Algorithm for Multi Keyword Search Over Encrypted Data in Cloud Environment

Debasis Das
Department of CSE
Indian Institute of Technology(IIT) Jodhpur
Jodhpur,Rajasthan, India
debasis@iitj.ac.in

Ruhul Amin
Department of CSE
IIIT Naya Raipur
Raipur, India
amin_ruhul@live.com

Sumit Kalra
Department of CSE
Indian Institute of Technology(IIT) Jodhpur
Jodhpur, Rajasthan, India
sumitk@iitj.ac.in

***Abstract*—Cloud Computing offers storage resources as well as network and computing resources to the organizations. This eliminates the high infrastructure cost for the organizations that are using these services as they can now dynamically pay for these services, i.e., pay per use model, which is followed by most of the cloud providers. As the organization does not locally host these resources, these are comparatively far easier to manage and use than the traditional infrastructural resources. As a result of these factors, the popularity of cloud computing is increasing continuously. But this transfer of data and applications to the cloud server also creates some challenges. It poses problems that must be dealt with properly to ensure a secure cloud computing environment. As more and more sensitive data is being uploaded on the cloud in the present scenario, the privacy and security concerns associated with the data is continuously increasing. To address this, issue the data is stored on the cloud in the encrypted form. Also, as the amount of data stored is usually tremendous, so an efficient search scheme is also necessary. So here, we deal with two significant aspects of cloud computing: Encryption and Searching. We are proposing a secure and efficient encryption scheme to encrypt the data stored in the cloud as well as the queries along with a multi-keyword search scheme to search over the encrypted cloud data.**

***Index Terms*—Cloud Computing; Multi- Keyword Searching; Encryption.**

I. Introduction

The cost and implementation advantages offered by cloud computing are enormous. Due to these advantages, both individual users and the enterprise users are increasingly outsourcing the data to the cloud [1], instead of spending a significant amount in procuring the required hardware themselves. Despite the many distinct advantages of cloud computing, some concerns are also there. Sensitive and personal information of people such as medical records, bank records, private photos are outsourced and stored on a cloud, which causes privacy concerns. For enterprise users, company financial data, government documents, user databases are outsourced, which gives rise to security concerns [2]. Also, the cloud service provider, which is storing this sensitive data, may access it without the requisite authorization. Therefore, to protect data confidentiality, data is encrypted before outsourcing. However, this gives rise to some significant problems. Most of the keyword-based retrieval techniques, which work very efficiently on plain text data, cannot be applied at all on the ciphertext. Downloading all the data on the cloud, decrypting it, and then searching is logistically impractical. Data Mining Over Encrypted Data [3,4] poses three main problems - First is the confidentiality of the encrypted data. The second main problem is the confidentiality of the user's query record, and the third problem is hiding data access patterns. Data perturbation and secure multi-party computation approaches are a few of the many possible solutions [5,6,7] which being considered for solving this problem. While encrypting the highly sensitive data, these techniques cannot be used. Also, these techniques dont produce very accurate results. Researchers have designed homomorphic encryption schemes that allow search over encrypted data. However, the extremely high computational cost associated with these methods makes them practically unfeasible on big data in cloud computing. Some search schemes [8,9,10] require exact keyword matching and therefore have a huge disadvantage in terms of data usability and user compatibility. In this paper, we propose a new scheme for tree-based secure multi-keyword ranked search over encrypted data and dynamic operation on the document collection.

The Key Generation, Encryption, and Decryption [11,12] of the data in the cloud environment and checking its integrity after the transfer of data involves three types of efficient algorithms. First is Symmetric key-based algorithms in which both encryption and decryption are done using the same key(e.g., DES, AES). It is less complicated and hence is computationally faster. It is one of the commonly used types of algorithms. Second is Asymmetric Key Algorithms in which encryption is done by the sender using the public key, and decryption is done by the receiver using the private key corresponding to the public key (i.e., RSA, ECC). They are mostly used for encrypting data of small size. Third, are Hash Algorithms which use hashing functions to produce a message digest of fixed length. Hash functions should have fixed output length, pre-image resistance, second pre-image resistance, and collision resistance. They are used to check whether the transfer of data has taken place correctly by checking the hash values before and after the transfer. In this paper, we propose an encryption scheme that uses a combination of all three algorithms to encrypt the data and

978-1-7281-3129-0/20/$31.00 ©2020 IEEE

make sure the transfer has taken place correctly. In this paper, we have used SHA-3 as our hashing algorithm. It is the most recent and secure hashing algorithm.

