International Journal of Research Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

A Secure and Dynamic Multi-key-word Ranked Search Scheme over Encrypted Cloud Data

[1]S. SHAHEENA, [2]U. SANDHYA.
[1]Pg Scholar, Besant Theosophical College, Madanapalli
[2]Assisatant Professor, Besant theosophical college Madanapalli

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*

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. In order to address the above problem, researchers have designed some general-purpose solutions with fully-homomorphic encryption [3] or oblivious RAMs[4]. However, these methods are not practical due to their high computational overhead for both the cloud sever and user. On the contrary, more practical specialpurpose 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 ciphertext 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, multikeyword ranked search achieves more and more attention 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 multikeyword ranked search.

II. RELATED WORK

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 [5], [6] or symmetric key based cryptography [7], [8], [9], [10]. Song *et al.* [7] proposed 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 cardinality 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). These early works are single keyword boolean search schemes, which are very simple in terms of functionality. Afterward, abundant works have been proposed under different threat models to achieve various search functionality, such as single keyword search, similarity search [11], [12], [13], [14], multi-keyword boolean search [15], [16], [17], [18], [19], [20], ranked search and multi-keyword ranked search etc. Multi-keyword boolean search allows the users to input multiple query keywords to request suitable documents. Among these works, conjunctive keyword search schemes [15], [16], [17] only return the documents that contain all of the query keywords. Disjunctive keyword search schemes [18], [19] return all of the documents that contain a subset of the query keywords. Predicate search schemes [20]are proposed to support both conjunctive and disjunctive search. All these multikeyword 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. Cao *et al.* realized the first privacy-preserving multi-keywordcranked search scheme, in which documents and queriesare represented as vectors of dictionary size. With the "coordinate matching", the documents are ranked according to the number of matched query keywords. However, Cao *et al.*'s

scheme does not consider the importance of the different keywords, and thus is not accurate enough. In addition, the search efficiency of the scheme is linear with the cardinality of document collection. Sun *et al.* presented a secure multi-keyword search scheme that supports similarity-based ranking. The authors constructed a searchable index tree based on vector space model and adopted cosine measure together with TF×IDF to provide ranking results. Sun *et al.*'s search algorithm achieves better-than-linear search efficiency but results in precision loss. O¨ rencik *et al.* proposed a secure multi-keyword search method which utilized local sensitive hash (LSH) functions to cluster the similar documents. The LSH algorithm is suitable for similar search but cannot provide exact ranking.

3 PROBLEM FORMULATION

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

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.

Known Ciphertext 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 ciphertext-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. This statistical information records how many documents are there for each term frequency of a specific keyword in the whole document collection , which could be used as the keyword identity. Equipped with such statistical information, the cloud server can conduct TF statistical attack to deduce or even identify certain keywords through analyzing histogram and value range of the corresponding frequency distributions .

3.3 Design Goals

To enable secure, efficient, accurate and dynamic multikeyword ranked search over outsourced encrypted cloud

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.

4 THE PROPOSED SCHEMES

In this section, 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) are constructed against two threat models, respectively The system model in this paper involves three different entities: data owner, data user and cloud server, as

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 capabilityto 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", which is employed by lots of works on secure cloud data search. Specifically, the cloud server honestly and correctly executes search control (trapdoors) access control (data decryption keys)

Known Ciphertext 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 ciphertext-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. This statistical information records how many documents are there for each term frequency of a specific keyword in the whole document collection, which could be used as the keyword identity. Equipped with such statistical information, the cloud server can conduct TF statistical attack to deduce or even identify certain keywords through analyzing histogram and value rangeof the corresponding frequency distributions .

3.3 Design Goals

To enable secure, efficient, accurate and dynamic multikeyword ranked search over outsourced encrypted cloud construction process of the tree index, we first generate leaf nodes from the documents. Then, the internal tree nodes are generated based on the leaf nodes. This figure also shows an example of search process, in which the query vector *Q* is equal to (0*;* 0.92*;* 0*;* 0.38). In this example, we set the parameter $k = 3$ with the meaning that three documents will be returned to the user. According to the search algorithm, the search starts with the root node, and reaches the first leaf node A through $r11$ and $r22$. The relevance score of A to the query is 0.92. After that, the leaf nodes B and D are successively reached with the relevance scores 0.038 and 0.67. Next, the leaf node A is reached with score 0.58 and replace B in *RList*. Finally, the algorithm will try to search subtree rooted by $r12$, and find that there are no reasonable results in this subtree because the relevance score of $r12$ is 0.52, which is smaller than

the smallest relevance score in *RList*. 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,

4 THE PROPOSED SCHEMES

In this section, 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) are constructed against two threat models, respectively

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. Following are some other notations, and the GDFS algorithm is described in Algorithm.

