*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. IV (Sep – Oct. 2014), PP 18-24*
*www.iosrjournals.org*

# Facilitating Efficient Encrypted Document Storage and Retrieval in a Cloud Framework

[1]Samantha Susan Mathew, [2]Hafsath C A
*[1]PG Scholar, Dept. of CSE,Ilahia College Of Engineering and Technology, Kerala, India*
*[2]Assistant Professor, Dept. of CSE,Ilahia College Of Engineering and Technology, Kerala, India*

***Abstract:****In the recent years, cloud computing has gained a lot of attention and big IT giants are providing cloudservices to the clients. As a part of this service user data are stored in remote server machines which are managed by the cloud service provider.This necessitates the encryption of user's sensitive information before outsourcing to the commercial public cloud. Huge amount of information present in the cloud calls for an efficient data retrieval mechanism.We are proposing a novel approach for keyword based search on the encrypted cloud data which facilitates easy retrieval of information from multiple encrypted data sources. Some of the existing approaches for the keyword based search focuses on Boolean search which will consider only the presence of the keyword in the file collection. Some other methods has server-side ranking wherethe encrypted relevance score may maintain some specific order which may lead to leakage of information. This paper uses latest information retrieval procedures along with vector space model and homomorphic encryption to ensure the search accuracy and secrecy while concealing the access pattern and search pattern. It provides a multi-user environment that supports storage of data belonging to multiple data owners andsearch request from multiple users. A salient feature of the suggested method is it allows efficient update operation of the file collection by the data owners. This process avoids the overhead of uploading/calculating the index scores for each of the existing documents from scratch for any file collection change.*

***Index Terms:****Homomorphicencryption,Vector-space model,Secure searchable index*

## I. Introduction

The advent of cloud computing has helped the companies to reduce their huge spending on IT infrastructure cost which in turn helped it to focus on strategic projects. In cloud computing, the information is accessed from a centralized storage and does not need any user to be in a specific place for information retrieval. Now a day, more and more sensitive information are being centralized into the cloud, such as e-mails, personal health records, banking information, company finance data, government documents, etc. By storing their data into the cloud, the data owners can be relieved fromthe burden of data storage and maintenance so as to enjoy the on-demandhigh quality data storage service. Since the data owner, data users and the cloud server doesn't belong to the same trusted domain ensuring the privacy and security of the unencrypted outsourced data is a problem. There may be some risk that the cloud server may leak data information to unauthorized entities or sometimes even the cloud server might be hacked. Therefore, to safeguard the data privacy and prevent unauthorized accesses, the sensitive data is encrypted before outsourcing. The encryption provides data security at the cost of ease of use. In cloud computing, data owners may share huge volume of data with a number of users, who actually might be interested to retrieve only certain specific data files during a given session.

Keyword based retrieval is an efficient way to retrieve files of some particular interest from a large data collection and has been widely used in plaintext search scenarios. It reduces large amount of unwanted data transfer which in turn results in less communication cost. But the data present in the cloud is in encrypted format and performing search operation on it is a challenging issue. In the context of cloud computing, the main issue to be addressedis efficientsearching and retrieval of data in a most secure and privacy preserving manner. This paper proposes an efficient mechanism to search on the encrypted data which is outsourced to the cloud server. Searching can be based on a single keyword or multiple keywords and number of results can be controlled by the requesting user.

The concept of similarity relevance, vector space model and homomorphic encryption are used for making the searching efficient and fast. The server is having high computational capability than the client and hence the score calculation is performed on the server side which involves huge computation. In order to avoid ranking information leakage the ranking is performed on the user side.

An effective mechanism for updating the document set by the data owner is suggested in this paper. In the existing mechanism when a data owner needs to add or remove a document from the collection he needs the existing documents in hand which has to be uploaded again for the score value updation. Through secure access of the index structure and the existing encrypted score values the new score values for the updated data collection can be performed. This will avoid unnecessary transfer of the data files to the cloud server which in turn reduces the communication cost.

## II. Relatedworks

To apply the searchable encryption to cloud computing, some researchers have been studying on how to search over encrypted cloud data securely and efficiently.

