSE (ranked)	$O(2mn+m+Qm)$	$O(Q+1)$	$O(2Qm+mn)$
Proposed FRSE (unranked)	$O(mn+Qm)$	$O(Q+1)$	$O(2Qm+n+1)$

asymptotic analysis helps to measure the upper bound time complexity of the schemes. The complexity associated to a set of documents is represented by n and for the keywords by m. As most of the schemes comprise of all the 5 phases *i.e.*, KeyGen, Build_Index, Build_Trap, Search_Outcome and the Dec phase, therefore, we analyze these phases separately to perform the asymptotic comparative analysis. It may also be observed that since the KeyGen and Dec phases are identical to any other scheme, we do not take their complexities into consideration. Our analysis also considers ranking and non-ranking separately because the well known existing fuzzy scheme presented in [146] does not rank the documents. Therefore, this will help readers to easily relate the complexities of the proposed scheme against similar existing schemes. Table 5.1 shows the comparative algorithmic analysis against similar existing schemes.

From the complexity analysis it can be observed that the proposed FRSE scheme, in the Build_Index phase, builds two indexes *i.e.* the inverted index table (I) and the fuzzy index table (FI). The inverted index table I enabling ranking is asymptotically bound by $O(2mn+m)$ and unranked inverted index generation requires $O(mn+m)$. The ranked or unranked searching does not have an impact on the fuzzy index table FI, therefore, the complexity induced is $O(Qm)$, where Q represents the random permutations used for Min Hashing. As a result the total complexity of the ranked Build_Index phase is $O(2mn+n+Qm)$. The Build_Trap phase is asymptotically bound by $O(Q+1)$. As mentioned in Section 5.4, the Search_Outcome is performed at the CS side. This phase is asymptotically bound by $(2Qm+mn)$ in terms of ranked searching and $(2Qm+n+1)$ for unranked searching. Apart from the complexities already mentioned, S is the complexity associated to [83] and represents the wildcard-based fuzzy set construction. It may be observed that the proposed FRSE scheme does not outperform the existing schemes, the reason being ranked searching, probabilistic trapdoors and eliminating the need of a predefined dictionary. It may be noted in the next section that in terms of the storage complexity, the proposed FRSE scheme outperforms the state-of-the-art, therefore, the performance analysis needs to be measured in relation with the asymptotic analysis and the storage overhead.

5.6.2 Storage Overhead

Another important metrics for measuring the performance of the scheme is the storage overhead analysis. This is dependent upon the data structure and helps analyze the amount of data stored either on the client-side or the server-side. We analyze both of the storages (*i.e.*, the client and the server) separately. Referring to the proposed FRSE scheme, it may be observed that the client stores the master key K and the session key k_s. Having the security parameter λ, the size of the keys are 128bit. Here, we consider the dataset contains only lowercase letters so the client stores a randomized shingle vector $\mathbf{S}'$ having the size $26^2 = 676$ bits. Furthermore, Q permutations are also stored that are used to form a signature vector $\mathbf{SV}$. As mentioned earlier the variable Q increases/decreases the accuracy of the results. Considering $Q = 10 \times 676$ bits, the total storage required at the client's side is $128+128+676+10\times 676 = 7016$ bits$= 7016/8$ bytes. This means the client requires only 877 bytes in terms of storage overhead.

Referring to the storage overhead of the CS, firstly the encrypted documents need to be stored. Having n documents and avg representing the average size of documents, this storage can be represented as $n \times D_{avg}$. The fuzzy index table FI requires a storage space of $(m \times Q)$, where m are the total number of keywords. The inverted index table I incurs a storage overhead of $8(m \times n)$. The session key k_s is also stored at the CS. The total storage overhead at the CS is $n \times D_{avg} + m \times Q + 8(m \times n) + 128$ bits.

We now discuss the storage overhead of the scheme presented in [146]. For clear and concise comparison we use abbreviations similar to the proposed FRSE scheme. Having the security parameter λ, the scheme requires to store the secret key K which is a combination of (M_1, M_2, S), where M_1, M_2 are matrices of the order λ x λ and **S** is a vector of the order λ. So having $\lambda = 128$ bits, the client requires $128 \times 128 + 128 = 2064$ bits. The CS stores the secure index that is similar to a per document bloom filter *i.e.*, $n \times (n \times m \times 128)$ bit. Whereas, the outsourced encrypted documents require the same storage as the proposed scheme.

It can be observed that the proposed FRSE scheme outperforms the scheme presented in [146] in terms of client and CS storage overhead.

5.7 Computational Analysis

Before discussing the computational overhead, it is important to discuss the BTCS architectural details, the dataset related information and the system specifications that are presented previously in the Sections 3.6.1, 3.6.2 and 3.6.3 respectively. This also helps to give more insight on the performance of the proposed scheme.

Figures 5.3 and 5.4 illustrate the activity diagrams depicting the flow of events when the proposed FRSE scheme is deployed onto the BTCS using the AE service.

5.7.1 Computation Overhead

This section analyzes the computational time for the different phases of the proposed FRSE scheme. As discussed in the Section 5.4, the proposed scheme comprises of five polynomial time algorithms. We skip the computational time analysis of the

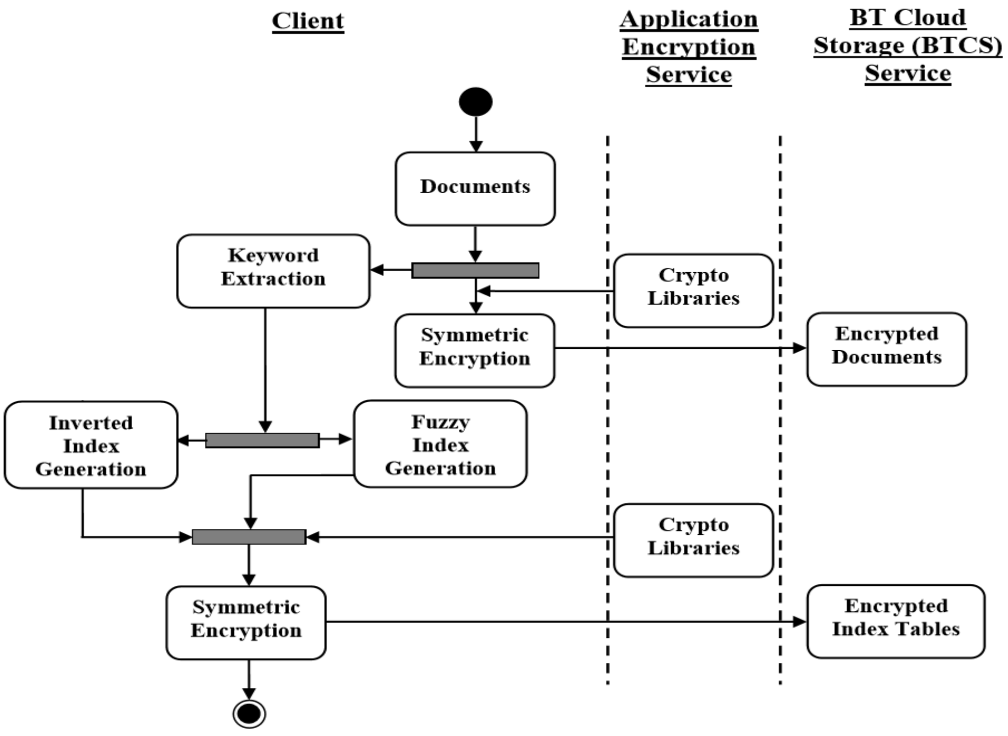

Fig. 5.3 FRSE Activity Diagram: Setup.

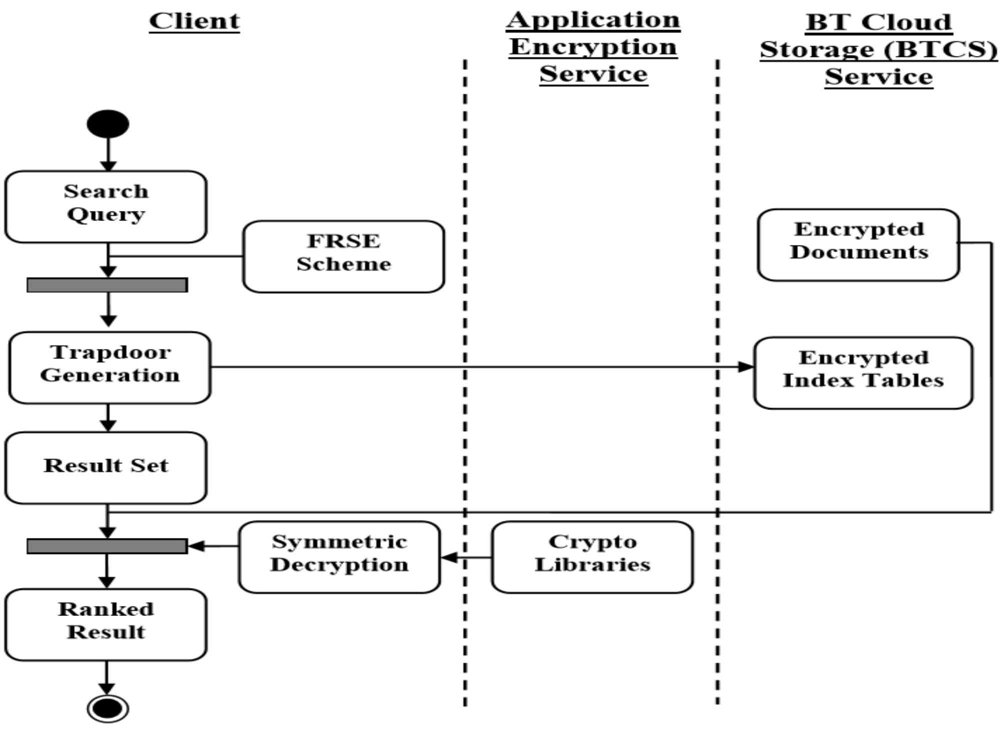

Fig. 5.4 FRSE Activity Diagram: Searching.

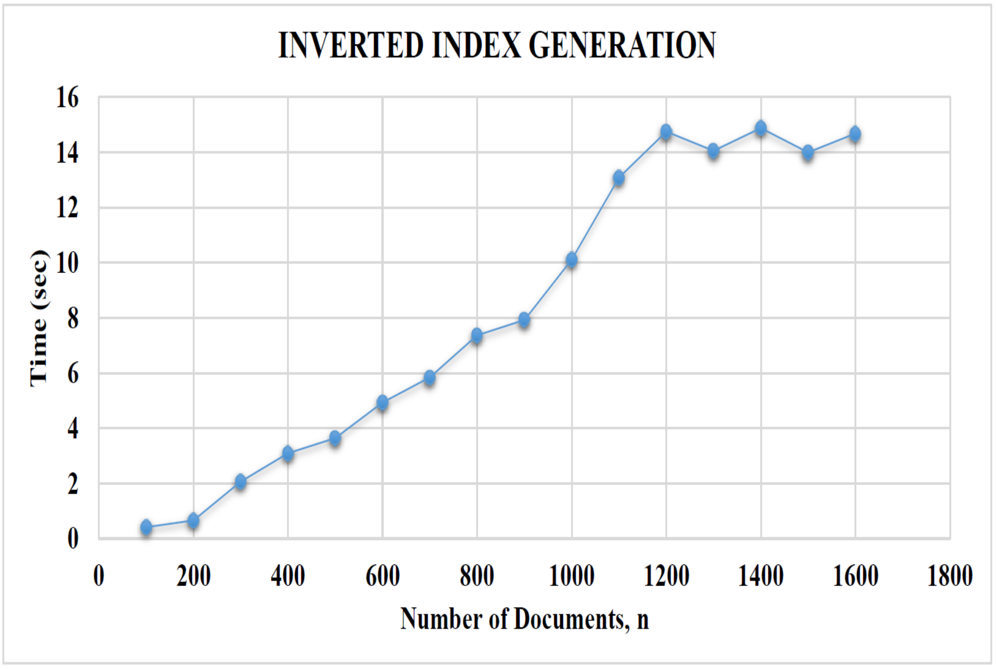

Fig. 5.5 FRSE: Computational Time for the Inverted Index Generation.

KeyGen phase and the Dec phase. It is to be mentioned here that these graphs are generated directly from actual results obtained from the experiments and no data normalization techniques have been applied on the results.

Build_Index Phase

Starting with the Build_Index phase, two main tasks are performed in this phase, *i.e.*, the inverted index generation and the fuzzy index generation. It is again emphasized that the Build_Index is a one-time process. The computational cost for the inverted index generation mainly comes from the RF generator. Figure 5.5 highlights the computational cost of the algorithm. The inverted index is generated for fixed number of keywords, 120,000, and variable amount of documents. The experiment starts with 100 documents that are incremented by 100 on every iteration to a maximum of 1600 documents. The number of documents are presented along the x-axis and the time in seconds is presented along the y-axis. The trend followed by the graph is dependent upon the size of the documents at hand. Our corpus contains documents placed in ascending order of their sizes. The size of the documents continues to grow up until 1200 documents after which the size of the documents is static. This fact is clearly visible by the trend followed by the graph Figure 5.5. For 1600 documents, inverted index generation takes 14.68 seconds. The curve fitting/ linear interpolation also signifies the graph showing a linear trend with $y = 0.0112x - 1.2583$ and $R^2 = 0.954$.

In the next step we analyze the computational time for the fuzzy index generation. This computation is mainly incurred due to the shingling of keywords and applying Min Hashing. The fuzzy index is not effected by the number of documents, therefore, only the number of keywords are varied. The results are presented in the form of a graph in Figure 5.6. The experiment starts with 10,000 keywords that are scaled to a maximum of 120,000 by gradually adding 10,000 on every iteration. The number of keywords are along the x-axis and the time in seconds is along the y-axis. It is observed that the fuzzy index shows a fairly linear growth with the increase in the number of keywords. The curve fitting/ linear interpolation highlights this linear trend with $y = 2E - 05x - 0.0389$ and $R^2 = 0.9927$. For 120,000 keywords, the fuzzy index generation requires approximately 2.84 seconds.

