Global Journal of Computer Science and Technology: E
Network, Web & Security
Volume 15 Issue 1 Version 1.0 Year 2015
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Performance Study of Cryptography Based Dynamic Multi-Keyword Searchable Security Algorithm in Cloud using CRSA /B+ Tree

By Prasanna B T & C B Akki

East Point College of Engineering and Technology, India

Abstract- Today, Cloud computing is a buzz word in IT industry. Cloud, a shared pool of computing resources, allows access to needed resources on demand through internet and web applications. Since data is outsourced to third party, user needs to maintain the accountability of their data in cloud. Hence preserving the confidentiality and securing the sensitive data in cloud is a major concern. Many cryptographic techniques have been proposed by researchers to assure the confidentiality of the user's data in cloud. But, the challenging task is to provide the secure search over this preserved data which has been encrypted so as to retrieve the effective data. Hence, we are proposing a system to have a secure search over the encrypted data on the cloud which preserves its confidentiality. In our system, a noble approach has been made using the Commutative-RSA algorithm, a cryptographic technique where the dual encryption takes place thus reducing the overall computation overhead. The search operation over the encrypted data is based on the tree search algorithm which supports multi-keyword search. Based on the relevance score, the more appropriate data is retrieved on the search operation. Using this approach, the information is not leaked when the encrypted data is searched by users and also the queries are handled in an efficient way. Finally, we demonstrate the effectiveness and efficiency of the proposed schemes through extensive experimental evaluation.

Keywords: cloud, searchable encryption, CRSA, B+ tree.

GJCST-E Classification : C.2.0

Strictly as per the compliance and regulations of:

Performance Study of Cryptography Based Dynamic Multi-Keyword Searchable Security Algorithm in Cloud using CRSA /B+ Tree

Prasanna B T[α] & C B Akki[σ]

Abstract- Today, Cloud computing is a buzz word in IT industry. Cloud, a shared pool of computing resources, allows access to needed resources on demand through internet and web applications. Since data is outsourced to third party, user needs to maintain the accountability of their data in cloud. Hence preserving the confidentiality and securing the sensitive data in cloud is a major concern. Many cryptographic techniques have been proposed by researchers to assure the confidentiality of the user's data in cloud. But, the challenging task is to provide the secure search over this preserved data which has been encrypted so as to retrieve the effective data. Hence, we are proposing a system to have a secure search over the encrypted data on the cloud which preserves its confidentiality. In our system, a noble approach has been made using the Commutative-RSA algorithm, a cryptographic technique where the dual encryption takes place thus reducing the overall computation overhead. The search operation over the encrypted data is based on the tree search algorithm which supports multi-keyword search. Based on the relevance score, the more appropriate data is retrieved on the search operation. Using this approach, the information is not leaked when the encrypted data is searched by users and also the queries are handled in an efficient way. Finally, we demonstrate the effectiveness and efficiency of the proposed schemes through extensive experimental evaluation.

Keywords: cloud, searchable encryption, CRSA, B+ tree.

I. Introduction

Cloud computing is getting popularity because of its user friendly nature. IT industry leaders believe that cloud will change the approach of IT business. Reduced cost for storing data in and retrieving data from cloud is the biggest driver for its expected growth. This technological methodology can save a lot of infrastructure cost. Pay-as-you-use model can also be offered through the cloud computing solutions. According to Gartner Inc. Cloud computing is a disruptive technology, with the potential to make IT organizations more responsive than ever [8]. Cloud computing promises scalability, reliability, flexibility, availability and security of data along with economic advantages to the end user. Through web based applications one can access the shared resources from cloud on demand. IDC India lead analyst (software and services research), Kamal Vohra in an interview with leading paper during 2010 quoted, "The most attractive feature of this new technology is the prospect of converting large, upfront capital investments in IT infrastructure into smaller, manageable 'pay-per-use' annuity payments." In 2013 IDC in a press release revealed that spending on public IT cloud services will reach $58.4 billion in 2015 and is expected to be more than $107 billion in 2017. Over the 2013–2017 forecast period, public IT cloud services will have a compound annual growth rate (CAGR) of 23.5%, five times that of the industry overall. Software as a service (SaaS) will remain the largest public IT cloud services category, capturing 59.7% of revenues in 2017. IDC predicts that by 2017, 80%+ of new cloud apps will be hosted on six PaaS platforms [1].