Searchable encryption is a new developing information security technique which can enable users search over encrypted data through keywords without decrypting it. Searchable encryption can be divided into two categories:symmetric-key and public-key version.Song et al. [1] is the first one who proposes the searchable encryption scheme in the symmetric setting. They proposed different techniques for searching in the encrypted data while ensuring the data confidentiality. This approach has high computational complexity in search.In a real time cloud environment this construction would not provide accurate result, since they are developed as crypto primitives. Here searches across encrypted keywordswithin a file with an overhead linear to the file size.

D. Boneh, proposed a method [2] known as Public Key Encryption with keyword Search (PEKS), which deals with the problem of searching in encrypted data by using a public key version or asymmetric version of searchable encryption. The owner of the email encrypts the email and specifies few keywords that the email gateway that the mail gateway can search for which are encrypted by PEKS which is similar to the Identity based encryption. The advantages are Gateway should learn nothing else about the encrypted email and hencethe user privacy will not be violated. The users who wants to search provides a trapdoor which will encrypted again with the PEKS and a test function will be performed by the gateway to find out a match. Disadvantages of this method are the sender of the mail needs to explicitly mention what the keywords are and the keywords may not be relevant to the message at all. If there is a system that contains n documents each of which contains utmost s keywords then one keyword search compares utmost ns times. It supports only Boolean keyword search and only few number of keywords are supported.

In [3] a method for securable searching in encrypted data is proposed. In the traditional keyword search, a Boolean search is performed which identifies the presence or absence of a particular keyword. Here a ranking technique is proposed by the author which is based on Order Preserving Mapping in order to protect sensitive score information. The order preserving mapping flattens the original relevance score distribution and increase its randomness but still preserve the plaintext order. The set of data is securely transformed so that the order is preserved but the distribution and domain are changed. This method is highly encrypted and provides better data relevance based on term frequency. But it supports only single keyword based top-k retrieval.

To provide ranked search functionality on encrypted data, Cao et al. [4] proposed a privacy-preserving multi-keyword ranked search scheme. The concept of co-ordinate matching is used where search result is ranked according to the number of matched keywords, which is not accurate enough.Inner product similarity and co-ordinate matching is used for obtaining the similarity between the documents and query. It supports multi-keyword search but ranking is based only on the number of retrieved keywords. The search complexity is linear with the number of documents in dataset.

A method for searching and retrieving Top-k documents of encrypted cloud data using a Confidential Index is proposed [5]. It is construct a ranking model which allows top-k retrieval from confidential outsources inverted index without information leakage and preserve the privacy of encrypted data. The relevance score of a term q in a document d is calculated as rscore (q, d) $=\frac{\mathrm{TFq}}{|d|}$where TFq is number of times a particular keyword q appears in document d and |d| is the document length. To improve the security of the indexed data, a relevance score transformation function(RSTF) is used to make the relevance scores of different terms indistinguishable.The client may send a follow up request if desired number of elements is not received. Here the interdependency between the files is not considered.

A framework for rank-ordered search and retrieval over large document collection is proposed in [6].It handles two different scenarios individually- searching of the document by the content owner and the searching in the data center by people other than the content owners. A confidentiality-preserving baseline model handles the search request from the content owner. The system assumes that a secure computing unit is present in the data center which is trusted by the content owner. It has an inner layer of encryption which is an order preserving encryption which will be applicable to the users other than the content owners and an outer layer of encryption which is applicable for everyone using the system. This encryption is used to encrypt the Term Frequency (TF) values. The Inner Layer Encryption performs computations and ranking directly on term frequency data in its encrypted form which is an order preserving encryption. But uniform distribution of posting elements alone does not hide the document frequency and thus allows an adversary to recover encrypted terms. During searching by a non-content owner performs stemming and sends the stemmed words to the content owner. The content owner checks whether the requester has the required permissions, derives the word-keys from the master key and encrypts the stemmed words, calculates the hash value gives to requester who forwards it to data center. The data center searches the TF table and identify the target row in SCU (Secure

Computing Uni) encrypts and decodes it to obtain the inner layer encryption protected TF value. The SCU performs the computation to find the relevance score from the encrypted domain, rank orders the document and sends back the most relevant document identifiers with their ranking. The security issues like protecting the communication link and combating traffic analysis are not addressed here. Also it does not support efficient index inserts and updates such that, at least in some cases, the posting list has to be completely rebuilt.

