Enabling Protected and Proficient Ranked Keyword Discover In Excess of Outsourced Cloud Data

B.Manju vani [1] N.Phani Kumar [2]

1 M.Tech, CSE, S.V. College of Engineering.
2 Asst. Professor of Department of CSE, S.V. College of Engineering

Abstract—Cloud computing economically enables the paradigm of data service outsourcing. However, to protect data privacy, sensitive cloud data has to be encrypted before outsourced to the commercial public cloud, which makes effective data utilization service a very challenging task. Although traditional searchable encryption techniques allow users to securely search over encrypted data through keywords, they support only Boolean search and are not yet sufficient to meet the effective data utilization need that is inherently demanded by large number of users and huge amount of data files in cloud. In this paper, we define and solve the problem of secure ranked keyword search over encrypted cloud data. Ranked search greatly enhances system usability by enabling search result relevance ranking instead of sending undifferentiated results, and further ensures the file retrieval accuracy. Specifically, we explore the statistical measure approach, i.e. relevance score, from information retrieval to build a secure searchable index, and develop a one-to-many order-preserving mapping technique to properly protect those sensitive score information. The resulting design is able to facilitate efficient server-side ranking without losing keyword privacy. Thorough analysis shows that our proposed solution enjoys "as-strong-as-possible" security guarantee compared to previous searchable encryption schemes, while correctly realizing the goal of ranked keyword search. Extensive experimental results demonstrate the efficiency of the proposed solution.

Keywords— Ranked search, searchable encryption, order-preserving mapping, confidential data, cloud computing

I. INTRODUCTION

Cloud Computing is the long dreamed vision of computing as a utility, where cloud customers can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources . The benefits brought by this new computing model include but are not limited to: relief of the burden for storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc .As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud, such as emails, personal health records, com- pany finance data, and government documents, etc. The fact that data owners and cloud server are no longer in the same trusted domain may put the outsourced unencrypted data at risk . the cloud server may leak data information to unauthorized entities or even be hacked . It follows that sensitive data has to be encrypted prior to outsourcing for data privacy and com- bating unsolicited accesses. However, data encryption makes effective data utilization a very challenging task given that there could be a large amount of outsourced data files. Besides, in Cloud Computing, data owners may share their outsourced data with a large number of users, who might want to only retrieve certain specific data files they are interested in during a given session. One of the most popular ways to do so is through keyword-based search. Such keyword search technique allows users to selectively retrieve files of interest and has been widely applied in plaintext search scenarios. Unfortunately, data encryption, which restricts user 's ability to perform keyword search and further demands the protection of keyword privacy, makes the traditional plaintext search methods fail for encrypted cloud data. Although traditional searchable encryption schemes allow a user to securely search over encrypted data through keywords without first decrypting it, these techniques support only con- vent ional Boolean keyword search , without capturing any relevance of the files in the search result. When directly applied in large collaborative data outsourcing cloud environment, they may suffer from the following two main drawbacks. On the one hand, for each search request, users without pre-knowledge of the encrypted cloud data have to go through every retrieved file in order to find ones most matching their interest, which demands possibly large amount of post processing over-

Therefore, how to enable a searchable encryption sys- tem with support of secure ranked search, is the problem tackled in this paper. Our work is among the first few ones to explore ranked search over encrypted data in Cloud Computing. Ranked search greatly enhances sys- tem usability by returning the matching files in a ranked order regarding to certain relevance criteria (e.g., key- word frequency), thus making one step closer towards practical deployment of privacy-preserving data hosting services in the context of Cloud Computing. To achieve our design goals on both system security and usability, we propose to bring together the advance of both crypto and IR community to design the ranked searchable symmetric encryption scheme, in the spirit of "as-strong- as-possible" security guarantee. Specifically, we explore the statistical measure approach from IR and text-mining to embed weight information (i.e. relevance score) of each file during the establishment of searchable index before outsourcing the encrypted file collection. As directly outsourcing relevance scores will leak lots of sensitive frequency information against the keyword privacy, we then integrate a recent crypto primitive [14] order- preserving symmetric encryption (OPSE) and properly modify it to develop a one-to-many order-preserving mapping technique for our purpose to protect those sensitive weight information, while providing efficient ranked search functionalities. Our contribution can be summarized as follows:

1) For the first time, we define the problem of se- cure ranked keyword search over encrypted cloud data, and provide such an effective protocol, which fulfills the secure ranked search functionality with little relevance score information leakage against keyword privacy.

