Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# Encrypted Data Storage in Cloud Environment

**Treesa Maria Vincent**
*Department of computer science*
*Anna University Chennai, India.*

**Mrs.J.Sakunthala**
Assistant professor
*Nandha Enginnering college, Erode, India.*

***Abstract— Cloud Computing is the use of computing resources that are deliverable as a service over a network. Cloud computing relies on sharing of resources to achieve coherence. The cloud server may leak data information due to unauthorised entities or even be hacked. Encrypted storage and retrieval model is used. Searchable encryption technique supports only Boolean search process. Relevance score is used in secure searchable index preparation process. One- to-many order preserving mapping technique is used to properly protect sensitive score information. Relevance score is used in secure searchable index preparation process. The privacy enabled data searching scheme provides solution for secure ranked keyword search over encrypted cloud data. The privacy enabled data searching scheme provides solution for secure ranked keyword search over encrypted cloud data. Ranked search greatly enhances system usability by enabling search result relevance ranking instead of sending undifferentiated results, and further ensures the file retrieval accuracy.***

***Keywords— Encrypted storage, Searchable encryption Technique, Boolean search process, Relevance score, One-to-many order preserving mapping, Ranked keyword.***

## I. INTRODUCTION

Cloud Computing enables cloud customers to remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. In information retrieval, inverted index is a widely used indexing structure that stores a list of mappings from keywords to the corresponding set of files that contain this keyword, allowing full text search. For ranked search purposes, the task of determining which files are most relevant is typically done by assigning a numerical score, which can be pre-computed, to each file based on some ranking function introduced below. We will use this inverted index structure to give our basic ranked searchable symmetric encryption construction. 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]. 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 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. An organization that is risk-averse enough to avoid the public cloud should be building a secure cloud--possibly the company should be building its dream cloud, which contains all the security controls that it thinks are missing from a public environment. Since the company physically owns the private cloud, incident response can be very swift. Detection capabilities need to be cloud-specific (for example, sensors need to monitor inside the cloud, not just at its perimeter) and operational capabilities such as patch management must be sharp. A vulnerable service that's in a cloud might have greater exposure and risk than the same service in a standard server farm thanks to the shared nature of cloud resources. 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 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.

## II. RELATED WORK

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]. 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.

Following our research on secure ranked search over encrypted data, very recently, Cao et al. [10] propose a privacy-preserving multi-keyword ranked search scheme, which extends our previous work in [1] with support of multi-keyword query. They choose the principle of "coordinate matching," i.e., as many matches as possible, to capture the similarity between a multi-keyword 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.

Encrypted cloud data hosting service involving three different entities as illustrated in Figure 2.1, data owner, data user, and cloud server. Data owner has a collection of n data files $C = \{F_1, F_2, \ldots, F_n\}$ 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 $W = \{w_1, w_2, \ldots, w_m\}$ extracted from the file collection C, and store both the index I and the encrypted file collection C on the cloud server.

The requirements of balancing privacy and confidentiality with efficiency and accuracy pose significant challenges to the design of search schemes for a number of search scenarios. This problem has attracted interests from the cryptography community in recent years to investigate theories and techniques for "searchable encryption." However, existing work only supports Boolean searches to identify the presence/absence of terms of interests in encrypted documents. Advances in information retrieval have gone well beyond Boolean searches; scoring schemes have been

 

widely employed to quantify and rank-order the relevance of a document to a set of query terms. The goals of this paper are to explore a framework to securely rank-order documents in response to a query, and develop techniques to extract the most relevant document(s) from a large encrypted data collection. To our best knowledge, this is the first attempt in the research community to explore secure rank-ordered search. As an initial step, we focus in this paper on modelling common scenarios of secure rank-ordered search and exploring indexing and search techniques built upon existing established cryptographic primitives. The understandings obtained from this exploration will pave ways to bring together researchers from information retrieval and applied cryptography to establish a bridge between these areas.

