Scientific Journal of Impact Factor (SJIF): 5.71

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

International Journal of Advance Engineering and Research Development

Volume 5, Issue 04, April -2018

Phrase Search Using Bloom Filter for Encrypted Cloud Storage

[1]Mr.K.Vijiyakumar[2]K.Bakkiyalakshmi[3]J.Kirthana[4]R.Swathi

[1]Professor and [2 3 4]Student, Department of Information Technology,
Manakula Vinayagar Institute of Technology, Puducherry, India

Abstract - *The advent of cloud technologies promises greater scalability and accessibility, but it also highlights the need for greater privacy and security. Many researchers investigated solutions to search over encrypted documents stored on remote cloud servers. In this paper, we introduced Bloom filters technique for a phrase search scheme that is significantly faster than existing solutions, with similar or better storage and communication cost. Our technique uses a series of n-gram filters to support the functionality solution makes use of symmetric encryption, which provides computational and storage efficiency over many existing solutions.*

Keywords *– Bloom filter, Phrase search, Privacy, Security, Symmetric Encryption, Storage cost.*

I. INTRODUCTION

Cloud computing was the long dreamed vision of computing as a utility, where cloud customers can able to their data remotely into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources and providing required services to the users with the "pay only for use" approach where the users pay only for the number of service units they consume. As organizations and individuals adopt cloud technologies, many have become aware of the serious concerns regarding security and privacy of accessing personal and confidential information over the internet. In particular, the recent times data breaches show up the need for more secure cloud storage systems. While it is generally agreed that encryption is necessary, cloud providers often perform the encryption and maintain the private keys instead of the data owners. That is, the cloud can read any data it desired, providing no privacy to its users. The storage of private keys and encrypted data by the cloud provider is also problematic in case of data breach. Hence, researchers have actively been exploring solutions for secure storage on private and public clouds where private keys remain in the hands of data owners. A computing Cloud is a set of network enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing infrastructures on demand, which could be accessed in a simple and pervasive way. Cloud computing is able to be defined as the provision of computing services via the internet such as Applications (software-as-a-service, or SaaS), Platforms, Infrastructure (IaaS), Process orchestration and integration. Cloud computing led to an inventive approach in the way in which IT infrastructures, applications, and services are designed, developed, and delivered. This vision opens new opportunities that significantly change the relationship that enterprises, academia, and end-users have with software and technology. Cloud computing provides an on-demand model for IT resource provisioning where a resource can be a virtual server, a services, or an application platform.

In this paper we described a symmetric scheme for encryption. The shortcomings of the many existing solutions. We now present a symmetric key based scheme for creating encrypted and searching process. Our scheme is based on the bloom filter technique using phrase search that was significantly faster than existing solutions, with similar or better storage and communication cost.

Waters et al. [2] investigated the problem for searching over encrypted audit logs. Many of the early works focused on single keyword searches. Recently, researchers have proposed solutions on conjunctive keyword search, which involves multiple keywords [4]. Other interesting problems, such as the ranking of search results [5], and searching with keywords that might contain errors [8] termed fuzzy keyword search, had also been considered. The ability to search for phrases was also recently investigated [10]. Some [14] have examined the defense of the proposed solutions and, where flaws were found, solutions were proposed [15].

In this paper, we proposed a phrase search scheme which can achieves much faster response time than existing solutions. The scheme is also very scalable, and the documents can simply be removed and added to the corpus. We also portray some modifications to the existing scheme to provide lower storage cost, faster response time and to protect against cloud providers with statistical knowledge on stored data. We begin by presenting the backgrounds in section 2 and including related works in section 3. Although phrase search scheme was processed autonomously using our technique, they are typically a specialized function in a keyword search scheme, where the primary function is to provide conjunctive keyword searches.

Therefore, we describe the bloom filter technique in section 4 and also we describe both the basic conjunctive keyword search algorithm and the basic phrase search algorithm in section 5.

