This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

A New Secure and Lightweight Searchable Encryption Scheme over Encrypted Cloud Data

Shahzaib Tahir, Sushmita Ruj, Senior Member, IEEE, Yogachandran Rahulamathavan, Member, IEEE, Muttukrishnan Rajarajan, Senior Member, IEEE, Cornelius Glackin

***Abstract*— Searchable Encryption is an emerging cryptographic technique that enables searching capabilities over encrypted data on the cloud. In this paper, a novel searchable encryption scheme for the client-server architecture has been presented. The scheme exploits the properties of the modular inverse to generate a probabilistic trapdoor which facilitates the search over the secure inverted index table. We propose indistinguishability that is achieved by using the property of a probabilistic trapdoor. We design and implement a proof of concept prototype and test our scheme with a real dataset of files. We analyze the performance of our scheme against our claim of the scheme being light weight. The security analysis yields that our scheme assures a higher level of security as compared to other existing schemes.**

***Index Terms*— Searchable Encryption, Modular Inverse, Extended Euclidean Algorithm, Indistinguishability, Privacy Preservation, Inverted Index, Database as a Service (DaaS).**

I. Introduction

THE cloud is an environment that provides the utility of on-demand resource sharing and data access to its clients and their devices remotely. Apart from the core categories of cloud services i.e. SaaS, PaaS, IaaS, nowadays, Database as a Service (DaaS) 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 with DaaS relating to security, trust, expectations, regulations and performance issues [1]. The concerns above 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 i.e. Searchable Encryption (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 data, while enabling the search over the encrypted documents. There are three main challenges associated with SE as discussed in [2] i.e. (1) efficiency, (2) security and (3) query expressiveness. These three terms can be assumed to be the vertices of a triangle and an idealistic SE scheme should be developed in such a way that it transforms the triangle into an equilateral triangle. In other words, a balance needs to be maintained between the faced challenges while designing a SE scheme.

The National Security Agency (NSA) has highlighted concerns related to security in the cloud and has proposed the use of homomorphic encryption [3]. Homomorphic encryption enables the processing of encrypted data. Although homomorphic encryption has revolutionized the field of cryptography, there are still major concerns related to its performance. In [4] the authors have conducted a survey and comparison of different homomorphic and non-homomorphic SE schemes. Their result yields that non-homomorphic SE schemes out-perform homomorphic SE schemes in terms of efficiency.

Until now the use of SE has been explored in connection with E-mail servers [5], to conduct searches on confidential emails. In the healthcare domain [6][7], SE has been researched as an effective method of providing keyword search on patients health records. SE could have a profound impact on areas related to telecom, e-commerce, warfare, big data analysis and cloud storage.

In this paper, we present a novel lightweight ranked SE scheme. We develop and implement a proof of concept prototype and test it on a database containing more than 100,000 documents. To validate our scheme in a practical real-life scenario we have implemented and tested it in a telecom environment. We use the Switchboard-1 speech database [8] that is a corpus of spontaneous conversations which addresses the growing need for large multi-speaker databases of telephone 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 of text, spoken by over 500 speakers of both genders from every major dialect of American English. This database consists of 120,000 distinct keywords. Hence, we prove that our scheme can perform efficient keyword search on telephone speech. Furthermore, our scheme

S. Tahir is with the Information Security Group, School of Mathematics, Computer Science and Engineering, City, University of London, UK, EC1V 0HB, on leave from the National University of Sciences and Technology (NUST), Islamabad, Pakistan (e-mail: shahzaib.tahir@city.ac.uk; shahzaib.tahir@mcs.edu.pk).

S. Ruj is with Indian Statistical Institute, 203 B.T. Road, Kolkata 700108, India. (e-mail: sush@isical.ac.in).

Y. Rahulamathavan is with the Loughborough University Epinal Way, Loughborough, Leicestershire LE11 3TU, UK. (email: y.rahulamathavan@lboro.ac.uk)

M. Rajarajan is with the Information Security Group, School of Mathematics, Computer Science and Engineering, City, University of London, UK, EC1V 0HB, (email: r.muttukrishnan@city.ac.uk)

C. Glackin is with the Intelligent Voice Limited, St Clare House, 30-33 Minories, London, EC3N 1BP, (email: neil.glackin@intelligentvoice.com)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

can be equally helpful for performing SE in the aforementioned domains.

Random Oracle Model (ROM) is based on the basic assumption that the cryptographic primitives are replaced with idealized versions. We prove the security of our scheme in the standard model that only limits the adversary by the resources available i.e. time and computational resources.

A. Our Contributions

Following contributions to the field of SE have been made in this work:

- Our foremost contribution is that we enumerate the properties of a "secure" ranked SE scheme by formally defining keyword-trapdoor indistinguishability and trapdoor-index table indistinguishability.
- We design and present a novel Ranked based Searchable Encryption scheme that is completely based on a probabilistic encryption algorithm to address the passive attacks.
- We design and implement a proof of concept prototype and 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.

B. Organization

Section II presents the literature review in which existing SE schemes are discussed. Section III discusses the Ranked Searchable Encryption Scheme (RSE) model by formally defining our construction. In Section IV, we revisit the security definitions related to searchable encryption, and propose new definitions for our proposed RSE scheme. Finally, in Section V, we present our RSE scheme followed by a security analysis. In Section VI, we perform a comparative analysis of the existing scheme against our scheme in terms of complexity. We also develop a proof of concept prototype and test our scheme with a live dataset of documents by analyzing the computational time. The computational time along with the storage overhead is analyzed in Section VI. The conclusions along with the future work are drawn towards the end of the paper, in Section VII.

II. Literature Review

A state of the art searchable encryption scheme must maintain a balance between security, efficiency and query effectiveness. Previous research fails to maintain this balance thus resulting in a system that lacks adaptability. In this section, we discuss some significant schemes.

Wang *et al.* in [9][10] 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 was an extension of [11] and they added secure ranking to it. Both schemes enable the server to perform ranked keyword search on a 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 indistinguishability. There is an advantage of their later scheme in that it provides a a dynamic inverted index i.e. whenever a new file is added to the server the re-ranking does not need to be done, but this comes at an increased computational cost, which will be discussed in Section VI. Furthermore, the latter scheme helps to keep the ranking score encrypted that will help to avoid leakage of occurrence of a particular keyword to the server. However, in [12] 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 [13] 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 [11]. Their scheme facilitates the adding, 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. Although this can be termed as a breakthrough in the field of searchable encryption, there is a drawback of this scheme i.e. the generated trapdoor is deterministic, and the same trapdoor is generated for the same keyword every time, hence it cannot resist distinguishability attacks. Furthermore, they have also analyzed that their scheme leaks even more information as compared to the previous scheme, hence this scheme cannot be termed as an ultimate solution.

Wang *et al.* in [14] 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, resulting in extra network traffic. Furthermore, with the increase in the outsourced data, the size of the bloom filter is increased, resulting in a slowing down of the search. 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 search comes with a tradeoff of privacy as the tree may reveal the path pattern. Hence this scheme does not provide the desired level of security and privacy and reveals too much information.

Tang in [15] have proposed a symmetric searchable multiparty encryption scheme (MPSE) that is an extension of [16]. In their scheme, they introduce a 'Follow' algorithm that allocates a token to the writer to be distributed among the readers (users) of the index table. This token authorizes the reader to perform the search on the index table. This scheme facilitates dynamic users but does not allow dynamic databases. The authors assume that there is a secure channel between the user and the cloud server to transmit the trapdoors. Although, the secure channel hides the leakage of the trapdoor during transmission, the trapdoor is based on a one-way hash function,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

due to which the server itself can learn the access pattern and the keyword being searched for, since the same trapdoor is generated for the same keyword. In other words, the trapdoor is distinguishable. Their scheme uses a forward index i.e. an index for each file due to which the ranking cannot be performed.

