Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 8, August -2017

A Safe and Runtime Multi-Keyword Ranked Search for Encrypted Information over Cloud Data

[1]Dr .Mohammed Abdul Waheed , [2]Sandeep

Associate Professor, Department of Studies in Computer Science and Engineering Visvesvaraya Technological University CPGS, Kalaburagi, Karnataka (India)

M tech Student, Department of Studies in Computer Science and Engineering Visvesvaraya Technological University CPGS, Kalaburagi, Karnataka (India)

ABSTRACT:-*Due to the increasing popularity of cloud computing, more and more data owners are motivated to outsource their data to cloud servers for great convenience and reduced cost in data management. However, sensitive data should be encrypted before outsourcing for privacy requirements, which obsoletes data utilization like keyword-based document retrieval. In this paper, we present a secure multi-keyword ranked search scheme over encrypted cloud data, which simultaneously supports dynamic update operations like deletion and insertion of documents. Specifically, the vector space model and the widely-used TF_IDF model are combined in the index construction and query generation. We construct a special tree-based index structure and propose a "Greedy Depth-first Search" algorithm to provide efficient multi-keyword ranked search. The secure kNN algorithm is utilized to encrypt the index and query vectors, and meanwhile ensure accurate relevance score calculation between encrypted index and query vectors. In order to resist statistical attacks, phantom terms are added to the index vector for blinding search results . Due to the use of our special tree-based index structure, the proposed scheme can achieve sub-linear search time and deal with the deletion and insertion of documents flexibly. Extensive experiments are conducted to demonstrate the efficiency of the proposed scheme.*

Keywords: *Searchable encryption, multi-keyword ranked search, dynamic update, cloud computing.*

I . INTRODUCTION

Cloud computing has been considered as a new model of enterprise IT infrastructure, which can organize huge resource of computing, storage and applications, and enable users to enjoy ubiquitous, convenient and on-demand network access to a shared pool of configurable computing resources with great efficiency and minimal economic overhead [1]. Attracted by these appealing features, both individuals and enterprises are motivated to outsource their data to the cloud, instead of purchasing software and hardware to manage the data themselves. Despite of the various advantages of cloud services, outsourcing sensitive information (such as e-mails, personal health records, company finance data, government documents, etc.) to remote servers brings privacy concerns. The cloud service providers (CSPs) that keep the data for users may access users' sensitive information without authorization. A general approach to protect the data confidentiality is to encrypt the data before outsourcing [2]. However, this will cause a huge cost in terms of data usability. For example, the existing techniques on keyword-based information retrieval, which are widely used on the plaintext data, cannot be directly applied on the encrypted data. Downloading all the data from the cloud and decrypt locally is obviously impractical, secure tree-based search scheme over the encrypted cloud data, which supports multikey word ranked search and dynamic operation on the document collection n order to address the above problem, researchers have designed some general-purpose solutions with fully-homomorphic encryption [3] or oblivious RAMs[4]. Our contributions are summarized as follows:

1) We design a searchable encryption scheme that supports both the accurate multi-keyword ranked search and flexible dynamic operation on document collection.

2) Due to the special structure of our tree-based index, the search complexity of the proposed scheme is fundamentally kept to logarithmic. And in practice, the proposed scheme can achieve higher search efficiency by executing our "Greedy Depth-first Search" algorithm. Moreover, parallel search can be flexibly performed to further reduce the time cost of search process.

II. METHODS AND MATERIAL

Searchable encryption schemes enable the clients to store the encrypted data to the cloud and execute keyword search over ciphertext domain. Due to different cryptography primitives, searchable encryption schemes can be constructed using public key based cryptography or symmetric key based cryptography single keyword boolean searchs chemes, which are very simple in terms of functionality. Afterward, abundant works have been proposed underdifferent threat models to achieve various search functionality, Multi-keyword boolean search allows the users to input multiple query

keywords to request suitable documents Among these works, conjunctive keyword search schemes only return the documents that contain all of the query keywords. Disjunctive keyword search schemes return all of the documents that contain a subset of the query keywords. Predicate search scheme are proposed to support both conjunctive and disjunctive search. All these multi keyword search schemes retrieve search results based on the existence of keywords, which cannot provide acceptable result ranking functionality. Ranked search can enable quick search of the most relevant data. Sending back only the top-*k* most relevant documents can effectively decrease network traffic. Some early works have realized the ranked search using order-preserving techniques, but they are designed only for single keyword search. In addition, the search efficiency of the scheme is linear with the cardinality of document collection.The authors constructed a searchable index tree based on vector space model and adopted cosine measure together with TF×IDF to provide ranking results.