Algorithm 1 BuildIndexTree($\mathcal{F}$)

Input: the document collection $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ with the identifiers $\mathcal{FID} = \{FID | FID = 1, 2, ..., n\}$.
Output: the index tree $\mathcal{T}$

1: **for** each document f_{FID} in $\mathcal{F}$ **do**
2: Construct a leaf node u for f_{FID}, with $u.ID$ = GenID(), $u.P_l = u.P_r = null$, $u.FID = FID$, and $D[i] = TF_{f_{FID}, w_i}$ for $i = 1, ..., m$;
3: Insert u to $CurrentNodeSet$;
4: **end for**
5: **while** the number of nodes in $CurrentNodeSet$ is larger than 1 **do**
6: **if** the number of nodes in $CurrentNodeSet$ is even, i.e. $2h$ **then**
7: **for** each pair of nodes u' and u'' in $CurrentNodeSet$ **do**
8: Generate a parent node u for u' and u'', with $u.ID$ = GenID(), $u.P_l = u'$, $u.P_r = u''$, $u.FID = 0$ and $D[i] = max\{u'.D[i], u''.D[i]\}$ for each $i = 1, ..., m$;
9: Insert u to $TempNodeSet$;
10: **end for**
11: **else**
12: **for** each pair of nodes u' and u'' of the former $(2h - 2)$ nodes in $CurrentNodeSet$ **do**
13: Generate a parent node u for u' and u'';
14: Insert u to $TempNodeSet$;
15: **end for**
16: Create a parent node u_1 for the $(2h - 1)$-th and $2h$-th node, and then create a parent node u for u_1 and the $(2h + 1)$-th node;
17: Insert u to $TempNodeSet$;
18: **end if**
19: Replace $CurrentNodeSet$ with $TempNodeSet$ and then clear $TempNodeSet$;
20: **end while**
21: **return** the only node left in $CurrentNodeSet$, name-

5 PERFORMANCE ANALYSIS

We implement the proposed scheme using C++ language in Windows 7 operation system and test its efficiency on a real-world document collection: the Request for Comments (RFC) [39]. The test includes 1) the search precision on different privacy level, and 2) the efficiency of index construction, trapdoor generation, search, and update. Most of the experimental results are obtained with an Intel Core(TM) Duo Processor (2.93 GHz), except that the efficiency of search is tested on a server with two Intel(R) Xeon(R) CPU E5-2620 Processors (2.0 GHz), which has 12 processor cores and supports 24 parallel threads.

5.1 Precision and Privacy

The search precision of scheme is affected by the dummy keywords in EDMRS scheme. Here, the 'precision' is defined as that in [26]: $Pk = k'=k$, where k' is the number of real top-k documents in the retrieved k documents. If a smaller standard deviation _ is set for the random 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: $P'k = \Sigma\ /ri - r'i/=k2;$ (8) 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, which is illustrated in In the proposed scheme, data users can accomplish different requirements on search precision and privacy by adjusting the standard deviation _, which can be treated as a balance parameter. We compare our schemes with a recent work proposed by Sun *et al.* , which achieves high search efficiency. Note that our BDMRS scheme retrieves the search results through exact calculation of document vector and query vector. Thus, top-k search precision of the BDMRS scheme is 100%. But as a similarity-based multi-keyword ranked search scheme, the basic scheme in suffers from precision loss due to the clustering of sub-vectors during index construction. The precision test of basic scheme is presented in Table 2. In each test, 5 keywords are randomly chosen as input, and the precision of returned top 100 results is observed. The test is repeated 16 times, and the average precision is 91%. The larger rank privacy denotes the higher security of the scheme, which is illustrated in In the proposed schem of returned top 100 results is observed.

5.2 Efficiency

5.2.1 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. As a 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. that the time cost of index tree construction is almost linear with the size of document collection, and is proportional to the number of keywords in the dictionary. Due to the dimension extension, the index tree construction of EDMRS scheme is slightly more time-consuming than that of BDMRS scheme. Although the index tree

construction consumes relatively much time at the data owner side, it is noteworthy that this is a one-time operation. On the other hand, since the underlying balanced binary tree has space complexity $O(n)$ and every node stores two m-dimensional vectors, the space complexity of the index tree is $O(nm)$. As listed in Table 3, when the document collection is fixed (n = 1000), the storage consumption of the index tree is determined by the size of the dictionary.

5.2.2 Trapdoor Generation

The generation of a trapdoor incurs a vector splitting operation and two multiplications of a (m × m) matrix, thus the time complexity is O(m2), Typical search requests usually consist of just a fewkeywords. shows that the number of query keywordshas little influence on the overhead of trapdoor generation when the dictionary size is fixed. Due to the dimension extension, the time cost of EDMRS scheme is a little higher than that of BDMRS scheme.

5.2.3 Search Efficiency

During the search process, if the relevance score at nodeu is larger than the minimum relevance score in result list 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)

6 CONCLUSION AND FUTURE WORK

In this paper, 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.

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– 14 .

[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 Applied 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," in 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. 1045-9219 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See

[14] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multikeyword fuzzy search over encrypted data in the cloud," in IEEE INFOCOM, 2014.

[15] 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.

[16] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive 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.

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

[17] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in Proceedings of the 7th international conference on Information and Communications Security. Springer-Verlag, 2005, pp. 414–426.

[18] 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.

[19] 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.

[20] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in Advances in Cryptology–EUROCRYPT 2008. Springer, 2008, pp. 146–162