III. Ranked Searchable Encryption 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, Alice (Client) and a Cloud Server (CS). Alice intends to upload all of her documents $\mathcal{D}=\{D_1, D_2, \dots, D_N\}$ to the CS to enable remote access. The CS performs the searching of relevant documents on behalf of Alice. In the scheme, it is assumed that the CS is a trusted but curious server. Being trusted 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.

Alice identifies a set of keywords $\mathcal{W}=\{W_1, W_2, \dots, W_m\}$ from the set of documents $\mathcal{D}$, 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 single 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. Therefore, ranked searching is not effective in multi-keyword or expressive queries.

Equation 1 in [17] is commonly used for relevance frequency generation, and is widely adopted by researchers for designing ranked based SE schemes. For example in [10][9] the authors have used Equation 1 for the ranking in searchable encryption

$$RF(W,D) = \sum_{t\in W} \frac{1}{|D|} \cdot \left(1 + \ln f_{D,W}\right) \cdot \ln\left(1 + \frac{N}{f_W}\right) \qquad (1)$$

where W denotes the keyword to be searched; D denotes the document; $|D|$ denotes length of the document obtained by counting the words appearing in the document D; $f_{D,W}$ denotes the number of times a word W appears within a particular document D; f_W denotes the number of documents in the dataset that contain the word W and N denotes the total number of documents in the dataset.

Now Alice generates an index table I (see Section V for more details), and outsources I along with the encrypted documents $\mathcal{D}$ to the CS.

If Alice wants to search for a document containing a specific keyword, she simply generates a probabilistic trapdoor T and sends it to CS. CS uses the trapdoor T to search the index table I and return a set of relevant documents in a ranked order. Figure 1 shows the flow of events of 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.

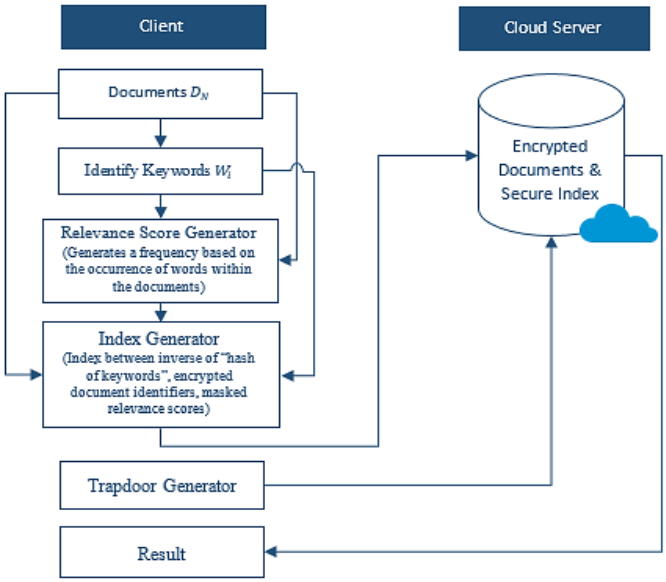


Fig. 1. The flow of events of proposed RSE scheme.

In order to highlight the advantage of indeterministic/ probabilistic trapdoors, we revisit the definition of probabilistic encryption also termed as randomized encryption [18].

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 $\Pi = \cup_{r\in R, k\in K, x\in M}\left(x, k, \psi(r, x, k)\right)$.

If the trapdoors are 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.

A. Scenario related to a Passive Attack

Probabilistic trapdoors mean that for the same keyword being searched twice, a unique search token (trapdoor) may be generated every time. One may appreciate the advantage of having a probabilistic trapdoor by considering an Adversary $\mathcal{A}$, capable of launching a passive attack. A passive attack over the network enables the adversary to monitor, read and capture data exchanges. 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, and over a longer period of time the adversary $\mathcal{A}$ may uncover the entire set of keywords. On the contrary, if the trapdoors are probabilistic then even if the adversary is able to uncover the keyword from the trapdoor, it may be effective only for that particular trapdoor and the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

adversary may not be able to predict future trapdoors. Section IV and Section V, G. explains the advantages of using probabilistic trapdoors, and how the existing schemes may be unsecure against a passive adversary.

We now formally define our proposed Ranked Searchable Encryption scheme (RSE) 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 Scheme (RSE)) A RSE comprises of five polynomial time algorithms* $\Pi = (\text{KeyGen}, \text{Build\_Index}, \text{Build\_Trap}, \text{Search\_Outcome}, \text{Dec})$ *such that*:

$(K, k_s, p) \leftarrow \text{KeyGen}(1^\lambda)$: *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, a session key* k_s *and a prime number* p.

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

$T_w \leftarrow$ Build_Trap*(*K, k_s, w*,num): is a probabilistic algorithm run by the client. The algorithm requires the master key* K, *a session key* k_s , *keyword w and the number (num) of documents D required as the input. 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 document identifiers encrypted* $Enc_K(id(D_i))$ *containing the keyword w in ranked order.*

$D_i \leftarrow \text{Dec}(K, X)$*: is a deterministic algorithm run by the client. The algorithm takes the client's master key and encrypted set of document identifiers* $Enc_K(id(D_i))$ *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 $\text{KeyGen}(1^\lambda)$, for (I) output by $\text{Build\_Index}(K, D)$, 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:
  $$\text{Search\_Outcome}(k_s, I, T_w) = D \cap Dec(K, X) = D_i, \text{where } 1 \leq i \leq n$$
- If $w \notin D_i$ then the following should hold with an overwhelming probability:
  $$\text{Search\_Outcome}(k_s, I, T_w) = D \cap Dec(K, X) = 0$$

Soundness: A RSE scheme is sound if for the security parameter λ, the master key K and the session key k_s generated by $\text{KeyGen}(1^\lambda)$, for (I) output by $\text{Build\_Index}(K, D)$, 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
  $$\text{Search\_Outcome}(k_s, I, T_w)=1$$
- If $w \notin D_i$ then the following should hold with an overwhelming probability
  $$\text{Search\_Outcome}(k_s, I, T_w)=0$$

IV. Security Definitions for Ranked Searchable Encryption (RSE)

In the context of searchable encryption, security is studied regarding the privacy preservation of the data outsourced to the CS [19][20].

A. 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 [21] were the first to come up with a practical way of searching symmetrically over encrypted data. Until then there was no formal definition regarding security for SE. Since 2000 several definitions and constructions related to SE have been presented. In 2003, for the first time Goh [22] 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 met his 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 the same in order to achieve indistinguishability, and if the index is indistinguishable the trapdoors need not be kept secure. Since their definitions were focused towards secure indices and not probabilistic trapdoors, their definitions could not be generalized.

In [20] the authors came up with an extension of IND-CKA that aimed to counter the assumption of same-size documents. They supported their definition by presenting a secure index construction called the z-index which was based on bloom filters. As highlighted in [11], this definition was not secure and would be fulfilled by any insecure searchable encryption scheme. Later Goh introduced extended definitions IND1/2-CKA, now for which the documents did not need to be of the same size, and the trapdoor was again not kept secure. Curtmola *et al.* in [11][19] claimed that all the previous definitions did not provide adequate security and proposed two new definitions termed Adaptive/Non-Adaptive Indistinguishability Security for SSE. Both of the newly proposed definitions have their weaknesses, and do not provide an adequate level of indistinguishability. We discuss the limitations of their slightly stronger definition i.e. Adaptive Indistinguishability below.