Build_Trap Phase

The Build_Trap phase is effected by the number of keywords to be searched, *i.e.*, the trapdoor generated for multi-keyword search will take more computational time as compared to single keyword search. Currently the implementation only facilitates single keyword search queries. In the experiment, the trapdoor is generated for the wrongly spelt keyword “aboot". The same keyword is used to analyze the computational time of proceeding phases. The proposed scheme takes an average time of 0.09 seconds for generating the trapdoor.

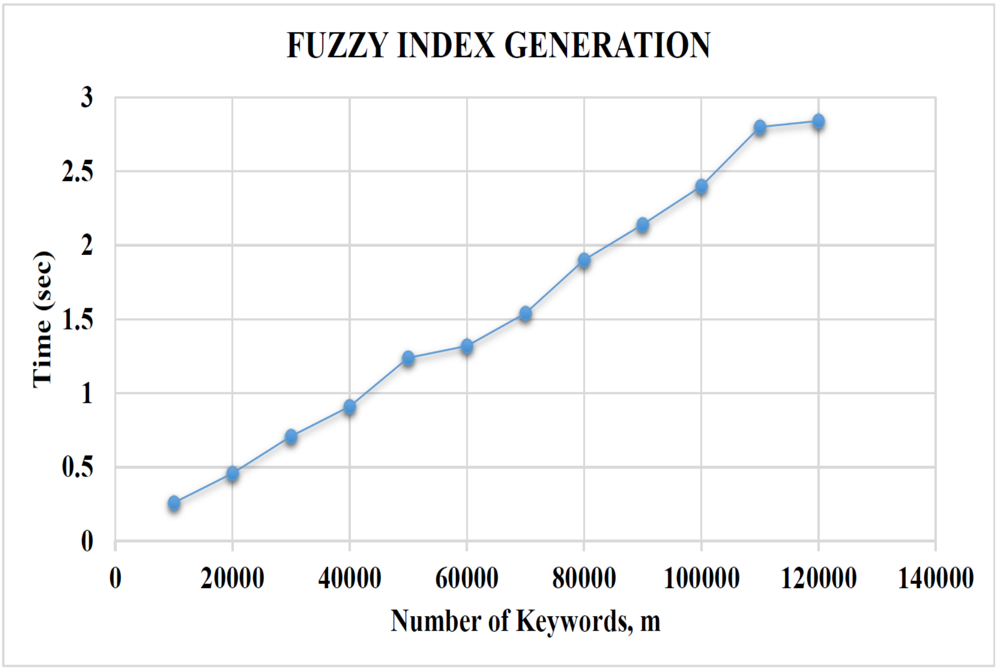

Fig. 5.6 FRSE: Computational Time for the Fuzzy Index Generation.

Search_Outcome Phase

The next phase is the Search_Outcome phase and performed on the BTCS. The computational time is analyzed threefold; firstly, the effect of varying number of keywords on the search results is analyzed on a stand alone workstation, secondly, the efficiency of search is analyzed by varying the number of documents on a stand alone workstation. Lastly, the result is analyzed that deploys the scheme onto the BTCS and hence includes the network latency incurred while retrieving the results directly from the BTCS.

It is observed that the Build_Index is highly affected by the change in the number of keywords and documents. This also means that with the change in the size of the index table the computational time of the Search_Outcome phase is also effected. Firstly, we analyze the effects of varying number of keywords on the scheme. Referring to the Figure 5.7, the number of keywords are changed ranging from 10,000 to 120,000. The number of keywords are presented along the x-axis and the time in seconds is along the y-axis. It is observed that for 100 documents and

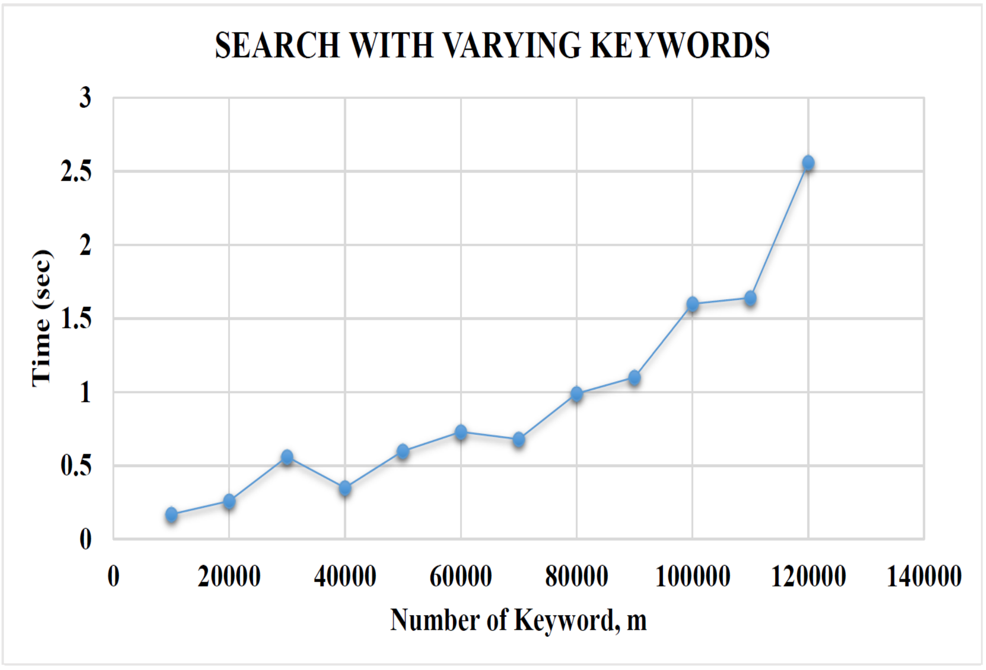

Fig. 5.7 FRSE: Computational Time for Search Having Fixed Number of Documents and Varying Amounts of Keywords.

with the increase in the number keywords, the efficiency decreases while searching for the keyword “aboot”. A non-uniform trend is observed because the trapdoors are probabilistic and more keywords show a Jaccard Similarity equal to 0, and therefore, more Euclidean Norms are to be calculated. However, curve fitting/ linear interpolation depicts a linear trend with $y = 2E-05x - 0.2192$ and $R^2 = 0.86$. It is also worth mentioning that the probabilistic trapdoors increase the security but at the cost of the increase in computation. Hence, for scenarios that require a few set of keywords, the search is very efficient. The search time for 10,000 and 120,000 keywords are 0.17 and 2.56 seconds respectively.

The effects of altering the number of documents is analyzed next. With the increase in the number of documents the performance decreases. This decrease is because with the increase in the documents, the search space increases and more computation is required for achieving ranking. The ranking is achieved through the sorting of the RFs which affects the computational cost. Searching for the keyword “aboot" takes around 2.17 seconds for 100 documents and 43.1 seconds for

1600 documents as shown in Figure 5.8. The graph tends towards a linear trend on applying curve fitting/ linear interpolation with $y = 0.0251x - 2.8$ and $R^2 = 0.89$.

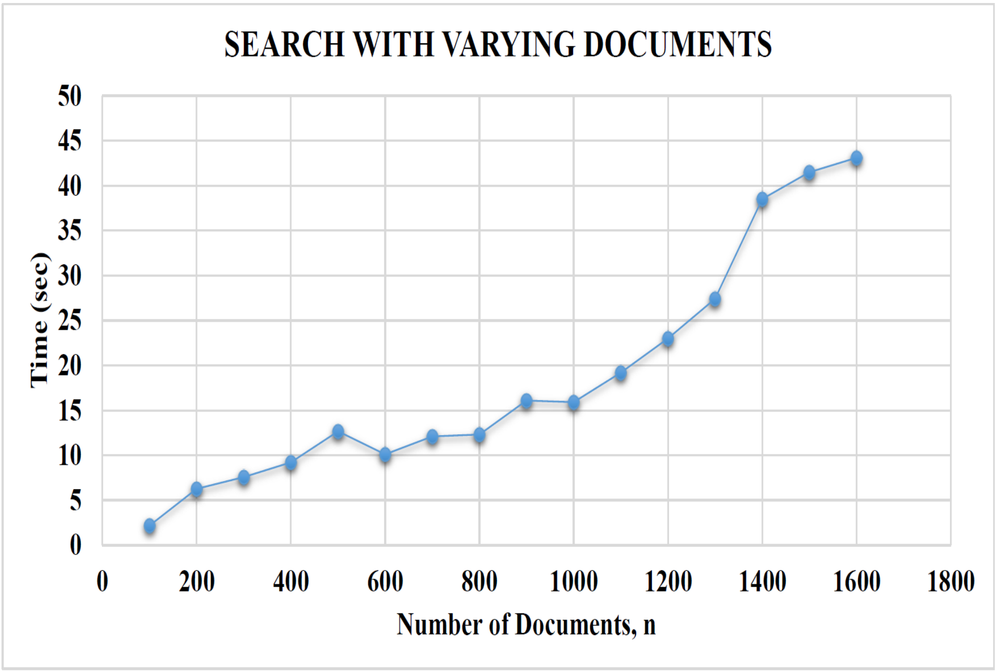

Fig. 5.8 FRSE: Computational Time for Search Having Varying Number of Documents and Fixed Number of Keywords.

On studying the influence of varying number of keywords and documents on the FRSE scheme, we search for the keyword “aboot”, where we have an inverted index table generated for 1600 documents and fuzzy index table comprising of 120,000 keywords. We only vary the number of required documents starting from 100 and incrementing in steps of 100 to attain a maximum of 1600. The results are illustrated in Figure 5.9, that includes the network latency incurred due to the communication between the BTCS and the client through sockets. While searching for the keyword “aboot” the fuzzy search result includes the documents containing the keywords “about, abort, abouts”. It is observed that the scheme shows a fairly linear growth that is also depicted on applying curve fitting/ linear interpolation that gives$y = 0.0065x + 0.0892$ and $R^2 = 0.9901$. The number of required documents are presented along the x-axis and the time in minutes is along the y-axis. The search time for 1600 documents is around 11.1 minutes. The network latency is the amount of time the result takes to be received by the client. This does not depend upon the scheme rather it is directly affected by the location of the CS. The farthest from the CS higher is the latency. Hence, the time in minutes is acceptable.

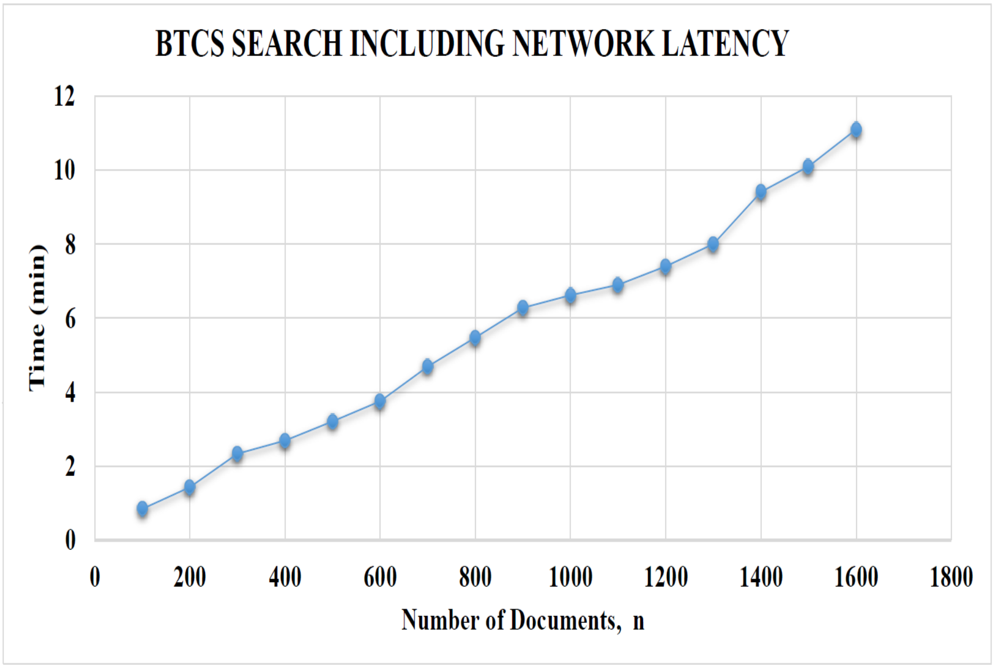

Fig. 5.9 FRSE: Computational Time for Search on the BTCS Including the Network Latency.