Along with benefits, cloud computing also has its own challenges and issues that need to be tackled. International Data Corporation (IDC) identified some of major challenges in cloud like Security, Performance, Availability, Integration and Cost. Among other challenges mentioned, privacy and security of data in cloud is the prime most concern where in research community needs to look in. Since cloud is basically based on a trust model, where client and provider must trust each other, we need to consider the issues related to security before making cloud a successful technology. Constant efforts have been put to ensure the privacy and confidentiality of the data at rest since long time.

Access to physical resources like servers is not under the control of an organization that outsources data on to the cloud for storage [18]. This in turn makes organization sensitive data vulnerable to risk [4] [19] [5]. Data in the cloud is typically in a shared environment along with the data from other resources. Cryptography is one effective way of securing user data in cloud. Before the data is been hosted or stored onto the cloud, the sensitive data is encrypted to protect the data privacy [2].To preserve privacy many techniques in the literature have been proposed to carry out work on encrypted data. In [23][24][25], extensive literature review has been done. Most of the research on encrypted data is concentrated on homomorphic encryption (HE) and searchable encryption (SE) techniques. The mathematical complexity involved in designing homomorphic encryption methods led

Author α σ: e-mail: prasi.bt@gmail.com

researchers to concentrate on searchable encryption methods. Searchable encryption schemes work on encrypted data at rest and perform search operation on encrypted data. There may be situations, where in an organization the end users may have to perform the search operation on the daily basis for their business needs which also may involve some queries or multiple-keyword [17] to search in a huge set of cloud data. As a result, privacy assured multiple-keyword search operation over the encrypted data must be performed in the cloud data which assures that the sensitive data is not leaked [16]. The existing Multi Ranked Keyword Searchable Encryption (MRSE) mechanism [6] is not able to process the performance of system where the computation is more.

The need for efficient high performance multi-keyword search over the encrypted cloud data which assures privacy is evident. Thus, here we aim at providing such mechanism in the cloud in an efficient way so that the most relevant document is retrieved. Achieving this is a challenging task. Due to the complexity of both, preserving the data and secure multi-keyword search, the methods we have proposed here is organised in an adaptive manner. A novel technique like commutative RSA algorithm for the encryption of document can be used to preserve privacy/security of the data. This enables the cloud service to efficiently process the encryption method on the documents to be outsourced on cloud which also proves through extensive tests that this approach is dynamic.

Now the cloud service should perform the search operation on the encrypted data in an efficient way, so as to retrieve only the particular data. Here we propose the tree search algorithm due to its performance, system usability and scalability which enables data users to find the most relevant information quickly, rather than in a group of retrieved documents [3]. The search operation is performed by the two main cloud workers. One worker is in charge of constructing the tree structure where each block (node) of data chunks represents whether corresponding keyword is contained in the document. The other worker takes care of the query keyword search on the tree based with the relevance score. The search result of the query represents whether corresponding keyword appears in this search data request, so the similarity could be exactly measured by the data chunks based on the query. The analysis of effectiveness of the privacy and efficiency of the proposed system is performed. Hence, the experimental result proves the low computational overhead and the high efficiency.

The research paper is organized as follows. Section II discusses the related work. A brief note on the Commutative-RSA along with the working of search operation based on tree is discussed in section III. Section IV discusses the search framework on Azure platform using tree. The analysis is done in section V. The experimental results and comparisons are presented in section VI, to prove the efficiency of our proposed system. The concluding remarks and references are provided in the last VII and VIII sections of the paper respectively.