B. Limitations of previous definitions

As mentioned earlier, Curtmola's definitions are widely accepted and used. They introduce four terms in [19][23] 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 represent the outcome, i.e. the documents contain a particular keyword. The Search pattern tells if the same keyword is being searched every time. The trace of a history

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

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 being 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 that their SSE-2 lacks in maintaining privacy associated with the trapdoor, and hence is prone to distinguishability attacks. Their construction (SSE-2) generates the same (deterministic) trapdoor every time the same keyword is queried. As a result the search pattern discloses which trapdoors correspond to the same underlying keywords, resulting in privacy concerns (*cf* Section 4.2 of [19]). The deterministic trapdoor reveals the corresponding history tuple "prior" to the search.

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 an indistinguishable index table and ensures trapdoor indistinguishability that results in an increase in the security and privacy of the entire system.

Now, we can formally state the privacy concerns associated to RSE. A RSE 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 the 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.

C. Security Definitions for Proposed RSE

We now revisit the existing definitions of SE that will be utilized to prove the security of our proposed scheme. We propose new definitions for indistinguishability in terms of ranked searchable encryption. An ideal searchable encryption scheme should fulfill all these definitions to ensure privacy. In Section V, G., we prove that our scheme complies with the following definitions.

1) Definition 1 (D_1)*: Non-Adaptive Indistinguishability for Searchable Encryption*

Non-Adaptive means that the adversary $\mathcal{A}$ cannot make queries based on the outcome of the previous query [11][19]. Therefore, the searchable scheme preserves the security in the sense of non-adaptive indistinguishability if for any two non-adaptively constructed histories (documents & keywords) with equal length and trace (documents length, search pattern and outcome) no adversary can distinguish between the view (encrypted documents, trapdoors & Index) of one history from the view of the other in polynomial time with non-negligible probability over $1/2$.

2) Definition 2 (D_2)*: Adaptive Indistinguishability for Searchable Encryption*

Adaptive means that the adversary A can make queries based on the outcome of the previous query [11][19]. Therefore, the searchable scheme preserves the security in the sense of adaptive indistinguishability if for any two adaptively constructed histories (documents & keywords) with equal length and trace (documents length, search pattern and outcome) no adversary can distinguish between the view (encrypted documents, trapdoors & Index) of one history from the view of the other in polynomial time with non-negligible probability over $1/2$.

3) Keyword-Trapdoor Indistinguishability for Ranked Searchable Encryption Scheme

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 (plain text) 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.

The challenger begins by generating an index table against a data collection $\mathcal{D}$. The adversary selects a keyword w and sends it to the challenger. The challenger generates a trapdoor and sends it back to the adversary. This continues until the adversary has submitted polynomial-many keywords. 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 guess and output, the bit b.

Definition 3 (D_3)(Keyword-Trapdoor Indistinguishability). *Let* RSE = (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) *be a Ranked Searchable Encryption scheme over a dictionary* $\mathcal{W}$, λ *be the security parameter,* $\mathcal{D}$ *be the set of documents and* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_{N+1})$ *be such that* $N \in \mathbb{N}$ *and consider the following probabilistic experiment* $\mathbf{Key\_Trap}_{\mathrm{RSE},\mathcal{A}}(\lambda)$**:**

$\mathbf{Key\_Trap}_{\mathrm{RSE},\mathcal{A}}(\lambda)$
$(K, k_s, p) \leftarrow \mathrm{KeyGen}(1^\lambda)$
$(I) \leftarrow \mathrm{Build\_Index}(K, D)$
$for\ 1 \le i \le N$
$(st_{\mathcal{A}}, w_i) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, T_{w_1}, \dots, T_{w_i})$
$T_{w_i} \leftarrow \mathrm{Build\_Trap}_K(w_i)$
$b \xleftarrow{\$} \{0,1\}$
$(st_{\mathcal{A}}, w_0, w_1) \leftarrow \mathcal{A}_0(1^\lambda)$
$T_{w_b} \leftarrow \mathrm{Build\_Trap}(K, k_s, w_b, \mathrm{num})$
$b' \leftarrow \mathcal{A}_{N+1}(st_{\mathcal{A}}, T_{w_b})$
$T_{w'} \leftarrow \mathrm{Build\_Trap}_K(w_j);\ j \in N$
$if\ b' = b, output\ 1$
otherwise $output\ 0$

where $st_{\mathcal{A}}$ is a string that represents and captures $\mathcal{A}'s$ state. We say that the keyword-trapdoor indistinguishability holds if for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

all polynomial-size adversaries $(\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_{N+1})$ such that $N = \text{poly}(\lambda)$,

$$\Pr[\mathbf{Key\_Trap}_{\text{RSE},\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \text{negl}(\lambda),$$

where probability is over the choice of b.

We explain this by designing a game in the standard model.

Game 1: Let RSE be a Searchable Encryption scheme (SE). Suppose that there are at most M keywords $\text{W} = (W_1, W_2, \dots, W_M)$ and N documents $\text{D} = (D_1, D_2, \dots, D_N)$, where $N, M \in \mathbb{N}$ (set of natural numbers) associated with an index table. The game is played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The game is divided into three phases as follows:

Phase 1: The adversary $\mathcal{A}$ sends a keyword to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ returns a trapdoor to $\mathcal{A}$. This continues between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ for a while.

Challenge Phase: The adversary $\mathcal{A}$ selects two keywords $W_1', W_2' \in \text{W}$ and send them to the challenger $\mathcal{C}$. The selection of the keywords can be done as follows:

a) $\mathcal{A}$ intends to search for unique keywords such that $W_1' \neq W_2'$;

The challenger $\mathcal{C}$ in response tosses a fair coin $b \leftarrow \{0,1\}$ and generates two trapdoors corresponding to the values of b i.e. $T'_{W_b'}$ such that $T'_{W_1'} \neq T'_{W_2'}$.

After the challenge has been completed, Phase 1 is run again. We allow the adversary to search for the same keywords again if interested.

Outcome Phase: $\mathcal{A}$ is given the generated trapdoors $T'_{W_1'}, T'_{W_2'}$. $\mathcal{A}$ will now have to guess and output $b' \in \{0,1\}$ and if $b = b'$ then the adversary wins. In other words, the adversary $\mathcal{A}$ has to output trapdoor T corresponding to W_1', W_2' to the challenger $\mathcal{C}$ in polynomial time. If the adversary $\mathcal{A}$ correctly guesses the trapdoor corresponding to the keyword then it has won, otherwise RSE provides keyword-trapdoor indistinguishability and the challenger $\mathcal{C}$ wins.

Therefore, the probability that the adversary $\mathcal{A}$ wins is $\frac{1}{2}$ which is according to the definition stated above.

4) Trapdoor-Index Indistinguishability for Ranked Searchable Encryption

Trapdoor-Index Indistinguishability relates to the complexity offered by a Searchable Encryption (SE) scheme. The keyword, 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.

The challenger begins by generating an index table against a data collection $\mathcal{D}$. The challenger sends the set of keywords $\mathcal{W}$, the trapdoors generated for all the keywords $\mathcal{W}$, along with the associated index table entries $I[0][W]$ 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.

Definition 4 (D_4)(Trapdoor-Index Indistinguishability). *Let* RSE = (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) *be a Ranked Searchable Encryption scheme over a dictionary* $\mathcal{W}$, λ *be the security parameter,* $\mathcal{D}$ *be the set of documents and* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *be such that* $M \in \mathbb{N}$ *and consider the following probabilistic experiment* $\mathbf{Trap\_Index}_{\text{RSE},\mathcal{A}}(\lambda)$**:**

$\mathbf{Trap\_Index}_{\text{RSE},\mathcal{A}}(\lambda)$
$(K, k_s, p) \leftarrow \text{KeyGen}(1^\lambda)$
$(I) \leftarrow \text{Build\_Index}(K, D)$
$for\ 1 \leq i \leq M$
$let\ I' = I[0][i]$
$let\ w = (w_1, \dots, w_i)$
$T_{w_i} \leftarrow \text{Build\_Trap}(K, k_s, w_i, \text{num})$
$b \xleftarrow{\$} \{0,1\}$
$(st_{\mathcal{A}}, w_0, w_1) \leftarrow \mathcal{A}_0(st_{\mathcal{A}}, 1^\lambda, w_M, I', T_{w_M})$
$T_{w_b} \leftarrow \text{Build\_Trap}(K, k_s, w_b, \text{num})$
$b' \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_{w_b})$
$if\ b' = b, output\ 1$
otherwise $output\ 0$

where $st_{\mathcal{A}}$ is a string that represents and captures $\mathcal{A}'$s state. We say that the trapdoor-index indistinguishability holds if for all polynomial-size adversaries $(\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_{N+1})$ such that $N = \text{poly}(\lambda)$,

$$\Pr[\mathbf{Trap\_Index}_{\text{RSE},\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \text{negl}(\lambda),$$

where probability is over the choice of b.