5.7.2 Result Accuracy

To discuss the accuracy of the scheme, we analyze the precision and recall as described in [99]. The precision is represented as $\frac{t_p}{t_p+f_p}$, whereas, the recall is defined as $\frac{t_p}{t_p+f_n}$. Before proceeding further, it is important to define false positive f_p and false negatives f_n. In the proposed construction, false positives are those keywords that are not required but appear in the search. Similarly, a false negative represents the set of keywords expected to appear in the search but do not show up. For the experiment we randomly pick 100 keywords from the possible 120,000 keywords and we generate the results for the keyword “aboot”. Figure 5.10 shows the performance of the proposed scheme while varying the threshold ε. If the keyword

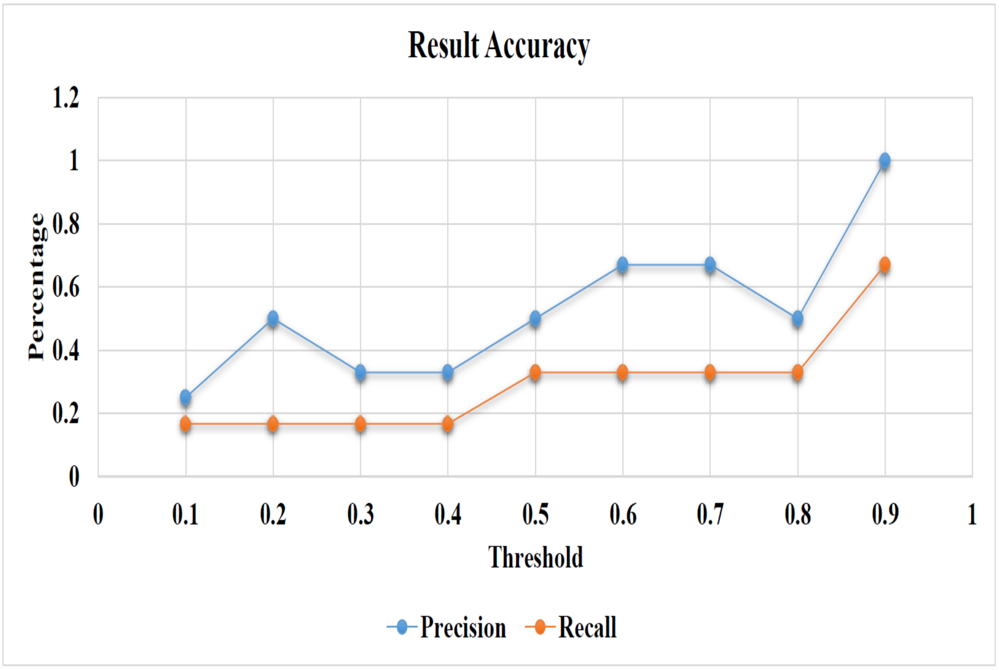

Fig. 5.10 FRSE: Performance Metrics Considering the Precision and Recall.

is correctly spelt then the exact matching takes place and that would lead to 100% accuracy, however, in fuzzy search we use Min Hashing and apply Euclidean Norm and Jaccard similarity to compute the similarity between the trapdoor and the fuzzy index. It is also worth mentioning that while doing exact matching we can get 100% accuracy but the recall is not 100%. This is because although we are searching for the exact keywords but the proposed algorithm performs fuzzy searching *i.e.*, even though we are searching for a correctly spelt keyword “about”, the search results should also include results for “abort”. By varying the ε value, the accuracy of the results changes. It is observed that with the increase in the threshold value the precision increases. For the threshold value of 0.1, the precision is 25% that increases to 67% when ε is 0.6 and to a maximum of 100% for ε value of 0.9. A slight fluctuation in the precision at different intervals is also observed that is due to the use of probabilistic trapdoor. During our experiment the fluctuation took place when the value of ε was 0.8. It is also observed that unlike [146] with the increase in ε, the recall also increases. This is due to the reason that we are seeing an increase in the true positives with the increase in the ε value. The proposed scheme attains a maximum of 67% recall value when the value of ε approaches 0.9. It is also observed that the accuracy of [146] is better as compared to the proposed scheme, the reason being the small dataset that the authors have used for the experiment comprising of only 20 keywords. The accuracy of the proposed FRSE scheme can also be increased by reducing the number of keywords in the fuzzy index table.

5.8 Summary

Human typographical errors are common when performing a search. In this chapter, a Fuzzy Ranked Searchable Encryption (FRSE) scheme has been presented that takes the human typographical errors into account. The FRSE scheme unifies the inverted index table and the fuzzy index table by making use of Min Hashing, Euclidean Norm, and Jaccard Similarity. The FRSE scheme shingles the keywords to breakdown the keyword into characters and then generates a probabilistic trapdoor. The probabilistic trapdoors generate a unique query each time a keyword is searched. The intensive security analysis yields that the proposed scheme is in accordance with the newly proposed definitions of keyword-trapdoor and trapdoor-index indistinguishability. Hence, the FRSE scheme does not leak the search pattern and resists against distinguishability attacks.

The asymptotic analysis and the storage overhead analysis yield that the scheme shows a better performance as compared to the state-of-the-art. To analyze the computational complexity, the PoC prototype has been simulated and deployed onto the British Telecommunication's cloud service (BTCS). The deployed FRSE scheme is then tested over a real corpus of encrypted data and the computational time is calculated. The analysis proves that the proposed scheme competes closely with the existing schemes in terms of computational time and accuracy, while enhancing the security and the query effectiveness.

Despite the fact that the proposed FRSE scheme enhances the query effectiveness to the maximum but it comes with a trade-off of lack of scalability and moderate amount of security and privacy as compared to a Homomorphic-based SE (HSE) Scheme. The next chapter proposes a novel HSE scheme that is aimed to enhance the security and privacy while supporting scalability by allowing dynamic updates to the database.

Chapter 6

Homomorphic-based Searchable Encryption Scheme

The emergence of cloud as Database-as-a-Service (DBaaS) [45] [69] platform to store large amounts of data poses the challenge of searching for keyword(s) over the encrypted data. Previous chapters aimed towards enhancing the query effectiveness. This enhancement in the query effectiveness comes with the trade-off of scalability. This chapter presents a novel SE scheme that is based on homomorphic encryption. The proposed construction aims to enhance the security of the system and present a highly scalable solution for the cross-cloud, cloud-of-clouds or nested cloud platforms [15][66]. The proposed Homomorphic-based Searchable Encryption (HSE) scheme presents probabilistic trapdoors and provides non-repudiation [42].

The proposed construction eliminates the requirement of a centralized data-structure such as an index table to achieve effective search. Different types of homomorphic encryption schemes are already discussed in the Section 2.2.5. To present a solution that is lightweight, practical and feasible, the proposed HSE scheme is based on a partial homomorphic encryption scheme.

The following contributions to the field of SE are made through this research:

- We design and present a novel SE scheme that is based on the partially homomorphic property of RSA [115].
- The proposed scheme may be termed an extension of the schemes proposed in the chapter 3 and 4 as it also explores the property of modular inverse to generate probabilistic trapdoors that makes the scheme privacy-preserving and prevents against the search pattern leakage. The proposed scheme provides non-repudiation and thwarts an adaptive adversary from successful distinguishability attacks.
- The proposed scheme eliminates the need of an index table and reduces the client-side computations that are incurred in the conventional index-based system whenever the documents are added, deleted or modified. The removal of an index table reduces the network latency and the storage overhead. We design and implement a proof of concept prototype over a real dataset of encrypted documents.

6.1 Problem Formulation

This section discusses the system model and highlights the design goals. This discussion further leads to presenting the security definitions.

6.1.1 System Model

Bob wants to upload his documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$ to the cloud server (CS). To maintain confidentiality of the documents he encrypts the documents prior to outsourcing them to the CS. The CS is "trusted-but-curious" or "honest-but-curious", *i.e.,* the CS is keen on learning any information related to the documents that can be extracted directly from the encrypted documents or the searched keywords.

In order to search for the document containing a particular keyword W, a naive method would be to download all the documents from the CS, decrypt all the documents and then search for a particular keyword, but this leads to increased network latency. Another method would be to pre-process the documents to generate an index table. The index generation results in increased client-side computations that become apparent whenever the database is modified, increased network latency and increased storage at the server-side. Therefore, Bob requires a Homomorphic-based Searchable Encryption scheme (HSE) to search for keywords directly over the encrypted documents. The HSE scheme should allow the addition, modification and deletion of the documents on run time.

Whenever Bob wants to search for a keyword, he generates a trapdoor and sends it to the CS. The CS searches directly over the encrypted documents and identifies the encrypted documents containing the required keyword. The CS then sends the corresponding encrypted document identifiers to Bob. Figure 6.1 illustrates the above scenario where Bob represents the client.

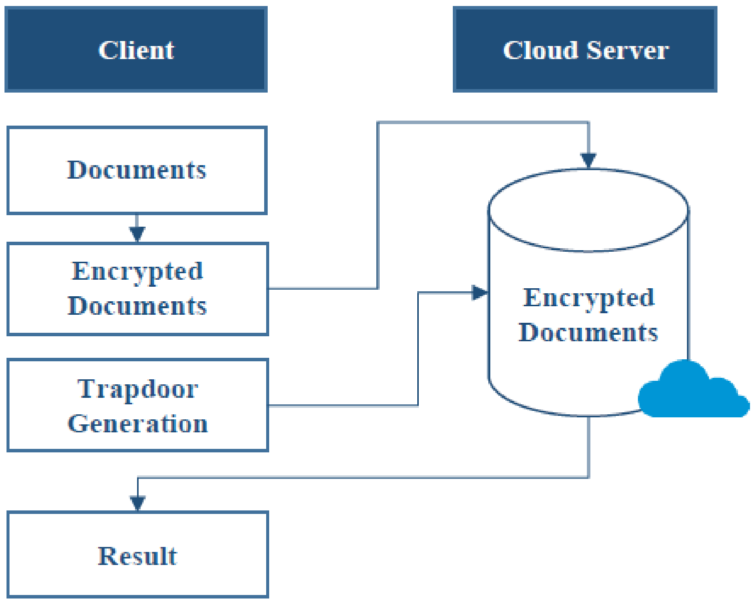

Fig. 6.1 The System Architecture for HSE Scheme

6.1.2 RSA and Homomorphic Encryption

Homomorphic encryption (HE) [60] is the property that allows mathematical computations on the encrypted data to generate an encrypted result. The decryption of the ciphertext matches the result set as if the operations were carried out directly on the plaintext.

An encryption scheme is homomorphic over the operation '$\star$' if the following property holds true:

$$Enc(M_1) \star Enc(M_2) = Enc(M_1 \star M_2), \forall M_1, M_2 \in \mathscr{M} \tag{6.1}$$

where $\mathscr{M}$ represents a set of messages.

RSA [115] is an asymmetric cryptographic algorithm that is partially homomorphic under multiplication. Although RSA is homomorphic, since it is a deterministic scheme it does not provide security against an adaptive adversary, *i.e.,* the adversary can identify the search pattern and launch successful distinguishability attacks leading to passive attacks. The proposed scheme couples symmetric encryption with RSA to provide indistinguishability and prevent for adaptive adversaries.

6.1.3 Design Goals

The proposed Homomorphic-based Searchable Encryption Scheme (HSE) should have the following design objectives:

- **Trapdoor Unlinkability:** The trapdoors should be probabilistic and able to resist distinguishability attacks. This will also prevent against search pattern leakage. The HSE scheme should provide security in the known-ciphertext model, *i.e.*, it should provide security against an adaptive polynomial time adversary.

- **Dynamic Database:** The scheme should allow update, insertion or deletion of the document to the database without the need of pre-processing and re-encrypting the entire database.

Definition (Homomorphic-based Searchable Encryption Scheme (HSE)): The proposed HSE comprises of five polynomial time algorithms $\Pi =$ (KeyGen, Encryption, Build_Trap, Search_Outcome, Decryption) such that:

$(k_{pub}, k_{pri}, K, k_s) \leftarrow$ KeyGen(p, q, λ): represents a probabilistic key generation algorithm. The algorithm takes the security parameter (λ) and large prime numbers (p, q) as the input. The output is a master key (K), session key (k_s), public key $k_{pub} = (e, n)$ and a private key $k_{pri} = (d, n)$. This algorithm is run by the client.

$(Enc(\mathscr{D})) \leftarrow$ Encryption$(K, k_{pri}, \mathscr{D})$: is a deterministic algorithm run by the client. The algorithm takes the master key (K), private key (k_{pri}) and the document set $\mathscr{D}$ as the input and outputs an encrypted document $(Enc(\mathscr{D}))$.

$T_W \leftarrow$ Build_Trap(K, k_s, k_{pri}, W): is a probabilistic algorithm run by the client. The algorithm takes the master key (K), session key (k_s), private key (k_{pri}) and the keyword (W) as the input and outputs a probabilistic trapdoor (T_W) generated for the keyword (W).

$X \leftarrow$ Search_Outcome$(k_s, k_{pub}, Enc(\mathscr{D}), T_W)$: is a deterministic algorithm run by the CS. The algorithm takes the session key (k_s), the public key (k_{pub}), encrypted documents $Enc(\mathscr{D})$, and the trapdoor (T_W) as the input and outputs a set (X) of encrypted document identifiers $Enc_K(id(D_i))$.

$D_i \leftarrow Dec(K, k_{pub}, X)$: is a deterministic algorithm that requires the client's master key (K), public key k_{pub} and encrypted set of document identifiers $Enc_K(id(D_i))$ to decrypt and recover the document id's. This algorithm is run by client.

6.2 Security Definitions

This section extends the security definitions proposed in the Section 3.2 to apply them to the HSE scheme. Although the security definition of Trapdoor-Index indistinguishability requires an index table, the proposed HSE scheme eliminates the need of an index table so the security definitions are also tuned accordingly.

6.2.1 Keyword-Trapdoor Indistinguishability for HSE

Keyword-Trapdoor Indistinguishability refers to the capability of a SE scheme to resist distinguishability attacks. Therefore, even for the same keyword searched again, a unique trapdoor is generated. This keeps the adversary from launching passive attacks. Keyword-Trapdoor indistinguishability should hold true even if the adversary is maintaining a history of searches and adaptively chooses the keyword to be searched.

Description

The challenger encrypts the documents using the HSE scheme. The adversary $\mathscr{A}$ selects a keyword W and sends it to the challenger. The challenger generates a trapdoor and sends it back to the adversary $\mathscr{A}$. This continues until the adversary $\mathscr{A}$ has submitted polynomial-many keywords. Now the adversary $\mathscr{A}$ has to submit two keywords (W_0, W_1) to the challenger and receives a trapdoor corresponding to the keyword W_b, where the selection of b is made by tossing a fair coin. The adversary $\mathscr{A}$ has to guess and output the bit b. If the guess is made with a probability greater than 1/2 then the adversary wins when the security parameter λ is negligible *i.e.*, λ is sufficiently small. Otherwise the HSE scheme provides keyword-trapdoor indistinguishability.