2) Thorough security analysis shows that our ranked searchable symmetric encryption scheme indeed enjoys "as-strong-as-possible" security guarantee compared to previous SSE schemes.

3) We investigate the practical considerations and enhancements of our ranked search mechanism, including the efficient support of relevance score dynamics, the authentication of ranked search results, and the reversibility of our proposed one-to- many order-preserving mapping technique.

4) Extensive experimental results demonstrate the effectiveness and efficiency of the proposed solution. The rest of the paper is organized as follows. Section 2 gives the system and threat model, our design goals, notations and preliminaries. Then we provide the frame- work, definitions and basic scheme in Section 3, followed by Section 4, which gives the detailed description of our ranked searchable symmetric encryption system. Section 5 gives the security analysis. Section 6 studies further enhancements and practical considerations, followed by Section 7 on performance evaluations. Related work for both searchable encryption and secure result ranking is discussed in Section 8. Finally, Section 9 gives the concluding remark of the whole paper.

II. PROBLEM STATEMENT

II.I The System and Threat Model

We consider an encrypted cloud data hosting service involving three different entities, as illustrated in Fig.1: data owner, data user, and cloud server. Data owner has a collection of n data files $C = (F_1, F_2, \ldots, F_n)$ that he wants to outsource on the cloud server in encrypted form while still keeping the capability to search through them for effective data utilization reasons. To do so, before outsourcing, data owner will first build a secure searchable index **I** from a set of m distinct keywords $W = (w_1, w_2, ..., w_m)$ extracted[2] from the file collection C, and store both the index **I** and the encrypted file collection C on the cloud server.

We assume the authorization between the data owner and users is appropriately done. To search the file collection for a given keyword w, an authorized user generates and submits a search request in a secret form a trapdoor T_w of the keyword w to the cloud server. Upon receiving the search request T_w, the cloud server is responsible to search the index **I** and return the corresponding set of files to the user. We consider the secure ranked keyword search problem as follows: the search result should be returned according to certain ranked relevance criteria (e.g., keyword frequency learn nothing or little about the relevance criteria as they exhibit significant sensitive information against keyword privacy. To reduce bandwidth, the user may send an optional value k along with the trapdoor T_w and cloud server only sends back the top-k most relevant files to the user 's interested keyword w.

We primarily consider an "honest-but-curious" server in our model, which is consistent with most of the previous searchable encryption schemes. We assume the cloud server acts in an

"honest" fashion and correctly follows the designated protocol specification, but is "cu- riots" to infer and analyze the message flow received during the protocol so as to learn additional information. In other words, the cloud server has no intention to actively modify the message flow or disrupt any other kind of services. However, in some unexpected events, the cloud server may behave beyond the "honest-but- curious" model. We specifically deal with this scenario in Section .

II.II Design Goals

To enable ranked searchable symmetric encryption for effective utilization of outsourced and encrypted cloud data under the aforementioned model, our system de- sign should achieve the following security and per- formance guarantee. Specifically, we have the follow- ing goals: i) Ranked keyword search: to explore differ- ent mechanisms for designing effective ranked search schemes based on the existing searchable encryption framework; ii) Security guarantee: to prevent cloud server from learning the plaintext of either the data files or the searched keywords, and achieve the "as- strong-as-possible" security strength compared to exist- ing searchable encryption schemes; iii) Efficiency: above goals should be achieved with minimum communication and computation overhead.

III THE DEFINITIONS AND BASIC SCHEME

In the introduction we have motivated the ranked key- word search over encrypted data to achieve economies of scale for Cloud Computing. In this section, we start from the review of existing searchable symmetric encryption (SSE) schemes and provide the definitions and frame- work for our proposed ranked searchable symmetric encryption (RSSE). Note that by following the same security guarantee of existing SSE, it would be very inefficient to support ranked search functionality over encrypted data, as demonstrated in our basic scheme. The discussion of its demerits will lead to our proposed scheme.