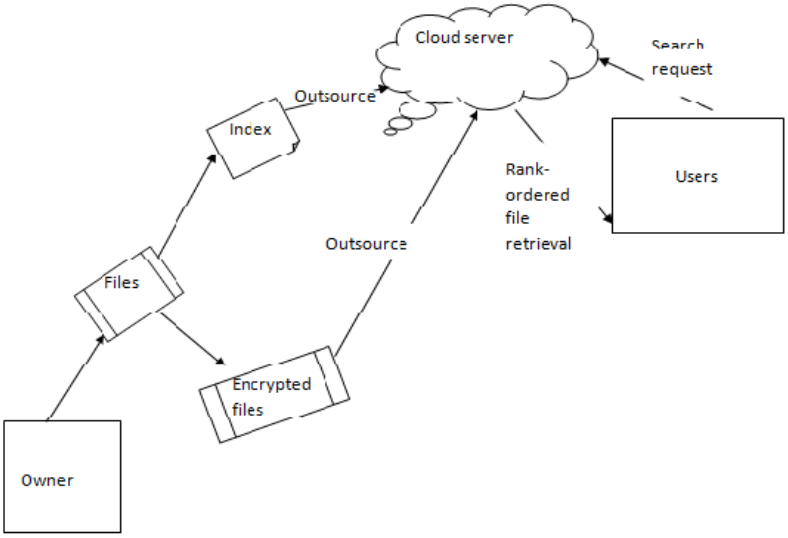


**Figure 2.1 Data Searching over Encrypted Cloud Data**

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 (e.g., keyword frequency-based scores, as will be introduced shortly), 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.

## III. SEARCHABLE SYMMETRIC ENCRYPTION SCHEME

This 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

### *A. Order Preserving Symmetric Encryption*

This 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 [7] 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.

The 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 cipher text 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*

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 cipher-text 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 range R. 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.

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 range R. 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 $\xleftarrow{R}$ $\{0, 1\}^k$, 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 $Tw = (\Pi_x(w), f_y(w))$ for an interested keyword w. Upon receiving the trapdoor $T_w$, the cloud server first locates the matching entries of the index via $\Pi_x(w)$, 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 $\xleftarrow{R}$ TapeGen(K, (D,R, 1||m, id(F)));
6: c $\xleftarrow{coin}$ 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 $\xleftarrow{R}$ TapeGen(K, (D,R, 0||y));
14: x $\xleftarrow{R}$ d + HYGEINV(coin,M,N, y - r);
15: if m ≤ 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

## IV. OBJECTIVES

Ranked searchable symmetric encryption for effective utilization of outsourced and encrypted cloud data under the aforementioned model, our system design should achieve the security and performance guarantee. 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.

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.

### *A. ZERBER$^{+R}$: Top-K Retrieval From A Confidential Index*

The number of access-controlled documents shared over enterprise intranets is growing rapidly. Collaboration groups within or across enterprises require facilities for effective and efficient retrieval of the top-k documents most relevant to a given query while shielding those documents from others' eyes. In the enterprise settings users can participate in a number of collaboration groups and need to obtain the most relevant top-k results from the whole document collection accessible to them. Top-k [8] is a standard IR technique which enables fast query execution on very large indexes and makes systems highly scalable. It prevents information overload by returning only highly ranked documents most relevant to the user query and allows reducing bandwidth in case group members use mobile devices to access the enterprise's search facilities. Whereas top-k retrieval of publicly available documents is wellstudied, indexing access-controlled information for top-k processing remains a challenging task. Even within a single enterprise, competitive working groups are not likely to agree on a single trusted server hosting the index or fully trusted system administrators. Therefore an index created over a set of confidential documents requires specific protection.

Inverted indexes are the standard choice for keyword (full-text) top-k document search[8]. An inverted index is a sequence of posting lists, each of which contains the posting elements. Every posting element represents a document which contains a particular term and includes the relevance score used for ranking. Posting elements within the list are sorted with respect to their scores which allows for efficient determination of the top-k results by pruning lower scored posting elements.

Unfortunately, an inverted index, which is a highly efficient data structure for top-k retrieval, does not preserve confidentiality of the indexed documents. The content of a document can be easily uncovered by a straightforward posting lists scan. Relevance scores within the posting elements disclose the number of the indexed documents highly relevant to a specific term. Even if the exact content of the elements is obscured, the number of highly ranked documents can give an industrial spy important insights, e.g., identification of compounds used in the development of a new chemical process. In general there is a tradeoff between retrieval effectiveness of the index and confidentiality it can provide. On the one hand in order to effectively answer a query, an index server requires possibly complete ranking information enclosed in its posting elements. On the other hand this information gives undesirable insights into the content of the indexed documents.

In this paper we present Zerber+R [8], a novel ranking model which allows for top-k retrieval from a confidential outsourced inverted index without information leakage. This paper makes the following contributions: (i) we introduce the problem of the confidential top-k retrieval from an outsourced inverted index; (ii) we propose Zerber+R, a ranking model which minimizes information leakage by top-k retrieval from a confidential outsourced inverted index by supporting sorted indexes which do not exhibit additional information to a potential adversary (iii) we propose a novel relevance score transformation function, **RSTF** for short, – a heuristic which hides term specific distribution of relevance score values, making scores of different terms indistinguishable. This heuristic enables inclusion of relevance scores in the posting elements on an un-trusted server to allow the server answering top-k queries

### *B. Confidentiality Preserving Rank Ordered Search*

The The requirements of balancing privacy and confidentiality with efficiency and accuracy pose significant challenges to the design of search schemes for a number of search scenarios. This problem has attracted interests from

 

the cryptography community in recent years to investigate theories and techniques for "searchable encryption." However, existing work only supports Boolean searches to identify the presence/absence of terms of interests in encrypted documents. Advances in information retrieval have gone well beyond Boolean searches; scoring schemes have been widely employed to quantify and rank-order the relevance of a document to a set of query terms. The goals of this paper are to explore a framework to securely rank-order documents in response to a query, and develop techniques to extract the most relevant document(s) from a large encrypted data collection. To our best knowledge, this is the first attempt in the research community to explore secure rank-ordered search. As an initial step, we focus in this paper on modeling common scenarios of secure rank-ordered search and exploring indexing and search techniques built upon existing established cryptographic primitives. The understandings obtained from this exploration will pave ways to bring together researchers from information retrieval and applied cryptography to establish a bridge between these areas.

To accomplish our goals, we collect term frequency information for each document in the collection to build indices, as in traditional retrieval systems for plaintext. We further secure these indices that would otherwise reveal important statistical information about the collection to protect against statistical attacks. During the search process, the query terms are encrypted to prevent the exposure of information to the data center and other intruders, and to confine the searching entity to only make queries within an authorized scope. Utilizing term frequencies and other document information, we apply cryptographic techniques such as order-preserving encryption to develop schemes that can securely compute relevance scores for each document, identify the most relevant documents, and reserve the right to screen and release the full content of relevant documents. The proposed framework has comparable performance to conventional searching systems designed for non-encrypted data in terms of search accuracy. There are a number of scenarios where the content owner may want to grant a user limited access to search a confidential collection. For example, the searcher could be a scholar or a low-level analyst who wants to identify relevant documents from a private/classified collection, and may need clearance only for the top-ranked documents

*C. Building an Encrypted and searchable Audit Log*

Secure versions of such logs, designed to defend against malicious tampering, allow the current state of the system to be audited even when that system has been under active attack by malicious insiders or outsiders. Correctly designed secure audit logging mechanisms can detect unauthorized past activity, even when the person performing that action goes to great lengths to cover their tracks. The existence of such logs can be used to enforce correct user behaviour, by holding users accountable for their actions as recorded in the audit log. Such logs can be used in a wide variety of systems, from a control system that logs the commands user issues, to a database system that logs the queries a user makes. Typically, when an organization wishes to inspect past activity it will search the audit log for relevant information. For example, if a certain user was suspected of behaving improperly the organization might search for all actions performed by that particular user. If the organization wishes to see all actions of a certain type, it might search for all log entries that match a given keyword. For an audit log to be useful in practice, it is critical that it be efficiently searchable for keywords of interest. At the same time, the contents of an audit log can be considered to be sensitive information. For instance, knowing what actions are made by a certain user could violate that individual's privacy. If the log contains information about not only what query was made, but what results were returned, access to the audit log would imply effective access to the database, circumventing database access controls. The organization that owns the system being logged might consider the information the log holds to be valuable and not wish to share it with others, while for robustness' sake, the organization may want to store backup copies of the audit log information at sites it may not completely control. In general, this means that the contents of the audit log must be encrypted. However, this makes it extremely difficult to search. Using traditional techniques, searching the log would require decrypting every record. This approach has several disadvantages. First, it requires decrypting the entire log data, regardless of what information one is looking for; this opens opportunities for unintended access to log records other than the ones relevant to the current investigation. Second, it requires the entity with the decryption key to interactively process all the log data, which can be quite large. In many applications, one would like to entrust the ability to decrypt audit logs to an entity or system with high levels of trust and assurance; requiring that system to also be able to process large quantities of log data in an on-line fashion limits one's choice of trusted parties. It would be preferable to be able to selectively delegate the ability to search the log to parties with the means to process the data. The key challenge to building a successful, secure audit logging system is to simultaneously protect the integrity of the audit log, control access to contents, and maintain its usefulness by making it searchable. We present a design for an encrypted audit log that allows a designated trusted party, the audit escrow agent, to construct keyword search capabilities, which allow investigators in possession of such capabilities to search for and decrypt entries matching a given keyword. The escrow agent can distribute a capability to an investigator if he deems it appropriate. Since we expect keyword search capabilities to be distributed rather infrequently, the escrow agent can be made to be very secure from attack.

## V. SECURITY AND PRIVACY ENSURED ENCRYPTED DATA SEARCH

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. 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. Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud, such as emails, personal health records, private videos and photos, company finance data, government documents, etc. By storing their data into the cloud, the data owners can be relieved from the burden of data storage and maintenance so as to enjoy the on-demand high quality data storage service. However, the fact that data owners and cloud server are not in the same trusted domain may put the our sourced data at risk, as the cloud server may no longer be fully trusted in such a cloud environment due to a number of reasons: the cloud server may leak data information to unauthorized entities or be hacked. It follows that sensitive data usually should 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.

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.

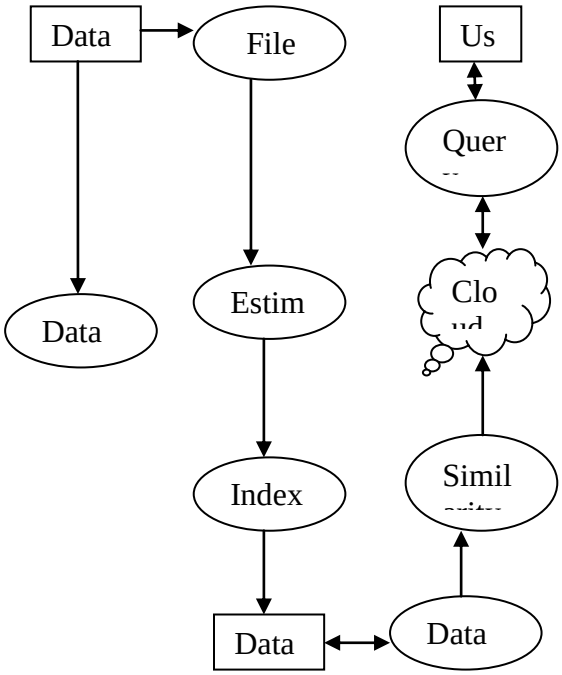


**Figure 5.1: Encrypted Search Model**

### *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. Score Assignment*

The score assignment module is designed to assign the score values for the transactions. The similarity value is estimated to assign the score values. The relevance score is used to rank the transaction data values. The data retrieval is

 

carried out with the score functions. The incremental data update initiates the dynamic score assignment process. The dynamic score assignment process updates the score values based on the new transaction data values.

### *D. 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.

### *E. Query Process*

The query process module is designed to fetch the transactional data values. Query keyword is collected from the client. The query keyword is encrypted and transferred to the data source. The data source performs the searching process. The transactional data values are compared and similarity values are estimated. The results are prepared using the similarity value and threshold levels. The client application decrypts the transactional data values and produces the results in a ranked way.

## VI. PROBLEM STATEMENT

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.

### *A. The System and Threat Model*

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.

## VII. CONCLUSION

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. The system is designed to solve the problem of supporting efficient ranked keyword search for achieving effective utilization of remotely stored encrypted data in Cloud Computing. We first give a basic scheme and show that by following the same existing searchable encryption framework, it is very inefficient to achieve ranked search. Extensive experimental results demonstrate the efficiency of our solution. The secure ranked keyword search framework can be enhanced to support distributed data retrieval mechanism. The integrity verification schemes can be integrated with the system to check the data over the storage and transmission process. We then

 

appropriately weaken the security guarantee, resort to the newly developed crypto primitive OPSE, and derive an efficient one-to-many order preserving mapping function, which allows the effective RSSE to be designed.

## VIII. 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.

 