We explain this by designing a game in the standard model.

Game 2: Let RSE be a Searchable Encryption (SE) scheme. Suppose that there are at most M keywords $\text{W} = (W_1, W_2, \dots, W_M)$ and N documents $\text{D} = (D_1, D_2, \dots, D_N)$, where $N, M \in \mathbb{N}$ (set of natural numbers) associated with an index table. The game is played between an adversary $\mathcal{A}$ and a Challenger $\mathcal{C}$. The game is divided into three phases as follows:

Phase 1: The challenger $\mathcal{C}$ generates an index table I corresponding to the set of documents. The challenger $\mathcal{C}$ generates and sends the trapdoors for all keywords W , the index table entries corresponding to the trapdoor and the keywords to the adversary $\mathcal{A}$.

Challenge Phase: The adversary $\mathcal{A}$ is allowed to select two keywords $W_1', W_2' \in \text{W}$ and send them to the challenger $\mathcal{C}$. The selection of the keywords can be done as follows:

a) $\mathcal{A}$ intends to search for unique keywords such that $W_1' \neq W_2'$;

The challenger $\mathcal{C}$ in response tosses a fair coin $b \leftarrow \{0,1\}$ and generates two trapdoors corresponding to the values of b i.e. $T'_{W_b'}$ such that $T'_{W_1'} \neq T'_{W_2'}$.

After the challenge has been completed, the adversary $\mathcal{A}$ is given access to the previously generated history that was sent in Phase 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

Outcome Phase: $\mathcal{A}$ is given the generated Trapdoors $T'_{W'_1}, T'_{W'_2}$. The adversary $\mathcal{A}$ will now have to guess and return the index entry corresponding to the Trapdoors $T'_{W'_1}, T'_{W'_2}$ and W'_1, W'_2 in polynomial time. The adversary $\mathcal{A}$ wins if the guess is correct, otherwise RSE provides trapdoor-index table indistinguishability, and the challenger $\mathcal{C}$ wins.

Therefore, the probability that the adversary $\mathcal{A}$ wins is $\frac{1}{2}$ which is in line with the above stated definition.

Corollary 1*: Keyword-Trapdoor Indistinguishability and Trapdoor-Index table 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 theorem.

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

Firstly, we assume that if there exists a polynomial-size adversary $\mathcal{A}$ that succeeds in an experiment $\mathbf{Key\_Trap}_{\mathrm{RSE},\mathcal{A}}(\lambda)$ with non-negligible probability over $1/2$, then there exists a polynomial size adversary $\mathcal{B}$ and a polynomial size distinguisher $\mathcal{D}$ that distinguishes between the output of the experiment $\mathbf{Trap\_Index}_{\mathrm{RSE},\mathcal{A}}(\lambda)$ with non-negligible probability over $1/2$.

Let adversary $\mathcal{B}$ sample $b \xleftarrow{\$} \{0,1\}$; computes $(st_{\mathcal{A}}, w_0, w_1) \leftarrow \mathcal{A}_0(1^{\lambda})$. The distinguisher $\mathcal{D}$ is given access to a history consisting of trapdoors and corresponding keywords. The adversary proceeds as follows:

1. It parses $(st_{\mathcal{A}}, w_i) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, T_{w_2}, \dots, T_{w_{i-1}})$ where $2 \le i \le N$; $N \in \mathbb{N}$
2. It computes $b' \leftarrow \mathcal{A}_{i+1}(st_{\mathcal{A}}, T_{w_b})$
3. It outputs 1 if $b' = b$, and 0 otherwise.

Since $\mathcal{A}_{i+1}$ is a polynomial size adversary, hence, $\mathcal{B}$ and $\mathcal{D}$ are also polynomial size adversaries. Now, we have to guess the probability of $\mathcal{D}$'s success. $\mathcal{D}$ will output 1 if and only if $\mathcal{A}_{i+1}(st_{\mathcal{A}}, T_{w_b})$ 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, hence the outcome is independent of b. Therefore, $\mathcal{A}_{i+1}$ will guess b with probability of at most $1/2$, which is according to the Definitions D_3. Therefore, our initial assumption of such an adversary who can succeed in the experiment $\mathbf{Key\_Trap}_{\mathrm{RSE},\mathcal{A}}(\lambda)$ with a non-negligible probability over $1/2$ is wrong. Hence the distinguisher $\mathcal{D}$ that distinguishes between the output of the experiment $\mathbf{Trap\_Index}_{\mathrm{RSE},\mathcal{A}}(\lambda)$ with non-negligible probability over $1/2$ does not exist, in accordance with our Definition D_4. 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 D_4, 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, resulting in privacy preservation.

V. Proposed Ranked Searchable Encryption (RSE) Framework

As discussed in Section III, our RSE scheme comprises of five phases. We now present and discuss each of the phases. (Table I shows the notations/abbreviations used in our scheme).

TABLE I
NOTATIONS AND ABBREVIATIONS

CS – Represents a Cloud Server.
D – Denotes a set of all possible documents to be outsourced to the cloud. That is $D = \{D_1, D_2, \dots, D_N\}$.
W – Denotes a set of unique Keywords extracted from D_N such that $W = \{W_1, W_2, \dots, W_M\}$.
$\lvert W \rvert$ – Denotes total number of identified distinct keywords.
$\lvert D \rvert$ – Denotes the size of a particular document, obtained by counting the words in the document D.
RF – Denotes the relevance frequencies of the keywords W among the documents D.
$Mask(RF)$ – Denotes the masked RF.
p – Denotes a prime number of size λ (security parameter) +1.
$id(D_N)$ – Denotes the set of unique identifiers for each D_N.
I – Denotes the secure inverted Index table stored on CS and provides ranked keyword searching.
T_{W_i} – Represents the unique trapdoors generated to identify documents D containing word W_i.
Enc – Denotes a probabilistic encryption algorithm such as AES-CBC.
Dec – Denotes the decryption algorithm corresponding to Enc.
$x \leftarrow A$ – Represents that x contains the content of the variable A.
$H(.)$ – Represents a keyed one-way hash function.
K – Represents the master key.
k_s – Represents the session key.
CSPRNG – Cryptographically Secure Pseudo Random Number Generator.

A. 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, $K \epsilon \{0,1\}^{\lambda}$, a session key k_s; where, $k_s \epsilon \{0,1\}^{\lambda}$ and a prime number p. 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.

Phase 1: KeyGen

a) Input: A security parameter λ.
b) KeyGen: Generate keys $K, k_s \leftarrow \{0,1\}^{\lambda}$, $p \leftarrow CSPRNG(1^{\lambda})$
c) Output: Master key K and session key k_s

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

B. Build_Index Phase

The client generates an index table I represented by a dynamic 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 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 W among the documents D. The relevance frequencies are calculated according to equation (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 the 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.