Definition 6.1 *Let* HSE=(KeyGen, Encryption, Build_Trap, Search_Outcome, Dec) *be a Homomorphic-based Searchable Encryption Scheme over a set of docu-*

ments $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$, *$\lambda$ be the security parameter, p, q be the prime numbers and $\mathscr{A} = (\mathscr{A}_1, \cdots, \mathscr{A}_{m+1})$ represent adversaries. Consider the following probabilistic experiment $Key\_Trap_{HSE,\mathscr{A}}(\lambda)$:*

```
$Key\_Trap_{HSE,\mathscr{A}}(\lambda)$
    $(k_{pub}, k_{pri}, K, k_s) \leftarrow KeyGen(p, q, \lambda)$
    $(Enc(\mathscr{D})) \leftarrow Encryption(K, k_{pri}, \mathscr{D})$
    $for\, 1 \leq i \leq m;\, where\, m = \mid \mathscr{W} \mid; \mathscr{W} = \{W_1, W_2, \cdots, W_m\}\, and\, \mathscr{W} \in \mathscr{D}$
        $(st_{\mathscr{A}}, W_i) \leftarrow \mathscr{A}_i(st_{\mathscr{A}}, T_{W_1}, \cdots, T_{W_i})$
        $T_{W_i} \leftarrow Build\_Trap(K, k_s, k_{pri}, W_i)$
    $b \xleftarrow{\$} \{0, 1\}$
    $(st_{\mathscr{A}}, W_0, W_1) \leftarrow \mathscr{A}_0(\lambda)$
    $(T_{W_b}) \leftarrow Build\_Trap_{K,k_{pri}}(k_s, W_b)$
    $b' \leftarrow \mathscr{A}_{m+1}(st_{\mathscr{A}}, T_{W_b})$
    $T'_W \leftarrow Build\_Trap_{K,k_{pri}}(k_s, W_j); 1 \leq j \leq m$
    $if\, b' = b, output\, 1$
    $otherwise\, output\, 0$
```

where $st_{\mathscr{A}}$ represents a string that captures $\mathscr{A}$'s state. The keyword-trapdoor indistinguishability holds for all the polynomial-time adversaries $(\mathscr{A}_0, \mathscr{A}_1, \cdots, \mathscr{A}_{m+1})$ such that $m = poly(\lambda)$,

$$Pr[Key\_Trap_{HSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \tag{6.2}$$

6.2.2 Trapdoor-Document Indistinguishability for HSE

Trapdoor-Document Indistinguishability refers to the complexity offered by a HSE scheme. The keywords, trapdoor and encrypted documents should be complex, and involved in such a way that the trapdoor should not reveal the corresponding document/ document identifiers prior to the search, and should not be distinguishable.

This holds true for the same keyword searched again and the trapdoor should not be distinguishable even if the history (keyword, trapdoor, encrypted document) is generated adaptively.

Description

The challenger encrypts the data collection $\mathscr{D}$. The challenger sends the set of keywords $\mathscr{W}$, the trapdoors generated for all the keywords $\mathscr{W}$, along with the associated document identifiers to the adversary, while maintaining the order in which they occur. The adversary adaptively submits two keywords (W_0, W_1) to the challenger and receives a trapdoor corresponding to the keyword W_b. The adversary has to now decide the corresponding document identifiers and is challenged to output the bit b. If the adversary is able to make a guess with a probability greater than 1/2 then the adversary has succeeded and the scheme lacks in providing trapdoor-document indistinguishability. The security parameter λ is negligible *i.e.*, λ is sufficiently small that it can be ignored to highlight the security of the scheme.

Definition 6.2 *Let* HSE= (KeyGen, Encryption, Build_Trap, Search_Outcome, Dec) *be a Homomorphic-based Searchable Encryption scheme over a set of documents* $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$, λ *be the security parameter and* p, q *be the prime numbers,* $\mathscr{A} = (\mathscr{A}_0, \mathscr{A}_1)$ *represent the adversaries. Consider the following probabilistic experiment* $Trap\_Doc_{HSE,\mathscr{A}}(\lambda)$:

```
$Trap\_Doc_{HSE,\mathscr{A}}(\lambda)$
    $(k_{pub}, k_{pri}, K, k_s) \leftarrow KeyGen(\lambda)$
    $(Enc(\mathscr{D})) \leftarrow Encryption(K, k_{pri}, \mathscr{D})$
    $for\, 1 \leq i \leq n;$
        $let\ D' = D' \cup Enc(D_i)$
        $let\ \mathscr{W} = (W_1, \cdots, W_i)$
        $T_{W_i} \leftarrow Build\_Trap(K, k_{pri}, k_s, W_i)$
```

$$b \xleftarrow{\$} \{0,1\}$$

$$(st_{\mathscr{A}}, W_0, W_1) \leftarrow \mathscr{A}_0(st_{\mathscr{A}}, \lambda, W_m, D', T_{W_m})$$

$$(T_{W_b}) \leftarrow Build\_Trap_{K,k_{pri}}(k_s, W_b)$$

$$b' \leftarrow \mathscr{A}_1(st_{\mathscr{A}}, D_{W_b}, k_{pub})$$

$$if\, b' = b, output\, 1$$

$$otherwise\, output\, 0$$

where $st_{\mathscr{A}}$ represents a string that captures $\mathscr{A}$'s state. The keyword-trapdoor indistinguishability holds if for the polynomial-time adversaries $(\mathscr{A}_0, \mathscr{A}_1)$,

$$Pr[Trap\_Doc_{HSE,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \qquad (6.3)$$

where probability is over the choice of b.

Theorem 6.1: *The proposed HSE scheme provides Keyword-Trapdoor Indistinguishability and Trapdoor-Document Indistinguishability if the trapdoors are probabilistic and the documents are homomorphically encrypted.*

6.3 Proposed HSE scheme

This section gives the details of the proposed HSE scheme already defined in the Section 6.1.1.

6.3.1 Scheme Construction

- **Phase 1-KeyGen** (p, q, λ)**:** Given a security parameter λ, two prime numbers p and q, and generate random keys $K, k_s \leftarrow \{0,1\}^{\lambda}$. Compute:
  - $N = p * q$ and $\varphi(N) = (p-1)(q-1)$.
  - a random integer $e \mid (gcd(e, \varphi) = 1, ed \equiv 1 \bmod \varphi(N))$.
  - $d \mid ed \equiv 1 \bmod \varphi(N))$.

Output the master key K, session key k_s, public key $k_{pub} = (e, N)$ and private key $k_{pri} = (d, N)$.

- **Phase 2-Encryption** $(K, k_{pri}, \mathscr{D})$**:**
  - For $1 \leq t \leq n$:
    - For $1 \leq u \leq |D_t|$:
      - let $a \leftarrow (Enc_K(W_u))$, where $W_u \in D_t$ and Enc_K represents a symmetric encryption algorithm.
      - Compute $r = a^d \bmod N$.
      - $W_u \leftarrow r$.
      - Compute $Enc - Prob_K(id(D_t))$, where $Enc - Prob_K$ represents a probabilistic symmetric encryption algorithm

Output encrypted documents $Enc(\mathscr{D})$ and outsource to the CS.

- **Phase 3-Build_Trap** (K, k_s, k_{pri}, W)**:**
  - let $b \leftarrow (Enc - Prob_K(W))$.
  - let $z \leftarrow (Enc_K(W))$.
  - let $c \leftarrow z^{-1} \bmod N$.
  - let $j \leftarrow c^d \bmod N$.
  - let $f \leftarrow b \cdot j \bmod N$.
  - let $g \leftarrow H_{k_s}(b).(Enc(1))^d \bmod N$.
  - $t \leftarrow (f, g)$.
  - $(T_W) \leftarrow t^d \bmod N$.

Transmit T_W to the CS.

- **Phase 4- Search_Outcome** $(k_s, k_{pub}, T_W))$**:** Identify the documents $D_i \in \mathscr{D}$ as the outcome of the search as follows:

```
– Initialize dynamic array $X$.
– $(f,g) \leftarrow (T_W)^e \bmod N$
– for $1 \leq h \leq n$:
        - for $1 \leq i \leq |Enc(D_h)|$:
          - if $(g == H_{k_s}(f \cdot W_i) \bmod N)$:
          - $X \leftarrow Enc_K(id(D_h))$.
```

Output X, a set of encrypted document identifiers and return to the client.

- **Phase 5-Dec** (K, k_{pub}, X)**:** Given X a set of encrypted document identifiers, decrypt X using the public key k_{pub} and the master key K to uncover the outcome of the search.

6.3.2 Description

This section briefly discusses each of the phases presented in the previous section, in detail below:

KeyGen Phase

The client generates a master key K used for the symmetric key encryption, a session key k_s used for the computing the cryptographic keyed hash function and the asymmetric key pairs (k_{pub}, k_{pri}) for the RSA. The session key k_s and the public key k_{pub} is shared with the CS.

Encryption Phase

The client encrypts the documents $\mathscr{D}$. The encryption is done such that the client accesses each keyword within each document. Firstly the symmetric encryption (AES-ECB) of the keyword using the master key K is performed, then the RSA using the private key (k_{pri}) is computed. The document identifiers are also encrypted using

AES-CBC, where AES-CBC represents a probabilistic encryption algorithm. The entire encryption algorithm can be processed in parallel by using the advancements made in multi-threading. This algorithm also supports the update, addition or removal of a document from the database as a predefined data structure is not required in the proposed HSE scheme.

Build_Trap Phase

The client generates a trapdoor (search query) for a keyword and sends it to the CS. The trapdoor is generated such that the client computes b using a probabilistic symmetric encryption algorithm and then computes z using a deterministic symmetric encryption algorithm. It may be observed that z is already computed in the encryption phase and represented as a. Now the client computes the modular inverse $c \leftarrow z^{-1} \bmod N$ and performs the RSA represented as j. To make the trapdoor probabilistic, the client multiplies b and j. The trapdoor is encrypted using the private key k_{pri} and T_W transmitted to the CS.

Search_Outcome Phase

The CS using the public key k_{pub} decrypts the trapdoor. This provides non-repudiation as it proves the authenticity and integrity of the data originating from the source. Using the trapdoor T_W, the CS searches for the documents containing the required keyword. The search is done against every encrypted keyword within the documents. The desired keyword is identified such that $g == H_{k_s}(f \cdot r) \bmod N$. The searching is entirely based on the partially homomorphic property of RSA (equation 6.1), *i.e.*, it is observed that f and r are actually inverses of each other so the $E(f) \cdot Enc(r) = Enc(f \cdot r) = Enc(1)$. The hash is computed in order to identify the keyword when the trapdoor is probabilistic. Therefore, in such a way every time a unique trapdoor appears before the CS even for the same keyword searched repeatedly. This helps resist distinguishability attacks and prevent from the search pattern leakage. Upon the identification, the encrypted document identifiers are sent to the client. Since the underlying encryption and inverses are distinct, this leads to the correctness of the scheme. Furthermore, since the trapdoor generation is based on the mechanism proposed in the RSE and RMSE schemes presented in the Chapters 3 and 4 respectively, therefore, the correctness of the HSE scheme flows directly from the correctness of the RSE and RMSE schemes.

Decryption Phase

This phase is run by the client to uncover the underlying document identifiers. Later on, the identified documents can be downloaded from the CS and may be decrypted using the public key k_{pub}.

6.4 Security Analysis

This section analyses the security of the proposed HSE scheme according to the security definitions presented in the Section 6.2. Before proceeding towards the formal security proofs the leakage profiles are analyzed.

6.4.1 Leakage Profiles

The leakage profiles highlight all the information given out by the artifacts evolved during the execution of phases of the HSE scheme that may lead to possible security and privacy concerns. The analysis includes the artifacts obtained from the five polynomial time algorithms explained previously, *i.e.,* encrypted documents, trapdoor T_W and the outcome of the search. The leakage focuses on the information that is revealed within polynomial time.

Leakage $L_{6.1}$

Description: The leakage $L_{6.1}$ is associated to the encrypted documents $Enc(\mathscr{D})$. It is assumed the encrypted documents are revealed to all the stakeholders, *i.e.*, the client, the CS and the adversary $\mathscr{A}$. This leakage is defined as:

$$L_1(Enc(\mathscr{D})) = \begin{Bmatrix} Encrypted\, keywords\,(r),\, total\, number\, of\, documents\,(n), \\ Encrypted\, document\, identifiers\,(Enc - Prob_K(id(D_n))), \\ Total\, number\, of\, encrypted\, keywords\,(m), \\ Encrypted\, documents\, Enc(\mathscr{D}) \end{Bmatrix} \tag{6.4}$$

Leakage $L_{6.2}$

The leakage $L_{6.2}$ is associated to the Trapdoor T_W generated for a particular keyword W to be searched. It is assumed that T_W is generated by the client and revealed to all the stakeholders, *i.e.,* the CS and the adversary $\mathscr{A}$.

$$L_{6.2}(T_W) = \begin{Bmatrix} f \leftarrow (Enc - Prob_K(W_m)) \cdot z^{-1} \mathrm{mod}\, N, \\ g \leftarrow H_{k_s}(Enc - Prob_K(W_m)) \end{Bmatrix} \tag{6.5}$$

Leakage $L_{6.3}$

The leakage $L_{6.3}$ is associated to search outcome (SO) of the trapdoor generated for a particular keyword T_W.The search outcome is revealed to all the stakeholders, *i.e.,* the client, CS and the adversary $\mathscr{A}$. This leakage is defined as:

$$L_{6.3}(SO) = \left\{ \mathrm{OC}(W),\, Enc_K(id(D_i))_{\forall T_W \in \mathscr{D}} \right\} \tag{6.6}$$

where OC represents the relevant outcome corresponding to the searched keyword.

