An Efficient Disjunctive Query enabled Ranked Searchable Encryption Scheme

Shahzaib Tahir; Muttukrishnan Rajarajan
School of Mathematics, Computer Science and Engineering
City, University of London
London, United Kingdom
{Shahzaib.Tahir, R.Muttukrishnan}@city.ac.uk

Sushmita Ruj
Indian Statistical Institute
203 BT Road
Kolkata, India
sush@isical.ac.in

***Abstract*—Cloud computing motivates data owners to economically outsource large amounts of data to the cloud. To preserve the privacy and confidentiality of the documents, the documents need to be encrypted prior to being outsourced to the cloud. In this paper, we propose a lightweight construction that facilitates ranked disjunctive keyword (multi-keyword) searchable encryption based on probabilistic trapdoors. The security analysis yields that the probabilistic trapdoors help resist distinguishability attacks. Through the computational complexity analysis we realize that our scheme outperforms similar existing schemes. We explore the use of searchable encryption in the telecom domain by implementing and deploying our proof of concept prototype onto the British Telecommunication's Public Cloud offering and testing it over a real corpus of audio transcriptions. The extensive experimentation thereafter validates our claim that our scheme is lightweight.**

***Index Terms*—Probabilistic Trapdoors; Indistinguishability; Privacy Preservation; Inverted Index.**

I. Introduction

Over the past decade the use of file hosting services (such as Amazon, Dropbox, etc.) has seen a trend due to which Cloud has emerged as Data-as-a-Service (DaaS) platform. There are many advantages of outsourcing data to the cloud including agility, availability and cost effectiveness but they lead to security and privacy concerns. Searchable Encryption (SE) has gained importance with the advent of DaaS that allows a client to encrypt the documents prior to outsourcing them to the cloud in such a way that the search queries can be generated by the client and the Cloud Server may be delegated to conduct the search on the client's behalf.

Three main challenges associated with SE as discussed in [1] are (a) security and privacy (b) efficiency and (c) query expressiveness. Security, privacy and efficiency are discussed in detail throughout this paper. Query expressiveness refers to 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 enabled SE scheme is more expressive as compared to an unranked SE scheme. In order to deal with the challenges associated to security and privacy we require a SE scheme.

In this paper, we propose a index-based, ranked multi-keyword searchable encryption (RMSE) scheme that is not only efficient but equally secure and privacy preserving. Ranked search is desirable for many applications where only the most relevant documents are needed, instead of all the documents which contain the keyword (search query). Ranked disjunctive multi-keyword search poses a more difficult challenge. Often search queries are not single word but a group of words. Our aim is to provide a solution for ranked multi-keywords search on encrypted data. We consider disjunctive queries, meaning that the search returns a document which contains at least one keywords in the query. However, the more relevant document is one which has all the keywords, instead of the one which has only one keyword. For example, if a client searches for "A brown fox", the documents containing all of the three keywords will be given priority over the document containing fewer number of these keywords.

Apart from generic applications of SE on Cloud, SE has been proposed in the healthcare domain [2] to perform search over the encrypted medical records. In [3], authors have explored the use of SE in performing secure search over the genomic signatures outsourced to the cloud server. Whereas, SE over email servers has been carried out in [4]. Apart from these domains SE could have a profound impact on IoT, e-commerce, business intelligence and data science.

A. Our Contribution

In this paper, we make the following contributions to the field of SE:

- We design and present a novel and lightweight ranked based multi-keyword searchable encryption (RMSE) scheme that supports disjunctive queries. The trapdoors are probabilistic that helps to maintain privacy of the search by resisting distinguishability attacks.
- We introduce stronger notions of security in searchable encryption and evaluate the security of our scheme.
- We implement and test our proof of concept prototype over an encrypted Telephone Speech corpus (real dataset) by deploying it to the British Telecommunication's Public Cloud offering.

B. Organization

Section II explains the proposed RMSE architecture, the threat model along with the associated security and privacy concerns. Section III, highlights the literature review and discusses the existing relevant schemes. Finally, in Section IV, we present our RMSE scheme. In Section V, we perform the security analysis. In Section VI, we present the computational complexity analysis of similar schemes. The same section, explains the implementation details and the computational time of our scheme. Conclusions are drawn towards the end.

II. A HIGH LEVEL VIEW OF OUR PROBLEM

We now describe the system architecture and present our RMSE scheme thereafter.

A. The System and Threat Model

The architecture of our ranked multi-keyword searchable encryption (RMSE) scheme comprises of three entities: Alice, Bob (client/ data owner), cloud server (CS). Bob converts all the conversation that he has with Alice to transcriptions. Hence Bob's corpus contains n documents $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$. Bob outsources the encrypted corpus to the CS. He wants to perform disjunctive multi-keyword search over the encrypted documents while preserving the privacy of the documents and the search. Bob identifies a set of unique keywords $\mathcal{W} = \{W_1, W_2, \ldots, W_m\}$ from the documents $\mathcal{D}$. There is a high probability that Bob's trapdoor (search query) may be disjunctively mapped to multiple documents, therefore, he wishes retrieve documents based on some ranking mechanism.