II. Related Work

More research has been done in privacy preserving single-keyword and multi-keyword search on encrypted data in cloud. In [5], a practical symmetric searchable encryption method is proposed for the first time. The first public key encryption with keyword search (PEKS)was proposed by in [15]. In [7] [9] [10] [11] [12] [13][15][20] extensive research has been done to make searchable encryption practical. Later, the schemes in [6] [22] use multi keyword search technique for search.

In [23][24][25], authors have discussed about the different techniques used to work on encrypted data, extensive survey of different searchable and homomorphic encryption schemes with their benefits and limitations.

To overcome all these issues, we have proposed a system of privacy preserving multiple-keyword ranked search over encrypted cloud data which is secure and has efficient search method. Thus, a high performance security model with multi-keyword search evaluation mechanism is proposed.

III. Problem Formulation

Searchable Encryption (SE) schemes maintain the confidentiality and privacy of owner's data by facilitating searching keywords directly on encrypted data. Users can upload their encrypted data to cloud. Later, the authorized users can perform private keyword search on encrypted data in cloud. Multiple domains like cryptography, indexing, storage etc. are involved in devising efficient, secure, SE algorithms over encrypted files. The participants of a secure search model in a cloud, typically involves data owner, data user and cloud server. Data owner encrypts the files and corresponding keywords based index files by using any known cryptographic algorithms. Both the encrypted files and index files are uploaded to the cloud server. The trapdoors (encrypted keywords) are used to search encrypted files by cloud server in cloud database.

a) System Model

Our system consists of 3 entities data owner, data user and the cloud server as shown in Figure 1.

1. Data owner encrypts the data files for securing the data in cloud using Commutative RSA (CRSA) before uploading into the cloud. They also define the access rights for the user who want to access those documents. The access right is a 2-state variable: permission granted or permission denied.

Data owner creates an index tree based on B tree and encrypts the tree using CRSA.

2. Cloud server stores the encrypted data files and encrypted index tree. It accepts the encrypted keywords (trapdoor) and returns the matching data file based on their relevance.
3. Data user can search for encrypted data files in cloud with encrypted keywords (trapdoor). The purpose of using encrypted keywords is that even the cloud server must not be able to infer the contents of data files.

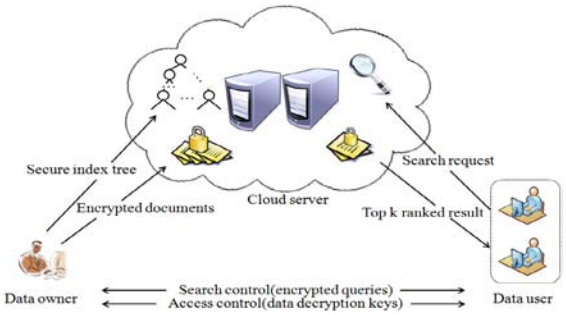

Figure 1 : Searchable Encryption Architecture using CRSA

b) Design Goals

The proposed solution addresses the following requirements

1. The search on encrypted document/file must be fully secure and cloud server must not be able to infer the contents of the documents in any way.
2. The search results must be ranked in order of relevance.

To enable ranked searchable 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.

i. *Existing systems*

Existing searchable encryption schemes [6] [15] [38] allow a user to securely search over encrypted data through keywords. These techniques support multi keyword search. The similarity measure "coordinate matching" in MRSE [6] has some drawbacks when used to evaluate the document ranking order. First, it takes no account of term frequency i.e. any keyword appearing in a document will present in the index vector as binary value 1 for that document, irrespective of the number of its appearance. Obviously, it fails to reflect the importance of a frequently appeared keyword to the document. Second, it takes no account of the term scarcity. Usually a keyword appearing in only one document is more important than a keyword appearing in several ones. In addition, long documents with many terms will be favored by the ranking process because they are likely to contain more terms than short documents. Hence, due to these limitations, the heuristic ranking function, "coordinate matching", is not able to produce more accurate search results. More advanced similarity measure should be adopted from plaintext information retrieval community. On the other hand, the search complexity of MRSE is linear to the number of documents in the dataset, which becomes undesirable and inefficient when a huge amount of documents are present.

ii. *Proposed system*

For our system, we choose the B+-tree as indexing data structure to identify the match between search query and data documents. Specially, we use inner data correspondence, i.e., the number of query keywords appearing in document, to evaluate the similarity of that document to the search query. Each document is converted to a balanced B+-tree according to the keywords and encrypted using CRSA. Whenever user wants to search, he/she creates a trapdoor for the keywords. Our aim is to design and analyze the performance of multiple-keyword ranked search scheme using Commutative RSA algorithm and B+ tree data structure for searchable index tree.

