Scientific Journal Impact Factor (SJIF): 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

MULTIKEYWORD ARRANGEMENT PURSUIT METHOD IN ENCRYPTED CLOUD DATA WITH VULNERABILITY AND VIBRANT

Ms. Athira Mohan [1], Mr. S.P.Santhoshkumar[2],
Ms. V.Anuradha[3], Ms. G.Mariya Lavanya Sangeetha[4]
[1,2,3,4]*UG Student, Department of CSE, Shree Sathyam College of Engg. and Tech, Sankari, India*

Abstract: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources 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. 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: Cloud Computing, Data Management, Greedy Depth-first Search, kNN algorithm, TF_IDF model

I. INTRODUCTION

Cloud computing, also known as on-demand computing, is a kind of Internet-based computing that provides shared processing resources and data to computers and other devices on-demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers. It relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network. At the foundation of cloud computing is the broader concept of converged infrastructure and shared services.

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort.

Researchers have designed some general-purpose solutions with fully homomorphism encryption [4] or oblivious RAMs [5]. However, these methods are not practical due to their high computational overhead for both the cloud sever and user. On the contrary, more practical special-purpose solutions, such as searchable encryption (SE) schemes have made specific contributions in terms of efficiency, functionality and security. Searchable encryption schemes enable the client to store the encrypted data to the cloud and execute keyword search over cipher text domain. So far, abundant works have been proposed under different threat models to achieve various search functionality, such as single keyword search, similarity search, multi keyword Boolean search, ranked search, multi-keyword ranked search, etc. Among them, multi-keyword ranked search achieves more and more atten-tion for its practical applicability. Recently, some dynamic schemes have been proposed to support inserting and deleting operations on document collection. These are significant works as it is highly possible that the data owners need to update their data on the cloud server. But few of the dynamic schemes support efficient multi-keyword ranked search.

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

This paper proposes a secure tree-based search scheme over the encrypted cloud data, which supports multi-keyword ranked search and dynamic operation on the document collection. Specifically, the vector space model and the widely-used “term frequency (TF) × inverse document frequency (IDF)” model are combined in the index construction and query generation to provide multi-keyword ranked search. In order to obtain high search efficiency, we construct a tree-based index structure and propose a “Greedy Depth-first Search” algorithm based on this index tree.

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.

II. RELATED WORK

Searchable encryption schemes enable the clients to store the encrypted data to the cloud and execute keyword search over cipher text domain. Due to different cryptography primitives, searchable encryption schemes can be constructed using public key based cryptography [5], [6] or symmetric key based cryptography [10], [11]. The first symmetric searchable encryption (SSE) scheme, and the search time of their scheme is linear to the size of the data collection. Goh [8] proposed formal security definitions for SSE and designed a scheme based on Bloom filter. The search time of Goh’s scheme is O (n), where n is the cardi-nality of the document collection. Curtmola et al. [10] proposed two schemes (SSE-1 and SSE-2) which achieve the optimal search time. Their SSE-1 scheme is secure against chosen-keyword attacks (CKA1) and SSE-2 is secure against adaptive chosen-keyword attacks (CKA2).

Multikeyword Boolean search allows the users to input multiple query keywords to request suitable documents. Among these works, conjunctive keyword search schemes [17] only return the documents that contain all of the query keywords. Disjunctive keyword search schemes [19] return all of the documents that contain a subset of the query keywords. Predicate search schemes [22] 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.

III. PROBLEM FORMULATION

A. Notations and Preliminaries

- W – The dictionary, namely, the set of keywords, denoted as W = {w1; w2; :::; wm}.
- m – The total number of keywords in W.
- Wq – The subset of W, representing the keywords in the query.

- F – The plaintext document collection, denoted as a collection of n documents F = {f1; f2; :::; fn}. Each document f in the collection can be considered as a sequence of keywords.
- n – The total number of documents in F.
- C – The encrypted document collection stored in the cloud server, denoted as C = {c1; c2; :::; cn}.
- T – The unencrypted form of index tree for the whole document collection F.
- I – The searchable encrypted tree index generated from T .
- Q – The query vector for keyword set Wq.
- TD – The encrypted form of Q, which is named as trapdoor for the search request.
- Du – The index vector stored in tree node u whose dimension equals to the cardinality of the dictionary
- W. Note that the node u can be either a leaf node or an internal node of the tree.
- Iu – The encrypted form of Du.