In [5], a formula (equation 1) has been presented that is commonly used for the relevance frequency generation by researchers [6][7].

$$RF(W,D) = \sum_{t=1}^{W} \frac{1}{|D|} \cdot (1 + \ln f_{(WD)}) \cdot \ln(1 + \frac{N}{f_W}) \quad (1)$$

where W denotes the keyword to be searched; D denotes the document; $|D|$ denotes length of the document obtained by counting the words appeared in the document D; $f_{(D,W)}$ denotes 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.

Bob generates a secure ranked index table I and outsources it to the CS along with the encrypted documents $\mathcal{D}$. The CS is assumed to be "trusted but curious", *i.e.* the CS does not attempt to modify or delete the encrypted documents, index table or trapdoor, but instead, the CS is curious to learn additional information about the outsourced documents or queries that are being sent by Bob to the CS. In order to search for multiple keywords, Bob using his private key generates a valid probabilistic trapdoor and sends it to the CS. The CS searches over the secure index table I on Bob's behalf and returns the encrypted document identifiers in the ranked order. From here on, Bob will be represented as a client, throughout the rest of the paper.

B. Privacy Concerns

Two threat models widely used in literature [8][9][10] explain the client's privacy concerns :

- *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, frequency of the encrypted keywords with which they appear within a document.

The confidentiality of the document's is achieved by encrypting them before outsourcing them to the CS. Therefore, we are more concerned about the privacy that can be violated by a SE scheme in itself. 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.
- *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 twice.

Section V presents the security definitions that take the aforementioned privacy concerns into account to provide a greater level of security. We now formally define our RMSE Scheme.

Definition (Ranked Multi-Keyword Searchable Encryption Scheme (RMSE)) Our RMSE comprises of five polynomial time algorithms Π = (KeyGen, Build_Index, Build_Trap, Search_Outcome, Dec) such that:

$(K, k_s) \leftarrow$ KeyGen (1^λ): is a probabilistic key generation algorithm run by the client. The algorithm takes a security parameter λ as the input and returns a master key K and a session key k_s.

$(I) \leftarrow$ Build_Index$(K, \mathcal{D})$: is a deterministic algorithm run by the client to generate a secure index table I. The algorithm as input takes a master key K and a collection of documents $\mathcal{D}$ to be outsourced to the CS. 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, as the input requires the master key K, a session key k_s, a set of disjunctive keywords W, 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 document identifiers encrypted $Enc_K(id(D_j))$ containing the set of keywords W in ranked order.

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

Correctness: A RMSE scheme is correct if for the security parameter λ, master key K and the session key k_s generated by KeyGen(1^λ), for (I) output by Build_Index$(K, \mathcal{D})$, the search using the trapdoor T_W generated for the set of keywords W, always returns the correct set of encrypted document identifiers $E_K(id(D_j))$ containing the disjuctive keywords W in ranked order. A RMSE scheme is correct if the following are true:

- If $W \in D_j$ then the following should hold with a non-negligible probability
$$\text{Search\_Outcome}(k_s, I, T_W) = \mathcal{D} \cap \text{Dec}(K, X) = D_j, \text{where } 1 \leq j \leq n$$
- If $W \notin D_j$then the following should hold with a non-negligible probability
$$\text{Search\_Outcome}(k_s, I, T_W) = \mathcal{D} \cap \text{Dec}(K, X) = 0$$

$Soundness$: A RMSE scheme is sound if for the security parameter λ, master key K and the session key k_s generated by KeyGen(1^λ), for (I) output by Build_Index$(K, \mathcal{D})$, the search using the trapdoor T_W generated for disjunctive set of keywords W, always returns sound results *i.e.* the result should not contain any false positives or false negatives.

A RSE scheme is sound if the following are true:

- If $W \in D_j$ then the following should hold with a non-negligible probability
$$\text{Search\_Outcome}(k_s, I, T_W) = 1$$
- If $W \notin D_j$ then the following should hold with a non-negligible probability
$$\text{Search\_Outcome}(k_s, I, T_W) = 0$$

III. LITERATURE REVIEW

In [7][6], authors for the first time introduce the concept of ranking in SE. The authors present two schemes for single keyword ranked search over encrypted text that are an extension of [11]. There is an advantage of their later scheme as it supports dynamic inverted index *i.e.* whenever a new file is uploaded to the CS the re-ranking of the entire index table is not required. Furthermore, the scheme helps to keep the ranking score encrypted that helps 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 and demonstrated that the scheme still leaks the relevance scores to the adversary resulting in distinguishability attacks.