We designed a scheme based on secured ranked multiple-keyword search over encrypted cloud data using CRSA. Further, we analyzed its performance over B+ tree based searchable index tree. In [6] [38], authors have studied the performance of RSA algorithm on binary tree. We have used Microsoft's Azure platform to emulate the proposed system and to study its performance.

c) Preliminaries

i. *Commutative Encryption (CRSA)*

The RSA cryptosystem is one of the optimum public key cryptography approaches. However, its overall robustness gets limited due to one way encryption and majority of existing RSA schemes suffer from reorder issues. Therefore, in order to make this system least complicated and more efficient, an approach called Commutative RSA has been proposed. In this scheme, the order in which encryption has been done would not affect the decryption if it is done in the same order. Encryption is the standard method for making a communication private. With the many cryptographic approaches, our system follows the commutative RSA algorithm. The mathematical scheme

for performing this encryption is described by a pseudo algorithm presented below.

Let us consider two prime numbers $Prime\_P_p^{CRSA}$ and $Prime\_Q_q^{CRSA}$ initialized amongst all the group members. Let G_A and G_B represent the group members required to communicate over the documents. To compute the encryption keys and decryption key pairs of the commutative RSA algorithm the parameters $Param\_N^{CRSA}$ and $Param\_\phi^{CRSA}$ are computed using the following

$$Param\_N^{CRSA} = \left[\left(Prime\_P_p^{CRSA}\right) \times \left(Prime\_Q_q^{CRSA}\right)\right]$$

$$Param\_\phi^{CRSA} = \left[\left(Prime\_P_p^{CRSA} - 1\right) \times \left(Prime\_Q_q^{CRSA} - 1\right)\right]$$

From the above equations it is clear that

$Param\_N_X{}^{CRSA} = Param\_N_Y{}^{CRSA}$ and

$Param\_\phi_X{}^{CRSA} = Param\_\phi_Y{}^{CRSA}$ for X and Y.

The encryption key pair of X and Y are represented as $(Param\_N_X{}^{CRSA}, Param\_E_X{}^{CRSA})$ and $(Param\_N_Y{}^{CRSA}, Param\_E_Y{}^{CRSA})$ is to be obtained.

The $Param\_E^{CRSA}$ is obtained by randomly selecting numbers such that it is a co-prime of $Param\_\phi^{CRSA}$ or in other terms

$$\mathcal{Fn}_{GCD}(Param\_E^{CRSA}, Param\_\phi^{CRSA}) = 1$$

Where $\mathcal{Fn}_{GCD}(u, v)$ represents the greatest common divisor function between two variables u and v.

The decryption key pair of X and Y is represented by $(Param\_N_X{}^{CRSA}, Param\_D_X{}^{CRSA})$ and $(Param\_N_Y{}^{CRSA}, Param\_D_Y{}^{CRSA})$ and the parameter $Param\_D^{CRSA}$ is computed based on the following equation

$$Param\_D^{CRSA} = (Param\_E^{CRSA})^{-1} Mod(Param\_N^{CRSA})$$

Let Enc_U represent the encrypted data U. The encryption operation is defined as follows

$$Enc_U = U^{Param\_E^{CRSA}} Mod(Param\_N^{CRSA})$$

The commutative RSA decryption operation on the encrypted data V is defined

$$Dec_V = V^{Param\_D^{CRSA}} Mod(Param\_N^{CRSA})$$

ii. *B+ Tree*

A B+ tree is a data structure as shown in Figure 2. The tree contains index nodes and leaf nodes. All leaf nodes are at the same level (same depth). Each index nodes contain only keywords. Each node except root node in a B+-tree with order n must contain keys between n to 2n keys. Each node also contains (number of keys + 1) pointers to its child nodes. If the root node is an index node then it must have at least 2 children. The insertion, deletion, search operations takes only logarithmic time. Due to high fan-out B+ tree reduces I/O operations time to search an element.