Discussion on Leakage: It may be observed that majority of the leakages $L_{6.1}$, $L_{6.2}$ and $L_{6.3}$ are either hashed or encrypted. Furthermore, the proposed scheme is based on probabilistic trapdoors that resist distinguishability attacks and prevent against search pattern leakage.

The next section extends the security analysis by taking these leakage profiles into consideration.

6.4.2 Formal Security Proofs

Lemma 6.1. *The Homomorphic-based Searchable Encryption Scheme (HSE) presented is "privacy-preserving" as it is* $(L_{6.1}, L_{6.2}, L_{6.3})$*-secure (represented as equations 6.46.56.6) and according to Definition* 6.1, 6.2*, where,* $L_{6.1}$ *is associated with the encrypted documents* $Enc(\mathscr{D})$ *and leaks the encrypted keywords, total number of documents, total number of encrypted keywords and the encrypted documents.* $L_{6.2}$ *is associated to the trapdoor* T_W*, generated for a keyword and leaks* f, g *and* $L_{6.3}$ *leaks the outcome of a search against the trapdoor* T_W *and the encrypted document identifiers.*

Proof: The proof of this lemma flows directly from the proof of the Theorem 6.1. This is done by taking a game-based approach and by simulating the keyword-trapdoor indistinguishability and trapdoor-document indistinguishability definitions for the proposed HSE scheme. The proof requires an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. The proof is based on the assumption that if the adversary is able to distinguish between the outcome of the algorithms and between the keywords, trapdoors and the encrypted document identifiers, it will result in compromising the privacy-preserving property of HSE scheme. The game-based approach is divided into three phases, *i.e.*, the setup phase, challenge phase and the outcome phase.

Keyword-Trapdoor Indistinguishability in HSE:

Let HSE be a Homomorphic-based Searchable Encryption scheme. Given n documents $\mathscr{D} = \{D_1, D_2, \cdots, D_n\}$ and m keywords $\mathscr{W} = \{W_1, W_2, \cdots, W_m\}$ where $\mathscr{W} \in \mathscr{D}$. The game is played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$ as follows:

- **Setup Phase:** The adversary $\mathscr{A}$ selects a keyword $W \in \mathscr{W}$ and sends it to the challenger $\mathscr{C}$. The challenger $\mathscr{C}$ returns a trapdoor T_W corresponding to the keyword W to the adversary $\mathscr{A}$. This continues between the adversary $\mathscr{A}$ and the challenger $\mathscr{C}$ until all the trapdoors and associated keywords have not been shared with the adversary.

- **Challenge Phase:** The adversary $\mathscr{A}$ selects two keywords $W_0', W_1' \in \mathscr{W}$ and sends them to the challenger $\mathscr{C}$. The selection of the keywords is done; such that the keywords are unique *i.e.*, $W_0' \neq W_1'$. The challenger $\mathscr{C}$ in response tosses a fair coin $b \leftarrow \{0, 1\}$ and generates a trapdoor $T'_{W_b'}$. After the challenge has been completed, the setup phase is run again and the adversary is allowed to query the same keywords again.

- **Outcome Phase:** Adversary $\mathscr{A}$ is given the generated Trapdoor $T'_{W_b'}$ and it will now have to guess and output $b' \in \{0, 1\}$ and if $b = b'$ then the adversary wins. In other words the adversary $\mathscr{A}$ has to correctly guess the keyword associated to the trapdoor $T'_{W_b'}$ in polynomial time. If the adversary $\mathscr{A}$ correctly guesses the keyword corresponding to the trapdoor then it has won otherwise HSE provides keyword-trapdoor indistinguishability and the challenger $\mathscr{C}$ wins.

Therefore, as the trapdoors are probabilistic, the probability that the adversary $\mathscr{A}$ wins is 1/2 which is according to the definition 6.1 and the equation 6.2

Trapdoor-Document Indistinguishability in HSE:

Let HSE be a Homomorphic-based Searchable Encryption scheme. Let $\mathscr{D}$ represent the set of documents $\{D_1, D_2, \cdots, D_n\}$ and $\mathscr{W}$ represent a set of keywords $\{W_1, W_2, \cdots, W_m\}$ contained in $\mathscr{D}$. The game is played between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. The game is divided into three phases as follows:

- **Setup Phase:** The adversary $\mathscr{A}$ chooses a keyword $W \in \mathscr{W}$ and sends it to the challenger $\mathscr{C}$. The challenger generates a trapdoor T_W and sends the associated encrypted document identifiers $Enc(id(D_i))$ to the adversary $\mathscr{A}$. This continues until the adversary $\mathscr{A}$ has not queried all the keywords.

- **Challenge Phase:** The adversary $\mathscr{A}$ selects two keywords $W_0', W_1' \in \mathscr{W}$ and sends them to the challenger $\mathscr{C}$. The selection of the keywords is done such that $W_0' \neq W_1'$. The challenger $\mathscr{C}$ in response tosses a fair coin $b \leftarrow \{0,1\}$ and generates a trapdoor corresponding to the value of b, *i.e.,* $T'_{W_b'}$. After the challenge has been completed, the adversary $\mathscr{A}$ is given access to the previously generated history that was sent in setup phase and allowed to query the same keywords again.

- **Outcome Phase:** $\mathscr{A}$ is given the generated Trapdoor $T'_{W_b'}$. Adversary $\mathscr{A}$ will now have to guess and return the document identifiers corresponding to the Trapdoor $T'_{W_b'}$ in polynomial time. The adversary $\mathscr{A}$ wins if the guess is correct, otherwise HSE provides trapdoor-document indistinguishability, and the challenger $\mathscr{C}$ wins.

Therefore, the probability that the adversary $\mathscr{A}$ wins is 1/2 which is in-line with the above stated definition 6.2 and equation 6.3.

Proof of the Theorem 6.1 leads to the following corollary:

Corollary 6.1: *Keyword-Trapdoor Indistinguishability and Trapdoor-Document Indistinguishability results in a Privacy Preserving Homomorphic-based Searchable Encryption Scheme.*

Proof Sketch: Let HSE=(KeyGen, Encryption, Build_Trap, Search_Outcome, Dec) be a Homomorphic-based Searchable Encryption scheme. It is already proved that the scheme provides Keyword-Trapdoor Indistinguishability and Trapdoor-Document Indistinguishability. It is now to prove that the leakages $L_{6.1}$,$L_{6.2}$ and $L_{6.3}$ do not effect the security of the HSE scheme and is privacy-preserving. It may be observed that the proposed HSE scheme is based on trusted atomic primitives such as Hash, RSA, AES etc. The only leakage associated to $L_{6.1}$ is the total number of documents and the total number of keywords that are contained within the dataset. However, the documents and the keywords themselves are fully encrypted and the adversary cannot uncover them. Whereas, the leakages $L_{6.2}$ and $L_{6.3}$ are fully encrypted or hashed and do not leak any information that would weaken the Keyword-Trapdoor Indistinguishability or Trapdoor-Document Indistinguishability properties of the HSE.

Therefore, the proposed HSE is privacy-preserving and provides Keyword-Trapdoor Indistinguishability or Trapdoor-Document Indistinguishability.

6.5 Performance Metrics

This section analyses the performance of the proposed HSE scheme. This is achieved two-fold; firstly the asymptotic analysis is performed and then the storage overhead is discussed. The performance analysis highlights the theoretical feasibility of the proposed HSE scheme.

6.5.1 Algorithmic Analysis

This section further studies the proposed scheme by performing the asymptotic analysis. The analysis includes the upper bound complexity analysis of the individual phases that are involved in the scheme. The complexity associated to the keywords and the documents is represented by m and n respectively. As discussed in Section 6.3, the proposed HSE scheme comprises of 5 polynomial time algorithms.

The computational complexity of the KeyGen phase comprises of the multiplication of two large prime numbers p and q represented by $O(l^2)$ where l represents the size of the prime numbers p and q. The greatest common divisor (GCD) is also computed twice which is represented by $O(2log(N))$. As a result the total computational complexity of the KeyGen phase is $O(l^2 + 2log(N))$.

The computational complexity of the Encryption phase is calculated such that it requires a nested for-loop over encryption of the keywords, performing the exponentiation and encrypting the document's identifiers. Thus, the complexity is $O(E(Umn+n))$, where E represents the complexity associated to the encryption and U represents the exponentiation complexity against the random number e.

The computational complexity for the trapdoor generation is $O(2E + 2Ulog(m) + 2)$. The complexity for the Search_Outcome is $O(mn+1)$. Table 6.1 includes the algorithmic complexities of the proposed HSE scheme against a scheme that will be based on the standard RSA without probabilistic trapdoors. This clearly highlights that the proposed HSE scheme provides higher levels of security and privacy guarantees at the cost of increased computations.

Table 6.2 highlights the computational complexities of the proposed HSE schemes against some of the similar and relevant schemes that have been presented in the literature review (Chapter 2). It is evident that by introducing probabilistic trapdoors to the algorithm, the complexity has also increased.

Table 6.1 Algorithmic Comparative Analysis HSE vs. RSA-based SE

Schemes	Encryption	Build_Trap	Search_Outcome
Proposed HSE	$O(E(Umn+n))$	$O(2E+U+2)$	$O(mn+1)$
RSA-based SE	$O(Umn+En))$	$O(U+2)$	$O(mn+1)$

Table 6.2 Algorithmic Comparative Analysis HSE vs. Existing Schemes

Schemes	Encryption	Build_Trap	Search_Outcome
[34] & [35]	$O(3\lambda mn)$	$O(\lambda m)$	$O(2n(\lambda m+m+num))$
[102]	$O(E(3mn+n))$	$O(E(3m+1)$	$O(mn+n)$
Proposed HSE	$O(E(Umn+n))$	$O(2E+U+2)$	$O(mn+1)$

Note: λ represents the security parameter, *num* represents the number of occurrences of a substring within a path.

6.5.2 Storage Overhead

This section discusses the storage overhead of the proposed HSE scheme. As mentioned in the Section 6.3, the client has to store the master key K, a session key k_s and asymmetric key pairs (k_{pub}, k_{pri}).The storage overhead for storing the master key K, a session key k_s is 128 bits each. Whereas, the asymmetric key pairs (k_{pub}, k_{pri}) require 2048 bits. Therefore, the storage at the client-side is $128 \times 2 + 2048 = (2304/8)$ bytes. Referring to the storage at the CS, it has to store a session key k_s and a public key k_{pub} requiring 128+1024=1152 bits. Suppose the average storage required by an encrypted document is represented by D_{avg}, the storage at CS will be $1152 + n \times D_{avg}$ bits.

6.6 Computational Analysis

This section implements the proof of concept prototype and tests it over a real-world dataset already presented in the Section 3.6.2.

6.6.1 System Specification

The implementation is done in JAVA and the workstation used is an Intel Core i5 CPU running at 3.00 GHz and 8GB of RAM. The symmetric encryption algorithm used is 128-bit AES-ECB and AES-CBC mode. RSA-1024 is used to achieve partially homomorphic asymmetric encryption. The cryptographic hash function used is SHA-256. The client side and the server side are implemented on the same machine, and the graphs are generated using MATLAB2016.

6.6.2 Computation Overhead

The KeyGen phase and the Decryption phase are similar to nearly all the existing schemes, therefore, the computational analysis focuses mainly on the Encryption phase, Build_Trap phase and the Search_Outcome phase. We analyze the phases below:

To analyze the computational time of the encryption phase we run our implementation over 10 documents and scale it to 100 documents. In the Figure 6.2, the number of documents are represented along the x-axis and time in seconds is along the y-axis. It is observed that the scheme shows a linear growth with the increase in the number of documents and takes a total of 2806 seconds for encrypting 100 documents. The linear trend is depicted by applying curve fitting/ linear interpolation that shows $y = 27.879x - 13.333$ and $R^2 = 0.9986$. It may be observed that unlike the index-based schemes presented in the chapters 3,4,5, the HSE scheme is more resource consuming, however, it is scalable and allows dynamic databases.

To measure the computational time for the trapdoor generation, we generate a trapdoor for the keyword “about" and it takes a constant time of 0.78 seconds.

To analyze the time for searching required by the CS, we search for the keyword “about" over the encrypted documents. As shown in Figure 6.3, we range the search space between 10 to 100 by varying the number of documents. The number of

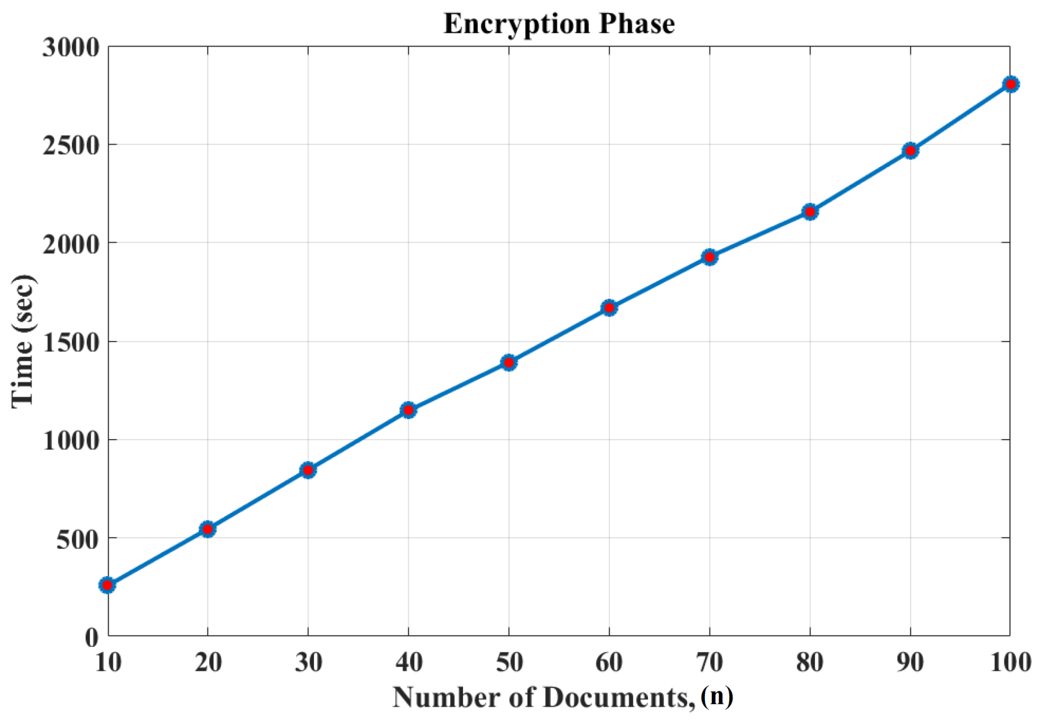

Fig. 6.2 HSE: Computational Time for the Encryption Phase.