The rest of the paper is organized as follows: Section II provides the related works. Section III presents the objectives for Multi Keyword Search Over Encrypted Data in the cloud environment. Section IV presents the Proposed Scheme and Section V provides Proposed Algorithm for Multi Keyword Search Over Encrypted Data. Section VI provides Result Analysis for the proposed scheme. Section VII finally concludes the work.

II. Related Work

Goldwasser-Micali (GM) [1] proposed an encryption scheme in 1982. Pascal Paillier, in 1999, proposed the Paillier encryption scheme [2], which is a probabilistic public-key algorithm. The computation of the nth residue class is considered to be difficult. Because of its homomorphic encryption properties, this scheme [3] is considered to be highly malleable in the sense that it is susceptible to adaptive chosen-ciphertext attacks. The homomorphic encryption system consists of mainly three components Key Generation Algorithm, which outputs public key, (P_k) and secret key, (S_k), Encryption Algorithm that takes the public key Pk and a message m and encrypts it to ciphertext c. Decryption Algorithm uses the secret key Sk to convert the ciphertext to message m. There has been significant work in the field of Cryptography and homomorphic encryption over the past few years. Due to the advent of Quantum Computing, the security threats have increased multi-fold, which has led to the discovery of new encryption algorithms like Elliptic Curve Cryptography and Secure Hash Algorithm. The use of Elliptic Curve in cryptography was proposed independently by Neal Koblitz [4] and Victor S. Miller [5] in 1985.

Searchable encryption schemes for cloud users to store encrypted data over the cloud on which queries can be executed without decryption. Song [7] proposed the first symmetric searchable encryption, where the time complexity of their scheme is proportional to the cardinality of the data collection. Goh [8] proposed the formal security considerations for the searchable symmetric encryption scheme based on the Bloom filter. Its time complexity is O(n), where n is the number of documents. Curtmola [9] proposed two different schemes that achieve optimal search time. In conjunctive multi-keyword search schemes [8,9,10,11,22,26], only the documents that contain all of the query keywords are returned.

III. Objectives

• The significant trade-off of using highly secure encryption algorithms is the resulting significant computational cost, which can, in turn, lead to degradation in performance and an increase in the cost of cloud computing. In this paper, we propose a secure and efficient encryption algorithm to encrypt the data stored in the cloud as well as the queries.

• We aim to design a searchable encryption scheme that supports multi-keyword search on the encrypted data without decrypting it. To provide a multi-keyword ranked search, we have used the TF-IDF (Term Frequency - Inverse Document Frequency) model along with the vector space model for index construction as well as query generation. We have constructed a tree-based index to achieve higher search efficiency.

IV. Proposed Scheme

A. Encryption

While proposing a solution for encrypting data in the cloud, we must keep in mind the various shortcomings for symmetric as well as asymmetric ciphers available [12,13,14,15]. RSA alone cannot be used for large size data, and AES, being a symmetric cipher [15], has inherent problems of key exchange [16]. Thus here we will use a hybrid solution for our problem. The proposed solution is a combination of both symmetric as well as asymmetric ciphers along with hashing [17,18,19]. The solution uses RSA to overcome limitations [20,21,22] of key exchange, AES, to efficiently encrypt a large quantity of data, which is a necessity for cloud storage and SHA-3 hashing to ensure data integrity [23,234,25]. We propose the complete encryption scheme below: 1. Convert PlainText (PT) to CipherText (CT) using AES with Key=K_{aes} (K_{aes} can be 128,192 or 256 bit long).

2. Encrypt K_{aes} using RSA. We wil call this encrypted key of AES $\rightarrow$ENK$_{aes}$.

3. Calculate Hash of PlainText (PT) using SHA-3. Call this $H_{sha}(PT_i)$.

4. When user uploads a document to the cloud the above three steps are performed. This action is performed once for each document. Each encrypted document will therefore have associated with it ENK_{aes} and $H_{sha}(PT_i)$.