III.I Background on Searchable Symmetric Encryption

Searchable encryption allows data owner to outsource his data in an encrypted manner while maintaining the selectively-search capability over the encrypted data. Generally, searchable encryption can be achieved in its full functionality using an oblivious RAMs [16]. Al- though hiding everything during the search from a ma- licious server (including access pattern), utilizing oblivi- ous RAM usually brings the cost of logarithmic number of interactions between the user and the server for each search request. Thus, in order to achieve more efficient solutions, almost all the existing works on searchable encryption literature resort to the weakened security guarantee, i.e., revealing the access pattern and search pattern but nothing else. Here access pattern refers to the outcome of the search result, i.e., which files have been retrieved. The search pattern includes the equality pattern among the two search requests (whether two searches were performed for the same keyword), and any information derived thereafter from this statement. We refer readers to [12] for the thorough discussion on SSE definitions.

III.II Definitions and Framework of RSSE System

We follow the similar framework of previously proposed searchable symmetric encryption schemes [12] and adapt the framework for our ranked searchable encryption sys- tem. A ranked searchable encryption scheme consists of four algorithms (KeyGen, Build Index, Trapdoor, Search Index). Our ranked searchable encryption sys- tem can be constructed from these four algorithms in two phases, Setup and Retrieval:

- Setup: The data owner initializes the public and se- cret parameters of the system by executing KeyGen, and pre-processes the data file collection C by using Build Index to generate the searchable index from the unique words extracted from C. The owner then encrypts the data file collection C, and publishes the index including the keyword frequency based relevance scores in some encrypted form, together with the encrypted collection C to the Cloud. As part of Setup phase, the data owner also needs to distribute the necessary secret parameters (in our case, the trapdoor generation key) to a group of authorized users by employing off-the-shelf public key cryptography or more efficient primitive such as broadcast encryption.

- **Retrieval**: The user uses Trapdoor to generate a secure trapdoor corresponding to his interested keyword, and submits it to the cloud server. Upon receiving the trapdoor, the cloud server will derive a list of matched file IDs and their corresponding encrypted relevance scores by searching the index via Search Index. The matched files should be sent back in a ranked sequence based on the relevance scores. However, the server should learn nothing or little beyond the order of the relevance scores.

Note that as an initial attempt to investigate the secure ranked searchable encryption system, in this paper we focus on single keyword search. In this case, the IDF factor in equation 1 is always constant with regard to the given searched keyword. Thus, search results can be accurately ranked based only on the term frequency and file length information contained within the single file using equation

III.III The Basic Scheme

Before giving our main result, we first start with a straightforward yet ideal scheme, where the security of our ranked searchable encryption is the same as previous SSE schemes, i.e., the user gets the ranked results without letting cloud server learn any additional information more than the access pattern and search pattern. However, this is achieved with the trade-off of efficiency: namely, either should the user wait for two round-trip time for each search request, or he may even lose the capability to perform top-k retrieval, resulting the unnecessary communication overhead. We believe the analysis of these demerits will lead to our main result. Note that the basic scheme we discuss here is tightly pertained to recent work [12], though our focus in on secure result ranking. Actually, it can be considered as the most simplified version of searchable symmetric

Also note that in this way, server still learns nothing about the value of relevance scores, but it knows the requested files are more relevant than the unrequited ones, which inevitably leaks more information than the access pattern and search pattern.

IV. Efficient Ranked Searchable Symmetric Encryption Scheme

The above straightforward approach demonstrates the core problem that causes the inefficiency of ranked searchable encryption. That is how to let server quickly perform the ranking without actually knowing the over encrypted file collection, we now resort to the newly developed cryptographic primitive – order preserving symmetric encryption (OPSE) [14] to achieve more practical performance.

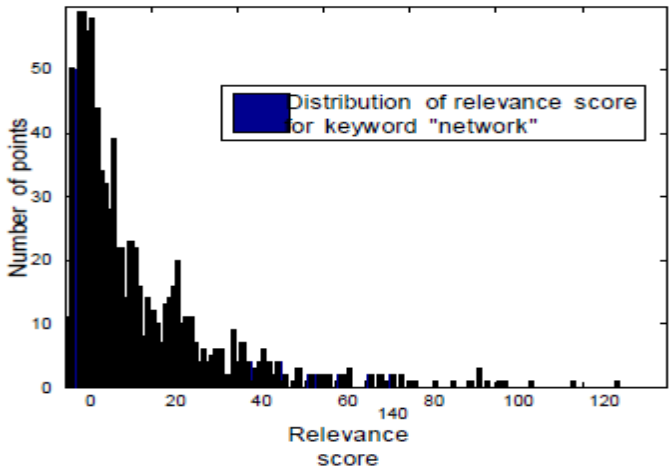

Fig. 2: An example of relevance score distribution.