In [7] a holistic and efficient solution that comprises a secure traversal framework and an encryption scheme based on privacy homomorphism is proposed. This method is scalable to large datasets and efficiency of the query processing is high. But it does not support for all query types such as top- queries, skyline queries and multi-way joins. It tries to solve the problem of top-k multi-keyword search over the encrypted data. But it faces two issues the Boolean representation where the files are ranked based on number of retrieved keywords and maintaining the balance between security and efficiency.

The method proposed in [8] also supports Boolean based keyword retrieval. When the user wants to search for a particular document he provides a capability for the word W and provides to the server which will identify the documents that satisfy the user's request. Here the capability is generated based on the fields and hence mainly applicable for searching in the encrypted emails.

Searchable symmetric encryption (SSE) [9] is a cryptographic technique that allows a user to outsource the storage of its data to a server in a confidential way, while maintaining the ability for keyword-based search over it. While most of the system is based on a single user setting [9] describes a procedure that can be applied on a multiuser environment. It constructs schemes provably secure against non-adaptiveand adaptive adversaries. The method is not dynamically scalable.

Most of the existing system considers only the term frequency for the ranking purpose which is not the accurate one. The server side ranking is supported by many of the systems which will leak additional information.

## III. Proposed system

The key entities involved in a cloud computing system which provides data services are data owner, data user and the cloud server. The data owner has a collection of file which will be outsourced into the cloud storage. Inorder to ensure the confidentiality of the file collection the cloud server stores the same in encrypted manner. The data user can be any authorized entity who can access the data present in the cloud. The suggested method ensures that a multi-keyword search by any data user on the data uploaded by multiple dataowners will return a set of relevant documents whose limit can be set by the requester.

### 3.1 Data Owner

Multiple data owners will be present in a cloud computing scenario where each one will have a collection of files to outsource. Consider a data owner having a file collection C= $\{f_1,f_2,\ldots f_n\}$. To handle the search request efficiently and quickly, an index of the file is maintained by the data owner. For creating the index, a set of keywords are extracted from the file collection. For limiting the index size stop words and the words that appear less frequently in the document are removed. Stemming is applied on these words set to further reduce the index size. Let W= $\{w_1,w_2\ldots w_l\}$ be the final word list thus generated.

#### 3.1.1 Index Building

The searchable index is created by calculating the scores of each keyword,w in the above generated wordlist. Scoring is a way to measure the relevance. We use the tf-idf weighing mechanism for scoring in our approach. This scheme considers two attributes – the term frequency (tf) and inverse document frequency (idf). Term frequency denotes the number of occurrence of a word w in a file f. Since every document is different in length, it is possible that a word would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length N. Thus TF (w) = (Number of times word w appears in a document) / (Total number of words in the document). Inverse document frequency measures how important a word which weigh down the frequent terms while scale up the rare ones. IDF (w) = log (Total number of documents / document frequency). Document frequency is the number of files that contains the particular word w. Then the tf-idf weighing scheme assigns to word w in the wordlist W, a weight in file f as $\text{tf-idf}_{w,f}= tf_{t,f}*idf_w$. This weight depicts the relevance pertaining to the particular keyword in the file collection.

To represent the score of the files on multi-keyword a vector space model is used. It is an algebraic model for representing the text documents as vectors. It is having many dimension corresponds to each word in the word list. If a word occurs in the document, its value in the vector is its tf-idf value otherwise it will be zero.So for each file fi the data owner builds a (l+1) dimensional vector $v_i=\{id_i,t_{i,1},t_{i,2},\ldots t_{i,l}\}$ where l is the length of the word list ,$id_i$ is the identifier of the file and $t_{i,j}=\text{tf-idf}_{wj},f_i$. The searchable index $I=\{v_{i\ 1}i\leq i \leq l)$.

We make the searchable index secure by encrypting the same. A special type of encryption called homomorphic encryption is used to create the secure searchable index. Homomorphic encryption is a form of encryption which allows specific types of computations to be carried out on cipher text and generate an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. In the setup phase the data owner generates a secret key SK and a set of public keys PK for the homomorphic encryption scheme. These will be shared with authorized data users. The secure searchable index is created by encrypted each value in the searchable index by any of the key from the public key set i.e.$I'=\{v'_i|1 \leq i \leq n\}$, where $v'_i = \{id'_i, t'_{i,1}, t'_{i,2}, \ldots, t'_{i,l}\}$ , $id'_i = Encrypt(R_{i,0}, id_i)$ and $t_{i,j}' = Encrypt(R_{i,j}, t_{i,j})$ ($R_{i,0} \in PK, Ri,j \in PK, 1 \leq j \leq l$). Since different keys from the public key set are used for the generation of the encrypted score values access pattern and search pattern analysis by attackers can be prevented.