B. Decryption

1. Use the private key of RSA corresponding to the public key which was used for encrypting K_{aes} to decrypt it.

$ENK_{aes} \rightarrow$ (RSA + Private key) $\rightarrow K_{aes}$.

2. Now use this K_{aes} to decrypt CT:

CT $\rightarrow$ Decryption$_{AES}$ + $K_{aes} \rightarrow$ PT.

3. Finally use PT to generate hash using SHA-3, call this $H_{sha}(PT_f)$.

PT $\rightarrow$ SHA-3 $\rightarrow H_{sha}(PT_f)$.

4. Operation successful if and only if:

$H_{sha}(PT_i)$ equals to $H_{sha}(PT_f)$.

C. Searching

We aim to devise a secure and efficient multi-keyword search over the encrypted index tree [26]. The search scheme given in detail below is designed, keeping in mind the main aim of minimizing exposures of the user's data to the

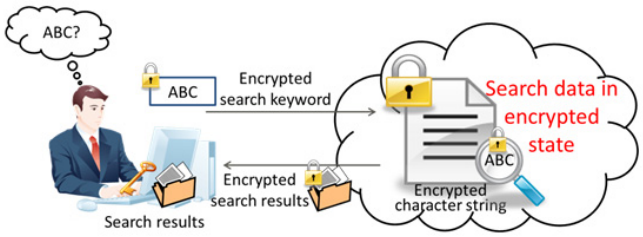

Fig. 1. Encryption in Cloud computing.

semi-trusted cloud service provider. The scheme provided below has the following design goals.

1. Dynamic update: Our proposed multi-keyword search scheme is such that it enables the user to update on the document collection in the cloud environment dynamically.

2. Efficient and Effective Search: Multi- keyword-based proposed encryption scheme for cloud environment aims to achieve search efficiency using a special tree-based index and an efficient and effective search algorithm.

3. Privacy-preserving: The proposed scheme is designed to maintain data confidentiality as a semi-trusted cloud service provider is assumed.

To make the index tree for efficient searching over encrypted data the first, we need to calculate the optimum value of m in the m-way tree. Earlier index tree-based search schemes have directly used a binary tree for searching. On the contrary, here a detailed proof is provided to show that a ternary tree is more efficient than using a binary tree.
Calculation of the optimum value of m in m-may tree:
Order of searching will be - $m \log_n m$
Here, n represent the total number of nodes. Now optimization is as follows:

$$y = m \log_n m$$

$$\frac{dy}{dm} = 0$$

$$m \frac{d}{dm} \log_n = 0$$

$$m \frac{d}{dm} (\log_m n) + \log_m n = 0$$

$$m \log_e n \frac{d}{dm} \left(\frac{1}{\log_e^m} \right) + \log_m n = 0$$

$$\frac{-m \ln(n) (\ln(m))^{-2}}{m} + \log_m n = 0$$

$$\ln(m) = 1$$

$$\Rightarrow m = e$$

Now, verifying if the obtained value of m is a minima by doing the double derivative test:

$$\frac{dy}{dm} = \log_m n \left(1 - \frac{1}{\log_e m}\right)$$

$$\frac{d^2y}{dx^2} = \frac{d}{dm} \left(\log_m n - \frac{\log_e n}{(\log_e m)^2} \right)$$

$$= \frac{\log_e n}{m} \left(2 (\log_e m)^{-3} - (\log_e m)^{-2} \right)$$

Here, it is clear that after substituting m = e, we'll get a positive value of the double derivative, which proves that m = e is minima, and thus this will be used in our tree construction (3-way tree). Hence we have proved that the search order for the above scheme will be minimum when a 3-way tree is used for index construction.