SYSTEM ARCHITECTURE

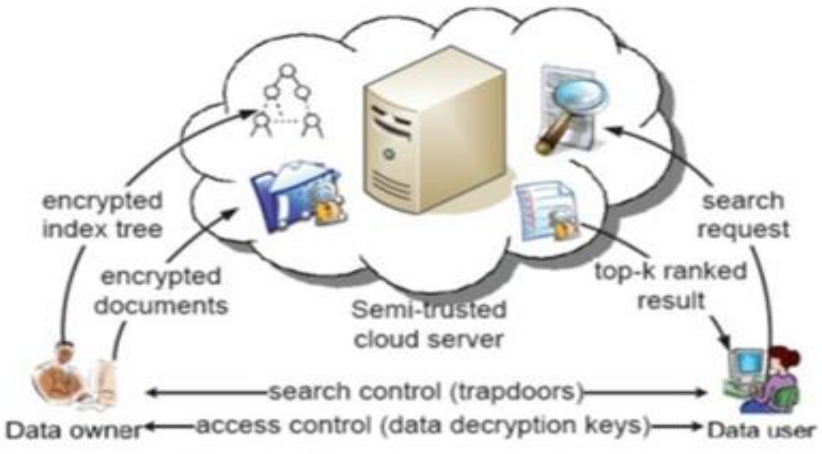

Fig. 1 The architecture of ranked search over encrypted cloud data

THE SYSTEM MODELS

The system model in this paper involves three different entities: data owner, data user and cloud server, as illustrated in Fig. 1.

Data owner has a collection of documents *F ={f1; f2; :::; fn}* that he wants to outsource to the cloud server in encrypted form while still keeping the capability to search on them for effective utilization. In our scheme, the data owner firstly builds a secure searchable tree index *I* from document collection *F*, and then generates an encrypted document collection *C* for *F*. Afterwards, the data owner outsources the encrypted collection *C* and the secure index *I* to the cloud server, and securely distributes the key information of trapdoor generation (including keyword IDF values) and document decryption to the authorized data users. Besides, the data owner is responsible for the update operation of his documents stored in the cloud server. While updating, the data owner generates the update information locally and sends it to the server.

Data users are authorized ones to access the documents of data owner. With *t* query keywords, the authorized user can generate a trapdoor *TD* according to search control mechanisms to fetch *k* encrypted documents from cloud server. Then, the data user can decrypt the documents with the shared secret key

.

Cloud server stores the encrypted document collection *C* and the encrypted searchable tree index *I* for data owner. Upon receiving the trapdoor *TD* from the data user, the cloud server executes search over the index tree *I*, and finally returns the corresponding collection of top- *k* ranked encrypted documents. Besides, upon receiving the update information from the data owner, the server needs to update the index *I* and document collection *C* according to the received information. The cloud server in the proposed scheme is considered as "honest-but-curious",

To enable secure, efficient, accurate and dynamic multi keyword ranked search over outsourced encrypted cloud data under the above models, our system has the following design goals.

Dynamic: The proposed scheme is designed to provide not only multi-keyword query and accurate result ranking, but also dynamic update on document collections.

Search Efficiency: The scheme aims to achieve sublinear search efficiency by exploring a special tree-based index and an efficient search algorithm.