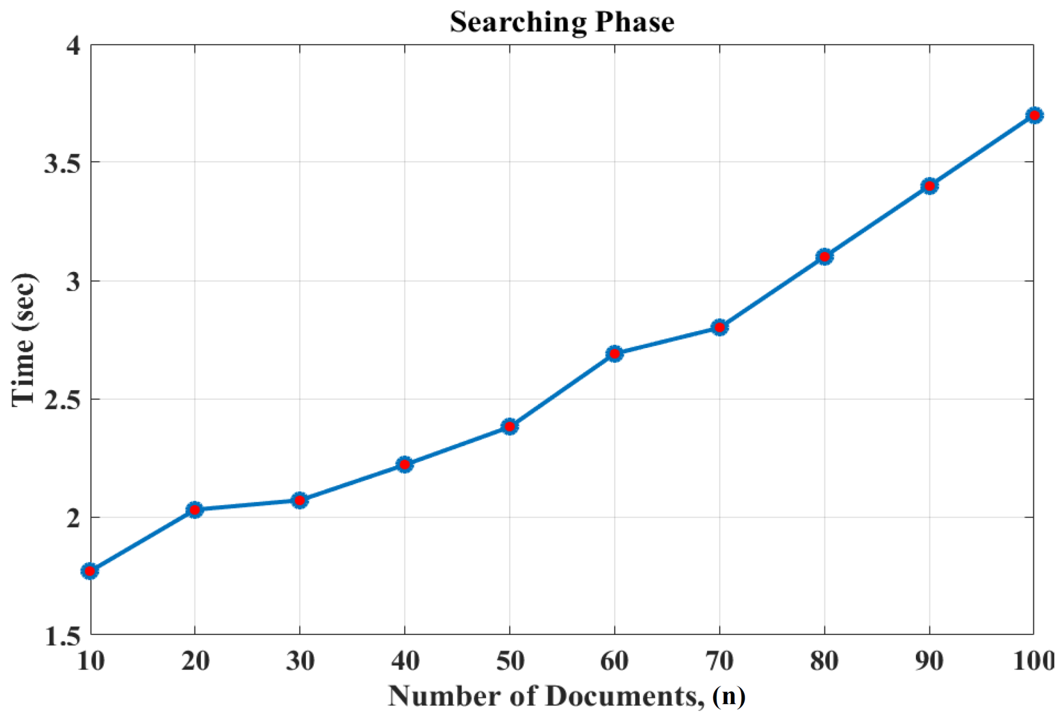

Fig. 6.3 HSE: Computational Time for the Searching Phase.

documents are presented along the x-axis and the time in seconds is along the y-axis. The search phase shows a linear growth by taking 3.7 seconds over 100 documents. The linear trend is again observed by applying curve fitting/ linear interpolation that shows $y = 0.0204x + 1.5013$ and $R^2 = 0.9731$.

6.7 Summary

This chapter presented a novel Homomorphic-based SE scheme. The proposed scheme utilized the partial homomorphic property of the RSA algorithm. RSA is an asymmetric cryptographic algorithm and by default it does not provide security against adaptive adversaries, hence, it is prone to distinguishability attacks. The proposed scheme used RSA over symmetrically encrypted keyword that helped to mitigate distinguishability attacks. The proposed scheme used the modular inverses to introduce probabilistic trapdoors that resist distinguishability attacks and prevented search pattern leakage. Benefiting from the RSA algorithm, the proposed HSE scheme also provided non-repudiation of the trapdoors that were sent by the client. Therefore, the proposed scheme achieved higher levels of security and privacy as compared to a SE scheme based on the standard RSA. The designed and developed proof-of-concept prototype was tested over a real-world speech corpus. The computational complexity showed that the scheme eliminates the need of the pre-processing of the data as an index table is no more required.

A very important aspect of the proposed HSE scheme is that it removed the need of a centralized data-structure, resulting in the scalability across cross-cloud and nested-cloud platforms. Since the client did not have to form an index table and outsource it to the cloud, it reduced the network latency and the storage overhead respectively. However, these advantages came at the cost of increased computations on the client side and the server side. So the proposed scheme can be deployed in scenarios where data security is very important and the data is dispersed on the cloud hosted across different geographic locations.

Chapter 7

Conclusions and Future Directions

With the advancements made in the field of cloud computing and cloud storage, clients are determined to outsource their data to the cloud. The outsourcing of unencrypted data on its own is quite straight forward, however, it leads to the lack of data confidentiality and results in security and privacy concerns. These factors prevent clients from fully benefiting from the advantages the cloud offers and discourages them from storing sensitive data on the cloud. Encrypting the data before outsourcing it to the cloud mitigates the risks associated to the data theft and misuse.

SE is a technique that enables clients to search and sift through the encrypted data stored on the cloud. This research presents a SE framework that makes the security and privacy of the data stored in the cloud achievable. This chapter gives an overview of the work done and goals achieved through this research. This chapter also sheds light on the challenges encountered during this research and how they can lead to impacting research in the future.

7.1 Overview of Research

When it comes to the utilization of the cloud as a platform to store large amounts of data, the security and privacy of the document's and the client's need to be dealt with accordingly and simultaneously. Current SE schemes are based on deterministic trapdoors, and hence do not preserve the privacy of the search. Majority of the existing SE schemes are prone to distinguishability attacks or extremely resource consuming due to which they cannot be deployed on to the existing cloud architectures. To counter this fundamental yet common problem, we introduce lightweight constructions of SE schemes that are based on randomized cryptographic primitives to generate probabilistic trapdoors.

The contributions of this research in the domain of SE have helped in developing state of the art SE schemes that are privacy preserving and resist distinguishability attacks. Perhaps the greatest advantage of this research is that the designed schemes are lightweight and can inter-operate with existing cloud architectures and consume very less client-side and server-side computational resources. The next section gives a summary of the contributions made through this research.

7.2 Summary of Contributions

The cloud is used to store large amounts of data for the on-demand data and resource sharing. Although the cloud storage has many advantages, the other side of the picture is quite grim as the clients lose control of the data once it is outsourced and stored into the cloud. This implies that sophisticated solutions are required that allow the client to store encrypted data on the cloud while facilitating keyword(s) search.

This thesis presents novel SE schemes based on probabilistic trapdoors. The term probabilistic trapdoor is associated to the randomization offered by a SE scheme by generating unique search queries for the same keywords searched repeatedly. A limitation of the existing SE schemes is that they are deterministic and generate same tokens/ trapdoors for the keywords searched repeatedly. Thus lead to successful distinguishability and passive attacks.

As mentioned in the Chapter 2, the existing security definitions adopted in SE have limitations due to which they cannot be used to validate the security of the schemes presented in this thesis when the trapdoors are probabilistic. Thus, this thesis presents new security definitions of keyword-trapdoor indistinguishability and trapdoor-index indistinguishability to appreciate the advantage of having probabilistic trapdoors.

Typically, a SE scheme is designed by taking into consideration the underlying infrastructures and target cloud architectures. In relation to this, Chapter 2 also discusses the existing SE schemes, their design primitives and highlights their pros and cons.

To address the problem of performing keyword search over the encrypted data stored on the cloud, Chapter 3 proposes a novel ranked single-keyword Searchable Encryption (RSE) scheme. The scheme is based on the property of modular inverse that generates probabilistic trapdoors. The probabilistic trapdoors provide indistinguishability and prevents against the search pattern leakage. Therefore, based on the new security definitions, the proposed scheme provides higher levels of security and privacy. The scheme is implemented and deployed onto the British Telecommunications alpha cloud testbed. The proof-of-concept (PoC) prototype is deployed and tested over a real-world dataset. The chapter analyses the storage overhead and algorithmic complexities of the scheme.

Disjunctive query allows a client to perform more targeted search for independent keywords, hence allowing to narrow down the outcome of the search. Chapter 4 extends the RSE scheme to facilitate ranked disjunctive query-based multi-keyword SE. To enhance the usability and efficiency of the scheme, the algorithm is designed such that it can explore the advantages of multi-cores and multi-threading to allow parallelized searching. The scheme in effect will allow to search across sub-folders and sub-directories. The scheme is not only privacy-preserving but it is also efficient. The RMSE scheme is also deployed on the BT cloud server and tested over a dataset of encrypted documents.

The previous SE schemes presented in the chapters 3 and 4 do an exact matching and do not take human typographical errors into account. The novel Fuzzy ranked SE scheme is presented in the Chapter 5 and takes the human typographical errors into consideration and performs search based on the similarity of keywords. The scheme is also based on probabilistic trapdoors and utilizes the concepts of Shingling, Minhashing, Jaccard Similarity and Euclidean Norm to achieve successful search. The scheme makes use of two index tables; fuzzy index table and inverted index table. The FRSE scheme is in accordance with the definitions of the Keyword-Trapdoor and Trapdoor-Index indistinguishability, hence, it is privacy preserving. The FRSE scheme provides correctness and soundness. The scheme is deployed onto the BTCS and the computational complexity is analyzed over a real-world dataset.

Chapter 6 presents a novel homomorphic-based SE scheme. The scheme eliminates the need of a predefined index-table, therefore, allowing the modification, updation and deletion of the documents on runtime. The HSE scheme is based on the partial homomorphic property of RSA and generates probabilistic trapdoors. The removal of an index table reduces the storage overhead and the network latency. The HSE scheme also provides non-repudiation of the trapdoor transmitted to the CS. Since the HSE scheme proposed in the Chapter 6 does not require an index table, therefore, the security definitions are tuned accordingly and provide Keyword-Trapdoor and Trapdoor-Document Indistinguishability. The scheme is implemented and tested over the real-world dataset.

7.3 Challenges and Future Directions

This section discusses the challenges we came across during this research. These challenges will be addressed in the future research.

7.3.1 Malicious Cloud Server

SE is a technique that gives users full control of their personal data stored on the cloud. This thesis has presented SE schemes that focus on enhancing the security and privacy of the clients and their data. These schemes are based on the assumption of the cloud server being trusted-but-curious or honest-but-curious that requires the trapdoors to be probabilistic. In order to further enhance the usability of the SE schemes, the future research should be aimed at towards designing SE schemes for scenarios where the CS is malicious. This may require the SE schemes to provide Proof of Deletion (POD) for data removal in the cloud and Provable Data Possession (PDP) for integrity verification [160].

7.3.2 Multi-user setting

With the growth in the global cloud market, enterprises have started to realize the advantages cloud has to offer and now they have started to migrate their workloads into the cloud. An enterprise is formed by individuals, stakeholders and employees thriving towards the goals of the enterprise. Since, enterprises offer a collaborative environment, in order to fully benefit from the cloud and the resource sharing, a SE scheme can function in a multi-user setting. However, this is not straight-forward and may require system monitoring, governance and access control in place along with sophisticated SE schemes. The SE schemes presented in this thesis are designed for the Single Writer/ Single Reader (S/S) architectures. As discussed in the Section 3.6.1, an Application Encryption (AE) Server can provide the key management for the multi writer/ multi reader (M/M) settings. Therefore, the incorporation of role-based access control [117][53] with the AE server can significantly enhance the usability of the SE schemes in multi-party settings. This amalgamation in effect could give enterprises and their customers trust onto the cloud and the benefit from the extensive resources cloud has to offer.

7.3.3 Dynamic Databases

With the reliance and trust onto the cloud, people will store more and more data onto the cloud. The thesis has focused mainly on presenting SE schemes that are based on probabilistic trapdoors to prevent distinguishability attacks and the proposed schemes are lightweight. The index-based SE schemes that are presented in this thesis require the pre-processing of the data to generate an index table. The proposed index-based SE schemes can be extended to generate sub-indexes whenever a document is added. In this way, the cloud server can benefit from the multi-core processes and execute parallel searching. Although this is achievable at the cost of the pre-processing the data, the ranking of the documents in the true sense cannot be achieved until the entire index tables are not recomputed. This requires to design a SE scheme that allows the index generation on runtime. This research can be extended to facilitate modification, deletion and updation of the index table on run time. Currently, we achieve this by presenting a homomorphic-based SE scheme in the Chapter 6.

7.3.4 ICMetrics Technology and SE

SE allows only an authorized person (in possession of the cryptographic keys) to perform the search. Apparently, SE does not provide protection against the key-theft. ICMetrics [6][80] technology prevents the key theft by extracting the device features and generating the cryptographic keys on run time. Therefore, eliminating the need of storing the cryptographic keys as the same key can be generated whenever required.

Extensive research is being done in the field of ICMetrics for the key generation in one-to-one and group communication [131][127][128]. This leads to a convincing case to combine the ICMetrics technology to prevent key theft and further enhancing the security of the entire system. The merger of ICMetrics and SE could have a profound impact on different verticals including healthcare and the IoT sectors where the devices have very limited resources to securely store the cryptographic keys.