Kamara *et al.* in [13] have proposed a dynamic searchable symmetric encryption scheme that is an extension of their previous work presented in [11]. Their scheme facilitates the addition, deletion or modification of documents on run time 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 get to modify the file. Though this can be termed as a breakthrough in the field of SE, there is a drawback of their scheme *i.e.* the generated trapdoor is deterministic and the same trapdoor is generated for the same word every time it is searched. Hence, their scheme cannot resist distinguishability attacks.

In [10], Cao *et al.* have proposed a ranked based multi-keyword SE scheme. The authors for the first time introduce the concept of "coordinate matching" that takes "as many matches as possible" into consideration. They use a vector for the trapdoor generation, where each element of the vector represents a particular keyword. To preserve the privacy of the documents, they add some dummy keywords to a specific document at random. This is not feasible as it increases the size of the index. For the ranking they compute the Euclidean Norm against every document vector, which is again not feasible. Although, they are achieving ranking but due to the dummy keywords the precision of the results decreases.

Li *et al.* in [9] have proposed a ranked multi-keyword SE scheme that is efficient as compared to the constructions proposed by Cao *et al.* in [10]. It is surprising that they use the term "conjunctive keywords" but perform the search just like disjunctive keywords. Hence, they loose the benefit of conjunctive keywords. They use a relevance score generation formula quite similar to the one that we use. They also use inverted index for the blind storage. The authors represent the trapdoor by a vector, where each element represents the presence/ occurrence of a keyword. They add dummy integers to the vector to increase the privacy of the trapdoor without measuring the effect it has on the precision of the results.

IV. PROPOSED RMSE SCHEME

As discussed in Section II, our RMSE scheme comprises of five polynomial time algorithms that are explained in this section. (Table I shows the notations used in our scheme).

TABLE I
NOTATIONS AND ABBREVIATIONS

CS –Represents a Cloud Server.
$\mathcal{D}$ –Denotes a set of all possible documents to be outsourced to the cloud. That is $\mathcal{D}$=$D_1, D_2, \ldots, D_n$.
$\mathcal{W}$ –Denotes a set of unique Keywords extracted from $\mathcal{D}$ such that $\mathcal{W} = W_1, W_2, \ldots, W_m$.
W –Denotes a set of unique disjunctive Keywords that are to be searched such that $W = W_1, W_2, \ldots, W_i$.
$\|\mathcal{W}\|$–Denotes total number of identified distinct keywords.
$\|D\|$–Denotes the size of a particular document, obtained by counting the words appeared in the document $\mathcal{D}$.
RF –Denotes the relevance frequencies of the keywords $\mathcal{W}$ among the documents $\mathcal{D}$.
$Mask(RF)$ –Denotes the masked RF.
P –Denotes a prime number of the size λ (security parameter) +1.
$id(D_i)$ –Denotes the set of unique identifiers for $\mathcal{D}$.
I –Denotes the secure inverted Index table stored on CS that provides ranked keyword searching.
T_W –Represents the unique trapdoors generated to identify documents D containing disjunctive keywords W.
Into_Integer –Represents the conversion of a value from Hexadecimal to positive Integer.
Enc –Denotes a probabilistic encryption algorithm such as AES-CBC.
Dec –Denotes the decryption algorithm corresponding to Enc.
$x \leftarrow$ –Denotes x contains the content of the variable A.
$H(\cdot)$ –Represents a keyed one-way hash function.
K –Represents the master key.
k_s –Represents the session key.

A. KeyGen Algorithm

The KeyGen algorithm is triggered by the client. The client provides the security parameter λ against which he receives the master key K; where, $K \in \{0,1\}^{\lambda}$. The master key K is kept secret. The client generates a session key k_s; where, $k_s \in \{0,1\}^{\lambda}$ and sends it to the CS.

Algorithm 1: *KeyGen*

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

B. Build_Index Algorithm

The client generates an index table I that is represented by a dynamic array A. The client uses a cryptographic Hash function

$$H : \{0,1\}^{\lambda} \times W \rightarrow \{0,1\}^{L}$$

where L is the length of the output. The keyed Hash function H uses the master key K to generate hash of the keywords and convert them to positive integers.