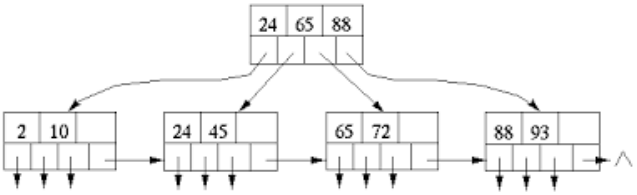

Figure 2 : B+ tree data stucture

IV. B+Tree Algorithm Search Framework Using Microsoft Windows Azure

To enable effective, efficient and secure multi-keyword ranked search over encrypted cloud data under many models, our mechanism is aiming to achieve the following design goals. The proposed framework is mainly on the tree which is index format, hence balanced binary search trees. In this section, we define the framework of multi-keyword ranked search over encrypted cloud data and establish various strict system-wise privacy requirements for such a secure cloud data utilization system.

The Cloud service architecture (CSA) chosen enable us to realize the framework using a modularized approach. The CSA shown in Figure 3 could be considered as a complex system of $\mathcal{p} : q$ dependencies, where $\mathcal{p}$ represents the services offered and q represents the applications offered by the CSA system. In CSA an application may need multiple service offerings or varied applications need similar services or similar applications may be provided by varied services. The searchable encryption utilizes a similar application of keyword search provided by the n workers hence it could be said that the searchable encryption depends on the availability of the keyword search application offered by the n workers. Multi-keyword search management tends to be cumbersome if it is done manually. In order to automate the multi-keyword search management we need a common syntax and a common searchable encryption to interoperate. The Commutative RSA have standardized the syntax definition through the searchable encryption.

Let us consider a set of all multi-keyword search S_K defined as

$$S_K = \{s_{k_1}, s_{k_2}, s_{k_3}, \ldots\ldots\ldots s_{k_a}\}$$

where s_{k_a} represents the a^{th} search keyword. The s_{k_a} is a keyword derived from both the encrypted tree data $r_{k_a} \in R_K$ from tree search algorithm (TSA) and the encrypted keyword contents $o_{k_a} \in O_K$ from the encrypted tree data R_K of the TSA. In other words

$$s_{k_a} = f_k(r_{k_a}, o_{k_a})$$

where $o_{k_a} = f_o(R_K, r_{k_a})$, f_o represents the tree builder function.

The tree builder function extracts all the related keywords of r_{k_a} present in the encrypted tree data R_K of the TSA.

The B^+ tree represents a complex CSA hence the encrypted tree data R_K data set is available with

$$R_K = \{r_{k_{11}}, r_{k_{12}}, \ldots\ldots r_{k_{1a}}\} \cup \{r_{k_{21}}, r_{k_{22}}, \ldots\ldots r_{k_{2a}}\} \cup \ldots\ldots \{r_{k_{n1}}, r_{k_{n2}}, \ldots\ldots r_{k_{na}}\}$$

The encrypted keyword contents extracted from the encrypted tree data could be defined as $O_K = O_{K1} \cup O_{K2} \cup \ldots\ldots \cup \ldots\ldots O_{Kn}$, where O_{Kn} the encrypted set available with n^{th} search service provided by the worker. The locally available encrypted data could be defined as $O_{Kn} \propto R_K$

From the above definition it is clear that encrypted data available with search service n provided by the worker may not contain all the possible keywords as the complete encrypted tree data set R_K is unavailable with the n^{th} search. This is the problem that exists in the current search deployments available [9]. The purpose of the B^+ tree is to overcome the short comings by using efficient searching algorithms and search encryption compositions.