7.3.5 Blockchain and SE

Searchable Encryption is a technology that maintains the confidentiality of the data stored on the cloud. Although, SE has many benefits, it lacks in providing transparency, provenance and immutability of the data. To address this issue, SE can be merged with the blockchain technology. Bitcoin was proposed by Satoshi Nakamoto in [95]. Bitcoin is a peer-to-peer version of electronic cash that does not require a financial institution to govern all the transactions. Blockchain is the underlying technology that turns the concept of Bitcoin into reality. Blockchain has attracted a lot of attention owing to the advantages it offers over the centralized cloud server environment. Blockchain is a distributed architecture that maintains all the data and transactions in the form of a ledger distributed across the nodes. Blockchain offers integrity, provenance and immutability but does not provide confidentiality. The amalgamation of SE with blockchain would allow the users to benefit from the security advantages that both of the technologies offer. To further emphasize on the benefits of this fusion, the areas of application include financial services by bringing trust, simplicity and efficiency to the financial transactions [21]. Blockchain and SE in the field of healthcare provides secure and efficient availability of the data across the peers [89]. Similarly blockchain and SE could have a profound impact in the domains of digital voting [98], driverless cars [37], Internet of Things (IoT) [40] and the education sector [67].

References

[1] (2017). Top 15 most popular file sharing websites | july 2017. http://www.ebizmba.com/articles/file-sharing-websites [Accessed; 30-April-2018].

[2] (2018). Cloud: Driving business transformation. https://www.cloudindustryforum.org/content/cif.

[3] (2018). Data breach statistics. https://breachlevelindex.com/.

[4] Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. (2017). A survey on homomorphic encryption schemes: Theory and implementation. *arXiv preprint arXiv:1704.03578*.

[5] Ahmad, K., Verma, S., Kumar, N., and Shekhar, J. (2011). Classification of internet security attacks. In *Proceeding of the 5th National Conference INDIACom-2011Bharti Vidyapeeth's Institute of Computer Applications and Management, New Delhi ISSN*, pages 0973–7529.

[6] Andrew, Klaus, and Gareth (2010). Device to generate a machine specific identification key.

[7] Arndt, J. (2010). *Generating Random Permutations*. Australian National University.

[8] Asif, W., Ghosh Ray, I., Tahir, S., and Rajarajan, M. (2018). Privacy-preserving anonymization with restricted search (pars) on social network data for criminal investigations. *19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 329–334.

[9] Azgomi, H. and Ghasemi Mahsayeh, M. (2014). Finding similar items in data mining: Shingling method. *Presentation, Iran.*

[10] Bačíková, M. and Porubän, J. (2014). Domain usability, user's perception. In *Human-Computer Systems Interaction: Backgrounds and Applications 3*, pages 15–26. Springer.

[11] Bagherzandi, A., Mohajeri, J., and Salmasizadeh, M. (2008). Comparison based semantic security is probabilistic polynomial time equivalent to indistinguishability. *IJ Network Security*, 6(3):354–360.

[12] Bao, F., Deng, R. H., Ding, X., and Yang, Y. (2008). Private query on encrypted data in multi-user settings. In *International Conference on Information Security Practice and Experience*, pages 71–85. Springer.

[13] Belland, M., Xue, W., Kurdi, M., and Chu, W. (2017). Somewhat homomorphic encryption.

[14] Benaloh, J., Chase, M., Horvitz, E., and Lauter, K. (2009). Patient controlled encryption: ensuring privacy of electronic medical records. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 103–114. ACM.

[15] Bessani, A., Correia, M., Quaresma, B., André, F., and Sousa, P. (2013). Depsky: dependable and secure storage in a cloud-of-clouds. *ACM Transactions on Storage (TOS)*, 9(4):12.

[16] Bijwe, S. and Ramteke, P. (2015). Database in cloud computing-database-as-a service (dbaas) with its challenges'. *International Journal of Computer Science and Mobile Computing*, 4(2):73–79.

[17] Blustein, J. and El-Maazawi, A. (2002). Bloom filters. a tutorial, analysis, and survey. *Halifax, NS: Dalhousie University*, pages 1–31.

[18] Boldyreva, A., Chenette, N., Lee, Y., and O'neill, A. (2009). Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 224–241. Springer.

[19] Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques*, pages 506–522. Springer.

[20] Boneh, D. and Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. *Theory of cryptography*, pages 535–554.

[21] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. (2015). Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE Symposium on Security and Privacy (SP)*, pages 104–121. IEEE.

[22] Bösch, C., Hartel, P., Jonker, W., and Peter, A. (2015). A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR)*, 47(2):18.

[23] Božović, V., Socek, D., Steinwandt, R., and Villányi, V. I. (2012). Multi-authority attribute-based encryption with honest-but-curious central authority. *International Journal of Computer Mathematics*, 89(3):268–283.

[24] Brakerski, Z. and Vaikuntanathan, V. (2014). Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871.

[25] Broder, A. (2005). Algorithms for duplicate documents. *Lecture Notes, Feb*, 18.

[26] Cameron, A. C. and Windmeijer, F. A. (1997). An r-squared measure of goodness of fit for some common nonlinear regression models. *Journal of econometrics*, 77(2):329–342.

[27] Canetti, R., Halevi, S., and Katz, J. (2003). A forward-secure public-key encryption scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 255–271. Springer.

[28] Cao, N., Wang, C., Li, M., Ren, K., and Lou, W. (2014). Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems*, 25(1):222–233.

[29] Carmel, D., Amitay, E., Herscovici, M., Maarek, Y. S., Petruschka, Y., and Soffer, A. (2001). Juru at trec 10-experiments with index pruning. In *TREC*.

[30] Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., and Steiner, M. (2013). Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in cryptology–CRYPTO 2013*, pages 353–373. Springer.

[31] Celebi, M. E., Celiker, F., and Kingravi, H. A. (2011). On euclidean norm approximations. *Pattern Recognition*, 44(2):278–283.

[32] Chang, Y.-C. and Mitzenmacher, M. (2005). Privacy preserving keyword searches on remote encrypted data. In *International Conference on Applied Cryptography and Network Security*, pages 442–455. Springer.

[33] Chase, M. and Kamara, S. (2010). Structured encryption and controlled disclosure. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 577–594. Springer.

[34] Chase, M. and Shen, E. (2015a). Substring-searchable symmetric encryption. *Proceedings on Privacy Enhancing Technologies*, 2015(2):263–281.

[35] Chase, M. and Shen, E. (2015b). Substring-searchable symmetric encryption.

[36] Chatterjee, A. and Sengupta, I. (2015). Searching and sorting of fully homomorphic encrypted data on cloud. *IACR Cryptology ePrint Archive*, 2015:981.

[37] Chavez-Dreyfuss, G. (2017). Toyota, tech firms explore blockchain for driverless cars. https://uk.reuters.com/article/us-toyota-selfdriving-blockchain/toyota-tech-firms-explore-blockchain-for-driverless-cars-idUKKBN18I2G4. [Online; accessed 23-November-2017].

[38] Chen, E., Gomez, I., Saavedra, B., and Yucra, J. (2015). Cocoon: Encrypted substring search.

[39] Cheney, W. and Kincaid, D. (2009). Linear algebra: Theory and applications. *The Australian Mathematical Society*, 110.

[40] Christidis, K. and Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303.

[41] Clark, M. R. and Hopkinson, K. M. (2013). Towards an understanding of the tradeoffs in adversary models of smart grid privacy protocols. In *Power and Energy Society General Meeting (PES)*, pages 1–5. IEEE.

[42] Coffey, T., Saidha, P., and Burrows, P. (2003). Analysing the security of a non-repudiation communication protocol with mandatory proof of receipt. In *Proceedings of the 1st international symposium on Information and communication technologies*, pages 351–356. Trinity College Dublin.

[43] Cohen, D., Amitay, E., and Carmel, D. (2007). Lucene and juru at trec 2007: 1-million queries track.

[44] Columbus, L. (2017). Forrester's 10 cloud computing predictions for 2018. https://www.forbes.com/sites/louiscolumbus/2017/11/07/forresters-10-cloud-computing-predictions-for-2018/#3caa25af4ae1 [Accessed; 30-April-2018].

[45] Curino, C., Evan, J. P. C., Popa, R. A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. (2011). Relational cloud: A database-as-a-service for the cloud. In *5th Biennial Conference on Innovative Data Systems Research*, pages 235–240.

[46] Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2006). Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88. ACM.

[47] Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2011). Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934.

[48] da Silva, E. A. (2016). Practical use of partially homomorphic cryptography.

[49] Ding, S., Li, Y., Zhang, J., Chen, L., Wang, Z., and Xu, Q. (2016). An efficient and privacy-preserving ranked fuzzy keywords search over encrypted cloud data. In *International Conference on Behavioral, Economic and Socio-cultural Computing (BESC)*, pages 1–6. IEEE.

[50] Dobraunig, C., Eichlseder, M., and Mendel, F. (2014). Analysis of sha-512/224 and sha-512/256. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 612–630. Springer.

[51] Du, M., Wang, Q., He, M., and Weng, J. (2018). Privacy-preserving indexing and query processing for secure dynamic cloud storage. *IEEE Transactions on Information Forensics and Security*, 13(9):2320–2332.

[52] Emura, K., Hayashi, T., Kunihiro, N., and Sakuma, J. (2017). Mis-operation resistant searchable homomorphic encryption. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 215–229. ACM.

[53] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. (2001). Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274.

[54] Force, J. T. and Initiative, T. (2013). Security and privacy controls for federal information systems and organizations. *NIST Special Publication*, 800(53):8–13.

[55] Foundation, A. S. (2011). Apache lucene - scoring. last access: 17 April 2018.

[56] Fu, Z., Sun, X., Liu, Q., Zhou, L., and Shu, J. (2015). Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. *IEICE Transactions on Communications*, 98(1):190–200.

[57] Fu, Z., Wu, X., Guan, C., Sun, X., and Ren, K. (2016). Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics and Security*, 11(12):2706–2716.

[58] Fuhr, T. and Paillier, P. (2007). Decryptable searchable encryption. In *International Conference on Provable Security*, pages 228–236. Springer.

[59] Gavin, G. (2016). An efficient somewhat homomorphic encryption scheme based on factorization. In *International Conference on Cryptology and Network Security*, pages 451–464. Springer.

[60] Gentry, C. and Boneh, D. (2009). *A fully homomorphic encryption scheme*, volume 20. Stanford University Stanford.

[61] Glackin, C., Chollet, G., Dugan, N., Cannings, N., Wall, J., Tahir, S., Ray, I. G., and Rajarajan, M. (2017). Privacy preserving encrypted phonetic search of speech data. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6414–6418. IEEE.

[62] Godfrey, J. and Holliman, E. (1993). Switchboard-1 Release 2 LDC97S62-Philadelphia: Linguistic Data Consortium.

[63] Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992). Switchboard: Telephone speech corpus for research and development. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520. IEEE.

[64] Goh, E.-J. et al. (2003). Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216.

[65] González-Burgueño, A., Santiago, S., Escobar, S., Meadows, C., and Meseguer, J. (2015). Analysis of the pkcs# 11 api using the maude-npa tool. In *International Conference on Research in Security Standardisation*, pages 86–106. Springer.

[66] Gorelik, E. (2013). *Cloud computing models*. PhD thesis, Massachusetts Institute of Technology.

[67] Grech, A. and F. Camilleri, A. (2017). Blockchain in education. In *JRC Science for Policy Report*, pages 1–136. European Commission.

[68] Grubbs, P., Sekniqi, K., Bindschaedler, V., Naveed, M., and Ristenpart, T. (2017). Leakage-abuse attacks against order-revealing encryption. In *IEEE Symposium on Security and Privacy*, pages 655–672. IEEE.

[69] Hacigumus, H., Iyer, B., and Mehrotra, S. (2002). Providing database as a service. In *18th International Conference on Data Engineering*, pages 29–38. IEEE.

[70] Han, F., Qin, J., and Hu, J. (2016). Secure searches in the cloud: A survey. *Future Generation Computer Systems*, 62:66–75.

[71] Huth, A. and Cebula, J. (2011). The basics of cloud computing. *United States Computer*.

[72] Hwang, R.-J., Lu, C.-C., and Wu, J.-S. (2014). Searchable encryption in cloud storage. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 8(7):1080–1083.

[73] Hwang, Y. and Lee, P. (2007). Public key encryption with conjunctive keyword search and its extension to a multi-user system. *Pairing-Based Cryptography–Pairing 2007*, pages 2–22.

[74] Islam, M. S., Kuzu, M., and Kantarcioglu, M. (2012). Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Ndss*, volume 20, page 12.

[75] Ji, J., Li, J., Yan, S., Tian, Q., and Zhang, B. (2013). Min-max hash for jaccard similarity. In *13th International Conference on Data Mining (ICDM)*, pages 301–309. IEEE.

[76] Jie, W., Xiao, Y., Ming, Z., and Yong, W. (2014). A novel dynamic ranked fuzzy keyword search over cloud encrypted data. In *12th International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pages 91–96. IEEE.

[77] Kamara, S. and Papamanthou, C. (2013). Parallel and dynamic searchable symmetric encryption. In *International Conference on Financial Cryptography and Data Security*, pages 258–274. Springer.

[78] Kamara, S., Papamanthou, C., and Roeder, T. (2012). Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976. ACM.

[79] Kim, K. S., Kim, M., Lee, D., Park, J. H., and Kim, W.-H. (2017). Forward secure dynamic searchable symmetric encryption with efficient updates. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1449–1463. ACM.

[80] Kovalchuk, Y., McDonald-Maier, K. D., and Howells, G. (2011). Overview of icmetrics technology–security infrastructure for autonomous and intelligent healthcare system. *International Journal of u-and e-Service, Science and Technology*, 4(3):49–60.