Phase 2: Build_Index

a) Input: A set of documents D and a master key K, a Hash function $H(.)$.
b) Initialization:
   - Initialize dynamic 2D Array A.
   - Scan D and build W, a set of unique and distinct keywords occurring in D.
   - Initialize Prime number p of the size $\lambda + 1$ bits.
c) Build Index I:
   - for $1 \leq t \leq |W_i|$:
     - let $a \leftarrow H_K(W_i)$
     - 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_i occurring in D_N using equation (1) and store the value at the respective locations within A;
   - $Mask(RF)$:
     for $1 \leq m \leq number\ of\ columns\ in\ A$
     - for $1 \leq n \leq number\ of\ rows\ in\ A$
       - $A[n+1][m+1] = A[n+1][m+1] * random\ values$
d) Output: Index table I

C. 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(.) 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.

Phase 3: Build_Trap

a) Input: The master key (K), the session key (k_s), a keyword (w_i), a Hash functions $H(.)$, desired number of documents (num).
b) Trapdoor Generation:
   - let $a \leftarrow (H_K(W_i))$
   - let $b \leftarrow Enc_K(W_i)$
   - let $c \leftarrow a * b$
   - let $d \leftarrow H_{k_s}(b)$.
   - Set Trapdoor $T_{W_i} \leftarrow (d, c, num)$.
c) Output: Transmit T_{W_i} to CS.

D. Search_Outcome Phase

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

Phase 4: Search_Outcome

a) Input: A trapdoor T_{W_i} transmitted by the client, a session key k_s, a Hash functions $H(.)$ (the same as for the Build_trap phase) and the index table I.
b) Initialization:
   - Dynamic Array X.
c) Searching:
   - for $1 \leq l \leq size\ of\ I$:
     - if $\left(d == H_{k_s}(c * a^{-1})\right)$:
       - for $1 \leq m \leq num$:
         - find highest RF, return $Enc_K(id(D_i))$;
     - $X[\,] \leftarrow Enc_K(id(D_i))$;
d) Output: X; the set of encrypted document identifiers stored in ranked order.

E. Dec Phase

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

Phase 5: Dec

a) Input: The master key (K), A set X of encrypted document identifiers stored in ranked order
b) Decryption:
   - for $1 \leq o \leq size\ of\ X$:
     - $Dec_K(X[o])$;
c) Output: Documents identifiers $id(D_i)$

Remark 1: The index table I needs to be regenerated whenever the database is modified, but this can be avoided if we remove

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

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).

F. Analysis of the proposed RSE scheme

We now prove that our proposed RSE scheme provides correctness and soundness (defined in Section III).

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

- Given $T_w = \text{Build\_Trap}(K, k_s, w, num)$, the following equality holds with a probability 1:

$$T_w = \begin{Bmatrix} H_{k_s}\big(\big(Enc_K(w)\big) * \big(H_K(w)\big)\big), \\ \big(\big(Enc_K(w)\big) * \big(H_K(w)\big)\big), \\ num \end{Bmatrix}$$

- Given $T_w = \text{Build\_Trap}(K, k_s, w', num)$, and $w' \neq w$, the following inequality holds with an overwhelming probability:

$$T_w \neq \begin{Bmatrix} H_{k_s}\big(\big(Enc_K(w')\big) * \big(H_K(w')\big)\big), \\ \big(\big(Enc_K(w')\big) * \big(H_K(w')\big)\big), \\ num \end{Bmatrix}$$

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 file identifiers $Enc_K\big(id(D)\big)$ for every document that maps to the keywords, therefore, as a result, the value of the Search_Outcome phase corresponds to the value outlined in the correctness and soundness definitions mentioned in Section III. Therefore, the proposed RSE scheme is correct and sound.

G. Security Analysis

All of the previously known searchable encryption constructions leak some information because they were based on a deterministic trapdoor [13][23]. In [24] authors have studied the access pattern disclosure of the previously known searchable encryption schemes that were based on deterministic trapdoors. Our proposed scheme is based on a probabilistic trapdoor. So before mapping our scheme against the security definitions stated in Section IV, we would like to formally highlight any information that our scheme leaks. We analyze any possible leakage of information significant or insignificant, encrypted or unencrypted based on a set of assumptions. We analyze all the 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 $\mathcal{A}$ 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:

1) Leakage L_1
   Description: The leakage L_1 is associated to the index table I.
   Assumption: We assume that I is revealed to all the stakeholders i.e. the client, the Cloud Server and the adversary $\mathcal{A}$.
   Definition: The Leakage L_1 is defined as:

$$L_1(I) = \begin{Bmatrix} \Big(\big(H_K(W_i)\big)\Big)^{-1}, Enc_K\big(id(D_i)\big), \\ Mask(RF), \big(H_K(W_i)\big) \end{Bmatrix}$$

2) Leakage L_2
   Description: The leakage L_2 is associated to the Trapdoor T_w generated for a particular keyword w to be searched.
   Assumption: We assume that T_w is generated by the client and revealed to all the stakeholders i.e. the Cloud Server and the adversary $\mathcal{A}$.
   Definition: The Leakage L_2 is defined as:

$$L_2(T_w) = \begin{Bmatrix} a \leftarrow H_K(W_i) * Enc_K(W_i), \\ b \leftarrow H_{k_s}\big(Enc_K(W_i)\big), \\ num \end{Bmatrix}$$

3) Leakage L_3
   Description: The leakage L_3 is associated to search outcome (SO) of the Trapdoor generated for a particular keyword (T_w).
   Assumption: The search outcome is revealed to all the stakeholders i.e. the client, Cloud Server and the adversary $\mathcal{A}$.
   Definition: The Leakage L_3 is defined as:

$$L_3(\text{SO}) = \Big\{\text{OC}(w), Enc_K\big(id(D_i)\big)_{\forall T_w \in D_i}\Big\}$$

   where OC is the outcome.

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 consider it. In other words, suppose if an adversary is accidently given access to the trapdoor oracle then all the future searches are still secure. We explain this with the help of the leakage profile of the scheme presented in [23]. Their scheme gives away the Search pattern and Access pattern (*cf* Section 2 [23]). 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's scheme [23], our scheme also leaks the access pattern.

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 unknown 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

Therefore, it is evident that L_1 and L_3 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.

Lemma 1. *The Ranked Searchable Encryption Scheme (RSE) presented above is "secure" as it is* (L_1, L_2, L_3)*-secure and according to Definition* D_1, D_2, D_3, D_4*, where* L_1 *is associated with the index table* I *and leaks the encrypted file identifiers, masked relevance frequencies, and the inverse of the hash of the keyword. Whereas,* L_2 *leaks a,b and the number of required documents, and* L_3 *leaks the outcome of a trapdoor and the encrypted file identifiers.*

Proof Sketch. 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 V. The $KeyGen$ phase generates two keys $(K, k_s, p) \leftarrow KeyGen(\lambda)$. The $Setup$ phase generates an index table $(I) \leftarrow Setup(K, D_N)$ 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 the *Search_Outcome(*k_s, I, T_w*)* represents the outcome of the search. In order to prove that our scheme satisfies this lemma, we first prove that our scheme satisfies the security definitions D_1, D_2, D_3, D_4. 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 twice. 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 of D_1, D_2, D_3, D_4.

Now we need to prove the security of our scheme against the leakages L_1, L_2 and L_3. We argue that the leakages L_1, L_2 and L_3 are meaningless and do not affect our scheme. 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_1, L_2, L_3)-secure against adaptive/non-adaptive indistinguishability attacks and provides Keyword-Trapdoor Indistinguishability & Trapdoor-Index table indistinguishability.