### 3.1.2 Data Outsourcing

Once the secure index is generated, the file collection will be encrypted by the data owner. Any symmetric encryption schemes like AES (Advanced Encryption Standard) can be used for this purpose. The encrypted file collection and the secure index will be outsourced to the cloud server.

### 3.1.3 Data Modification

In real world cloud computing scenario, frequent changes in the file collection is common. For such modifications a reliable and efficient mechanism without any burden in the user side is desirable. The system provides a facility to the data owner to add or delete documents to his existing file collection. The index will be updated automatically without any manual intervention. Since the data owner generates private key and public key, he can retrieve /decrypt the index structure stored in the cloud. The additional words identified from the new file collection are added to the retrieved index and the tf-idf values are computed/recalculated using the tf and idf values obtained from the retrieved index structure. This process avoids the overhead of uploading/calculating the index scores for each of the existing documents from scratch for any file collection change. In the case of deletion of a document from the document collection, the secure searchable index is decrypted by the data owner, finds out the words in the word list that corresponds to the file has to be deleted and updates the index structure accordingly.

Figure1 explains the actions performed by the data owner.

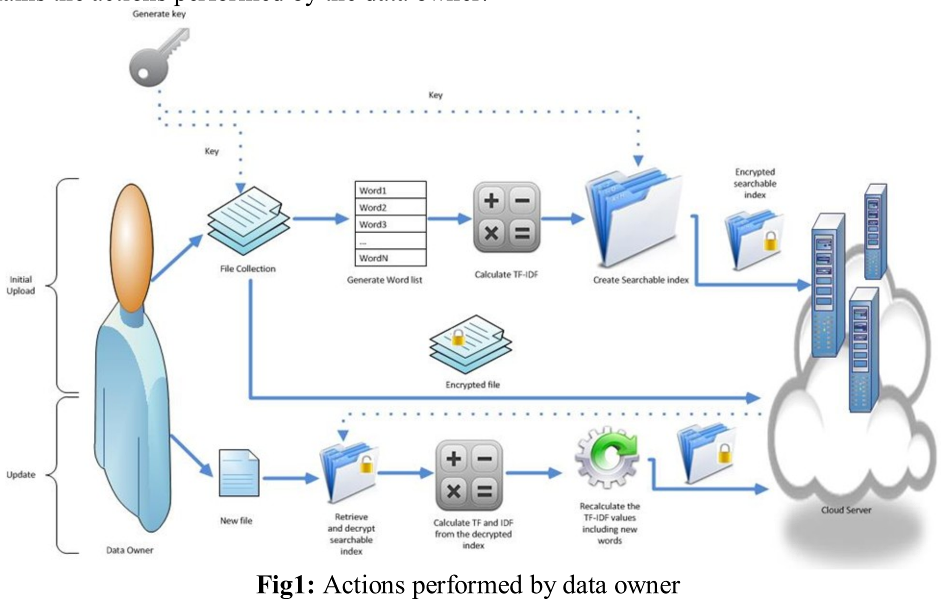

**Fig1:** Actions performed by data owner

### 3.2 Data User

The data user can perform multi-keyword search on the encrypted data. The documents based on relevance will be sending back to the user. The ranking is performed on the user side which will prevent inevitable leakage of sensitive information due to server side ranking.

### 3.2.1 Request Generation

When a user performs a multiple keyword search, a query vector is constructed from it using the concept of vector space model. The word list is transferred securely to the user side and a query vector is generated by inserting either 1 or 0 based on the presence of the word in the users search request. This request is encrypted using the public keys provided by the data owner. For a search requestREQ ={w$_1$', w$_2$', . . . , w$_s$'} , the query vector $T_\omega$= {m$_1$,m$_2$, . . .,m$_l$} is generated where $m_i$=1 ($1 \le i \le l$) if $t_i \in$ REQelse it will be 0. The trapdoor is generated by encrypting it as $T_{\acute{\omega}}$ = {c$_1$, c$_2$, . . . , c$_l$}, where $c_i$ =Encrypt(R,m) and R$\in$PK. The user sends the trapdoor to the server.

### 3.2.2 Post processing of Response

The reply from the cloud server is the result vector$\aleph$'={(id$_1$',p$_1$'),( id$_2$',p$_2$'),….(id$_n$',p$_n$'))} where id$_i$' is the identifier of the file i and p$_i$' is its score relevant to the query vector both in encrypted format. Using the private key provided by the data owner the data user decrypts it to get the score of each file. The ranking is performed in the user side by sorting the files in the descending order of their scores. The user can decide the number of files required and suppose it is k, he sends the identifiers of the top k files to the cloud server. The server sends back the encrypted files to the user which can be decrypted and used by the user.

Figure2 explains the actions performed by the data user.

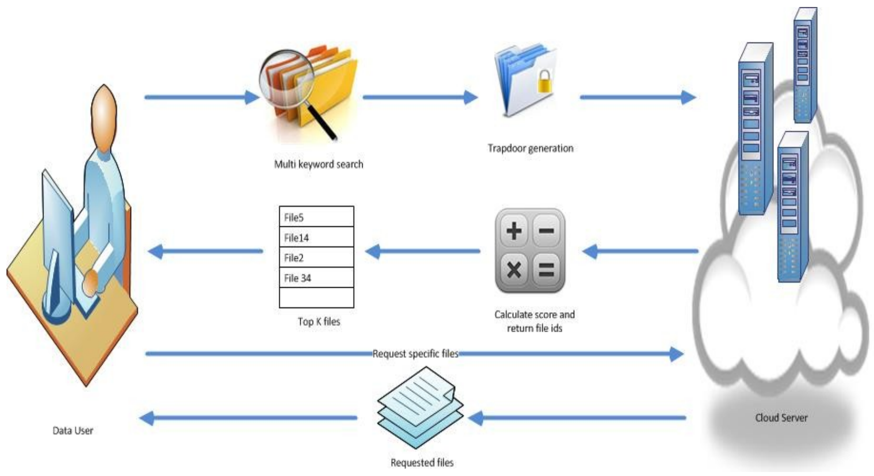

**Fig2:** Actions performed by data user

### 3.3 Cloud Server

The data user and owner are in direct contact with the cloud server for satisfying their request. The encrypted tf-idf values and data files from the data owner are stored in the cloud server. The server is having high computational capability than the client. On receiving the request vector from the authorized data user, the cloud server will perform an inner product operation using the request vector and encrypted tf-idf values to obtain the relevant score. These operations will be performed with the index of all data owners that has provided access to the particular data user. Since the operation is performed on the encrypted data and the result obtained is also encrypted, no information is leaked to the server. This is possible because of the property of homomorphic encryption which allows the computation to be performed on the encrypted data and the decrypted value of the result will be same as what would obtain if the same operation is applied on the plain text. These score values will be sending back to the user. Upon receiving the identifiers of the top-k files from the user after ranking in user side, the cloud server will provide those files in encrypted format to client.

## IV. Solution methodology

The homomorphic encryption has been used for making the index secure[10]. This encryption can be well described as four stages: KeyGen, Encrypt, Evaluate, and Decrypt.

KeyGen($\lambda$): The secret key and the set of public key are generated here.The secret key SK is an odd ŋ-bit number from interval $[2^{ŋ-1},2^{ŋ})$.The set of public keys PK=$\{k_0,k_1,\ldots k_Ţ\} \subseteq \{pq + xr,\ q \in [0, 2^{\gamma}/p), r \in (-2^{ß},2^{ß})\}$ where ŋ-bit-length of the secret key , $\eta = \Theta(\lambda^2)$, Ţ -number of integers in the public key, p-secret key, q- a multiple parameter, r-noise to achieve proximity against brute force attack. The noise parameter x is considered as $x = 2^{2||m||}$ where $||m||$ is the bit length of the cipher text.
Encrypt(PK,m): The cipher text is generated using the public key and the plain text, c=pq+xr+m
Evaluate(c1,c2,…ct): Apply the addition and multiplication in the input cipher text and return the resulting integer X.
Decrypt(SK,X): It will provide the output m'=(X mod SK) mod x.

The overall process involved in the multi-keyword based search can be divided into an initialization phase, updation phase and retrieval phase.

Initialization Phase:

1. Using the KeyGen(λ) the data owner generates the secret key and the public key setand provides the same to authorized users.
2. From the collection of files the data owner extracts the set of keywords say l, stem it and finds the TF-IDF values of those words. For each file an (l+1) dimensional vector is generated using the calculated TF-IDF values to form the searchable index
3. The searchable index is encrypted to secure searchable index using the public key set.
4. The collection of files in encrypted format and the secure searchable index is uploaded into the cloud server.

Updation Phase:

1. When an existing data owner wants to add or delete documents in his file set, extract the encrypted tf-idf values from the cloud server
2. Decrypt with the private key which is owned by him.
3. Find out the old tf and id values separately from the index
4. Calculate the tf value for the new documents and the new idf values.
5. Create new searchable index using the above generated values, encrypt it and upload to cloud.

Retrieval Phase:

1. The data users generate a set of keywords from which the query vector is generated by putting 1 if the term in the word list is present in the request otherwise zero.
2. The query vector is encrypted by the public key set to generate the trapdoor which is send to the cloud server.
3. An inner product of the query vector and secure searchable index is calculated to form the result vector containing the id of the files and the encrypted score value which is send to the user.
4. The user decrypts it using the secrete key and ranking is performed and the top-k highest scoring file identifiers are send to the cloud server.
5. The cloud server returns the encrypted k files back to the data user.

## V. Conclusion

This paper solves the problem of multi-keyword search over encrypted data in a secure manner. It encompasses a homomorphic encryption technique to ensure the secrecy and confidentiality. The score calculation is performed at the server side which is having high computational capability. Only the ranking performed at the user side and it prevents a type of information leakage. The set of public keys ensures no leakage in the access and search pattern. Even if it is a two round scheme, the security and privacy is guaranteed. The update operation can also be performed with minimum overhead on the data owner side. The size of the cipher text as well as the key size is high in the current implementation. The homomorphic encryption is a growing branch in the cryptography community. Now researchers in cryptography community are making attempts for more practical full homomorphic encryption. A development in this area will further improve the efficiency of the proposed method.

## Acknowledgment

The authors wish thanks the Management and Principal and Head Of the Department (CSE) of Ilahia College of Engineering and Technology for their support and help in completing this work

## References

[1]. D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," Proc. IEEE Symp. Security and Privacy, 2000.

[2]. D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, "Public- Key Encryption with Keyword Search," Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (Eurocrypt), 2004

[3]. C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," Proc. IEEE 30th Int'lConf. Distributed Computing Systems (ICDCS), 2010.

[4]. N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multikeyword Ranked Search over Encrypted Cloud Data," Proc.IEEE INFOCOM, 2011.

[5]. S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-kRetrieval from a Confidential Index," Proc. 12th Int'l Conf.Extending Database Technology: Advances in Database Technology(EDBT), 2009.

[6]. A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A.L. Varna, S. He, M.Wu, and D.W. Oard, "Confidentiality-Preserving Rank-OrderedSearch," Proc. Workshop Storage Security and Survivability, 2007.

[7]. H. Hu, J. Xu, C. Ren, and B. Choi, "Processing Private Queries over Untrusted Data Cloud through Privacy Homomorphism,"Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE), 2011.

[8]. P. Golle, J. Staddon, and B. Waters, "Secure Conjunctive KeywordSearch over Encrypted Data," Proc. Second Int'l Conf. AppliedCryptography and Network Security (ACNS), pp. 31-45, 2004.

[9]. R. Curtmola, J.A. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," Proc. ACM 13th Conf. Computer and Comm. Security (CCS), 2006.

[10]. M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "FullyHomomorphic Encryption over the Integers," Proc. 29th Ann. Int'lConf. Theory and Applications of Cryptographic Techniques, H. Gilbert,pp. 24-43, 2010.

[11]. Toward Secure Multikeyword Top-k Retrieval overEncrypted Cloud Data. Jiadi Yu, Member, IEEE, PengLu, Yanmin Zhu, Member, IEEE, GuangtaoXue,Member, IEEE Computer Society, and Minglu Li.