II. BACKGROUND

Song et al. [11] addresses the problem of searching on encrypted data in a symmetric-key situation. In a symmetric key based scheme, the keys that are used to create the encrypted entries also allow search and decryption of the audit log. Thus, servers that construct audit log entries possess keys capable of decrypting log entries. Tang [11] addresses the privacy concerns by proposing a solution with provable security using normalization. The technique uses an index table that allows for verifications of chains of keywords instead of having the word locations stored. However, the index table requires significant storage, which hinders its practicality. The key difference between conjunctive keyword search and phrase search is that, in addition to containing the requested keywords, they must also appear contiguously in the specified order in the document. In another word, we must have some knowledge on the locations of the keywords. In [10], all keyword locations are stored alongside encrypted keywords in an index. Waters [2] proposed schemes for searching over encrypted audit logs. Although early works focused on single keyword searches, more recent works have focused on conjunctive keyword searches involving multiple keywords [3], [4]. In addition, some considered the problem of ranking of search results [5], [6]. Other researchers provided solutions for searching with keywords that might contain errors [8], [9], termed fuzzy keyword search. Boneh et al. [2] have also recently examined the problem of searching on publicly encrypted data. They independently devised a scheme based on the Identity-Based Encryption scheme of Boneh and Franklin [3]. Their scheme is similar to our underlying cryptographic scheme in its construction and security properties. Ding et al. [3] extended Boneh et al.'s scheme using bilinear mapping to perform multiple keyword search and described a solution that did not include expensive pairing operations in the encryption and trapdoor generation phase. Kerschbaum et al. [4] considered the search of unstructured text, where positions of keywords are unknown. The use of encrypted index for keyword search was examined in [22] and a scheme secure against chosen keyword attack was proposed. The ranking of search results was looked at by Wang et al. in [17]. Liu et al. [23] considered the search for potentially erroneous keywords termed fuzzy keyword search. The index-based solution makes use of fuzzy dictionaries containing various misspelling of keywords including wildcards. Solutions for searching for phrases over encrypted data were only recently proposed by researchers. The main difference between conjunctive keyword search and phrase search was that the queried keywords must appear contiguously in the specified order in addition to all being present in the document. Zittrower et al. [10] were the first to investigate the problem. His solution uses a keyword-to-document index and a keyword location index. The keyword-to-document index provides the mapping of keywords to the documents which contain them while the location index contains the position of the keywords within each document. In [13], Poon et al. proposed a phrase search scheme that achieved further reduction in storage cost. The technique exploits the space-efficiency of Bloom filters to perform conjunctive keyword search and phrase search. Similar to other techniques, a set of keyword to document Bloom filters and a set of keyword location filters are used with symmetric key encryption.

III. MODEL FOR KEYWORD SEARCH OVER ENCRYPTED DATA

The communication model for a keyword search protocol generally involves up to three parties: The data owner, the cloud server and the user. In a private cloud, the user is simply the data owner. For most of our discussions, we will be considering the public cloud scenario involving three different parties. A typical protocol is illustrated in figure 1 where the user initiates the search request by passing the keywords to the data owner. The data owner then sends a trapdoor to the cloud to initiate a protocol to search over the requested keywords. Finally, the cloud responds to the user with the indexes to the requested documents. The cloud server usually wields significant computational power compared to the data owner and users. Therefore, it is desirable that the computational and storage cost be asymmetrically placed on the cloud.

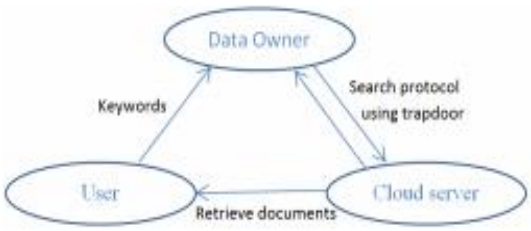

Figure 1. Communication model for keyword search over encrypted data

A. SECURITY

The cloud server contains the encrypted documents, $E_{KDi}(D_i)$, the keyword index, $I(E_K(kw_j)) = \{D_a, D_b,...,D_y\}$ and the keyword location indexes, $\{H(D_i|kw_i), j_1, j_2 ...j_n\}$. The security of the encrypted documents at rest is equivalent to that of the symmetric encryption E() and cryptographic hash function, H(). However, the keyword and location indexes are particularly susceptible to statistical attacks, since certain words are more common than others in every natural language. The location of the keywords may also reveal information to an adversary with partial knowledge [10]. To alleviate the problem, Tang [11] opted to encrypt the single word index and have the keyword chain index normalized by filling in random data so that every entry contains the same number of elements as the largest entry prior to normalization. Encrypting the single word index is sufficient to provide provable security for hiding single word statistics. However, hiding the statistical properties of the keyword chain index used to provide phrase search capability came at a high cost in storage. It is known that most natural languages roughly follows Zipf's law, which states the word frequency is inversely proportional to its rank in the frequency table [17]. By filling in entries up to the maximum occurrence of a word in the entire document set, the resulting index comprises almost entirely of random data. Figure 2 shows the distribution of a particular research paper with approximately 6000 words, which follows roughly the distribution of documents of English language. We'll refer to this example for the remaining of this section to explain the design of our scheme. If all words in the sample document are indexed, the most common word is 'the', with 445 instances, ≈ 7% of all words. It was found that the ten most common words comprise 25% of all words in the document. In fact, almost 1200 of the 1300 distinct words occurred less than 10 times in the document. Using Tang's scheme, the keyword chain table for this document would 510. Increasing distinct word distribution of a sample document contains 446*1300 entries where over 436*1200 contains random data. Thus, less than 10% is actually used for searching. If the most common English words are excluded, we would have 1066 distinct words indexed and the 20 most common words comprising 25% of all words indexed. The most common word occurring 84 times and over 1000 words occurring less than 10 times, resulting in less than 26% of the data useful for search. During query, one must also account for the statistical property of search terms, that was users may be more likely to perform a search for a subset of keywords.

IV. BLOOM FILTER

The Bloom filter is a space-efficient probabilistic data structure that supports set membership queries. The data structure was conceived by Burton H. Bloom in 1970. Bloom filters are space-efficient probabilistic data structure used to test whether an element is a member of a set. A Bloom filter contains m bits, where k hash functions, $H_i(x)$, are used to map elements to the m-bits in the filter. The Bloom filter is initially set to all zeros. To add an element, a, to the filter, we compute $H_i(a)$ for i = 1 to k, and set the corresponding positions in the filter.

The structure offers a compact probabilistic way to represent a set that can result in false positives, but never in false negatives. This makes Bloom filters useful for many different kinds of tasks that involve lists and sets. The fundamental operations involved in adding elements to the set and querying for element membership in the probabilistic set representation. The accuracy of a Bloom filter depends on the size of the filter, the number of hash functions used in the filter, and the number of elements added to the set. The more elements were added to a Bloom filter, the higher probability that the query operation reports false positives. A Bloom filter was an array of m bits for a set $S = \{x_1, x_2, \ldots, xn\}$ of n elements. Initially all the bits in the 2 filter are set to zero.

The key idea was to use k hash functions is $hi(x)$, $1 \leq i \leq k$ to map the items $x \in S$ are random numbers uniform in the range 1,...m. The hash functions were assumed to be uniform. The MD5 hash algorithm is a popular choice for the hash functions. An element $x \in S$ was inserted into the filter by setting the bits $hi(x)$ to one for $1 \leq i \leq k$. Algorithm 1 presents the pseudocode for the insertion operation. The weak point of Bloom filters is the possibility for a false positive. False positives are elements that were not part of S but are reported being in the set by the filter size.

Algorithm 1: Pseudocode for Bloom filter insertion

Data: x was the object key to insert into the Bloom filter.

Function: insert(x)

for j : 1 . . . k do

/* Loop all hash functions k */

i ← hj(x);

if Bi == 0 then

/* Bloom filter had zero bit at position

i */

Bi ← 1;

end

end

Algorithm 2: Pseudocode for Bloom member test

Data: x was the object key for which membership was tested.
Function: ismember(x) returns true or false to the membership test
m ← 1;
j ← 1;
while m == 1 and j ≤ k do
i ← h_j (x);
if Bi == 0 then
m ← 0;
end
j ← j + 1;
end
return m;

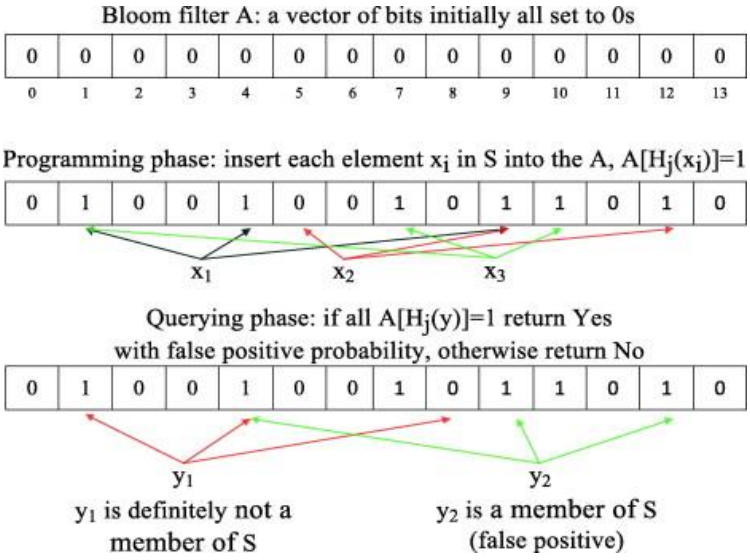

Figure.2: Process of Bloom Filter

It presents an overview of a Bloom filter. The Bloom filter consists of a bitstring of length 32. Three elements had been inserted, namely x, y, and z. Each of the elements had been hashed using k = 3 *hash* functions to bit positions in the bitstring. The corresponding bits had been set to 1. We observe that for the inserted elements, the hashed positions correctly report that the bit was set in the bit string.A practical example of a Bloom filter through adding and querying elements. In this example, While Bloom filters have no false-negatives, it can falsely identify an element as member of a set. Given k hash functions, n items inserted and m bits used in the filter, the probability of false positives is approximately p = $(1 - e^{-kn/m})$ k and minimum false positive rate is achieved when k = m/n ln2.

V. PHRASE SEARCH SCHEME BASED ON BLOOM FILTERS

In a keyword search scheme, Bloom filters can be used to test whether a keyword was associated with a document. Many existing phrase search schemes [10], [11] use a keyword-to-document index and a location/chain index to map keywords to documents and match phrases. We describe another method using Bloom filters to support this functionality with an emphasis on response time. Our scheme can be summarized as the use of multiple n-gram Bloom filters, B^n_{Di} , to provide conjunctive keyword search and phrase search.

A. Conjunctive keyword search protocol

To provide conjunctive keyword search capability, each document, Di , was parsed for a list of keywords kw_j . A Bloom filter of size m was initialized to zeros. Each keyword was hashed using a secret key to produce H_{kc} (kw_j) and passed into k Bloom filter hash functions to set k bits in the Bloom filter. This results in a 1-gram Bloom filter for each document: B_1D_i = {b_1, b_2, ...b_m} where b_i∈ {0, 1}. The document collection, D = {D_1, D_2, . . . ,D_n}, was encrypted and uploaded along with the Bloom filters to the cloud server. The Bloom filters are then prepared into a matrix with the first row containing the filter B_1D_1 for

the first document and the last row containing B_1D_N . Its transpose was stored as a Bloom filter index IBF where each row corresponds to a bit in the Bloom filters. Note that the ith row in IBF contains information on which document's filter has its ith bit set. This arrangement allows us to quickly identify the documents for a specific query by working only with bits that are set. To perform a conjunctive keyword search for a set of keywords kw0 = {kw_1, kw_2...kw_q}, the data owner performs the Bloom filter hash computation to regulate the set of bit locations, Q = {q_1, q_2, ...q_x}, that would be set in the query filter and sends them to the server. The server then computes T = IBF,q_1&IBF,q_2 ...&IBF,q_x , where IBF,q_i was the qthi row in IBF . The index of bits that were set in T were identified as the matched documents. Once the matches are identified, the cloud server can then return the matched document identifiers or the encrypted documents depending on the application requirements. Note that the size of the set Q was much smaller than m since the query filter contains only a few keywords while a conjunctive keyword Bloom filter contains all the keywords in a document. Therefore, this approach can identify the matched documents much faster, performing fewer operations than individual filter verification. Note that an entry in the Bloom filter index has as many bits as the number of documents. A query generally involves only a few words and very few bits set. These lead to only a few rows being extracted for matching. Furthermore, when performing the bit-wise AND testing, computer processors would generally test 32 or 64 bits at a time. Should a test results in all zeroes for any subset of bits in a row, the corresponding documents are no longer candidates and the subset of bits no longer require testing in subsequent rows.

B. Phrase search protocol

To provide phrase search capability, each document was parsed for lists of keyword pairs and triples. For example, 'Snow Flakes, Snow White' would yield the pairs, 'Snow Flakes', 'Flakes Snow' and 'Snow Night', and the triples, 'Snow Flakes Snow' and 'Flakes Snow White'. A keyed hash for each keyword pair was computed, H_{kp} (kw_j |kw_{j+1}), and passed into k hash functions and the result was used to set k bits in the Bloom filter, B_2 D_i. Keyword triples were similarly hashed to generate the Bloom filter, B_3D_i . The resulting Bloom filters for pairs and triples were organized into matrices with the first rows containing the filters B_x D_1 for the first document. The matrices are then transposed to produce the pairs and triples Bloom filter indexes, IBF 2 and IBF 3, which are stored alongside the encrypted documents on the cloud. To perform a phrase search for the keyword sequence, kw0 = {kw_1, kw_2...kw_q}, the data owner must first perform the Bloom filter hash computation of the pair, $H_{kp}(kw_1|kw_2)$, to establish the set bits in the query filter if the phrase contains two keywords. If the phrase contains more than two keywords, the hashes of triples within the phrase, H_{kp} (kw_j |kw_{j+1}|kw_{j+2}) where j = 1 to q − 2, are evaluated instead. The set bit locations are sent to the server, who then computes T = IBF 2,q1&IBF2,q2 ...&IBF 2,qx , where IBF 2,qi was the q thi row in IBF 2 if the phrase contains two keywords, and similarly using IBF 3 for longer phrases. The set bits in T identify the matched documents. That was, for each set bit index, i, in T, the following is true:

$\{H_{kp} (kw_1|kw_2)\} \in B_2D_i$ (1)

for pairs and

$\{H_{kp} (kw_j |kw_{j+1}|kw_{j+2})\} \in B_3D_i$, where j = 1 to q − 2, (2)

for triples.

Once the matches were identified, the cloud server returns the matched document identifiers or the encrypted documents depending on the application requirements. Our phrase search scheme requires only 2 messages to be sent: a) The initial message to the cloud server containing the set bit locations of the query Bloom filter T for pairs or triples and b) The response to the data owner containing the query results from the phrase search performed locally by the cloud. Performing the phrase search requires k(q − 2) hash computations for phrases of length q > 2 and a simple bit-wise AND operations. The protocol is computationally efficient. Its performance is dependent on the length of the phrase and largely independent of the size of the document set. Due to the space efficiency of Bloom filters, our scheme also requires less storage than index based schemes. Since filters are assigned per document, adding or removing documents consists simply of adding or removing the associated filters, providing a scalable solution. While a document containing a phrase will always be correctly identified as such, our scheme can falsely identify documents as containing a phrase when it doesn't. The source of the false positive was not only the natural property of Bloom filter, but also in how a phrase match was determined. If a user queries n-grams for n = 2 or n = 3, our scheme has no false positives other than ones arising from the use of Bloom filters. For n > 3, however, it was possible that keyword triples within a phrase appear in different parts of a document without the complete phrase being present. Using the previous example of 'Snow Flakes Snow White', a false positive would occur if a document does not contain the phrase but instead contains 'Snow Flakes Snow White' and 'Snow Flakes Snow White'. The soundness of the scheme was based on low occurrence of such scenarios in practical settings.

Symmetric Key Algorithm

In this paper we proposed two symmetric encryption algorithm, they are AES is to encrypt the document and triple DES is to encrypt the keywords with base 64 hashing technique.

AES (Advanced Encryption Standard)

- Advanced encryption standard is a block cipher intended to replace DES for commercial applications.
- It uses a 12~-bit block size and a key size of 128, 192 or 256 bits.

Steps

```
Cipher(byte in[16], byte out[16], key_arrayround_key[Nr+1])
begin
byte state[16];
state = in;
AddRoundKey(state, round_key[0]);
fori = 1 to Nr-1 stepsize 1 do
SubBytes(state);
ShiftRows(state);
MixColumns(state);
AddRoundKey(state, round_key[i]);
end for
SubBytes(state);
ShiftRows(state);
AddRoundKey(state, round_key[Nr]);
End
```

TRIPLE DES (Data Encryption Standard)

- Triple Data Encryption Standard (DES) is a symmetric key encryption technique where block cipher algorithms are applied three times to each data block.
- The key size is increased in Triple DES to guaranteeadditional benefit security through encryption capabilities.
- Each block contains 64 bits of data. Three keys are referred to as bundle keys with 56 bits per key.

Steps

The encryption algorithm is:

DES encrypt with K_1, DES *decrypt* with K_2, then DES encrypt with K_3.

Cipher text = $E_{K3}(D_{K2}(E_{K1}(\text{plain text})))$

Decryption is the reverse:

DES decrypt with K_3, *encrypt* with K_2, then decrypt with K_1.

plaintext = $D_{K1}(E_{K2}(D_{K3}(\text{cipher text})))$

Where,

E-Encryption
D-Decryption
K-Key

VI. CONCLUSION

In this paper, we proposed a phrase search scheme based on Bloom filter technique which is significantly faster than existing approaches, requiring just a single round of communication and Bloom filter verifications. Then the solution also addresses phrase search scheme as n-gram verification rather than a location search or a sequential chain verification. Unlike [10], [12], our schemes consider only the existence of a phrase, omitting any information of its location. Unlike [11], our schemes do not require sequential verification, is parallelizable and has a practical storage requirement. Our approach is also allows a phrase search to run autonomously without first performing a conjunctive keyword search to identify candidate documents. The technique Bloom filter index introduced in phrase search enables fast verification of Bloom filters in the same manner as indexing, it also achieves a lower storage cost than all existing solutions except [13], where a higher computational cost was exchanged in favor of lower storage. While exhibiting similar communication cost to leading existing solutions, the proposed solution is capable of adjusting to achieve maximum speed or high speed with a reasonable storage cost depending on the application.

REFERENCES

[1] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in In proceedings of Eurocrypt, 2004, pp. 506–522.

[2] B. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log," in Network and Distributed System Security Symposium, 2004.

[3] M. Ding, F. Gao, Z. Jin, and H. Zhang, "An efficient public key encryption with conjunctive keyword search scheme based on pairings," in IEEE International Conference onNetwork Infrastructure and Digital Content, 2012, pp. 526–530.

[4] F. Kerschbaum, "Secure conjunctive keyword searches for unstructured text," in International Conference on Network and System Security, 2011, pp. 285–289.

[5] C. Hu and P. Liu, "Public key encryption with ranked multikeyword search," in International Conference on Intelligent Networking and Collaborative Systems, 2013, pp. 109–113.

[6] Z. Fu, X. Sun, N. Linge, and L. Zhou, "Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query," IEEE Transactions on Consumer Electronics, vol. 60, pp. 164–172, 2014.

[7] C. L. A. Clarke, G. V. Cormack, and E. A. Tudhope, "Relevance ranking for one to three term queries," Information Processing and Management: an International Journal, vol. 36, no. 2, pp. 291–311, Jan. 2000.

[8] H. Tuo and M. Wenping, "An effective fuzzy keyword search scheme in cloud computing," in International Conference on Intelligent Networking and Collaborative Systems, 2013, pp. 786–789.

[9] M. Zheng and H. Zhou, "An efficient attack on a fuzzy keyword search scheme over encrypted data," in International Conference on High Performance Computing and Communications and Embedded and Ubiquitous Computing, 2013, pp. 1647–1651.

[10] S. Zittrower and C. C. Zou, "Encrypted phrase searching in the cloud," in IEEE Global Communications Conference, 2012, pp. 764– 770.

[11] Y. Tang, D. Gu, N. Ding, and H. Lu, "Phrase search over encrypted data with symmetric encryption scheme," in International Conference on Distributed Computing Systems Workshops, 2012, pp. 471–480.

[12] H. Poon and A. Miri, "An efficient conjunctive keyword and phrase search scheme for encrypted cloud storage systems," in IEEE International Conference on Cloud Computing, 2015.

[13]"A low storage phrase search scheme based on bloom filters for encrypted cloud services," to appear in IEEE International Conference on Cyber Security and Cloud Computing, 2015.

[14] H. S. Rhee, I. R. Jeong, J. W. Byun, and D. H. Lee, "Difference set attacks on conjunctive keyword search schemes," in Proceedings of the Third VLDB International Conference on Secure Data Management, 2006, pp. 64–74.

[15] K. Cai, C. Hong, M. Zhang, D. Feng, and Z. Lv, "A secure conjunctive keywords search over encrypted cloud data against inclusion-relation attack," in IEEE International Conference on Cloud Computing Technology and Science, 2013, pp. 339–346.

[16] Y. Yang, H. Lu, and J. Weng, "Multi-user private keyword search for cloud computing," in IEEE Third International Conference on Cloud Computing Technology and Science, 2011, pp. 264–271.

[17] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in International Conference on Distributed Computing Systems, 2010, pp. 253–262.

[18] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia, "Practical oblivious storage," in Proceedings of the Second ACM Conference on Data and Application Security and Privacy, 2012, pp. 13–24.

[19] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in Proceedings of the 36th Annual Symposium on Foundations of Computer Science, 1995, pp. 41–50.

[20] S. Ruj, M. Stojmenovic, and A. Nayak, "Privacy preserving access control with authentication for securing data in clouds," in Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012, pp. 556–563.

[21] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Proceedings of the 2000 IEEE Symposium on Security and Privacy, 2000, pp. 44–55.

[22] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, Report 2003/216, 2003.

[23] C. Liu, L. Zhu, L. Li, and Y. Tan, "Fuzzy keyword search on encrypted cloud storage data with small index," in 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, 2011, pp. 269–273.

[24] Yinqi Tang, Dawu Gu, Ning Ding, and Haining Lu, "Phrase search over encrypted data with symmetric encryption scheme," in International Conference on Distributed Computing Systems Workshops, 2012, pp. 471–480.

[25] Ke Cai, Cheng Hong, Min Zhang, Dengguo Feng, and Zhiquan Lv, "A secure conjunctive keywords search over encrypted cloud data against inclusion-relation attack," in IEEE International Conference on Cloud Computing Technology and Science, 2013, pp. 339–346.

[26] H. Poon and A. Miri, "An efficient conjunctive keyword and phrase search scheme for encrypted cloud storage systems," in IEEE International Conference on Cloud Computing, 2015.