B. Vector Space Model and Relevance Score Function.

Vector space model along with TF×IDF rule is widely used in plaintext information retrieval, which efficiently supports ranked multi-keyword search. Here, the term frequency (TF) is the number of times a given term (keyword) appears within a document, and the inverse document frequency (IDF) is obtained through dividing the cardinality of document collection by the number of documents containing the keyword. Following are the notations used in our relevance evaluation function:

- $N_{f;wi}$ – The number of keyword w_i in document f.
- N – The total number of documents.
- N_{wi} – The number of documents that contain key-word w_i.
- $TF'_{f;wi}$ – The TF value of w_i in document f.
- IDF'_{wi} – The IDF value of w_i in document collection.
- $TF_{u;wi}$ – The normalized TF value of keyword w_i stored in index vector D_u.
- IDF_{wi} – The normalized IDF value of keyword w_i in document collection.

The relevance evaluation function is defined as:

$$\mathrm{RScore}(D_u; Q) = D_u \cdot Q = \sum_{w_i \in W_q} TF_{u;wi} \times IDF_{wi} : \quad (1)$$

(1)

If u is an internal node of the tree, $TF_{u;wi}$ is calculated from index vectors in the child nodes of u. If the u is a leaf node, $TF_{u;wi}$ is calculated as:

$$TF_{u,w_i} = \frac{TF'_{f,w_i}}{\sqrt{\sum_{w_i \in \mathcal{W}} (TF'_{f,w_i})^2}}, \quad (2)$$

where $TF'_{f;wi} = 1 + \ln N_{f;wi}$. And in the search vector Q, IDF_{wi} is calculated as:

$$IDF_{w_t} = \frac{IDF'_{w_t}}{\sqrt{\sum_{w_t \in \mathcal{W}_q}(IDF'_{w_t})^2}}, \qquad (3)$$

where $IDF'_{wi} = \ln(1 + N/N_{wi})$.

A. Keyword Balanced Binary Tree.

The balanced binary tree is widely used to deal with optimization problems. The keyword balanced binary (KBB) tree in our scheme is a dynamic data structure whose node stores a vector *D*. The elements of vector *D* are the normalized TF values. Sometimes, we refer the vector *D* in the node *u* to D_u for simplicity. Formally, the node *u* in our KBB tree is defined as follows:

$$u = ID; D; P_l; P_r; FID ; \qquad (4)$$

Where *ID* denotes the identity of node *u*, P_l and P_r are respectively the pointers to the left and right child of node *u*. If the node *u* is a leaf node of the tree, *FID* stores the identity of a document, and *D* denotes a vector consisting of the normalized TF values of the keywords to the document. If the node *u* is an internal node, *FID* is set to *null*, and *D* denotes a vector consisting of the TF values which is calculated as follows:

$$D[i] = max\{u{:}P_l \rightarrow D[i]; u{:}P_r \rightarrow D[i]\}; i = 1; :::; m: \qquad (5)$$

The detailed construction process of the tree-based index is illustrated in below, which is denoted as BuildIndexTree(*F*).

The System and Threat Models

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 capa-bility 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.

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

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.

2. **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.
3. **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.

Known Cipher text Model.

In this model, the cloud server only knows the encrypted document collection C, the searchable index tree I, and the search trapdoor TD submitted by the authorized user. That is to say, the cloud server can conduct cipher text only attack (COA) in this model.

Known Background Model.

Compared with known ciphertext model, the cloud server in this stronger model is equipped with more knowledge, such as the term frequency (TF) statistics of the document collection.

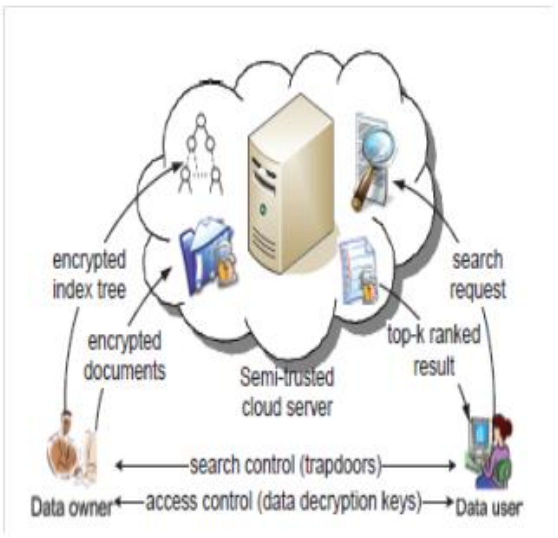

Fig. 1: The architecture of ranked search over encrypted cloud data

IV. PROPOSED SYSTEM

This paper proposes a secure tree-based search scheme over the encrypted cloud data, which supports multi-keyword ranked search and dynamic operation on the document collection. Specifically, the vector space model and the widely-used "term frequency (TF) × inverse document frequency (IDF)" model are combined in the index construction and query generation to provide multi-keyword ranked search. In order to obtain high search efficiency, we construct a tree-based index structure and propose a "Greedy Depth-first Search" algorithm based on this index tree.