Algorithm 2: *Build_Index*

```
a) Input: A set of documents $\mathcal{D}$ and a master key $K$, a Hash functions $H(\cdot)$.
b) Initialization:
    • Initialize dynamic 2D Array $A$.
    • Scan $\mathcal{D}$ and build $\mathcal{W}$, a set of unique and distinct keywords occurring in $\mathcal{D}$.
    • Initialize Prime number $P$ of the size $\lambda + 1$ bits.
c) Build Index $I$:
    • for $1 \leq t \leq |\mathcal{W}|$:
      – let $a \leftarrow$ Into_Integer$(H_K(W_t))$ mod $P$
      – 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 $W_t$ occurring in $D_n$ using equation (1) and store the value at the respective location within $A$;
    • $Mask(RF)$ :
      – for $1 \leq m \leq |\mathcal{W}|$:
        ◦ for $1 \leq n \leq \mathcal{D}$:
        Choose $R$ in $Z_p$
        $A[n+1][m+1] = A[n+1][m+1] * R$
d) Output: Index table $I$
```

The array A of dimensions $\mathcal{D} \times \mathcal{W}$ holds three attributes. The first row of the array consists of values that are generated by calculating the inverse of the hash of keywords $\mathcal{W}$ after converting it into positive integer under mod P. The first column consists of the encrypted document identifiers $Enc_K(id(\mathcal{D}))$ of all the outsourced documents. Whereas, the remaining entries of the array are the relevance frequencies of the keywords $\mathcal{W}$ among the documents $\mathcal{D}$. 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 R within Z_p, represented by $Mask(RF)$. This way the relevance frequencies are masked while maintaining proportion between the relevance scores of the keywords W occurring in different documents. However, in case document does not contain a keyword, the particular entry within the table is left blank. Therefore, the server from the index table can only learn the presence or absence of a keyword but can never deduce the keyword or launch a statistical attack. This helps to prevent frequency analysis attack and disclosure of document size. For further enhancing the security in case of active adversary, one may also use Order Preserving Hashing for masking the RF.

C. Build_Trap Algorithm

The client generates a trapdoor to search for documents containing a set of disjunctive keywords. The client using a probabilistic symmetric encryption algorithm such as AES, encrypts the search string and converts the results into positive integer under mod P, represented by b in the algorithm. The client using the master key K generates the hash $H(\cdot)$ of the keyword set and converts it into positive integer under mod P, represented by a in the algorithm. Now c is computed by multiplying a with b under mod P. The client initializes a dynamic array B having dimensions $1 \times W$ and stores c in the respective locations corresponding to all the keywords present in the string. The client uses a cryptographic keyed Hash function

$$H : \{0,1\}^{\lambda} \times W \rightarrow \{0,1\}^{L}$$

where L is the length of the output. 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, B and the desired number of documents represented by num.
The trapdoor T_W is transmitted to the CS. Using this trapdoor the CS performs search on the client's behalf.

Algorithm 3: *Build_Trap*

```
a) Input: The master key $(K)$, the session key $(k_s)$, keywords $(W)$, Hash function $H(\cdot)$, desired number of documents $(num)$.
b) Initialization:
    Initialize dynamic Array $B$.
c) Trapdoor Generation:
    • let $b \leftarrow Into\_Integer(Enc_K(W))$ mod $P$.
    • for $1 \leq u \leq W$
      – let $a \leftarrow Into\_Integer(H_K(W))$ mod $P$
      – let $c \leftarrow a * b$ mod $P$
      – $B[u] = c$
    • let $d \leftarrow H_{k_s}(b)$.
    • $T_W \leftarrow (d, B, num)$.
d) Output: Transmit $T_W$ to CS.
```

D. Search_Outcome Algorithm

CS now using the transmitted trapdoor searches for the documents containing the disjunctive multi-keywords. The server has d, B and num. The CS tries to find entries for which the following condition holds true $d == H_{k_s}(B[u] * a^{-1} mod P)$ where u represents the total number of keywords in the search query. On a positive hit, CS performs the addition of the frequencies represented as Z, the CS also records the number of words that add up to form a particular frequency within Z. This helps us achieve disjunctive search. In this way if a client has searched for the three disjunctively chosen keywords, the document containing all the three keywords will be given priority over the documents containing fewer number of the searched keywords. Now, within Z, CS searches for the highest value, records the index location and correspondingly stores the document identifier in the array X. This helps the CS achieve ranking of the documents. The total number of returned encrypted document identifiers are equal to the num mentioned in the trapdoor.

Algorithm 4: *Search_Outcome*

```
a) Input: A trapdoor $T_W$ transmitted by the client, a session key $k_s$, a Hash functions $H(\cdot)$ (same as Build_Trap phase) and the index table $I$.
b) Initialization:
  • Dynamic Arrays $X, Y, Z$.
c) Searching:
  • for $1 \leq q \leq W$
    – for $1 \leq l \leq sizeofI$:
      ◦ if $(d == H_{k_s}(B[q] * a^{-1} mod P))$ :
        ◇ $Y[q] = l$
        ◇ for $1 \leq m \leq Y.length$:
          ▷ Add all the $RF$ within a row, store it in $Z$. Also store the number of $RF$ that add up to form a particular index of $Z$.
          ▷ Find the highest value in $Z$ OR gate {the greatest number of keywords added}, return the $Enc_K(id(D_i))$ corresponding to the $I[l][identifiedvalue]$.
    – $X[\,] \leftarrow Enc_K(id(D_i))$
c) Output: Ranked encrypted document identifiers $Enc_K(id(D_i))$.
```

E. Dec Algorithm

The client after receiving the ranked encrypted document identifiers, decrypts them to uncover the document identifiers containing the required set of keywords.