Note that by resorting to OPSE, our security guarantee of RSSE is inherently weakened com- pared to SSE, as we now let server know the relevance order. However, this is the information we want to trade- off for efficient RSSE, as discussed in previous Section 3. We will first briefly discuss the primitive of OPSE and its pros and cons. Then we show how we can adapt it to suit our purpose for

ranked searchable encryption with an "as-strong-as-possible" security guarantee. Finally, we demonstrate how to choose different scheme parameters via concrete examples.

IV.I Using Order Preserving Symmetric Encryption

The OPSE is a deterministic encryption scheme where the numerical ordering of the plaintexts gets preserved by the encryption function. Boldyreva et al. [14] gives the first cryptographic study of OPSE primitive and provides a construction that is provably secure under the security framework of pseudorandom function or pseudorandom permutation. Namely, considering that defined by a combination of M out of N ordered items, an OPSE is then said to be secure if and only if an adversary has to perform a brute force search over all the possible combinations of M out of N to break the encryption scheme. If the security level is chosen to be 80 bits, then it is suggested to choose $M = N/2 > 80$ so that the total number of combinations will be greater than 2^{80}. Their construction is based on an uncovered relationship between a random order-preserving func- tion (which meets the above security notion) and the hyper geometric probability distribution, which will later be denoted as HGD. We refer readers to [14] for more details about OPSE and its security definition.

At the first glance, by changing the relevance score encryption from the standard indistinguishable symmet- ric encryption scheme to this OPSE, it seems to follow directly that efficient relevance score ranking can be achieved just like in the plaintext domain. However, as pointed out earlier, the OPSE is a deterministic encryption scheme. This inherent deterministic property, if not treated appropriately, will still leak a lot of information as any deterministic encryption scheme will do. One such information leakage is the plaintext distribution. Take Fig. 2 for example, which shows a skewed relevance score distribution of keyword "network", sampled from 1000 files of our test collection. For easy exposition, we encode the actual score into 128 levels in domain from 1 to 128. Due to the deterministic property, if we use OPSE directly over these sampled relevance scores, the resulting cipher text shall share exactly the same distribution as the relevance score in Fig. 2. On the other hand, previous research works [18], [22] have shown that the score distribution can be seen as keyword specific. Specifically, in [22], the authors have shown that the TF distribution of certain keywords from the Enron email corpus[3] can be very peaky, and thus result in significant information leak for the corresponding keyword. In [18], the authors further point out that the TF distribution of the keyword in a given file collection usually follows a power law distribution, regardless of the popularity of the keyword. Their results on a few test file collections show that not only different keywords can be differentiated by the slope and value range of their TF distribution, but even the normalized TF distributions, i.e., the original score distributions (see the equation 2), can be keyword specific. Thus, with certain background information on the file collection, such as knowing it contains only technical research papers, the adversary may be able to reverse-engineer the keyword "network" directly from the encrypted score distribution without actually breaking the trapdoor construction, nor does the adversary need to break the OPSE.

IV.II One-to-many Order-preserving Mapping

Therefore, we have to modify the OPSE to suit our purpose. In order to reduce the amount of information leakage from the deterministic property, an one-to-many OPSE scheme is thus desired, which can flatten or ob- furcated the original relevance score distribution, increase its randomness, and still preserve the plaintext order. To do so, we first briefly review the encryption process of original deterministic OPSE, where a plaintext m in domain D is always mapped to the same random-sized on-overlapping interval bucket in range R, determined by a keyed binary search over the range R and the result of a random HGD sampling function. A cipher text c is then chosen within the bucket by using m as the seed for some random selection function.

Our one-to-many order-preserving mapping employs the random plaintext-to-bucket mapping of OPSE, but incorporates the unique file IDs together with the plain- text m as the random seed in the final cipher text chosen process. Due to the use of unique file ID as part of random selection seed, the same plaintext m will no longer be deterministically assigned to the same cipher- text c, but instead a random value within the randomly assigned bucket in range R. The whole process is shown in Algorithm 1, adapted from [14]. Here TapeGen(·) is a random coin generator and HYGEINV(·) is the efficient function implemented in MATLAB as our instance for the HGD(·) sampling function. The correctness of

our one-to-many order-preserving mapping follows directly from the Algorithm 1. Note that our rational is to use the OPSE block cipher as a tool for different application scenarios and achieve better security, which is suggested by and consistent with [14]. Now, if we denote OPM as our one-to-many order-preserving mapping function.