a) Cloud Workers

The workers provide search encryption services which support the multi-keyword search application. The workers are defined as $WR = \{Wr_1, Wr_2, Wr_3, \ldots\ldots Wr_n\}$, where Wr_n is the n^{th} search provided by the worker. The system architecture of the azure cloud search over an encrypted data by the worker is shown in Figure 1. Each search provided by the worker possess the encrypted tree data based documents. The encrypted tree data records could be represented as $R_{KB} = \{rkb_1, rkb_2, rkb_3, rkb_4, \ldots\ldots\ldots rkb_r\}$, where rkb_r is the r^{th} encrypted tree data record available with the azure cloud search provided by the worker $Wr_n \in WR$ on the n^{th} search.

The encrypted tree data records are said to consist of triplets. Based on the record, rkb_r could be represented as $rkb_r = < trkb_{r_{sub}}, trkb_{r_{prd}}, trkb_{r_{obj}} >$ where $trkb_{r_{sub}}$ is the subject triplet, $trkb_{r_{prd}}$ is the predicate triple and $trkb_{r_{obj}}$ represents the object triplet.

The keywords extracted from the encrypted tree data include some complex relations that cannot be represented in encrypted tree data alone, hence the B^+ tree presented here adopts representation of the encrypted keyword contents through tree structure builder due to its benefits.

The data of the cloud search provided by the worker constitutes of both the encrypted tree data and encrypted keyword contents which are humongous in nature and size. A search executed on huge databases would affect the response times due to numerous disk read and disk write operations involved in the search operation. To compress the data and create cache the B^+ tree utilizes a hierarchical data ordering algorithm.

n^{th} search service provided by the workers. The encrypted tree data can be defined as

$$R_K = R_{K1} \cup R_{K2} \cup R_{K3} \cup \ldots\ldots R_{Kn}$$

where $R_{K1} \neq R_{K2} \neq R_{Kn}$

b) Azure Cloud Search Application

The search application is a user interface which accepts user search queries represented by SS_Q. The B^+tree search algorithm accepts logical, conditional and simple term based search queries. The response of the search is represented as SS_R. The cloud search application provides the search responses SS_R by using cloud search service composition techniques. The search response not only consists of search responses but additionally provides the encrypted relevance score used in ranking the search responses i.e. higher the encrypted relevance score greater is the rank of the search response. The encrypted data are constructed after consuming the search services provided by the n search service. These are provided by the search application from the cloud worker. The encrypted data are constructed by the possible keywords obtained after the cloud search service composition. This enables the B^+tree search algorithm to provide better search results and overcome the drawback currently discussed in the previous section of this paper.

Let us consider search keyword set S_K and two keywords $s_{k_x} \in S_K$ and $s_{k_y} \in S_K$. There exists 4 possible relations amongst keywords s_{k_x} and s_{k_y}. The possible relations could be defined by using the subsume represented by Sb_{sum} and defined as

$Sb_{sum} : (S_K \times S_K) \mapsto \{T, F\}$, where T represents the conditional true relation and F represents a conditionally false relation. Using the above definition we could define the first possible relation between the keywords s_{k_x} and s_{k_y} as $Sb_{sum}(s_{k_x}, s_{k_y}) = T$ holds if and only if the search keyword s_{k_x} is a generalization of the search keyword s_{k_y}. It could be stated that the search keyword s_{k_y} is a specialization of the search keyword s_{k_x}.

$Sb_{sum}(s_{c_y}, s_{c_x}) = T$ holds if and only if the search keyword s_{k_x} is a generalization of the search keyword s_{k_y} .It could be stated that the search keyword s_{k_y} is a specialization of the search keyword s_{k_x}.

If the search keywords s_{k_x} and s_{k_y} are not related then $Sb_{sum}(s_{k_x}, s_{k_y}) = F$ and $Sb_{sum}(s_{k_y}, s_{k_x}) = F$.

If the search keywords s_{k_x} and s_{k_y} are equal then $Sb_{sum}\left(s_{k_x}, s_{k_y}\right) = T$ and $Sb_{sum}\left(s_{k_y}, s_{k_x}\right) = T$.