Algorithm 5: *Dec*

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

V. Security Analysis

Firstly, we revisit the existing security definitions and explain their limitations. We propose and present new security definitions applicable to our scheme and in accordance with the threat model presented in Section II.

A. Limitations of Previous Definitions

Curtmola *et al.* in [11] claimed that all the previous definitions did not provide adequate amount of security, hence they were vulnerable to security attacks. Notwithstanding the previous definitions, they introduced two new definitions Adaptive and Non-Adaptive Indistinguishability. These definitions are widely accepted and used to date.

We remark that Curtmola's work was indeed a breakthrough and provides the desired level of security when the trapdoor is deterministic. In [14], again authors clearly state that *"A serious limitation of known SSE constructions (including ours) is that the token they generate are deterministic, in the sense that the same token will always be generated for the same keyword. Currently, it is not known how to design efficient SSE schemes with probabilistic trapdoors"*. Since our proposed SE construction is based on probabilistic trapdoors, therefore we term the definitions proposed by them a "Baseline" for any SE scheme and we require stringent definitions to prove the security of our proposed RMSE scheme.

B. Proposed Definitions

Definition 1:*(Keyword-Trapdoor Indistinguishability for RMSE- Informal Version) A RMSE scheme is secure in the sense of* Keyword-Trapdoor Indistinguishability *if for any two adaptively chosen keywords and constructed trapdoor for any one of the chosen keywords, no (probabilistic polynomial-time) adversary can distinguish the trapdoor of one keyword from the other with probability non-negligibly better than 1/2.*

Proof Sketch: We assume that the adversary chooses the disjunctive multi-keywords adaptively *i.e.* based on the outcome of the previous search the selection of multiple keywords can be made. Against the selected keywords the adversary is given access to the corresponding trapdoor. This way the adversary may form a history containing the keywords and associated trapdoors. Now from the past history, the adversary has adaptively select two distinct queries comprising of any number of multiple keywords. The adversary is given one trapdoor corresponding to any of the queries sent earlier. The adversary is successful if it guesses the query corresponding to the generated and transmitted trapdoor. Intuitively, this could lead the adversary to perform more sophisticated attacks and violate the privacy of the search.

Definition 2:*(Trapdoor-Index Indistinguishability for RMSE- Informal Version) A RMSE scheme is secure in the sense of* Trapdoor-Index Indistinguishability *if for any two adaptively chosen keywords, the constructed trapdoor and index entries for any of the chosen keywords, no (probabilistic-time) adversary can distinguish the index entry of one trapdoor from the other with the probability non-negligibly better than 1/2.*

Proof Sketch: The adversary is given access to the trapdoors and the associated index table entries. We assume that the adversary chooses the queries comprising of multiple keywords adaptively *i.e.* based on the previous history shared, the selection of the query can be made. Against the selected query the adversary is given access to the corresponding trapdoor and the associated index table entry. Now the adversary has to select two distinct queries adaptively making a selection from the past history. The adversary is given one trapdoor corresponding to any of the queries sent earlier. The adversary is successful if it guesses the index table entry corresponding to the trapdoor. Intuitively, this could lead the adversary to perform more sophisticated attacks and if the scheme lacks to provide indistinguishability, the adversary may violate the privacy of the search.

Theorem 1: RMSE *is a completely indistinguishable SE scheme.*

Proof Sketch: At a very high level, this proof for our RMSE automatically follows the proofs of definition 1 and definition 2. Since the trapdoors are probabilistic, *i.e.* for the same keywords queried twice, an adaptive adversary is not able to predict the result prior to performing the search. Therefore, the RMSE scheme ensures Keyword-Trapdoor and Trapdoor-Index indistinguishability. Having said this, the adversary cannot even guess the required document identifiers prior to the search. Hence, our proposed RMSE can be termed as completely indistinguishable.

The above definitions and theorem conclude that since our scheme is based on a probabilistic trapdoor hence it is secure under the Known Cipher-text Model and the known background model. Furthermore, the indistinguishability security proof assures that our scheme mitigates the privacy concerns associated to the keywords and trapdoors.

Theorem 2: RMSE *ensures correctness and soundness.*

Proof Sketch: At a very high level, we prove the correctness and soundness of our proposed RMSE scheme. It is straightforward to verify that for two distinct keywords, different hashes appear due to which different trapdoors are generated. Therefore, while searching for multiple keywords, there is a negligible probability that the same hashes may appear for different keyword. The soundness property is associated to the correctness. Since, using the inverses we are correctly able to map the trapdoor to the index entry, therefore, the correct set of encrypted document identifiers will definitely be returned to the client. This leads to the soundness of our proposed RMSE scheme. In other words, if a scheme does not ensure correctness, it cannot provide soundness.

Hence, the proposed RMSE scheme ensures a high level of security by defying any distinguishability attacks.

VI. PERFORMANCE ANALYSIS

Performance analysis helps validate the claim of our proposed scheme being "lightweight". This is a two-fold process *i.e.* we perform the asymptotic comparative analysis of our proposed scheme against similar existing schemes, then we implement a proof of concept prototype by deploying it to the British Telecommunication's Public Cloud offering and measuring the computational time of our scheme over a real corpus.

A. Complexity Analysis

It is observed that the KeyGen and the Dec phase of all the schemes is nearly identical, therefore, our analysis does not take these two phases of the schemes into account.

Starting with MRSE_I and MRSE_II [10], the main difference between MRSE_I and MRSE_II is that MRSE_II is more privacy preserving. For the Build_Index, MRSE_I requires two multiplications over the matrix of the order $(d+2)*(d+2)$ and a $(d+2)$-dimension vector. Hence, the total computational complexity of the Build_Index is $O(md^2)$. Similarly, MRSE_II requires $O(md^2+U^2)$, where U dummy keywords are added to preserve the privacy. The Build_Trap phase for both MRSE_I and MRSE_II requires two multiplications over the matrix of the order $(d+2)*(d+2)$ and a $(d+2)$-dimension vector. Hence the complexity of Build_Trap phase is $O(d^2)$. In case of MRSE_I, the Search_Outcome phase requires to compute the inner product of two $(d+2)$-dimension vectors twice for the dataset containing m documents, resulting in $O(md)$. The complexity for MRSE_II is $O(md+U)$.

TABLE II
COMPUTATIONAL COMPLEXITY ANALYSIS

Schemes	Build_Index	Build_Trap	Search_Outcome
MRSE_I [10]	$O(md^2)$	$O(d^2)$	$O(md)$
MRSE_II [10]	$O(md^2+U^2)$	$O(d^2)$	$O(md+U)$
EMRS [9]	$O(md^2+d^2)$	$O(d^2)$	$O(d^{'}ds^{'})$
This Paper	$O(d+dm)$	$O(d^{'})$	$O(dd^{'}+m^{'}d^{'})$

d represents the total number of keywords, (m) is the total number of documents, U represents the dummy keywords, $(d^{'})$ represents the number of keywords in the trapdoor, $s^{'}$ represents the number of a blocks in a document including dummy blocks, $m^{'}$ represents the documents containing the keyword $d^{'}$.

Analyzing the EMRS scheme [9], the Build_Index requires two multiplications over the matrix of the order $(d+2)*(d+2)$ and a $(d+2)$-dimension vector. For the relevance score generation a computation of $O(md)$ is required. Therefore, the total computational complexity of the Build_Index is $O(md^2+md)$. For the Build_Trap phase, two multiplications over the matrix of the order $(d+2)*(d+2)$ and a $(d+2)$-dimension vector are required. So the complexity of Build_Trap is $O(d^2)$. The search requires $O(d^{'}ds^{'})$.

Now, we present the computational complexity of our proposed RMSE scheme. The main task in the Build_Index is to calculate the Hash of all the keywords in the dictionary and compute their inverses. This leads to the computational complexity of $O(d)$. To enhance the privacy of the index, we have to mask the entries. Since the masking and relevance frequencies can be calculated simultaneously, the total computation complexity for the Build_Index of the proposed RMSE scheme is $O(d+dm)$. The computational complexity of the Build_Trap is dependent upon the total number of keywords $(d^{'})$ in the trapdoor. So, for every keyword in the trapdoor, we have to calculate the Hash. Hence, the Build_Trap requires $O(d^{'})$ computational resources. The searching is performed at the CS. For every keyword $(d^{'})$ in the trapdoor, the CS has to parse all the keywords (d) contained in the index table consuming $O(d*d^{'})$. After that the CS has to obtain the ranked documents so it has to parse the documents $m^{'}$ containing the keyword $d^{'}$. Therefore, the computational complexity on the whole for the Search_Outcome phase becomes $O(dd^{'}+m^{'}d^{'})$.

B. Dataset Description

We use the Switchboard-1 Telephone Speech Corpus (LDC 97S62) [15] for testing our implementation originally collected by Texas Instruments in 1990-1, under DARPA sponsorship.

The Switchboard-1 speech database [16] 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 more than 120,000 distinct keywords.

C. *System Specification*

We have implemented our algorithms in Java and presented the results in the form of graphs using Excel.We have deployed the client side on our local machine and the server side on the BT Public Cloud Server(BTCS) offering. The client and the server interact with each other through sockets.

- Client side: Encryption is achieved by implementing 128-bit AES-CBC and the keyed cryptographic hash function used is SHA-128. The workstation used for the demonstration runs with an Intel Core i5 CPU running at 3.00 GHz and 8 GB of RAM.
- Server side: Entire 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.

D. *Performance Metrics*

To analyze the computational time of our proposed RMSE scheme, we analyze each of the algorithms separately over the dataset. Since the KeyGen algorithm and the Dec algorithm are the same for any SE scheme, therefore we do not analyze them, instead we analyze the remaining three algorithms.

1) Build_Index Algorithm: We generate the index table for the dataset by gradually adding 100 documents to the dataset and analyzing the time required for the index generation. As ranking may not be required in all of the SE constructions. Therefore, our analysis includes ranked and non-ranked index generation. Figure 1, graphically represents the computational time of our proposed SE scheme. The time in seconds is represented along the y-axis, whereas, the number of documents are represented along the x-axis. The graph plotted using a solid line in blue represents the time required for our ranked SE scheme, whereas, the dashed line in red represents our unranked SE scheme. It can be seen that both the variations show a linear growth with the increase in the number of documents. However unranked scheme is inefficient as compared to the ranked one. The ranked index generation requires 15.45 seconds for 2000 documents whereas the unranked index generation requires 7.22 seconds.

Figure 2, illustrates the time required for the ranked index generation for 2000 documents, while gradually increasing the number of keywords starting with 10,000 and scaling them to 120,000. Therefore, in scenarios where only a small set of keywords are required to be searched, the index generation can be made extremely efficient. For 10,000 keywords, the ranked index generation requires only 4.83 seconds, whereas, for 120,000 keywords the time increases to 14.7 seconds.

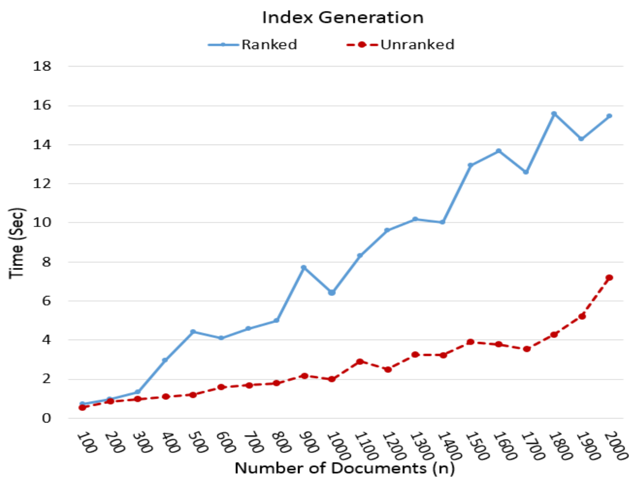

Fig. 1. Computational time for the index generation.

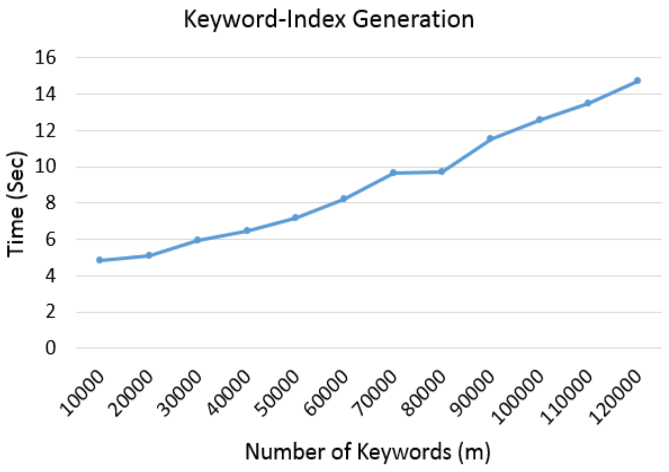

Fig. 2. Computational time for ranked index generation with variable keywords.

It is to be noted that the index generation is a one time process and does not need to be generated for every query.

2) Build_Trap Algorithm: Figure 3 demonstrates the computational time that our proposed RMSE scheme requires for trapdoor generation. We start with 1 keyword and scale it to 20 keywords. The time in seconds is along the y-axis, whereas, the number of keywords are along the x-axis. It can be seen that the RMSE scheme shows a linear growth with the increase in the number of disjunctive keywords. Hence for generating a trapdoor containing 20 keywords, we require 0.317seconds.

3) Search_Outcome Algorithm: Search_Outcome algorithm refers to the computational time required for performing disjunctive search and obtaining ranked documents.

Figure 4 shows the computational time while searching for the keyword “about time” at the BTCS. The time in seconds is along the y-axis, whereas, the number of documents are along the x-axis. We gradually increase the number of

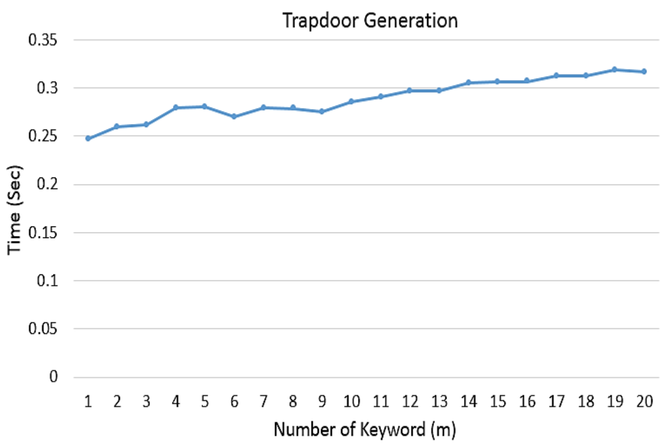

Fig. 3. Computational time for the trapdoor generation.

ranked documents required. The label placed with each node shows the number of documents that contain the desired disjunctive keywords. For example, out of 2000 documents, 1679 documents contain the keyword "about time". With the increase in the number of documents, searching shows a linear growth and for 2000 documents requires mere 0.040 seconds.

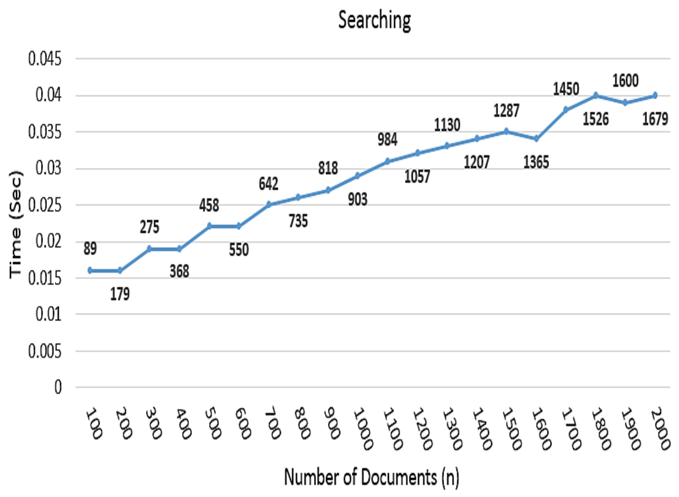

Fig. 4. Computational time for searching the keywords "about time".

Based on the above computational complexity analysis and the computational time analysis, it is obvious that our scheme is extremely lightweight as compared to existing schemes.

VII. Conclusion

In this paper, we have presented a ranked based disjunctive multi-keyword searchable encryption scheme. The scheme uses an inverted index for the searching and is based on a probabilistic trapdoor that helps resist distinguishability attacks and any privacy breeches. The security analysis yields that our scheme resists distinguishability attacks. We have successfully implemented and deployed our scheme onto the British Telecommunication's Public Cloud Server offering. We have tested our scheme over the Switchboard-1 Telephone Speech Corpus. The computational complexity analysis yields that our scheme is not only scalable but is also lightweight as compared to similar existing schemes. Our scheme can be adapted to the telecom domain for the efficient and effective multi-keyword disjunctive searching.

Acknowledgment

We are thankful to Intelligent Voice Ltd. for providing the dataset used in the implementation. We would also like to thank British Telecom Plc. for their valuable discussions during the deployment. This research has been supported through the REACH UK-INDIA research funds.

References

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

[2] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: ensuring privacy of electronic medical records," *Proceedings of the 2009 ACM workshop on Cloud computing security*, p. 103114, 2009.

[3] Y. Zhao, X. Wang, and H. Tang, "Secure Genomic Computation through Site-Wise Encryption." *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, vol. 2015, pp. 227–31, 2015.

[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," *Eurocrypt*, p. 506522, 2004.

[5] I. Witten, A. Moffat, and T. Bell, "Managing Gigabytes. Compressing and Indexing Documents and Images," *IEEE Transactions on Information Theory*, vol. 41, p. 519, 1999.

[6] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," *Proceedings - International Conference on Distributed Computing Systems*, pp. 253–262, 2010.

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

[8] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," *Proceedings - IEEE INFOCOM*, pp. 2112–2120, 2014.

[9] 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 Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 312–325, 2016.

[10] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 1 2014.

[11] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

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

[13] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," *ACM conference on Computer and communications security*, pp. 965–976, 2012.

[14] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7859 LNCS, pp. 258–274, 2013.

[15] J. Godfrey and E. Holliman, "Switchboard-1 Release 2 - Linguistic Data Consortium." [Online]. Available: https://catalog.ldc.upenn.edu/LDC97S62

[16] J. Godfrey, E. 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.