IV.III Choosing Range Size of R

We have highlighted our idea, but there still needs some care for implementation. Our purpose is to discard the peaky distribution of the plaintext domain as much as possible during the mapping, so as to eliminate the on the domain D. Clearly, according to our random one- to-many order-preserving mapping (Algorithm 1 line 6), the larger size the range R is set, the less peaky feature will be preserved. However, the range size |R| cannot be arbitrarily large as it may slow down the efficiency of HGD function. Here, we use the min-entropy as our tool to find the size of range R.

Algorithm 1 One-to-many Order-preserving Mapping-OPM

1: **procedure** $OPM_K(D, R, m, id(F))$
2: **while** $|D| != 1$ **do**
3: $\{D, R\} \leftarrow$ BinarySearch(K, D, R, m);
4: **end while**
5: coin $\leftarrow$ TapeGen(K, (D, R, 1||m, id(F)));
6: $C \xleftarrow{coin} R$;
7: return c;
8: **end procedure**

9: **procedure** BinarySearch(K, D, R, m);
10: $M \leftarrow |D|$; $N \leftarrow |R|$;
11: $d \leftarrow \min(D) - 1$; $r \leftarrow \min(R) - 1$;
12: $y \leftarrow r + \lceil N/2 \rceil$;
13: coin $\leftarrow$ TapeGen(K, (D, R, 0||y));
14: $x \leftarrow d +$ HYGEINV(coin, M, N, $y - r$);
15: **if** $m \leq x$ **then**
16: $D \leftarrow \{d + 1, \ldots, x\}$;
17: $R \leftarrow \{r + 1, \ldots, y\}$;
18: **else**
19: $D \leftarrow \{x + 1, \ldots, d + M\}$;
20: $R \leftarrow \{y + 1, \ldots, r + N\}$;
21: **end if**
22: return $\{D, R\}$;
23: **end procedure**

V. SECURITY ANALYSIS

We evaluate the security of the proposed scheme by analyzing its fulfillment of the security guarantee de- scribed in Section 2. Namely, the cloud server should not learn the plaintext of either the data files or the searched keywords. We start from the security analysis of our one-to-many order-preserving mapping. Then we analyze the security strength of the combination of one-to-many order-preserving mapping and SSE.

V.I Security Analysis for One-to-many Mapping

Our one-to-many order-preserving mapping is adapted from the original OPSE, by introducing the file ID as the additional seed in the final cipher text chosen process. Since such adaptation

only functions at the final cipher text selection process, it has nothing to do with the randomized plaintext-to-bucket mapping process in the original OPSE. In other words, the only effect of introducing file ID as the new seed is to make multiple plaintext duplicates m's no longer deterministically mapped to the same cipher text c, but instead mapped to multiple random values within the assigned bucketing range R. This helps flatten the cipher text distribution.

to some extent after mapping. However, such number of plaintext duplicates are not large. In case there are many duplicates of plaintext m, its corresponding cipher text distribution after mapping may still exhibit certain skew- ness or peaky feature of the plaintext distribution, due to the relative small size of assigned bucket selected from range R.

This is why we propose to appropriately enlarge R in Section 4.3. Note that in the original OPSE, size R is determined just to ensure the number of different combinations between D and R is larger than 2^{80}. But from a practical perspective, properly enlarging R in our one-to-many case further aims to ensure .

V.II Security Analysis for Ranked Keyword Search

Compared to the original SSE, the new scheme embeds the encrypted relevance scores in the searchable index in addition to file ID. Thus the encrypted scores are the only additional information that the adversary can utilize against the security guarantee, i.e., keyword privacy and file confidentiality. Due to the security strength of the file encryption scheme, the file content is clearly well protected. Thus, we only need to focus on keyword privacy. bucket assignment (inherited from OPSE) and the highly flattened one-to-many mapping still makes it difficult for the adversary to predict the original plaintext score distribution, let alone reverse-engineer the keywords. Also note that we use different order-preserving encryption keys for different posting lists, which further reduces the information leakage from an overall point of view. Thus, the keyword privacy is also well preserved in our scheme.

VI FURTHER ENHANCEMENTS AND INVESTIGATIONS

Above discussions have shown how to achieve an efficient RSSE system. In this section, we give further study on how to make the RSSE system more readily deployable in practice. We start with some practical considerations on the index update and show how our mechanism can gracefully handle the case of score dynamics without introducing re-computation overhead on data owners. For enhanced quality of service assurance, we next study how the RSSE system can support ranked search result authentication. Finally, we uncover the reversible property of our one-to-many order-preserving mapping, which may find independent use in other interesting application scenarios.

VI.I Supporting Score Dynamics

In Cloud Computing, outsourced file collection might not only be accessed but also updated frequently for various application purposes (see [19]–[21], for example). Hence, supporting the score dynamics in the searchable index for an RSSE system, which is reflected from the corresponding file collection updates, is thus of practical importance. Here we consider score dynamics as adding newly encrypted scores for newly created files, or mod- ifying old encrypted scores for modification of existing files in the file collection. Ideally given a posting list in the inverted index, the encryption of all these newly changed scores should be incorporated directly without affecting the order of all other previously encrypted scores, and we show that our proposed one-to-many order-preserving mapping does exactly that. Note that we do not consider file deletion scenarios because it is not hard to infer that deleting any file and its score does not affect the ranking orders of the remaining files in the searchable index. This graceful property of supporting score dynamics is inherited from the original OPSE scheme, even though we made some adaptations in the mapping process. This can be observed from the BinarySearch(•) procedure in Algorithm 1, where the same score will always be mapped to the same random-sized non-overlapping bucket, given the same encryption key and the same parameters of the plaintext domain D and cipher text range R. Because the buckets themselves are non-overlapping, the newly changed scores indeed do not affect previously mapped values.

Algorithm 2 Reversing One-to-many Order-preserving Mapping-ROPM

```
1: procedure OPM_K(D, R, c, id(F))
2:     while |D| != 1 do
3:         {D, R} ← BinarySearch(K, D, R, c);
4:     end while
5:     m ← min (D);
6:     coin ← TapeGen(K, (D, R, 1||m, id(F)));
7:     w ← r
8:     if w = c then return m;
9:     end if
10:    return ⊥;
11: end procedure

12: procedure BinarySearch(K, D, R, c);
13:    M ← |D|; N ← |R|;
14:    d ← min (D) − 1; r ← min (R) − 1;
15:    y CCC r + dN/2e;
16:      coin ← TapeGen(K, (D, R, 0||y));
17:    x d ← HYGEINV(coin, M, N, y − r);
18:    if c ≤ y then
19:        D ← {d + 1, . . . , x};
20:        R ← {r + 1, . . . , y};
21:    else
22:        D ← {x + 1, . . . , d + M};
23:        R ← {y + 1, . . . , r + N};
24:    end if
25:    return {D, R};
26: end procedure
```

VI.II Authenticating Ranked Search Result

In practice, cloud servers may sometimes behave beyond the semi-honest model. This can happen either because cloud server intentionally wants to do so for saving cost when handling large number of search requests, or there may be software bugs, or internal/external attacks. Thus, enabling a search result authentication mechanism that can detect such unexpected behaviors of cloud server is also of practical interest and worth further investigation. To authenticate a ranked search result (or Top-k re- trieval), one need to ensure: 1) the retrieved results are the most relevant ones; 2) the relevance sequence among the results are not disrupted. To achieve this two authentication requirements,

VI.III Reversing One-to-many Order-preserving Mapping

For any order-preserving mapping process, being re- eversible is very useful in many practical situations, especially when the underlying plaintext values need to be BinarySearch(·) procedure in Algorithm 1. The intuitionism that the plaintext-to-bucket mapping process of OPSE is reversible. Namely, as long as the cipher text is chosen from the certain bucket, one can always find through the BinarySearch(·) procedure to uniquely identify the plaintext value, thus making the mapping reversible. For completeness, which again we acknowledge that is adapted from [14]. The corresponding reversed mapping performance is reported in Section 7.2.

RELATED WORK

Searchable Encryption: Traditional searchable encryption [8]–[12], [24], [25] has been widely studied as a cryptographic primitive, with a focus on security definition formalizations and efficiency improvements. Song et al. [8] first introduced the notion of searchable encryption. They proposed a scheme in the symmetric key setting, where each word in the file is encrypted independently under a special two-layered encryption construction. Thus, a searching overhead is linear to the whole file collection length. Goh [9] developed a Bloom filter based per-file index, reducing the work load for each search request proportional to the number of files in the collection. Chang et al. [11] also developed a similar per-file index scheme. To further enhance search efficiency, Curtmola et al. [12] proposed a per-keyword based approach, where a single encrypted hash table index is built for the entire file collection, with each entry consisting of the trapdoor of a keyword and an encrypted set of related file identifiers. Searchable encryption has also been considered in the public-key setting. Bone et al. [10] presented the first public-key based searchable encryption scheme, with an analogous scenario to that of [8]. In their construction, anyone with the public key can write to the data stored on the server but only authorized users with the private key can search. As an attempt to enrich query predicates, conjunctive keyword search over encrypted data have also been proposed in [26]–[28]. Recently, aiming at tolerance of both minor typos and format inconsistencies in the user search input, fuzzy keyword search over encrypted cloud data has been proposed by Li. et al in [29]. Note that all these schemes support only Boolean keyword search, and none of them support the ranked search problem which we are focusing in this paper.

Following our research on secure ranked search over encrypted data, very recently, Cao et al. [30] pro- pose a privacy-preserving multi-keyword ranked search scheme, which extends our previous work in [1] with support of multi-keyword query. They choose the pri nipple of "coordinate matching", i.e., as many matches as possible, to capture the similarity between a multi- keyword search query and data documents, and later quantitatively formalize the principle by a secure inner product computation mechanism. One disadvantage of the scheme is that cloud server has to linearly traverse the whole index of all the documents for each search request, while ours is as efficient as existing SSE schemes with only constant search cost on cloud server.

Secure top-k retrieval from Database Community [18], [22] from database community are the most related work to our proposed RSSE. The idea of uniformly distributing posting elements using an order-preserving cryptographic function was first discussed in [22]. How-

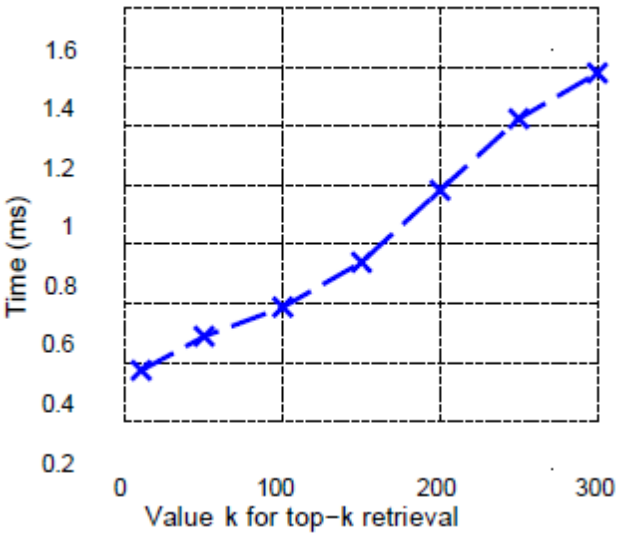

Fig. 7: The time cost for top-k retrieval.

ever, the order-preserving mapping function proposed in [22] does not support score dynamics, i.e.,

any insertion and updates of the scores in the index will result in the posting list completely rebuilt. [18] uses a different order-preserving mapping based on pre-sampling and training of the relevance scores to be outsourced, which is not as efficient as our proposed schemes. Besides, when scores following different distributions need to be inserted, their score transformation function still needs to be rebuilt. On the contrary, in our scheme the score dynamics can be gracefully handled, which is an impor- tant benefit inherited from the original OPSE. This can
be observed from the BinarySearch(·) procedure in Algo-rithm 1, where the same score will always be mapped to the same random-sized non-overlapping bucket, given the same encryption key. In other words, the newly changed scores will not affect previous mapped values. We note that supporting score dynamics, which can save quite a lot of computation overhead when file collection changes, is a significant advantage in our scheme. More- over, both works above do not exhibit thorough security analysis which we do in the paper. Other Related Techniques Allowing range queries over encrypted data in the public key settings has been studied in [31], [32], where advanced privacy preserving schemes were proposed to allow more sophisticated multi-attribute search over encrypted files while preserving the attributes' privacy. Though these two schemes provide provably strong security, they are generally not efficient in our settings, as for a single search request, a full scan and expensive computation over the whole encrypted scores corresponding to the keyword posting list are required. Moreover, the two schemes do not support the ordered result listing on the server side. Thus, they can not be effectively utilized in our scheme since the user still does not know which retrieved files would be the most relevant.

CONCLUSION

In this paper, as an initial attempt, we motivate and solve the problem of supporting efficient ranked keyword search for achieving effective utilization of remotely stored encrypted data in Cloud Computing. We first give a basic scheme and show that by following the same existing searchable encryption framework, it is very inefficient to achieve ranked search. We then appropri- ately weaken the security guarantee, resort to the newly developed crypto primitive OPSE, and derive an efficient one-to-many order-preserving mapping function, which allows the effective RSSE to be designed. We also investigate some further enhancements of our ranked search mechanism, including the efficient support of relevance score dynamics, the authentication of ranked search results, and the reversibility of our proposed one-to-many order-preserving mapping technique. Through thorough security analysis, we show that our proposed solution is secure and privacy-preserving, while correctly realizing the goal of ranked keyword search. Extensive experimental results demonstrate the efficiency of our solution.

REFERENCES

[1] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in Proc. of ACNS'05, 2005.

[2] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Search- able symmetric encryption: improved definitions and efficient constructions," in Proc. of ACM CCS'06, 2006.

[3] A. Singhal, "Modern information retrieval: A brief overview," IEEE Data Engineering Bulletin, vol. 24, no. 4, pp. 35–43, 2001.

[4] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order- preserving symmetric encryption," in Proc. of Eurocrypt'09, volume 5479 of LNCS. Springer, 2009.

[5] J. Zobel and A. Moffat, "Exploring the similarity space," SIGIR Forum, vol. 32, no. 1, pp. 18–34, 1998.

[6] O. Goldreich and R. Ostrovsky, "Software protection and simu- lation on oblivious rams," Journal of the ACM, vol. 43, no. 3, pp. 431–473, 1996.

[7] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in Proc. of Crypto'07, volume 4622 of LNCS. Springer, 2007.

[8] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k retrieval from a confidential index," in Proc. of EDBT'09, 2009.

[9] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Verifiability and Data Dynamics for

Storage Security in Cloud Computing," IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 22, no. 5, pp. 847–859, May 2011.

[10] C. Wang, Q. Wang, K. Ren, and W. Lou, "Towards Secure and Dependable Storage Services in Cloud Computing," IEEE Trans- actions on Service Computing (TSC), to appear.

[11] C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy- Preserving Public Auditing for Secure Cloud Storage," IEEE Transactions on Computers (TC), to appear.

[12] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A. L. Varna, S. He, M. Wu, and D. W. Oard, "Confidentiality-preserving rank-ordered search," in Proc. of the Workshop on Storage Security and Survivabil- ity, 2007.

[13] RFC, "Request For Comments Database," http://www.ietf.org/ rfc.html.

[14] B. Waters, D. Balfanz, G. Durfee, and D. Smetters, "Building an encrypted and searchable audit log," in Proc. of NDSS'04, 2004.

[15] F. Bao, R. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in Proc. of ISPEC'08, 2008.

[16] P. Golle, J. Staddon, and B. R. Waters, "Secure Conjunctive Key- word Search over Encrypted Data," in Proc. of ACNS'04, 2004, pp. 31–45.

[17] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in Proc. of ICICS'05, 2005.

[18] Y. H. Hwang and P. J. Lee, "Public Key Encryption with Conjunc- tive Keyword Search and Its Extension to a Multi-User System," in Proc. of Pairing'07, 2007, pp. 31–45.

[19] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in Proc. of IEEE INFOCOM'10 Mini-Conference, 2010.

[20] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in Proc. of INFOCOM'11, 2011.

[21] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in Proc. of TCC'07, 2007, pp. 535–554.

[22] E. Shi, J. Bethencourt, H. Chan, D. Song, and A. Perrig, "Multi- dimensional range query over encrypted data," in Proc. of IEEE Symposium on Security and Privacy'07, 2007.