Lemma 2. *The Ranked Searchable Encryption Scheme (RSE) presented above is "Privacy Preserving" as it is* (L_1, L_2, L_3)*-secure and according to Definition* D_1, D_2, D_3, D_4*, where* L_1 *is associated with the index table* I *and leaks the encrypted file identifiers, masked relevance frequencies, and the inverse of the hash of the keyword. Whereas,* L_2 *leaks a,b and the number of required documents and* L_3 *leaks the outcome of a trapdoor and the encrypted file identifiers.*

Proof Sketch. We extend the proof of Lemma 1 to establish proof of this lemma. We have already proved that our scheme is (L_1, L_2, L_3)-secure since the trapdoor of the proposed scheme is generated based on probabilistic encryption therefore our scheme satisfies the definitions D_3, D_4. Since the trapdoor T is indistinguishable over the keyword W and the index table I, therefore there is an equal probability that the generated trapdoor T may be generated for any keyword W_i, and may be mapped to any index table entry. Therefore, the outcome (prior to the search) will be completely indistinguishable. Hence the proposed RSE scheme is privacy preserving.

VI. Performance Evaluation

A. Algorithmic Analysis

The algorithmic analysis presented here is based on the complexity analysis of the target schemes. We analyze the algorithm of each scheme and perform the complexity analysis. This analysis is based on upper bound analysis of the set of keywords (W) and set of documents (D). In the asymptotic analysis, the complexities of a set of keywords (W) is denoted by m, whereas the complexity of the set of documents (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 perform the analysis of our scheme while considering ranking and no-ranking. This way the readers can easily relate and evaluate the efficiency of our scheme compared to other schemes under discussion.

From the complexity analysis of our scheme, it is evident that the Build_Index phase requires $\Theta(mn + n)$ where m represents the total number of keywords and n is the total number of documents in the dataset. The Build_Trap phase is bound by $\Theta(h + e)$, where h represents the complexity of computing a Hash and e represents the complexity of computing encryption. The Search_Outcome phase is bound by $\Theta(mn)$. 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 $\Theta(n)$. Whereas, the efficiency of the Search_Outcome phase can be increased to $\Theta(n + 1)$.

Table 2 shows the algorithmic comparative analysis of the schemes. From the table, it is evident that our scheme is efficient as compared to the other schemes.

TABLE II
Algorithmic Analysis

Scheme	Complexity				
	Build_Index Phase		Build_Trap Phase	Search_Outcome Phase	
ERSE[9]	$\Theta(mn + 3n)$		$\Theta(2hm)$	$\Theta(mn)$	
DSE[13]	$\Theta(mn + mh)$		$\Theta(m)$	$\Theta(3n)$	
GRSE[14]	$\Theta(mn + n)$		$\Theta(m)$	$\Theta(mn)$	
MPSE[15]	$\Theta(mn + n)$		$\Theta(m)$	$\Theta(mn)$	
Our work	$\Theta(mn + n)$	$\Theta(mn)$	$\Theta(2h + e)$	$\Theta(mn)$	$\Theta(n + 1)$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

We graphically represent the complexities of the schemes by analyzing their phases separately. In order to plot the graphs onto a two-dimensional plane we propose that the number of keywords are equal to the number of documents *i.e.* $n = m$. We also suppose that the Hashing and the encryption takes a unit of time. Our work is represented by (I) and (II), where (I) is for ranked searching and (II) is for unranked searching. We do the complexity analysis of our scheme by comparing it with the ranked and unranked schemes separately.

Figure 2 shows the Build_Index phase of the ranked schemes. It can be seen that the complexity of our proposed RSE scheme and existing ranked scheme increases with the increase in the number of documents. Even though our protocol also shows an exponential growth, it is more efficient and outperforms the other existing scheme.

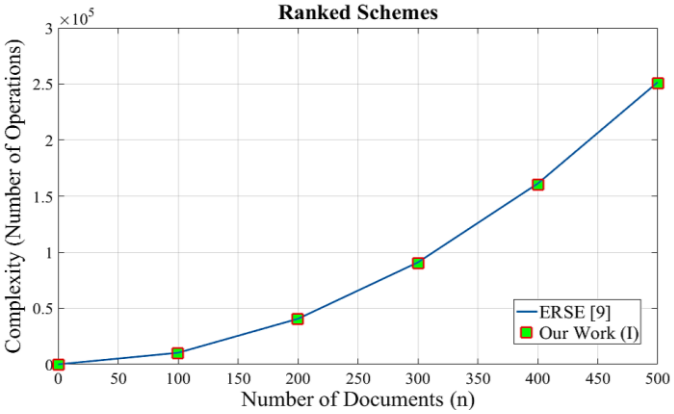


Fig. 2. Complexity analysis of Build_Index phase in ranked schemes.

On considering our scheme without ranking, it exhibits an exponential growth but is more efficient as compared to existing schemes. Figure 3, graphically represents the complexity of the unranked schemes. From the graph, it is evident that with the increase in the number of documents, there is an increase in the required number of operations.

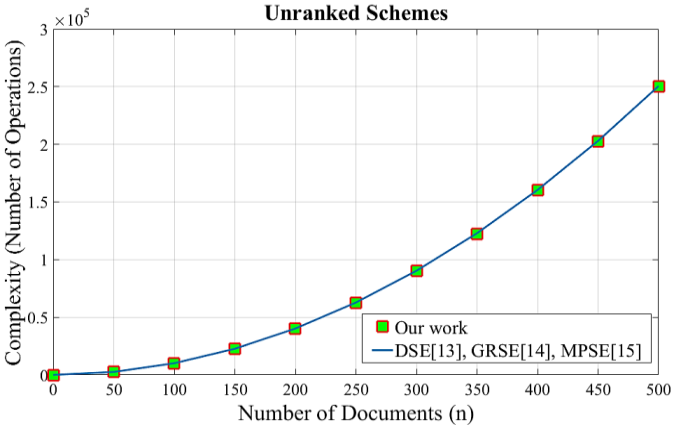


Fig. 3. Complexity analysis of Build_Index phase in unranked schemes.

The complexity of the Build_Trap phase is not effected by ranking or un-ranking. Therefore, Figure 4 represents a collective graph of the Build_Trap phase of ranked and unranked schemes. It can be seen that all the schemes show a linear growth but our proposed scheme outperforms other schemes in terms of complexity by maintaining the same efficiency even with the increase in the number of keywords being searched.

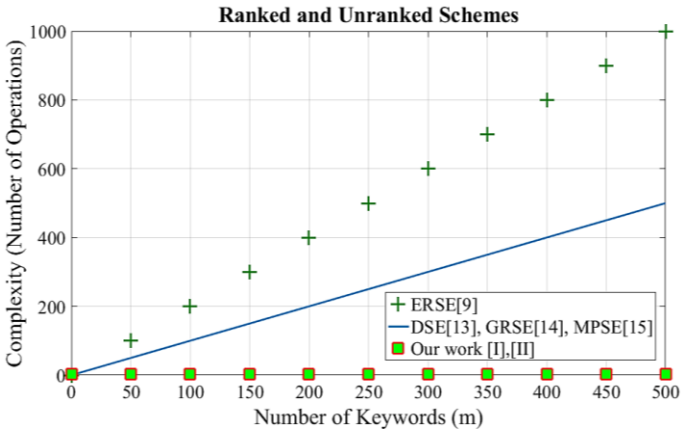


Fig. 4. Complexity analysis of Build_Trap phase.

Figure 5 illustrates a graph generated for the Search_Outcome phase of the RSE schemes. It can be seen that our proposed scheme and the existing ranked schemes are showing an exponential growth by depicting the same complexity.