The generalization, specialization and the subsume Sb_{sum} relations are transitive.

Let us consider a parameter p_x of the search service provided by the worker Wr_x and a parameter p_y of the search service provided by the worker Wr_y. If the parameters $p_x = p_y$ then the cloud search service could be called if only $Sb_{sum}(Wr_x, Wr_y) = T$. It could also be stated that the parameter p_x requires less or equal data than the parameter p_y. For the cloud search service composition required, there is no requirement for a demarcation amongst the keywords and the search keywords. Let's define a set of cloud search services available with the search application as follows

$$Sws_{Wr} = \{ sws_{Wr_1}, sws_{Wr_2}, \ldots\ldots sws_{Wr_n} \}$$

Where sws_{Wr_n} represents the n^{th} cloud search service offered by search service provided by the worker Wr_n.

Each cloud search service offered by search worker Wr_n required a set of inputs denoted as $ss_{Q_{Wr_n}}$ and if the set of inputs is provided in an orderly fashion the cloud search service provides a set of output keywords denoted by $ss_{R_{Wr_n}}$ and $ss_{R_{Wr_n}} \in S_K$. The efficient ranked keyword search cloud service composition algorithm discovers the cloud search services available on Sws_{Wr}. On successful execution of the cloud search service execution algorithm, the next cloud search service i.e. sws_{Wr_2} could be processed only if the execution of the previous sws_{Wr_1} (provided with the input parameters $ss_{Q_{Wr_1}}$ and the output keywords $ss_{R_{Wr_1}}$ are obtained in response) is processed successfully. Let the ranked keyword search cloud service composition be represented as $Comp_{Sws}(Sws_{Wr})$ then the cloud search service composition is said to successfully process all the requests if

$$Comp_{Sws}(Sws_{Wr}) \leftrightarrow \forall \mathcal{X} \in ss_{Q_{Wr_1}} \exists \mathcal{Y} \in SS_{Q_{Wr}} : Sb_{sum}(\mathcal{X}, \mathcal{Y}) \wedge \forall \mathcal{X} \in ss_{Q_{Wr_z}}, z \in \{2, 3, \ldots, n\}$$

$$\exists \mathcal{Y} \in SS_{Q_{Wr}} \cup ss_{R_{Wr_{z-1}}} \cup \ldots\ldots \cup ss_{R_{Wr_1}} : Sb_{sum}(\mathcal{X}, \mathcal{Y}) \wedge \forall \mathcal{X} \in ss_{R_{Wr_1}} \cup \ldots\ldots \cup ss_{R_{Wr_n}} \cup SS_{Q_{Wr}} : Sb_{sum}(\mathcal{X}, \mathcal{Y})$$

Let f_{SSWS} represent a service provided by worker on search function based on a keyword S_K which provides all the set of cloud search services available defined as

$$f_{SWr}(S_C) = Sws_{Wr}$$

Also it could be stated that

$$\forall s_{k_a} \in f_{SWr}(S_C) \exists ss_{R_{Wr_a}} \in SS_{R_{Wr}} : Sb_{sum}\left(S_C, ss_{R_{Wr_a}}\right)$$

The search application is an interface which provides the search criteria to the composed services, the results obtained are then there by provided to the user. On receiving the user's search query SS_Q the application of the B^+tree search algorithm performs the cloud services search function f_{SSWS}. The cloud service offerings amongst the varied workers are obtained by the process invoked by the f_{SSWS}. Based on the cloud services offered and the user query, appropriate cloud services are selected. The selected cloud service offerings Sws_{Pr} are composed using the cloud search service composition function $Comp_{Sws}(Sws_{Pr})$. On completing the composition, the cloud search services are invoked by parsing the required user parameters SS_Q. The results obtained are aggregated and ranked, based on the encrypted relevance score. Higher is the encrypted relevance score, higher is the rank. The ranking could be easily achieved using any sorting algorithm.

Let the cloud search response set be defined as

$SS_R = \{ ss_{R_{Wr_1}}, ss_{R_{Wr_2}}, ss_{R_{Wr_3}}, \ldots\ldots ss_{R_{Wr_n}} \}$, where $ss_{R_{Wr_n}}$ represents the search response received from the n^{th} search service by the worker for a given query set SS_Q.

As stated earlier the search algorithm available at the worker's end, provides the result page information, the encrypted data behind the search and the encrypted relevance score (ranked data). Based on this argument $ss_{R_{Wr_n}}$ could be defined as

$$ss_{R_{Wr_n}} = \{ r_1 ss_{R_{Wr_n}}, r_2 ss_{R_{Wr_n}}, r_3 ss_{R_{Wr_n}} \ldots\ldots r_m ss_{R_{Wr_n}} \},$$

where $r_m ss_{R_{Wr_n}}$ represents the m^{th} search result received from the n^{th} search service by the worker for a given query set SS_Q.

The cloud service composition is an important entity of the cloud search application. The next section of this paper discusses the B^+ tree search algorithm utilized in composing the cloud services Sws_{Wr} offered by the n search service provided by the worker.

c) Cloud Service Composition Using B^+ Tree Search Algorithm

The search framework B^+tree search algorithm introduced in this system utilizes the B^+ tree search algorithm for cloud service composition. The B^+ tree search algorithm is selected for the sole purpose of quicker responses it offers and it is computationally lighter when compared to other cloud service composition algorithms. The cloud service composition function introduced in the earlier section of this paper

$Comp_{Sws}$ (Sws_{Wr}) receives the set of cloud services Sws_{Wr} over which the composition has to be performed. The cloud services composition is performed using the B^+tree search algorithm.

Let us define a function f_{SWS-DS} which performs the B^+tree search algorithm is defined as

f_{SWS-DS} (ss_Q , ss_R , Sws_{tmp} , d_c) = sws, where ss_Q represents the input query set, ss_R is the desired response, Sws_{tmp} represents the current temporary cloud services identified, d_c represents the height and sws represents the resultant cloud service identified.

The f_{SWS-DS} is solved by the following algorithm

Step 01: START
Step 02: For Each $Var_1 \in ss_R$
Step 03: *For Each* $sws_{Wr} \in f_{SWr}(S_K) = Sws_{Wr}$
Step 04: Initialization $ss_{R_{tmp}} = ss_R$
Step 05: *For Each* $Var_2 \in ss_{R_{tmp}}$
Step 06: *IF* $\exists\, Var_3 \in ss_{R_{Wr}} : Sb_{sum}(Var_2, Var_3)$
Step 07: $ss_{R_{tmp}} = ss_{R_{tmp}} / Var_2$
Step 08: *End IF*
Step 09: *End For Each*
Step 10: *For Each* $Var_4 \in ss_{Q_{tmp}}$
Step 11: *IF* $\exists\, Var_5 \in ss_{Q_{Wr}} : Sb_{sum}(Var_4, Var_5)$
Step 12: $ss_{R_{tmp}} = ss_{R_{tmp}} \cup Var_4$
Step 13: *End If*
Step 14: *End For Each*
Step 15: $sws_{tmp} = sws_{Wr} \oplus Sws_{tmp}$
Step 16: *IF* $ss_{R_{tmp}} = \{\}$
Step 17: *Return* sws_{tmp}
Step 18: *End IF*
Step 19: *ELSE*
Step 20: *IF* $d_c < d_{max}$
Step 21: $ss_{R_{tmp}} = f_{SWS-DS}\left(ss_Q , ss_{R_{tmp}} , sws_{tmp} , d_c + 1 \right)$
Step 22: *End IF*
Step 23: *IF* $sws_{tmp} \neq \{\}$
Step 24: *Return* sws_{tmp}
Step 25: *End IF*
Step 26: *End ELSE*
Step 27: *End For Each*
Step 28: *End For Each*
Step 29: *Return* { }
Step 30: *END*

Where $Var_1, Var_2, Var_3, