Data Storage in Cloud Environment Enhance Privacy

Treesa Maria Vincent[#1], J.Sakunthala [#2]

M.E Student , Anna university Chennai, Thodupuzha, Kerala, India
Assistant Professor(Department of computer Science and Engineering), Anna University Chennai, Erode, India

Abstract— **Cloud data have to be encrypted to protect data privacy, before outsourced to the commercial public cloud. The encryption process makes effective data utilization service a very challenging task. Traditional searchable encryption techniques allow users to securely search over encrypted data through keywords. They support only Boolean search and are not yet sufficient to meet the effective data utilization need that is inherently demanded by large number of users and huge amount of data files in cloud. The system facilitates server side ranking without keyword privacy. Search result authentication is provided in the system. The similarity analysis scheme is used to identify the query results under the cloud data storage.**

Keywords— **Encryption, Searchable encryption technique, Boolean search, Server side ranking, Search result authentication, Similarity analysis scheme**

I. Introduction

Cloud computing is a long dreamed vision of computing as a utility, where cloud customers can remotely store their data into the cloud as to enjoy the on-demand high-quality application and services from a shared pool of configurable computing resources [2]. The benefits brought by this new computing model include but are not limited to: relief of the burden for storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc., [3].

As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud, such as e-mails, personal health records, company finance data, and government documents, etc. The fact that data owners and cloud server are no longer in the same trusted domain may put the outsourced unencrypted data at risk [4] the cloud server may leak data information to unauthorized entities [5] or even be hacked [6]. It follows that sensitive data have to be encrypted prior to outsourcing for data privacy and combating unsolicited accesses. However, data encryption makes effective data utilization a very challenging task given that there could be a large amount of outsourced data files. Besides, in Cloud Computing, data owners may share their outsourced data with a large number of users, who might want to only retrieve certain specific data files they are interested in during a given session. One of the most popular ways to do so is through keyword-based search. Such keyword search technique allows users to selectively retrieve files of interest and has been widely applied in plaintext search scenarios. Unfortunately, data encryption, which restricts user's ability to perform keyword search and further demands the protection of keyword privacy, makes the traditional plaintext search methods fail for encrypted cloud data.

Although traditional searchable encryption schemes allow a user to securely search over encrypted data through keywords without first decrypting it, these techniques support only conventional Boolean keyword search,1 without capturing any relevance of the files in the search result. When directly applied in large collaborative data outsourcing cloud environment, they may suffer from the following two main drawbacks. On the one hand, for each search request, users without pre-knowledge of the encrypted cloud data have to go through every retrieved file in order to find ones most matching their interest, which demands possibly large amount of post-processing overhead, On the other hand, invariably sending back all files solely based on presence/ absence of the keyword further incurs large unnecessary network traffic, which is absolutely undesirable in today's pay-as-you-use cloud paradigm. In short, lacking of effective mechanisms to ensure the file retrieval accuracy is a significant drawback of existing searchable encryption schemes in the context of Cloud Computing. Nonetheless, the state of the art in information retrieval (IR) community has already been utilizing various scoring mechanisms quantify and rank order the relevance of files in response to any given search query. Although the importance of ranked search has received attention for a long history in the context of plaintext searching by IR community, surprisingly, it is still being overlooked and remains to be addressed in the context of encrypted data search.

Therefore, how to enable a searchable encryption system with support of secure ranked search is the problem tackled in this paper. Our work is among the first few ones to explore ranked search over encrypted data in Cloud Computing. Ranked search greatly enhances system usability by returning the matching files in a ranked order regarding to certain relevance criteria, thus making one step closer toward practical deployment of privacy-preserving data hosting services in the context of Cloud Computing. To achieve our design goals on both system security and usability, we propose to bring together the advance of both crypto and IR community to design the ranked searchable symmetric encryption (RSSE) scheme, in the spirit of "as-strong-as-possible" security guarantee. Specifically, we explore the statistical measure approach from IR and text mining to embed weight information of each file during the establishment of searchable index before outsourcing the encrypted file collection [12]. As directly outsourcing relevance scores will leak lots of sensitive frequency information against the keyword privacy, we then integrate a recent crypto primitive order-preserving symmetric encryption (OPSE) and properly modify it to develop a one-to- many order-preserving mapping technique for our purpose to protect those sensitive weight information, while providing efficient ranked search functionalities. Our contribution can be summarized as follows:

1. For the first time, we define the problem of secure ranked keyword search over encrypted cloud data, and provide such an effective protocol, which fulfills the secure ranked search functionality with little relevance score information leakage against keyword privacy.
2. Thorough security analysis shows that our ranked searchable symmetric encryption scheme indeed enjoys "as-strong-as-possible" security guarantee compared to previous searchable symmetric encryption (SSE) schemes.
3. We investigate the practical considerations and enhancements of our ranked search mechanism, including the efficient support of relevance score dynamics, the authentication of ranked search results, and the reversibility of our proposed one-to- many order-preserving mapping techniques.
4. Extensive experimental results demonstrate the effectiveness and efficiency of the proposed solution.

II. RELATED WORK

Searchable encryption Traditional searchable encryption has been widely studied as a cryptographic primitive, with a focus on security definition formalizations and efficiency improvements. Song et al. first introduced the notion of searchable encryption. They proposed a scheme in the symmetric key setting, where each word in the file is encrypted independently under a special two-layered encryption construction. Thus, a searching overhead is linear to the whole file collection length. Goh developed a Bloom filter-based per-file index, reducing the workload for each search request proportional to the number of files in the collection. Chang and Mitzenmacher also developed a similar per-file index scheme. To further enhance search efficiency, Curtmola et al. proposed a per-keyword-based approach, where a single encrypted hash table index is built for the entire file collection, with each entry consisting of the trapdoor of a keyword and an encrypted set of related file identifiers. Searchable encryption has also been considered in the public-key setting. Boneh et al. presented the first public-key-based searchable encryption scheme, with an analogous scenario. In their construction, anyone with the public key can write to the data stored on the server but only authorized users with the private key can search. As an attempt to enrich query predicates, conjunctive keyword search over encrypted data have also been proposed. Aiming at tolerance of both minor typos and format inconsistencies in the user search input, fuzzy keyword search over encrypted cloud data has been proposed by Li et al. in [9]. Very recently, a privacy-assured similarity search mechanism over outsourced cloud data has been explored by Wang et al. in [11]. Note that all these schemes support only Boolean keyword search, and none of them support the ranked search problem which we are focusing on in this paper.

Following our research on secure ranked search over encrypted data, very recently, Cao et al. [10] propose a privacy-preserving multikeyword ranked search scheme, which extends our previous work in [1] with support of multikeyword query. They choose the principle of "coordinate matching," i.e., as many matches as possible, to capture the similarity between a multikeyword search query and data documents, and later quantitatively formalize the principle by a secure inner product computation mechanism. One disadvantage of the scheme is that cloud server has to linearly traverse the whole index of all the documents for each search request, while ours is as efficient as existing SSE schemes with only constant search cost on cloud server.

Secure top-k retrieval from Database Community from database community are the most related work to our proposed RSSE. The idea of uniformly distributing posting elements using an order-preserving cryptographic function. However, the order-preserving mapping function proposed does not support score dynamics, i.e., any insertion and updates of the scores in the index will result in the posting list completely rebuilt. Zerr et al. use a different order-preserving mapping based on pre-sampling and training of the relevance scores to be outsourced, which is not as efficient as our proposed schemes. Besides, when scores following different distributions need to be inserted, their score transformation function still needs to be rebuilt. On the contrary, in our scheme the score dynamics can be gracefully handled, which is an important benefit inherited from the original OPSE. This can be observed from the Binary Search(.) procedure in Algorithm 1, where the same score will always be mapped to the same random-sized non-overlapping bucket, given the same encryption key. In other words, the newly changed scores will not affect previous mapped values. We note that supporting score dynamics, which can save quite a lot of computation overhead when file collection changes, is a significant advantage in our scheme. Moreover, both works above do not exhibit thorough security analysis which we do in the paper.

Other related techniques. Allowing range queries over encrypted data in the public key settings, where advanced privacy-preserving schemes were proposed to allow more sophisticated multi-attribute search over encrypted files while preserving the attributes' privacy. Though these two schemes provide provably strong security, they are generally not efficient in our settings, as for a single search request, a full scan and expensive computation over the whole encrypted scores corresponding to the keyword posting list are required. Moreover, the two schemes do not support the ordered result listing on the server side. Thus, they cannot be effectively utilized in our scheme since the user still does not know which retrieved files would be the most relevant. The transactional data values are compared and similarity values are estimated.

The results are prepared using the similarity value and threshold levels.

III. PROBLEM STATEMENT

A. The System and Threat Model

We consider an encrypted cloud data hosting service involving three different entities, data owner, data user, and cloud server. Data owner has a collection of n data files C = {F_1, F_2, . . . , F_n} that he wants to outsource on the cloud server in encrypted form while still keeping the capability to search through them for effective data utilization reasons. To do so, before outsourcing, data owner will first build a secure searchable index I from a set of m distinct keywords W = {w_1, w_2, . . . ,w_m} extracted from the file collection C, and store both the index I and the encrypted file collection C on the cloud server.

We assume the authorization between the data owner and users is appropriately done. To search the file collection for a given keyword w, an authorized user generates and submits a search request in a secret form—a trapdoor T_w of the keyword w—to the cloud server. Upon receiving the search request T_w, the cloud server is responsible to search the index I and return the corresponding set of files to the user. We consider the secure ranked keyword search problem as follows: the search result should be returned according to certain ranked relevance criteria, to improve file retrieval accuracy for users without prior knowledge on the file collection C. However, cloud server should learn nothing or little about the relevance criteria as they exhibit significant sensitive information against keyword privacy. To reduce bandwidth, the user may send an optional value k along with the trapdoor T_w and cloud server only sends back the top-k most relevant files to the user's interested keyword w.

We primarily consider an "honest-but-curious" server in our model, which is consistent with most of the previous searchable encryption schemes. We assume the cloud server acts in an "honest" fashion and correctly follows the designated protocol specification, but is "curious" to infer and analyze the message flow received during the protocol so as to learn additional information. In other words, the cloud server has no intention to actively modify the message flow or disrupt any other kind of services. However, in some unexpected events, the cloud server may behave beyond the "honest-but-curious" model.

B. Design Goals

To enable ranked searchable symmetric encryption for effective utilization of outsourced and encrypted cloud data under the aforementioned model, our system design should achieve the following security and performance guarantee. Specifically, we have the following goals: 1) Ranked keyword search: to explore different mechanisms for designing effective ranked search schemes based on the existing searchable encryption framework, 2) Security guarantee: to prevent cloud server from learning the plaintext of either the data files or the searched keywords, and achieve the "as-strong-as-possible" security strength compared to existing searchable encryption schemes, 3) Efficiency: above goals should be achieved with minimum communication and computation overhead.

IV. SEARCHABLE ENCRYPTION SCHEME

In the introduction, we have motivated the ranked keyword search over encrypted data to achieve economies of scale for Cloud Computing. We start from the review of existing searchable symmetric encryption schemes and framework for our proposed ranked searchable symmetric encryption. Note that by following the same security guarantee of existing SSE, it would be very inefficient to support ranked search functionality over encrypted data, as demonstrated in our basic scheme. The discussion of its demerits will lead to our proposed scheme.

A. Searchable Symmetric Encryption

Searchable encryption allows data owner to outsource his data in an encrypted manner while maintaining the selectively search capability over the encrypted data. Generally, searchable encryption can be achieved in its full functionality using an oblivious RAMs. Although hiding everything during the search from a malicious server, utilizing oblivious RAM usually brings the cost of logarithmic number of interactions between the user and the server for each search request. Thus, in order to achieve more efficient solutions, almost all the existing works on searchable encryption literature resort to the weakened security guarantee. Here, access pattern refers to the outcome of the search result, i.e., which files have been retrieved. The search pattern includes the equality pattern among the two search requests and any information derived thereafter from this statement.

Having a correct intuition on the security guarantee of existing SSE literature is very important for us to define our ranked searchable symmetric encryption problem. As later, we will show that following the exactly same security guarantee of existing SSE scheme, it would be very inefficient to achieve ranked keyword search, which motivates us to further weaken the security guarantee of existing SSE appropriately and realize an "as-strong-as-possible" ranked searchable symmetric encryption. Actually, this notion has been employed by cryptographers in much recent work [7] where efficiency is preferred over security.

B. Framework of RSSE System

We follow the similar framework of previously proposed searchable symmetric encryption schemes and adapt the framework for our ranked searchable encryption system. A ranked searchable encryption scheme consists of four algorithms (KeyGen, BuildIndex, TrapdoorGen, SearchIndex). Our ranked searchable encryption system can be constructed from these four algorithms in two phases, Setup and Retrieval:

- Setup. The data owner initializes the public and secret parameters of the system by executing Key- Gen, and pre-processes the data file collection

C by using Build Index to generate the searchable index from the unique words extracted from C. The owner then encrypts the data file collection C, and publishes the index including the keyword frequency-based relevance scores in some encrypted form, together with the encrypted collection C to the Cloud. As part of Setup phase, the data owner also needs to distribute the necessary secret parameters to a group of authorized users by employing off-the-shelf public key cryptography or more efficient primitive such as broadcast encryption.

V. Efficient Ranked Searchable Symmetric Encryption Scheme

The above straightforward approach demonstrates the core problem that causes the inefficiency of ranked searchable encryption. That is how to let server quickly perform the ranking without actually knowing the relevance scores. To effectively support ranked search over encrypted file collection, we now resort to the newly developed cryptographic primitive—order preserving symmetric encryption achieve more practical performance. Note that by resorting to OPSE, our security guarantee of RSSE is inherently weakened compared to SSE, as we now let server know the relevance order. However, this is the information we want to trade off for efficient RSSE. We will first briefly discuss the primitive of OPSE and its pros and cons. Then, we show how we can adapt it to suit our purpose for ranked searchable encryption with an "as-strong-as-possible" security guarantee. Finally, we demonstrate how to choose different scheme parameters via concrete examples.

A. Using order Preserving Symmetric Encryption

The OPSE is a deterministic encryption scheme where the numerical ordering of the plaintexts gets preserved by the encryption function. Boldyreva et al. gives the first cryptographic study of OPSE primitive and provides a construction that is provably secure under the security framework of pseudorandom function or pseudorandom permutation. Namely, considering that any order-preserving function g(.) from domain D = {1, . . .,M} to range R = {1, . . .,N} can be uniquely defined by a combination of M out of N ordered items, an OPSE is then said to be secure if and only if an adversary has to perform a brute force search over all the possible combinations of M out of N to break the encryption scheme. If the security level is chosen to be 80 bits, then it is suggested to choose M = N/2 > 80 so that the total number of combinations will be greater than 2^{80}. Their construction is based on an uncovered relationship between a random order-preserving function and the hyper geometric probability distribution, which will later be denoted as HGD.

At the first glance, by changing the relevance score encryption from the standard indistinguishable symmetric encryption scheme to this OPSE, it seems to follow directly that efficient relevance score ranking can be achieved just like in the plaintext domain. However, as pointed out earlier, the

- Retrieval. The user uses TrapdoorGen to generate a secure trapdoor corresponding to his interested keyword, and submits it to the cloud server. Upon receiving the trapdoor, the cloud server will derive a list of matched file IDs and their corresponding encrypted relevance scores by searching the index via Search-Index. The matched files should be sent back in a ranked sequence based on the relevance scores. However, the server should learn nothing or little beyond the order of the relevance scores.

OPSE is a deterministic encryption scheme. This inherent deterministic property, if not treated appropriately, will still leak a lot of information as any deterministic encryption scheme will do. One such information leakage is the plaintext distribution. For example, which shows a skewed relevance score distribution of keyword "network," sampled from 1,000 files of our test collection. For easy exposition, we encode the actual score into 128 levels in domain from 1 to 128. Due to the deterministic property, if we use OPSE directly over these sampled relevance scores, the resulting ciphertext shall share exactly the same distribution as the relevance score. Specifically, the authors have shown that the TF distribution of certain keywords from the Enron e-mail corpus3 can be very peaky, and thus result in significant information leak for the corresponding keyword. In [8], the authors further point out that the TF distribution of the keyword in a given file collection usually follows a power law distribution, regardless of the popularity of the keyword. Their results on a few test file collections show that not only different keywords can be differentiated by the slope and value range of their TF distribution, but even the normalized TF distributions, i.e., the original score distributions can be keyword specific. Thus, with certain background information on the file collection, such as knowing it contains only technical research papers, the adversary may be able to reverse engineer the keyword "network" directly from the encrypted score distribution without actually breaking the trapdoor construction, nor does the adversary need to break the OPSE.

B. One-to-Many Order-Preserving Mapping

Therefore, we have to modify the OPSE to suit our purpose. In order to reduce the amount of information leakage from the deterministic property, an one-to-many OPSE scheme is thus desired, which can flatten or obfuscate the original relevance score distribution, increase its randomness, and still preserve the plaintext order. To do so, we first briefly review the encryption process of original deterministic OPSE, where a plaintext m in domain D is always mapped to the same random-sized nonoverlapping interval bucket in range R, determined by a keyed binary search over the range R and the result of a random HGD sampling function. A ciphertext c is then chosen within the bucket by using m as the seed for some random selection function.

Our one-to-many order-preserving mapping employs the random plaintext-to-bucket mapping of OPSE, but incorporates the unique file IDs together with the plaintext m as the random seed in the final ciphertext chosen process. Due to the use of unique file ID as part of random selection seed, the same plaintext m will no longer be deterministically assigned to the same cipher text c, but instead a random value within the randomly assigned bucket in rangeR. The whole process is shown in Algorithm 1. Here, TapeGen(.) is a random coin generator and HYGEINV(.) is the efficient function implemented in Matlab as our instance for the HGD(.) sampling function. The correctness of our one-to-many order-preserving mapping follows directly from the Algorithm 1. Note that our rational is to use the OPSE block cipher as a tool for different application scenarios and achieve better security, which is suggested by and consistent. Now, if we denote OPM as our one-to-many order-preserving mapping function with parameter: OPM: $\{0, 1\}^{l} * \{0, 1\}^{\log |D|} \rightarrow \{0, 1\}^{\log |R|}$, our proposed RSSE scheme can be described as follows:

In the Setup phase

1. The data owner calls KeyGen(1^k, 1^l, $1^{l`}$ , $1^p$, |D|, |R|), generates random keys x, y, z $\overset{R}{\leftarrow}$ {0, 1}$^k$, and outputs K = {x, y, z, $1^l$,$1^{l`}$, 1^p, |D|, |R|}.
2. The data owner calls BuildIndex(K, C) to build the inverted index of collection C, and uses $OPMf_z(w_i)$(.) instead of E(.) to encrypt the scores.

In the Retrieval phase

1. The user generates and sends a trapdoor Tw = (Π_x(w), f_y(w)) for an interested keyword w. Upon receiving the trapdoor T_w, the cloud server first locates the matching entries of the index via Π_x(w), and then uses f_y(w) to decrypt the entry. These are the same with basic approach.
2. The cloud server now sees the file identifiers <id(F_{ij})> and their associated order-preserved encrypted scores: $OPMf_z(w_i)(S_{ij})$.
3. The server then fetches the files and sends back them in a ranked sequence according to the encrypted relevance scores {$OPMf_z(w_i)(S_{ij})$}, or sends top-k most relevant files if the optional value k is provided.

Algorithm 1. One-To-Many Order-Preserving Mapping-Opm

```
1: procedure $OPM_K$(D,R,m, id(F))
2: while |D|! = 1 do
3: {D,R} BinarySearch (K,D,R,m);
4: end while
5: coin $\overset{R}{\leftarrow}$ TapeGen(K, (D,R, 1||m, id(F)));
6: c $\overset{coin}{\longleftarrow}$ R;
7: return c;
8: end procedure
9: procedure BinarySearch(K,D,R,m);
10: M |D|, N |R|;
11: d min(D) - 1, r min(R) – 1;
12: y←r+ $\lceil N/2 \rceil$ ;
13: coin $\overset{R}{\leftarrow}$ TapeGen(K, (D,R, 0||y));
14: x $\overset{R}{\leftarrow}$ d + HYGEINV(coin,M,N, y - r);
15: if m $\leq$ x then
16: D←{d + 1, . . . , x};
17: R←{r + 1, . . . , y};
18: else
19: D←{x + 1, . . . , d+M};
20: R←{y + 1, . . . , r + N};
21: end if
22: return {D, R},
23: end procedure
```

VI. Security and Privacy Ensured Data Search Model for Encrypted Storage

The cloud data center manages the transactional data values. The data values are maintained in encrypted format. The data values are queried using the encrypted query values. The system is designed to provide data security and privacy for the transactional data over the cloud environment. The order preserving mapping model is used for the encryption process. The score functions are used to fetch the data values in a ranked manner. The dynamic scoring mechanism is used in the system.

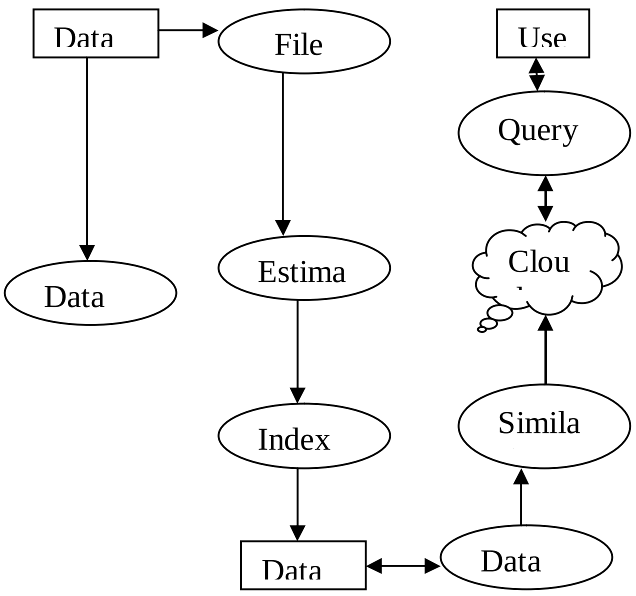

Figure.1: Encrypted Search Model

The system is divided into two applications. They are data source and client application. The data source manages the transactional data values. The client application issues the query value and collects the data from the data source. The data values are updated in the data source in an encrypted format. The data retrieval and ranking operations are carried out on the encrypted data format only. The system secures the data under the storage and query transmission process.

The system is divided into five major modules. They are data source, storage management, score assignment, client and query process. The data source module is designed to manage the data values. The storage management module is designed to perform the data encryption and update operations. The score assignment module is used to assign the relevance score the for the transactional data values. The client application is used to fetch the data value from the data source. The query process module is designed to submit and collect the data values.

A. *Data source*

The data source application is designed to manage the transactional and user information. The user information are updated with their access information. All the query history is maintained under the data source application. The transactional data values are maintained for different domains. The data values are updated in encrypted format. The data retrieval is performed under the data source application.

B. *Storage Management*

The storage management is designed to handle data encryption and update operations. The order preserving mapping technique is used to encrypt the data values. The system includes the reversible order preserving map model for the encryption process. The data update operation can be dynamically performed on the system. The data values are updated and stored in the encrypted format. The transactional data and its encryption process are carried out under the data source environment.

C. *Client*

The client application is designed to perform the data retrieval operations. The data values are collected from the server and updated into the client interface. Each client is authenticated with unique identification value. The client collects the data values with query keywords.

VII. CONCLUSIONS

Cloud customers can remotely store their data on a shared pool of configurable computing resources in cloud. Searchable Symmetric Encryption scheme is used to provide storage and retrieval security. Order Preserving Symmetric Encryption scheme is enhanced in reversible mechanism. The system is improved with result authentication and similarity based ranking model. The data storage and search process is carried out with encrypted query model. The system performs index operations on encrypted data values. The system also secures the search results. The system supports incremental data update scheme.

ACKNOWLEDGMENT

I express my sincere thanks to Mrs.J.Sakunthala ME, Assistant professor, Department of Computer Science and Engineering (PG), who has always been with me throughout the process of design and construction of this paper. I am very much thankful for her guidance, constant encouragement, support and valuable suggestions to successfully carryout this paper.

REFERENCES

[1] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, “Secure Ranked Keyword Search over Encrypted Cloud Data,” Proc. IEEE 30th Int’l Conf. Distributed Computing Systems (ICDCS ’10), 2010.
[2] P. Mell and T. Grance, “Draft Nist Working Definition of Cloud Computing,” http://csrc.nist.gov/groups/SNS/cloudcomputing/ index.html, Jan. 2010.
[3] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing,” Technical Report UCB-EECS-2009-28, Univ. of California, Berkeley, Feb. 2009.
[4] Cloud Security Alliance “Security Guidance for Critical Areas of Focus in Cloud Computing,” http://www.cloudsecurityalliance.org, 2009.
[5] Z. Slocum, “Your Google Docs: Soon in Search Results?” http://news.cnet.com/8301-17939_109-10357137-2.html, 2009.
[6] B. Krebs, “Payment Processor Breach May Be Largest Ever,” http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html, Jan. 2009.
[7] A. Boldyreva, N. Chenette, Y. Lee, and A. O’Neill, “Order-Preserving Symmetric Encryption,” Proc. Int’l Conf. Advances in Cryptology, 2009.
[8] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, “Zerber+r: Top-k Retrieval from a Confidential Index,” Proc. Int’l Conf. Extending Database Technology: Advances in Database Technology (EDBT ’09), 2009.
[9] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, “Fuzzy Keyword Search over Encrypted Data in Cloud Computing,” Proc. IEEE Infocom ’10, 2010.
[10] N. Cao, C. Wang, K. Ren, and W. Lou, “Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data,” Proc. IEEE Infocom ’11, 2011.
[11] C. Wang, K. Ren, S. Yu, K. Mahendra, and R. Urs, “Achieving Usable and Privacy-Assured Similarity Search over Outsourced Cloud Data,” Proc. IEEE INFOCOM, 2012.
[12] Cong Wang, Ning Cao, Kui Ren and Wenjing Lou, “Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data” IEEE Transactions On Parallel And Distributed Systems, Vol. 23, No. 8, August 2012.