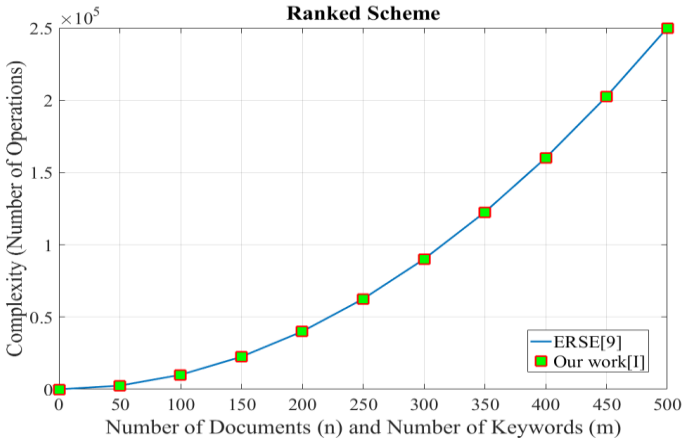


Fig. 5. Complexity analysis of Search_Outcome phase in ranked schemes.

When we compare our unranked scheme with the similar existing unranked schemes then our scheme performs much better and faster. Our unranked scheme shows a linear growth in terms of the complexity. Figure 6 shows a complexity analysis of the Search_Outcome phase of the unranked schemes.

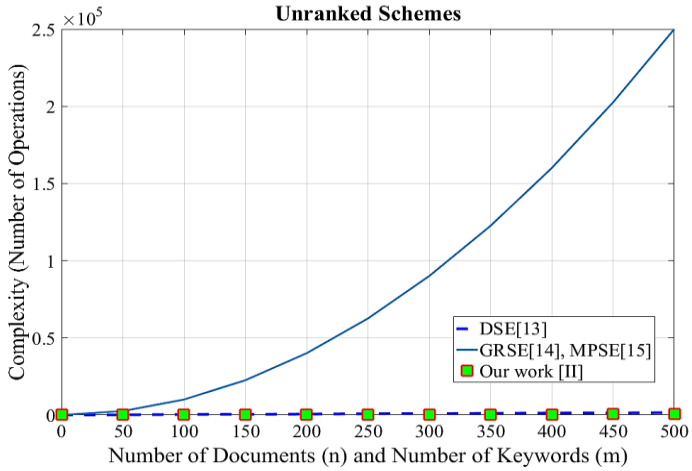


Fig. 6. Complexity analysis of Search_Outcome phase in unranked schemes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

The overall complexity analysis of our scheme against existing schemes yields that our scheme dominates the existing schemes in terms of efficiency and can be termed a lightweight scheme.

B. Computational Analysis

a) Dataset Description

The Switchboard-1 Telephone Speech Corpus (LDC97S62)[25] 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 [8] 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 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 searchable encryption presented in this paper.

b) Implementation Details

To demonstrate the feasibility of our RSE scheme, we have implemented our algorithms in Java and present the results in the form of graphs using MATLAB2016. 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 the client and server side implementation is done on the same machine. Hence, the analysis does not take the cost incurred while transferring the documents, index tables or trapdoor over the network, to the CS, into account.

The implementation uses all the algorithms presented in Section V. 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. The workstation used for the demonstration runs on an Intel Core i5 CPU running at 3.00GHz with 8GB of RAM.

c) Computation Overhead

To determine the performance of our RSE scheme, we analyze the performance of each individual phases that have been discussed throughout the paper. 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.

1) 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.

The index table is generated by the client and transmitted to the server. Our scheme facilitates both ranked and un-ranked searches depending upon the required functionality and area of application. As mentioned in Section V, our scheme provides ranking that comes with an increase in the number of computation needed, resulting in an increase in the computational time. Therefore, we execute ranked and un-ranked index generation seperately.

Figure 7 shows a graphical representation for the computational time of 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 depicting a linear growth.

The dotted line represents the computational time for the unranked index generation. For 10,000 documents the index generation takes only 0.43 minutes that gradually increases to 5.75 minutes over 100,000 documents.

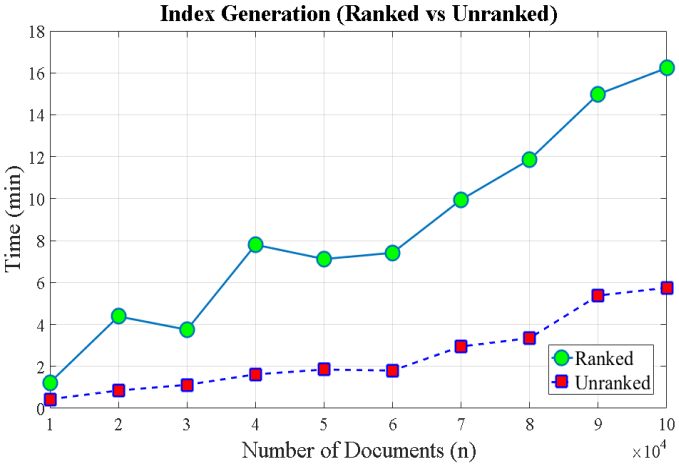


Fig. 7. Computational time for Index Generation.

2) 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 effected by the ranked or unranked searching so the computational time remains the same. The Build_Trap phase is executed for the keyword “about” and the trapdoor generation takes a constant time of a mere 0.016 seconds.

3) Search_Outcome Phase

Once the encrypted documents along with the index table are uploaded onto the CS and the trapdoor has been generated and transmitted to the CS, the next step is the searching of the relevant documents. Figure 8, represents the graph generated on executing the Search_Outcome phase against the trapdoor generated for the keyword “about”. The searching takes a total of mere 3.42 seconds against 100,000 documents and shows a 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2737789, IEEE Transactions on Emerging Topics in Computing

For example, out of the total 100,000 documents in the dataset, 98,144 documents contain the keyword "about".

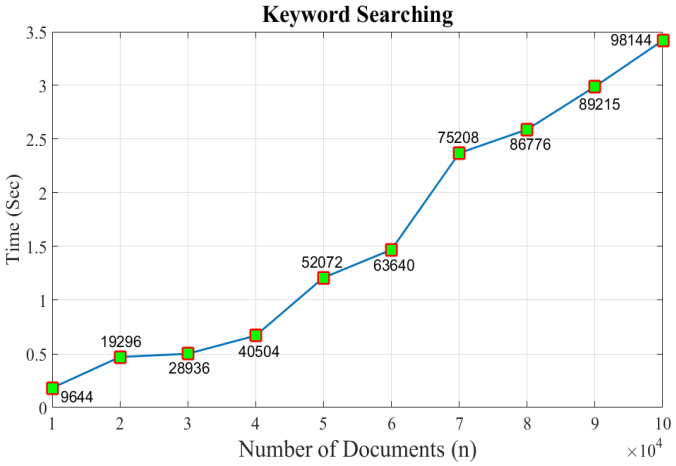


Fig. 8. Computational time for searching for the keyword "about".

d) *Storage Overhead*

We now study the storage overhead for our scheme. We evaluate the storage overhead in two parts: the client and the Cloud Server (CS).

As discussed in Section V, 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 $(2 * 128 + 161)/8$ bytes. Which means 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 $n * D_{avg}$, 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 $8(mn) + n * D_{avg}$. In order to compare our storage overhead with similar existing schemes, we refer readers to [26] that discusses the storage overhead of existing ranked SE schemes. It may be observed that our scheme also outperforms existing schemes in terms of storage overhead.

VII. Conclusion and Future Work

In this paper, we have readdressed the problem of supporting keyword search on encrypted data outsourced to the cloud. We make several contributions to this domain by presenting a novel ranked based searchable encryption scheme. Our construction exploits the properties of modulo prime to generate a probabilistic trapdoor. The greatest challenge in searchable encryption is to maintain a balance between security, efficiency and query expressiveness.

In order to perform the security analysis of our scheme, we revisit the existing definitions for searchable encryption and introduce the concept of indistinguishability. We prove the security of our scheme by giving formal proofs to the new definitions and designing games in the standard model. From the security analysis of our construction, it is demonstrated that the scheme provides greater security under these proposed definitions as compared to previous schemes. In order to prove the efficiency of our scheme, we perform an asymptotic analysis of existing schemes against our approach. The results yield that our scheme is lightweight and outperforms other existing approaches.

We designed and implemented a proof of concept prototype and successfully tested our scheme with real data. The analysis of the result yields that our scheme shows a linear growth with the increase in the input. Based on the results we can term our scheme to be extremely lightweight.

In our future work, we will extend our proposed scheme to support multi-keyword searching to further support query expressiveness and deploy it to a multi-client architecture.

Acknowledgements

We thank Intelligent Voice UK for providing the datasets to carry out the evaluation. We appreciate their useful comments and suggestions during the implementation of this work.

References

[1] J. Weis and J. Alves-Foss, "Securing Database as a Service: Issues and Compromises," *IEEE Secur. Priv.*, vol. 9, no. 6, pp. 49–55, 2011.

[2] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A Survey of Provably Secure Searchable Encryption," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 1–51, 2014.

[3] Research Directorate Staff, "Securing the cloud with homomorphic encryption," *Next Wave*, vol. 20, no. 3, pp. 1–4, 2014.

[4] B. T. Prasanna and C. B. Akki, "A Comparative Study of Homomorphic and Searchable Encryption Schemes for Cloud Computing," *Int. J. Adv. Stud. Comput. Sci. Eng. IJASCSE*, vol. 4, no. 5, 2015.

[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," *Eurocrypt*, pp. 506–522, 2004.

[6] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient dna searching through oblivious automata," in *Proceedings of the 14th ACM conference on Computer and communications security - CCS '07*, 2007, p. 519.

[7] P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker, "Computationally efficient searchable symmetric encryption," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6358 LNCS, pp. 87–100, 2010.

[8] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: telephone speech corpus for research and development," in *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992, pp. 517–520 vol.1.

[9] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, 2012.

[10] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," *Proc. - Int. Conf. Distrib. Comput. Syst.*, pp. 253–262, 2010.

[11] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption," *Proc. 13th ACM Conf. Comput. Commun. Secur. - CCS '06*, p. 79, 2006.

[12] K. Li, W. Zhang, C. Yang, and N. Yu, "Security Analysis on One-to-Many Order Preserving Encryption-Based Cloud Data Search," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 9, pp. 1918–1926, 2015.

[13] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," *2012 ACM Conf. Comput. Commun. Secur.*, pp. 965–976, 2012.

[14] B. Wang, S. Member, M. Li, and H. Wang, "Geometric Range Search on Encrypted Spatial Data," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 4, pp. 704–719, 2016.

[15] Q. Tang, "Nothing is for free: Security in searching shared and encrypted data," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 11, pp.

1943–1952, 2014.
[16] R. Popa and N. Zeldovich, "Multi-key searchable encryption," pp. 1–18, 2013.
[17] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes. Compressing and Indexing Documents and Images*. 1999.
[18] R. Rivest and A. Sherman, "Randomized Encryption Techniques," in *CRYPTO 82, Plenum Press*, 1983, pp. 1–20.
[19] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, 2011.
[20] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," *Appl. Cryptogr. Netw. Secur. Proc.*, vol. 3531, pp. 442–455, 2005.
[21] D. Wagner and A. Perrig, "Practical techniques for searches on encrypted data," *Proceeding 2000 IEEE Symp. Secur. Privacy. S&P 2000*, pp. 44–55, 2000.
[22] E.-J. Goh, "Secure Indexes," *An early version this Pap. first Appear. Cryptol. ePrint Arch. Oct. 7th*, pp. 1–18, 2003.
[23] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7859 LNCS, pp. 258–274, 2013.
[24] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," *Ndss*, vol. 20, pp. 1–15, 2012.
[25] J. Godfrey and E. Holliman, "Switchboard-1 Release 2 - Linguistic Data Consortium." [Online]. Available: https://catalog.ldc.upenn.edu/LDC97S62. [Accessed: 31-Aug-2016].
[26] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling Fine-Grained Multi-Keyword Search Supporting Classified Sub-Dictionaries over Encrypted Cloud Data," *IEEE Trans. Dependable Secur. Comput.*, vol. 13, no. 3, pp. 312–325, 2016.

S. Tahir received his B.E. degree in software engineering from Bahria University, Islamabad, Pakistan, in 2013. In 2015, he received his MS degree in information security from the National University of Sciences and Technology (NUST), Islamabad, Pakistan. He is currently pursuing a Ph.D. degree in information engineering at City, University of London, UK.

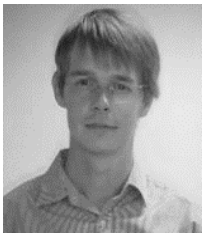
From June, 2015 to present, he is a Lecturer in the Department of Information Security, NUST, Islamabad, Pakistan and has been awarded a scholarship by NUST for pursuing his Ph.D. at City, University of London. He is currently also a Research Assistant at City, University of London. His research interests include applied cryptography and cloud security.

S. Ruj received her B.E. degree in computer science from Bengal Engineering and Science University, Shibpur, India and Masters and Ph.D. in computer science from the Indian Statistical Institute. She was an Erasmus Mundus Post-Doctoral Fellow at Lund University, Sweden and Post-Doctoral Fellow at University of Ottawa, Canada. She is currently an Assistant Professor at the Indian Statistical Institute, Indore, India. Prior to this, she was an Assistant Professor at IIT, Indore. She was a visiting researcher at INRIA, France, University of Wollongong, Australia, Kyushu University, Japan. KDDI labs, Japan and Microsoft Research Labs, India.

Her research interests are in applied cryptography, security, combinatorics and complex network analysis. She works actively in mobile ad hoc networks, vehicular networks, cloud security, security in smart grids. She has served as program co-chair of IEEE ICCC (P&S Track), IEEE ICDCS, IEEE ICC, etc and served on many TPCs. She won best papers awards at ISPA'07 and IEEE PIMRC'11. Sushmita is a Senior Member of the IEEE.

Y. Rahulamathavan received a B.Sc. degree (first-class honors) in electronic and telecommunication engineering from the University of Moratuwa, Sri Lanka, in 2008, and a Ph.D. degree in signal processing from Loughborough University, the UK in 2011. From April 2008 to September 2008, he was an Engineer at Sri Lanka Telecom, Sri Lanka and from November 2011 to March 2012, he was a Research Assistant with the Advanced Signal Processing Group, School of Electronic, Electrical and Systems Engineering, Loughborough University, UK. He has worked as a Research Fellow with the Information Security Group, School of Engineering and Mathematical Sciences, City University London, UK. Moreover, Dr. Rahulamathavan received a scholarship from Loughborough University to pursue his Ph.D. degree. He is currently working as a Faculty member with Loughborough University, UK. His research interests include signal processing, machine learning and information security and privacy. http://www.drrahul.uk/

M. Rajarajan is Professor of Security Engineering at the City, University of London, UK. He obtained his Ph.D. from City University London in 2001. His research expertise is in the areas of mobile security, intrusion detection and privacy techniques. He has chaired several international conferences in the area of information security and is involved in the editorial boards of several security and network journals. He is also a visiting fellow at British Telecommunications (BT) UK and is currently actively engaged in the UK Governments Identity Assurance programme (Verify UK). He is a Senior Member of the IEEE, a Member of ACM and Advisory board member of the Institute of Information Security Professionals UK.



C. Glackin graduated from the University of Ulster, School of Computing & Intelligent Systems with an MSc in Computing & Intelligent Systems in 2004. Cornelius completed PhD concerning Spiking Neural Network research at the University of Ulster in 2009. After six years post-doctoral research experience working at the University of Ulster and the University of Hertfordshire, he then moved to industry.

Cornelius is currently employed as a Research Scientist at Intelligent Voice Ltd working on machine learning approaches to signal processing, natural language processing and speech recognition. Cornelius' other research interests include: kernel machines, information theory, and robotics.