Privacy-preserving: The scheme is designed to prevent the cloud server from learning additional information about the document collection, the index tree, and the query. The specific privacy requirements are summarized as follows,

1) *Index Confidentiality and Query Confidentiality:* The underlying plaintext information, including keywords in the index and query, TF values of keywords stored in the index, and IDF values of query keywords, should be protected from cloud server;
2) *Trapdoor Unlinkability:* The cloud server should not be able to determine whether two encrypted queries (trapdoors) are generated from the same search request
3) *Keyword Privacy:* The cloud server could not identify the specific keyword in query, index or document collection by analyzing the statistical information like term frequency. Note that our proposed scheme is not designed to protect access pattern, i.e., the sequence of returned documents.

we firstly describe the unencrypted dynamic multi-keyword ranked search (UDMRS) scheme which is constructed on the basis of vector space model and KBB tree. Based on the UDMRS scheme, two secure search schemes (BDMRS and EDMRS schemes)

Index Construction of UDMRS Scheme: Introduced the KBB index tree structure, which assists us in introducing the index construction. In the process of index construction, we first generate a tree node for each document in the collection. These nodes are the leaf nodes of the index tree. Then, the internal tree nodes are generated based on these leaf nodes. An example of our index tree is shown in Figure the data structure of the tree node is defined as *(ID;D; Pl; Pr; FID)*, where the unique identity *ID* for each tree node is generated through the function GenID().

- *CurrentNodeSet* – The set of current processing nodes which have no parents. If the number of nodes is even, the cardinality of the set is denoted as $2h(h \in \mathbf{Z}+)$, else the cardinality is denoted as $(2h + 1)$.
- *TempNodeSet* – The set of the newly generated nodes.

Search Process of UDMRS Scheme: The search process of the UDMRS scheme is a recursive procedure upon the tree, named as "Greedy Depthfirst Search (GDFS)" algorithm. We construct a result list denoted as*RList*, whose element is defined as *(RScore; FID)*. Here, the *RScore* is the relevance score of the document *fFID* to the query, which is calculated according to Formula (1). The *RList* stores the *k*accessed documents with the largest relevance scores to the query. The elements of the list are ranked in descending order according to the *RScore*, and will be updated timely during the search process.

- RScore(*Du;Q*) – The function to calculate the relevance score for query vector *Q* and index vector *Du* stored in node *u*, which is defined in Formula (1).
- *kthscore* – The smallest relevance score in current *RList*, which is initialized as 0.
- *hchild* – The child node of a tree node with higher relevance score.

BDMRS Scheme : Based on the UDMRS scheme, we construct the basic dynamic multi-keyword ranked search (BDMRS) scheme by using the secure kNN algorithm [38]. The BDMRS scheme is designed to achieve the goal of privacy preserving in the known ciphertext model,the four algorithms included are described as follows:

- *SK* ← Setup() Initially, the data owner generates the secret key set *SK*, including 1) a randomly generated *m*-bit vector *S* where *m* is equal to the cardinality of dictionary,2) two $(m \times m)$ invertible matrices $M1$ and $M2$. Namely, $SK = \{S;M1;M2\}$.
- *I* ← GenIndex(*F; SK*) First, the unencrypted index tree *T* is built on *F* by using *T* ← BuildIndexTree(*F*). Secondly, the data owner generates two random vectors *{Du* Finally, the encrypted index tree *I* is built where the node *u* stores two encrypted index vectors*Iu*
- *TD* ← GenTrapdoor(*Wq; SK*) With keyword set *Wq*,the unencrypted query vector *Q* with length of *m* is generated. If $wi \in Wq$, $Q[i]$ stores the normalized *IDF* value of *wi*; else $Q[i]$ is set to 0. Similarly, the query vector *Q* is split into two random vectors Q' and Q''. The difference is that if $S[i] = 0$, $Q'[i]$ and $Q''[i]$ are set to two random values whose sum equals to $Q[i]$; else $Q'[i]$ and $Q''[i]$ are set as the same as $Q[i]$. Finally, the algorithm returns the trapdoor $TD = \{M^{-1}1\ Q';M^{-1}2\ Q''\}$.
- *RelevanceScore* ← SRScore(*Iu;TD*) With the trapdoor *TD*, the cloud server computes the relevance score of node *u* in the index tree *I* to the query

EDMRS Scheme: The security analysis above shows that the BDMRS scheme can protect the *Index Confidentiality and Query Confidentiality* in the known ciphertext model. However, the cloud server is able to link the same search requests by tracking path of visited nodes. In addition, in the known background model, it is possible for the cloud server to identify a keyword as the normalized TF distribution of the keyword can be exactly obtained from the final calculated relevance scores. The primary cause is that the relevance score calculated from *Iu* and *TD* is exactly equal to that from *Du* and *Q*. A heuristic method to further improve the security is to break such exact equality. Thus, we can introduce

some tunable randomness to disturb the relevance score calculation. In addition, to suit different users' preferences for higher accurate ranked results or better protected keyword privacy, the randomness are set adjustable.

The enhanced EDMRS scheme is almost the same as BDMRS scheme except that:

- *SK* ← Setup() In this algorithm, we set the secret vector *S* as a *m*-bit vector, and set *M*1 and *M*2 are $(m + m') \times (m + m')$ invertible matrices, where m' is the number of phantom terms.
- *I* ← GenIndex(*F; SK*) Before encrypting the index vector *Du*, we extend the vector *Du* to be a $(m+m')$- dimensional vector. Each extended element $Du[m+ j]$, $j = 1; :::;m'$, is set as a random number *"j* .
- *TD* ← GenTrapdoor(*Wq; SK*) The query vector *Q* is extended to be a $(m + m')$-dimensional vector. Among the extended elements, a number of m'' elements are randomly chosen to set as 1, and the rest are set as 0.
- *RelevanceScore* ← SRScore(*Iu;TD*) After the execution of relevance evaluation by cloud server, the final relevance score for index vector *Iu* equals to $Du \cdot Q + \Sigma''v$, where $v \in \{j|Q[m + j] = 1\}$.

Dynamic Update Operation of DMRS: Since the index of DMRS scheme is designed as a balanced binary tree, the dynamic operation is carried out by updating nodes in the index tree. Note that the update on index is merely based on document identifies, and no access to the content of documents is required *{I's; ci}* ← GenUpdateInfo(*SK; Ts; i; updtype*)) This algorithm generates the update information *{Is; ci}* which will be sent to the cloud server. If *updtype* is equal to *Del*, the data owner deletes from the subtree the leaf node that stores the document identity *i* and updates the vector *D* of other nodes in subtree *Ts*, so as to generate the updated subtree *T '* If *updtype* is equal to *Ins*, the data owner generates a tree node $u = (GenID(); D; null; null; i)$ for the document *fi*, where $D[j] = TFfi;wj$ for $j = 1; :::;m$. Then, the data owner inserts this new node into the subtree *Ts* as a leaf node and updates the vector *D* of other nodes in subtree*Ts* *{I'; C'}* ← Update(*I; C; updtype; I's; ci*) In thi algorithm,cloud server replaces the corresponding subtree *Is*(the encrypted form of *Ts*) with *I' s*, so as to generate a new index tree *I'*. If *updtype* is equal to *Ins*, cloud server inserts the encrypted document *ci* into *C*, obtaining a new collection *C'*. If *updtype* is equal to *Del*, cloud server deletes the encrypted document *ci* from *C* to obtain the new collection *C'*

III. RESULTS AND DISCUSSION

Precision and Privacy : The search precision of scheme is affected by the dummy keywords in EDMRS scheme. Here, the 'precision' is defined]: $Pk = k'=k$, where k' is the number of real top-*k* documents in the retrieved *k* documents.

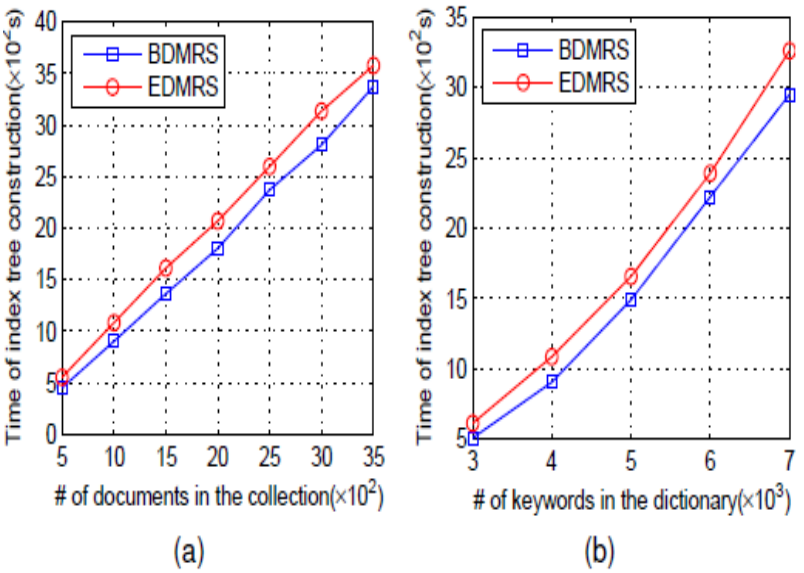

Fig. 2 Time cost for trapdoor generation: (a) for different sizes of dictionary with the fixed number of query keywords $t = 10$, and (b) for different numbers of query keywords with the fixed dictionary,$m = 1000$

In the EDMRS scheme, phantom terms are added to the index vector to obscure the relevance score calculation, so that the cloud server cannot identify keywords by analyzing the *TF* distributions of special keywords. Here, we quantify the obscureness of the relevance score by "rank privacy", which is defined as:

$$P'k = \Sigma|ri - r'i| = k2;$$

where *ri* is the rank number of document in the retrieved top-*k* documents, and *r'i* is its real rank number in the whole ranked results. The larger rank privacy denotes the higher security of the scheme.

Efficiency

1 Index Tree Construction The process of index tree construction for document collection *F* includes two main steps: 1) building an unencrypted KBB tree based on the document collection*F*, and 2) encrypting the index tree with splitting operation and two multiplications of a ($m \times m$) matrix.The index structure is constructed following a post order traversal of the tree based on the document collection *F*, and $O(n)$ nodes are generated during the traversal. For each node, generation of an index vector takes $O(m)$ time, vector splitting process takes $O(m)$ time, and two multiplications of a ($m \times m$) matrix takes $O(m2)$ time. Asa whole, the time complexity for index tree construction is $O(nm2)$. Apparently, the time cost for building index tree mainly depends on the cardinality of document collection *F* and the number of keywords in dictionary *W*.

2 Trapdoor Generation The generation of a trapdoor incurs a vector splitting operation and two multiplications of a ($m \times m$) matrix, thus the time complexity is $O(m2)$

3 Search Efficiency During the search process, if the relevance score at node *u* is larger than the minimum relevance score in resultlist *RList*, the cloud server examines the children of the node; else it returns. Thus, lots of nodes are not accessed during a real search. We denote the number of leaf nodes that contain one or more keywords in the query as _. Generally ,is larger than the number of required documents *k*, but far less than the cardinality of the document collection *n*. As a balanced binary tree, the height of the index is maintained to be log *n*, and the complexity of relevance score calculation is $O(m)$.

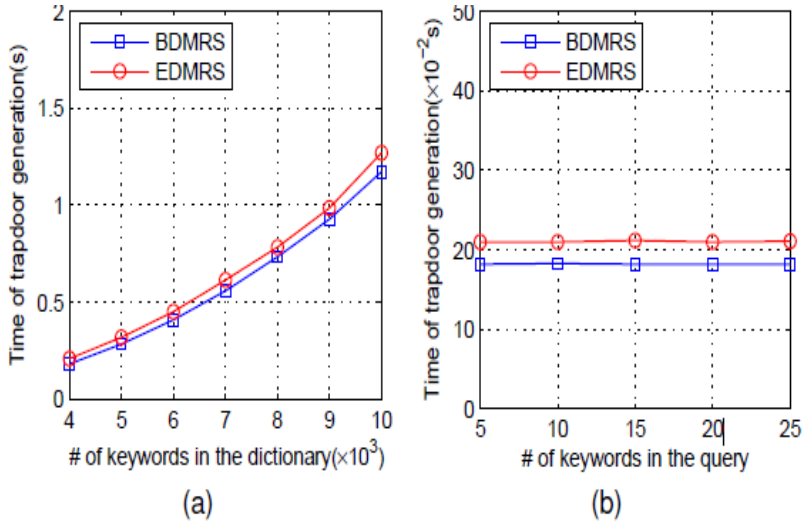

Fig.3 Time cost for trapdoor generation: (a) for different sizes of dictionary with the fixed number of query keywords, $t = 10$, and (b) for different numbers of query keywords with the fixed dictionary,$m = 4000$.

4 Update Efficiency In order to update a leaf node, the data owner needs to update log *n* nodes. Since it involves an encryption operation for index vector at each node, which takes $O(m2)$ time, the time complexity of update operation is thus $O(m2 \log n)$.

Figure 4 : shows that all details of file filename, keyword, file type upload details file size rank ,download, modify & delete

Fig 5:User can see status of file requested and download a file

IV. CONCLUSION

A secure, efficient and dynamic search scheme is proposed, which supports not only the accurate multi-keyword ranked search but also the dynamic deletion and insertion of documents. We construct a special keyword balanced binary tree as the index, and propose a "Greedy Depth-first Search" algorithm to obtain better efficiency than linear search. In addition, the parallel search process can be carried out to further reduce the time cost. The security of the scheme is protected against two threat models by using the secure kNN algorithm. Experimental results demonstrate the efficiency of our proposed scheme. There are still many challenge problems in symmetric SE schemes. In the proposed scheme, the data owner is responsible for generating updating information and sending them to the cloud server. Thus, the data owner needs to store the unencrypted index tree and the information that are necessary to recalculate the IDF values. Such an active data owner may not be very suitable for the cloud computing model. It could be a meaningful but difficult future work to design a dynamic searchable encryption scheme whose updating operation can be completed by cloud server only, meanwhile reserving the ability to support multi-keyword ranked search. In addition, as the most of works about searchable encryption, our scheme mainly considers the challenge from the cloud server. Actually, there are many secure challenges in a multi-user scheme. Firstly, all the users usually keep the same secure key for trapdoor generation in a symmetric SE scheme. In this case, the revocation of the user is big challenge. If it is needed to revoke a user in this scheme, we need to rebuild the index and distribute the new secure keys to all the authorized users. Secondly, symmetric SE schemes usually assume that all the data users are trustworthy. It is not practical and a dishonest data user will lead to many secure problems. For example, a dishonest data user may search the documents and distribute the decrypted documents to the unauthorized ones. Even more, a dishonest data user may distribute his/her secure keys to the unauthorized ones. In the future works, we will try to improve the SE scheme to handle these challenge problems.

V. REFERENCES

[1] K. Ren, C.Wang, Q.Wang et al., "Security challenges for the public cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69–73, 2012.
[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in Financial Cryptography and Data Security. Springer, 2010, pp. 136–149.
[3] C. Gentry, "A fully homomorphic encryption scheme," Ph.D.dissertation, Stanford University, 2009.
[4] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," Journal of the ACM (JACM), vol. 43, no. 3,pp. 431–473, 1996.
[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in Advances in Cryptology-Eurocrypt 2004. Springer, 2004, pp. 506–522.
[6] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III,"Public key encryption that allows pir queries," in Advances in Cryptology-CRYPTO 2007. Springer, 2007, pp. 50–67.
[7] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on. IEEE, 2000, pp. 44–55.
[8] E.-J. Goh et al., "Secure indexes." IACR Cryptology ePrint Archive, vol. 2003, p. 216, 2003.
[9] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in Proceedings of the Third international conference on Applie Cryptography and Network Security.Springer-Verlag, 2005, pp. 442–455.

[10] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006, pp. 79–88.
[11] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," INFOCOM, 2010 Proceedings IEEE. IEEE, 2010, pp. 1–5.
[12] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in Data Engineering (ICDE), 2012 IEEE 28th International Conference on. IEEE, 2012, pp. 1156–1167.
[13] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in INFOCOM, 2012 Proceedings IEEE. IEEE, 2012, pp. 451–459
[14] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security. ACM, 2013, pp. 71–82.
[15] C. Orencik, M. Kantarcioglu, and E. Savas, "A practical and secure multi-keyword search method over encrypted cloud data," in Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on. IEEE, 2013, pp. 390–397.
[16] W. Zhang, S. Xiao, Y. Lin, T. Zhou, and S. Zhou, "Secure ranked multi-keyword search for multiple data owners in cloud computing,"in Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on. IEEE, 2014, pp. 276–286.