[81] Kurosawa, K., Ito, T., and Takeuchi, M. (1988). Public key cryptosystem using a reciprocal number with the same intractability as factoring a large number. *Cryptologia*, 12(4):225–233.

[82] Li, H., Yang, Y., Luan, T. H., Liang, X., Zhou, L., and Shen, X. S. (2016). Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data. *IEEE Transactions on Dependable and Secure Computing*, 13(3):312–325.

[83] Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., and Lou, W. (2010). Fuzzy keyword search over encrypted data in cloud computing. In *Proceedings of INFOCOM*, pages 1–5. IEEE.

[84] Li, K., Zhang, W., Yang, C., and Yu, N. (2015). Security analysis on one-to-many order preserving encryption-based cloud data search. *IEEE Transactions on Information Forensics and Security*, 10(9):1918–1926.

[85] Liu, Z., Choo, K.-K. R., and Zhao, M. (2017). Practical-oriented protocols for privacy-preserving outsourced big data analysis: Challenges and future research directions. *Computers & Security*, 69:97–113.

[86] Machluf, S. (2008). Curve fitting in matlab. *Retrieved September*, 9:2014.

[87] Martins, P., Sousa, L., and Mariano, A. (2017). A survey on fully homomorphic encryption: An engineering perspective. *ACM Computing Surveys (CSUR)*, 50(6):83.

[88] Matousek, M., Bösch, C., and Kargl, F. (2016). Using searchable encryption to protect privacy in connected cars.

[89] Mettler, M. (2016). Blockchain technology in healthcare: The revolution starts here. In *IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), 2016*, pages 1–3. IEEE.

[90] Miller, F. P., Vandome, A. F., and McBrewster, J. (2009). Advanced encryption standard.

[91] Moataz, T. and Shikfa, A. (2013). Boolean symmetric searchable encryption. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 265–276. ACM.

[92] Moore, C., O'Neill, M., O'Sullivan, E., Doroz, Y., and Sunar, B. (2014). Practical homomorphic encryption: A survey. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2792–2795. IEEE.

[93] Moraga, C. (2005). Introduction to fuzzy logic. *Facta universitatis-series: Electronics and Energetics*, 18(2):319–328.

[94] Morris, L. (2013). Analysis of partially and fully homomorphic encryption. *Rochester Institute of Technology*, pages 1–5.

[95] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[96] Naveed, M., Kamara, S., and Wright, C. V. (2015). Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 644–655. ACM.

[97] Nieles, M., Dempsey, K., and Pillitteri, V. Y. (2017). An introduction to information security. *NIST Special Publication*, 800:12.

[98] Noizat, P. (2015). Blockchain electronic vote. *Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data*, page 453.

[99] Olson, D. L. and Delen, D. (2008). *Advanced data mining techniques*. Springer Science & Business Media.

[100] Owens, J. and Matthews, J. (2008). A study of passwords and methods used in brute-force ssh attacks. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*.

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

[102] Papadimitriou, A., Bhagwan, R., Chandran, N., Ramjee, R., Haeberlen, A., Singh, H., Modi, A., and Badrinarayanan, S. (2016). Big data analytics over encrypted datasets with seabed. In *OSDI*, pages 587–602.

[103] Parmar, P. V., Padhar, S. B., Patel, S. N., Bhatt, N. I., and Jhaveri, R. H. (2014). Survey of various homomorphic encryption algorithms and schemes. *International Journal of Computer Applications*, 91(8).

[104] Paverd, A., Martin, A., and Brown, I. (2014). Modelling and automatically analysing privacy properties for honest-but-curious adversaries. *Tech. Rep.*

[105] Pawar, M. V. and Anuradha, J. (2015). Network security and types of attacks in network. *Procedia Computer Science*, 48:503–506.

[106] Peres, P. L., Bonatti, I. S., and Borelli, W. C. (2003). The linear interpolation method: a sampling theorem approach. *Sba: Controle & Automação Sociedade Brasileira de Automatica*, 14(4):439–444.

[107] Phillips, J. M. (2013). Min hashing. https://www.cs.utah.edu/ jeffp/teaching/cs5955/L5-Minhash.pdf.

[108] Poh, G. S., Chin, J.-J., Yau, W.-C., Choo, K.-K. R., and Mohamad, M. S. (2017). Searchable symmetric encryption: designs and challenges. *ACM Computing Surveys (CSUR)*, 50(3):40.

[109] Popa, R. A. and Zeldovich, N. (2013). Multi-key searchable encryption. *IACR Cryptology ePrint Archive*, 2013:508.

[110] Prasanna, B. and Akki, C. (2015). A comparative study of homomorphic and searchable encryption schemes for cloud computing. *arXiv preprint arXiv:1505.03263*.

[111] PUB, F. (1994). Security requirements for cryptographic modules. *FIPS PUB*, 140.

[112] Rajan, A. P. (2013). Evolution of cloud storage as cloud computing infrastructure service. *arXiv preprint arXiv:1308.1303*.

[113] Ray, I. G. and Rajarajan, M. (2017). A public key encryption scheme for string identification. In *Trustcom/BigDataSE/ICESS, 2017 IEEE*, pages 104–111. IEEE.

[114] Rhee, H. S., Park, J. H., Susilo, W., and Lee, D. H. (2010). Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, 83(5):763–771.

[115] Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

[116] Rivest, R. L. and Sherman, A. T. (1983). Randomized encryption techniques. In *Advances in Cryptology*, pages 145–163. Springer.

[117] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.

[118] Securosis, L. (2013). Understanding and selecting a key management solution. Technical report, Technical Report, Feb.

[119] Shafagh, H., Hithnawi, A., Burkhalter, L., Fischli, P., and Duquennoy, S. (2017). Secure sharing of partially homomorphic encrypted iot data. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, page 29. ACM.

[120] Shafi, G. and Micali, S. (1984). Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299.

[121] Shekokar, N., Sampat, K., Chandawalla, C., and Shah, J. (2015). Implementation of fuzzy keyword search over encrypted data in cloud computing. *Procedia Computer Science*, 45:499–505.

[122] Silberschatz, A., Galvin, P. B., and Gagne, G. (2014). *Operating system concepts essentials*. John Wiley & Sons, Inc.

[123] Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55. IEEE.

[124] Stallings, W. (2006). *Cryptography and Network Security, 4/E*. Pearson Education India.

[125] Strizhov, M. and Ray, I. (2016). Secure multi-keyword similarity search over encrypted cloud data supporting efficient multi-user setup. *Transactions on Data Privacy*, 9(2):131–159.

[126] Sun, W., Lou, W., Hou, Y. T., and Li, H. (2014). Privacy-preserving keyword search over encrypted data in cloud computing. In *Secure cloud computing*, pages 189–212. Springer.

[127] Tahir, H. (2017). *An ICMetric based multiparty communication framework*. PhD thesis, University of Essex.

[128] Tahir, H., Tahir, R., and McDonald-Maier, K. (2018a). On the security of consumer wearable devices in the internet of things. *PloS one*, 13(4):e0195487.

[129] Tahir, S. and Rajarajan, M. (2018). Privacy-preserving searchable encryption framework for permissioned blockchain networks. In *2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics*, pages 1628–1633. IEEE.

[130] Tahir, S., Rajarajan, M., and Sajjad, A. (2017a). A ranked searchable encryption scheme for encrypted data hosted on the public cloud. In *International Conference on Information Networking (ICOIN)*, pages 242–247. IEEE.

[131] Tahir, S. and Rashid, I. (2016). Icmetric-based secure communication. In *Innovative Solutions for Access Control Management*, pages 263–293. IGI Global.

[132] Tahir, S., Ruj, S., Rahulamathavan, Y., Rajarajan, M., and Glackin, C. (2017b). A new secure and lightweight searchable encryption scheme over encrypted cloud data. *IEEE Transactions on Emerging Topics in Computing*, 99(99):1–1.

[133] Tahir, S., Ruj, S., and Rajarajan, M. (2017c). An efficient disjunctive query enabled ranked searchable encryption scheme. *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 425–433.

[134] Tahir, S., Ruj, S., Sajjad, A., and Rajarajan, M. (2018b). Fuzzy keywords enabled ranked searchable encryption scheme for a public cloud environment. *Computer Communications*.

[135] Tahir, S., Steponkus, L., Ruj, S., Rajarajan, M., and Sajjad, A. (2018c). A parallelized disjunctive query based searchable encryption scheme for big data. *Future Generation Computer Systems*.

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

[137] Tang, Q. (2014). Nothing is for free: security in searching shared and encrypted data. *IEEE Transactions on Information Forensics and Security*, 9(11):1943–1952.

[138] Tang, Q. and Chen, L. (2009). Public-key encryption with registered keyword search. In *European Public Key Infrastructure Workshop*, pages 163–178. Springer.

[139] Tarkoma, S., Rothenberg, C. E., and Lagerspetz, E. (2012). Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155.

[140] Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer.

[141] Van Rompay, C., Molva, R., and Önen, M. (2018). Secure and scalable multi-user searchable encryption. In *Proceedings of the 6th International Workshop on Security in Cloud Computing*, pages 15–25. ACM.

[142] Wang, B. (2016). *Search over Encrypted Data in Cloud Computing*. PhD thesis, Virginia Tech.

[143] Wang, B., Li, M., and Wang, H. (2016a). Geometric range search on encrypted spatial data. *IEEE Transactions on Information Forensics and Security*, 11(4):704–719.

[144] Wang, B., Song, W., Lou, W., and Hou, Y. T. (2015). Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 2092–2100. IEEE.

[145] Wang, B., Song, W., Lou, W., and Hou, Y. T. (2017). Privacy-preserving pattern matching over encrypted genetic data in cloud computing. In *INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE.

[146] Wang, B., Yu, S., Lou, W., and Hou, Y. T. (2014). Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *Proceedings of INFOCOM*, pages 2112–2120. IEEE.

[147] Wang, C., Cao, N., Li, J., Ren, K., and Lou, W. (2010). Secure ranked keyword search over encrypted cloud data. In *30th International Conference on Distributed Computing Systems (ICDCS)*, pages 253–262. IEEE.

[148] Wang, C., Cao, N., Ren, K., and Lou, W. (2012a). Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transactions on parallel and distributed systems*, 23(8):1467–1479.

[149] Wang, J., Chen, X., Ma, H., Tang, Q., Li, J., and Zhu, H. (2012b). A verifiable fuzzy keyword search scheme over encrypted data. *J. Internet Serv. Inf. Secur.*, 2(1/2):49–58.

[150] Wang, J., Ma, H., Tang, Q., Li, J., Zhu, H., Ma, S., and Chen, X. (2012c). A new efficient verifiable fuzzy keyword search scheme. *JoWUA*, 3(4):61–71.

[151] Wang, J., Ma, H., Tang, Q., Li, J., Zhu, H., Ma, S., and Chen, X. (2013a). Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. *Computer science and information systems*, 10(2):667–684.

[152] Wang, J., Wang, J., Yu, N., and Li, S. (2013b). Order preserving hashing for approximate nearest neighbor search. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 133–142. ACM.

[153] Wang, Q., He, M., Du, M., Chow, S. S., Lai, R. W., and Zou, Q. (2016b). Searchable encryption over feature-rich data. *IEEE Transactions on Dependable and Secure Computing*.

[154] Weis, J. and Alves-Foss, J. (2011). Securing database as a service: Issues and compromises. *IEEE Security & Privacy*, 9(6):49–55.

[155] Welch, D. and Lathrop, S. (2003). Wireless security threat taxonomy. In *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*, pages 76–83. IEEE.

[156] Wu, J., Ping, L., Ge, X., Wang, Y., and Fu, J. (2010). Cloud storage as the infrastructure of cloud computing. In *International Conference on Intelligent Computing and Cognitive Informatics (ICICCI)*, pages 380–383. IEEE.

[157] Wu, L., Chen, B., Choo, K.-K. R., and He, D. (2018). Efficient and secure searchable encryption protocol for cloud-based internet of things. *Journal of Parallel and Distributed Computing*, 111:152–161.

[158] Yang, H.-M., Xia, Q., Wang, X.-f., and Tang, D.-h. (2012). A new somewhat homomorphic encryption scheme over integers. In *International Conference on Computer Distributed Control and Intelligent Environmental Monitoring (CD-CIEM)*, pages 61–64. IEEE.

[159] Yang, Y., Lu, H., and Weng, J. (2011). Multi-user private keyword search for cloud computing. In *IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 264–271. IEEE.

[160] Yu, X. and Wen, Q. (2014). A multi-function provable data possession scheme in cloud computing. *IACR Cryptology ePrint Archive*, 2014:606.

[161] Zadeh, L. A. (2004). Fuzzy logic systems: Origin, concepts, and trends. *Computer Science Division Department of EECS UC Berkeley*.

[162] Zahrotun, L. (2016). Comparison jaccard similarity, cosine similarity and combined both of the data clustering with shared nearest neighbor method. *Computer Engineering and Applications Journal*, 5(1):11–18.

[163] Zhang, C., Li, J., Wang, S., and Wang, Z. (2017). An encrypted medical image retrieval algorithm based on dwt-dct frequency domain. In *15th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 135–141. IEEE.

[164] Zhao, R. and Iwaihara, M. (2017). Lightweight efficient multi-keyword ranked search over encrypted cloud data using dual word embeddings. *arXiv preprint arXiv:1708.09719*.

[165] Zhao, Y., Wang, X., and Tang, H. (2015). Secure genomic computation through site-wise encryption. *AMIA Summits on Translational Science Proceedings*, 2015:227.

[166] Zheng, M. and Zhou, H. (2013). An efficient attack on a fuzzy keyword search scheme over encrypted data. In *10th International Conference on High Performance Computing and Communications*, pages 1647–1651. IEEE.

[167] Zittrower, S. and Zou, C. C. (2012). Encrypted phrase searching in the cloud. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 764–770. IEEE.