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.

To resist different attacks in different threat models, we construct two secure search schemes: the basic dynamic multi-keyword ranked search (BDMRS) scheme in the known ciphertext model, and the enhanced dynamic multi-keyword ranked search (EDMRS) scheme in the known background model.

Based on the UDMRS scheme, we construct the basic dynamic multi-keyword ranked search (BDMRS) scheme by using the secure kNN algorithm. The BDMRS preserving in the known ciphertext model, and the four algorithms included are described as follows:

SK ← Setup() Initially, the data owner generates the secret key set SK, including 1) a randomly generat-ed m-bit vector S where m is equal to the cardinality of dictionary, and 2) two (m×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 gener-ates two random vectors {Du′; Du″} for index vector Du in each node u, according to the secret vector S. Specifically, if S[i] = 0, Du′[i] and Du″[i] will be set equal to Du[i]; if S[i] = 1, Du′[i] and Du″[i] will be set as two random values whose sum equals to Du[i]. Finally, the encrypted index tree I is built where the node u stores two encrypted index vectors

Iu = {M1T Du′; M2T Du″} (5)

TD ← GenTrapdoor(Wq; SK) With keyword set Wq, the unencrypted query vector Q with length of m is generated. If wi ∈ 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 = {M1−1Q′; M2−1Q″} (6)

RelevanceScore ← SRScore(Iu; TD) With the trap-door TD, the cloud server computes the relevance score of node u in the index tree I to the query. Note that the relevance score calculated from encrypted vectors is equal to that from unencrypted vectors as follows:

$$Iu \bullet TD \qquad (7)$$

V. 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.

Security analysis. We analyze the BDMRS scheme according to the three predefined privacy requirements in the design goals:

Query Unlinkability: The trapdoor of query vector is generated from a random splitting operation, which means that the same search requests will be transformed into different query trapdoors, and thus the query unlinkability is protected. However, the cloud server is able to link the same search requests according to the same visited path and the same relevance scores.

Keyword Privacy: In this scheme, the confidentiality of the index and query are well protected that the original vectors are kept from the cloud server. And the search process merely introduces inner product computing of encrypted vectors, which leaks no information about any specific keyword. Thus, the keyword privacy is protected in the known cipher text model. But in the known background model, the cloud server is supposed to have more knowledge, such as the term frequency statistics of keywords.

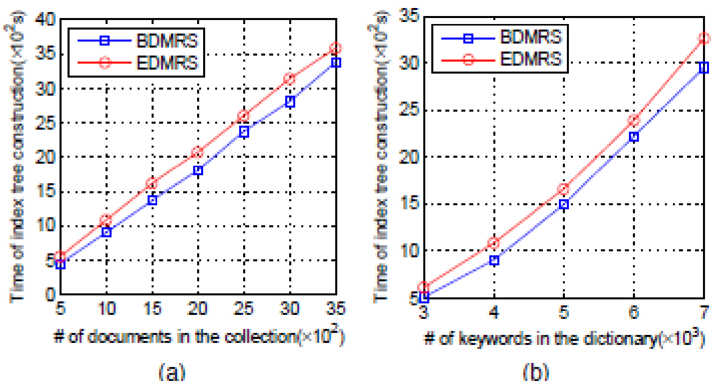

Fig 1. Time cost for index tree construction: (a) for the different sizes of document collection with the fixed dictionary, *m* = 4000, and (b) for the different sizes of dictionary with the fixed document collection, *n* = 1000.

TABLE 3: Storage consumption of index tree.

Size of dictionary	1000	2000	3000	4000	5000
BDMRS (MB)	73	146	220	293	367
EDMRS (MB)	95	168	241	315	388

1. 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). We illustrate the time cost for the deletion of a document. Fig. shows that when the size of dictionary is fixed, the deletion of a document takes nearly logarithmic time with the size of document collection. And Fig. 1. shows that the update time is proportional to the size of dictionary when the document collection is fixed.

In addition, the space complexity of each node is O(m). Thus, space complexity of the communication package of updating a document is O(m log n).