TABLE I
SYMBOLS

Symbol	Meaning
D	Set of keywords. D=$\{d_1, d_2, d_3, ..., d_m\}$
m	Total number of keywords in dictionary D
D_{query}	Set of keywords in the query
DOCS	Collection of plain text documents. DOCS = $\{doc_1, doc_2, doc_3, ..., doc_n\}$
n	Total number of keywords in collection of documents DOCS
EDOCS	Collection of ciphered text documents. EDOCS = $\{edoc_1, edoc_2, edoc_3, ..., edoc_n\}$
T	Index tree (unencrypted form)
ET	Searchable Index tree (encrypted form)
Q	Keyword set for Query vector D_{query}
TD	Encrypt(Q), which is trapdoor for the search request
I_u	Tree node u with Index vector. Dimension of this vector is equal to that of D.
EI_u	Encrypted form of I_u.

TABLE II
SYMBOLS OF EVALUATION FUNCTION

Symbol	Meaning
N_{doc,d_i}	Number of keywords $d_i indocumentdoc$
n	Total # of keywords in collection of documents DOCS
N_{d_i}	Number of documents conatining word d_i
TF_{doc,d_i}	Term frequency of $d_i indoc$
IDF_{d_i}	Normalized IDF value of keyword d_i

The relevance between Document and a query is evaluated as follows:
RScore(Doc_u, Q) = $Doc_u.Q = \sum_{d_i \in D} (TF_{u,d_i} \times IDF_{d_i})$

An internal node of the tree is u, $TF_{u,d_i i}$ is evaluated from index vectors in the child nodes of u. If the u is a leaf node,$TF_{u,d_i i}$is calculated as:

$$\text{TF}_{u,d_i i} = \frac{TF'_{doc \cdot di}}{\sqrt{\sum_{d_i \in D}(TF^1_{doc \cdot di})^2}}$$

where TF'$_{doc,d_i} = 1 + lnN_{doc,d_i}$

and in the search vector Q IDF$_{d_i}$ is calculated as:

$$IDF_{d_i} = \frac{IDF'_{d_i}}{\sqrt{\sum_{d_i \in D}(IDF^1_{doc \cdot di})^2}}$$

where IDF$_{w_i} = ln(1 + N/N_{w_i})$

Formally the node u can be represented as:

u = ID, D, P$_1, P_2, P_3, FID$

where ID denotes the identity of node u, $P_l$and$P_r$are pointers to left and right child of the node u. If the node u is a leaf node of the tree, FID stores the identity of the element and D denotes the vector consisting of normalized TF values of the keywords of 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:

$D[i] = max\{u.P_l \to D[i], u.P_r \to D[i]\}, i = 1, ..., m.$

D. System and Threat Models

This paper uses different entities in its system model: the data owner and cloud service provider. It also assumes that the data owner must be able to do operations like a multi-keyword search[27], [28] on the data stored on the cloud. The data owner has a collection of documents D = d1, d2, ..., dn, which he wants to store on cloud in encrypted form. But he also wants to search effectively on the encrypted data in order to use it efficiently. The cloud server is assumed to honestly and correctly execute instructions.

Algorithm 1: Pre-processing

Data: Document collection DOCS = { doc$_1, doc_2, doc_3, ..., doc_n$ with the identifiers FID = {FID—FID = 1,2,...,n .}

Result: The index tree T:

for *each document int* $f_{FID} inF$ **do**
  Construct a leaf node u for f_{FID}*, with u.ID = GenID(),*$u.P_1$ *= u.*P_2 *= u.*P_3 *= null, u.FID = FID and D[i] =* $TF_{\{}f_{FID}, d_i\}$ **for** *i = 1,.... m* **do**
    Insert u to CurrentNodeSet;
  end
  while *the number of nodes in CurrentNodesSet is larger than 1* **do**
    if *the number of nodes in CurrentNodesSet are of the form 3h* **then**
      for *nodes u', u" and u"' in CurrentNodesSet* **do**
        Generate a parent node u for u', u", u"' with u.ID = GenID(),$u.P_1$ *= u' ,* $u.P_2$ *= u" and* $u.P_3$ *= u"' u.FID = 0 and D[i] = max{u'.D[i], u".D[i], u"'D[i]} ;*
        Insert u to tempNodeSet;
      end
    else if *the number of nodes in CurrentNodesSet are of the form 3h+1: (h>0)* **then**
      for *nodes u', u", u"' of the former (3h-3) nodes in CurrentNodesSet* **do**
        Generate a parent node u for u', u", u"';
        Insert u to tempNodeSet;
      end
      Create a parent node for u1 for the (3h-2)th, (3h-1)th and (3h)th node and the create parent node u for u1, (3h+1)th and NULL node;
      Insert u to tempNodeSet;
    else
      for *nodes u', u", u"' of the former (3h-3) nodes in CurrentNodesSet* **do**
        Generate a parent node u for u', u", u"';
        Insert u to tempNodeSet;
      end
      Create a parent node for u1 for the (3h-2)th, (3h-1)th and (3h)th node and the create parent node u for u1, (3h+1)th and (3h+2)th node;
      Insert u to tempNodeSet;
    end
    Replace CurrentNodeSet with TempNodeSet and then clear TempNodeSet;
  end
end
return the only node left in CurrentNodeSet, i.e., the root index of tree T;

V. Proposed Algorithm for Multi Keyword Search Over Encrypted Data

In this section, we first describe the UDMRS Algorithm, which is built along with the basis of the KBB tree and vector space model. Founded on the UDMRS technique, two secure search algorithms(i.e., BDMRS and EDMRS Algorithm) are built against two threat models, respectively.

A. Index Construction of UDMRS Scheme

The conventional construction process of the index is presented in Algorithm 1. Following are some notations for Algorithm 1. Besides, the data structure of the tree node is defined as $< ID, D, P_l, P_r, FID >$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 $\in Z$),else the cardinality is denoted as $(2h + 1)$.

TempNodeSet - The set of the newly generated nodes.

1) Aearch Process of UDMRS Scheme: The search process of the UDMRS scheme is a recursive procedure upon the tree, named as the Greedy Depth-first Search algorithm. We construct a result list denoted as RList, whose element is defined as RScore; FIDi. Here, the RScore is the relevance score of the document f FID to the query, which is calculated according to Formula (1). The RList stores the k accessed documents with the largest relevancee 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. The following are some other notations, and the GDFS algorithm is described in Algorithm 2.

1. RScore(D_u, Q) - The function to calculate the relevance score for query vector Q and index vector $D_u$stored in node u, which is defined in Formula (1).

2. kthscore - The smallest relevance score in current RList, which is initialized as 0.

3. hchild - The child node of a tree node with higher relevance score.

4. mchild - The child node of a tree node with medium relevance score.

5. lchild - The child node of a tree node with lower relevance score.

Algorithm 2: GDFS

```
Data: Document collection DOCS = {
      doc_1, doc_2, doc_3, ..., doc_n with the identifiers FID
      = {FID—FID = 1,2,...,n
.}
if the node u is not a leaf node then
    if RScore (D_u, Q) > kth score then
        GDFS(u.hchild);
        GDFS(u.mchild);
        GDFS(u.lchild);
    else
        return;
    end
else
    if RScore (RScore(D_u, Q) > kth score then
        Delete the element with the smallest relevance
          score from RList;
        Insert a new element <RScore(D_u,Q), u.FID>
          and sort all the elements of RList;
    return;
end
```

2) Advanced Scheme: The advanced scheme is designed to achieve the goal of privacy-preserving in the known ciphertext model. The following are the algorithms involved along with the description:

1: SK $\leftarrow$ Setup(). Initially the data owner generates the secret key set SK, including - a randomly generated m bit vector S where m is equal to the cardinality of dictionary and two $m \times m$ invertible matrices $M_1 and$ M_2. Namely SK = {S, M_1, M_2}.

2: I $\leftarrow$ GenIndex(F, SK). First, the inecrypted index is built on F by using T $\leftarrow$BuildIndexTree(F). Second, the data owner generates two random vectors D'_u, D''_u for index vector D_u for index vector D_u in each node u, according to the secret vector S. Specifically, if S[i] = 0, D'_u and D''_uwill be set equal to D_u; if S[i] = 1, $D'_u[i]$ and $D''_u[i]$will be set as two random values whose sum equals to $D_u[i]$. Finally, the encrypted index tree I is built where the node u stores two encrypted index vectors $I_u = M_1 \times D'_u, M_2 \times D''_u$.

3 : $TD \leftarrow$ GenTrapdoor(W_q, SK) with keyword setW_q, the unencrypted query vector Q with length of m is generated. If $w_i \in W_q$, Q[i] stores the normalized IDF value of w_i; else Q[i] is set to 0. Similarly, the query vector Q is split into Q' and Q". The difference is that if S[i] = 0 then 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 same as Q[i]. Finally, the algorithm returns the trapdoor TD = { $M_1^{-1} \times$Q',$M_2^{-1} \times$Q" }.

4 : $RelevanceScore \leftarrow$ SRScore{I_u, TD}. With the trapdoor TD, the cloud server computes the relevance score of node u in the index tree I to the query.

VI. Results

Order of searching is $O(mlog_m n)$. The optimum value of 'm' to minimize the searching time has been derived from being 'e' (approximately 3).

A. Encryption

As we have used AES, which is an asymmetric cipher, the speed of the encryption scheme described above is higher as compared to only using asymmetric ciphers such as RSA or ECC. The security of the above scheme is as follows : AES 128 bit, RSA 128 bit, and SHA 128 bit. Thus, the minimum security level of these three combined will be 128 bit. Speed comparison:

m	N = 10^2	N = 10^3	N = 10^5	N = 10^7
2	13.28771238	19.93156857	33.21928095	46.50699333
3	12.57541965	18.86312947	31.43854911	44.01396876
4	13.28771238	19.93156857	33.21928095	46.50699333
5	14.30676558	21.46014837	35.76691395	50.07367953
6	15.42116651	23.13174976	38.55291627	53.97408278
7	16.56612527	24.84918791	41.41531319	57.98143846
8	17.71694984	26.57542476	44.2923746	62.00932444
9	18.86312947	28.2946942	47.15782367	66.02095314
10	20	30	50	70

Fig. 2. Graph of order vs m ways for different number of nodes

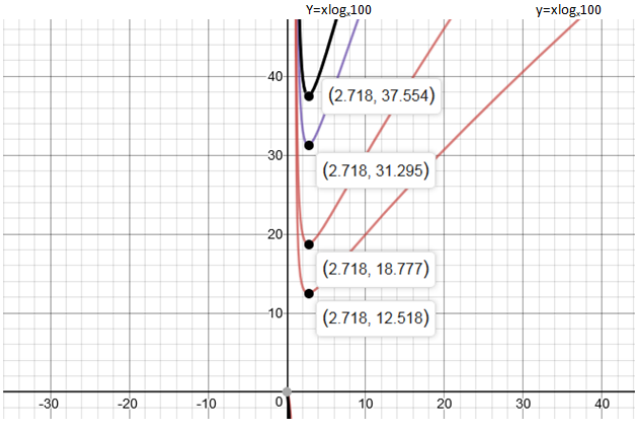

Fig. 3. Graph of order vs m ways for different number of nodes

VII. Conclusion

In this paper, we propose a secure tree-based multi-keyword ranked search scheme over encrypted data, and dynamic operation on the document collection. As more and more sensitive data is being uploaded on the cloud in the present scenario, the privacy and security concerns associated with the data is continuously increasing. The data is stored on the cloud in the encrypted form. Also, as the amount of data stored is usually very large, and efficient search scheme is also necessary. So here we deal with two major aspects of cloud computing: Encryption and Searching. We are proposing a secure and efficient encryption scheme to encrypt the data stored in the cloud as well as the queries along with a multi-keyword search scheme to search over the encrypted cloud data.

VIII. Acknowledgement

This research work is fully supported by ECRA from SERB, Department of Science Technology(DST), Govt. of India, New Delhi, India (Project Number: ECR/2015/000256/ES). All the authors have contributed equally for this work.

References

[1] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," IEEE Internet Comput., vol. 16, no. 1, pp. 6973, Jan-Feb. 2012.
[2] S. Kamara and K. Lauter, Cryptographic cloud storage, Proc. Financ. Cryptography Data Secur., 2010, pp. 136149.
[3] C. Gentry, A fully homomorphic encryption scheme, Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2009.
[4] O. Goldreich and R. Ostrovsky, Software protection and simulation on oblivious rams, J. ACM, vol. 43, no. 3, pp. 431473, 1996.
[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, Public key encryption with keyword search, in Proc. Adv. Cryptol.-Eurocrypt, 2004, pp. 506522.
[6] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III, Public key encryption that allows pir queries, in Proc. Adv. Cryptol., 2007, pp. 5067.
[7] D. X. Song, D. Wagner, and A. Perrig, Practical techniques for searches on encrypted data, in Proc. IEEE Symp. Secur. Privacy, 2000, pp. 4455.
[8] E.-J. Goh, Secure indexes, IACR Cryptol. ePrint Archive, vol. 2003, p. 216, 2003.
[9] Y.-C. Chang and M. Mitzenmacher, Privacy preserving keyword searches on remote encrypted data, in Proc. 3rd Int. Conf. Appl. Cryptography Netw. Secur., 2005, pp. 442455.
[10] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, Achieving usable and privacy-assured similarity search over outsourced cloud data, in Proc. IEEE INFOCOM, 2012, pp. 451459.
[11] B. Wang, S. Yu, W. Lou, and Y. T. Hou, Privacy-preserving multikeyword fuzzy search over encrypted data in the cloud, in Proc. IEEE INFOCOM, 2014, pp. 21122120.
[12] P. Golle, J. Staddon, and B. Waters, Secure conjunctive keyword search over encrypted data, in Proc. Appl. Cryptography Netw. Secur., 2004, pp. 3145.
[13] L. Ballard, S. Kamara, and F. Monrose, Achieving efficient conjunctive keyword searches over encrypted data, in Proc. 7th Int. Conf. Inf. Commun. Secur., 2005, pp. 414426.
[14] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, Achieving usable and privacy-assured similarity search over outsourced cloud data, in Proc. IEEE INFOCOM, 2012, pp. 451459.
[15] B. Zhang and F. Zhang, An efficient public key encryption with conjunctive-subset keywords search, J. Netw. Comput. Appl., vol. 34, no. 1, pp. 262267, 2011.
[16] J. Katz, A. Sahai, and B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, in Proc. Adv. Cryptol., 2008, pp. 146162.
[17] E. Shen, E. Shi, and B. Waters, Predicate privacy in encryption systems, in Proc. 6th Theory oCryptography Conf. Theory Cryptography., 2009, pp. 457473.
[18] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption, in Proc. 29th Annu. Int. Conf. Theory Appl. Cryptographic Tech., 2010, pp. 6291.
[19] 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. ACM Workshop Storage Security Survivability, 2007, pp. 712.
[20] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, Zerber+ r: Top-k retrieval from a confidential index, in Proc. 12th Int. Conf. Extending Database Technol.: Adv. Database Technol., 2009, pp. 439449.
[21] C. Wang, N. Cao, K. Ren, and W. Lou, Enabling secure and efficient ranked keyword search over outsourced cloud data, IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 8, pp. 14671479, Aug. 2012.
[22] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, in Proc. IEEE INFOCOM, Apr. 2011, pp. 829837.
[23] 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 Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. secur., 2013, pp. 7182.

[24] C. Orencik, M. Kantarcioglu, and E. Savas, A practical and secure multi-keyword search method over encrypted cloud data, in Proc. IEEE 6th Int. Conf. Cloud Comput., 2013, pp. 390397.
[25] 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 Syst. Networks (DSN), IEEE 44th Annu. IEEE/IFIP Int. Conf., 2014, pp. 276286.
[26] Xia, Z., Wang, X., Sun, X., Wang, Q. "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data", IEEE transactions on parallel and distributed systems, 27(2), 2016, pp. 340-352.
[27] Zhong, Hong, Zhanfei Li, Jie Cui, Yue Sun, and Lu Liu. "Efficient dynamic multi-keyword fuzzy search over encrypted cloud data." Journal of Network and Computer Applications 149 (2020): 102469.
[28] Yin, Hui, Zheng Qin, Jixin Zhang, Hua Deng, Fangmin Li, and Keqin Li. "A fine-grained authorized keyword secure search scheme with efficient search permission update in cloud computing." Journal of Parallel and Distributed Computing 135 (2020): 56-69.