Algorithm 1: K-Nearest Neighbor

a. A positive integer k is specied, along with a new sample
b. We select the k entries in our database which are closest to the new sample
c. We nd the most common classication of these entries
d. This is the classication we give to the new sample

Algorithm 1: GDFS(IndexTreeNode u)

```
1:  if the node u is not a leaf node then
2:      if RScore(Du; Q) > kthscore then
3:          GDFS(u:hchild);
4:          GDFS(u:lchild);
5:      else
6:          return
7:      end if
8:  else
9:      if RScore(Du; Q) > kthscore then
10:         Delete the element with the smallest relevance score from RList;
11:         Insert a new element RScore(Du; Q); u:FID and sort all the elements of RList;
12:     end if
13:     return
14: end if
```

Advantages

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.

VI. CONCLUSION AND FUTURE WORK

In this paper, a secure, efficient and dynamic search scheme is proposed. 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 multiuser scheme.

REFERENCES

[1] Zhihua Xia, Member, IEEE, Xinhui Wang, Xingming Sun, Senior Member, IEEE, and Qian Wang, Member, IEEE, "A Secure and Dynamic Multi-keyword Ranked
[2] Search Scheme over Encrypted Cloud Data", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS VOL: PP NO: 99 YEAR 2015.
[3] 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.
[4] S. Kamara and K. Lauter, “Cryptographic cloud storage,” in Financial Cryptography and Data Security. Springer, 2010, pp. 136– 149.
[5] C. Gentry, “A fully homomorphic encryption scheme,” Ph.D. dissertation, Stanford University, 2009.
[6] O. Goldreich and R. Ostrovsky, “Software protection and simula-tion on oblivious rams,” Journal of the ACM (JACM), vol. 43, no. 3, pp. 431–473, 1996.
[7] 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.
[8] 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.
[9] 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.
[10] E.-J. Goh et al., “Secure indexes.” IACR Cryptology ePrint Archive, vol. 2003, p. 216, 2003.
[11] Y.-C. Chang and M. Mitzenmacher, “Privacy preserving keyword searches on remote encrypted data,” in Proceedings of the Third in-ternational conference on Applied Cryptography and Network Security. Springer-Verlag, 2005, pp. 442–455.
[12] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient con-structions,” in Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006, pp. 79–88.
[13] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, “Fuzzy keyword search over encrypted data in cloud computing,” in INFOCOM, 2010 Proceedings IEEE. IEEE, 2010, pp. 1–5.
[14] 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.
[15] 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.
[16] B. Wang, S. Yu, W. Lou, and Y. T. Hou, “Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud,” in IEEE INFOCOM, 2014.
[17] P. Golle, J. Staddon, and B. Waters, “Secure conjunctive keyword search over encrypted data,” in Applied Cryptography and Network Security. Springer, 2004, pp. 31–45.
[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 Proceedings of the First international conference on Pairing-Based Cryptography. Springer-Verlag, 2007, pp. 2–22.
[19] L. Ballard, S. Kamara, and F. Monrose, “Achieving efficient con-junctive keyword searches over encrypted data,” in Proceedings of the 7th international conference on Information and Communications Security. Springer-Verlag, 2005, pp. 414–426.
[20] D. Boneh and B. Waters, “Conjunctive, subset, and range queries on encrypted data,” in Proceedings of the 4th conference on Theory of cryptography. Springer-Verlag, 2007, pp. 535–554.
[21] B. Zhang and F. Zhang, “An efficient public key encryption with conjunctive-subset keywords search,” Journal of Network and Computer Applications, vol. 34, no. 1, pp. 262–267, 2011.
[22] J. Katz, A. Sahai, and B. Waters, “Predicate encryption support-ing disjunctions, polynomial equations, and inner products,” in Advances in Cryptology–EUROCRYPT 2008. Springer, 2008, pp. 146–162.
[23] E. Shen, E. Shi, and B. Waters, “Predicate privacy in encryption systems,” in Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography. Springer-Verlag, 2009, pp. 457–473.
[24] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, “Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption,” in Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques. Springer-Verlag, 2010, pp. 62–91.
[25] 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 Proceedings of the 2007 ACM workshop on Storage security and survivability. ACM, 2007, pp. 7–12.

[26] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, “Zerber+ r: Top-k retrieval from a confidential index,” in Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. ACM, 2009, pp. 439–449.
[27] C. Wang, N. Cao, K. Ren, and W. Lou, “Enabling secure and efficient ranked keyword search over outsourced cloud data,” Parallel and Distributed Systems, IEEE Transactions on, vol. 23, no. 8, pp. 1467–1479, 2012.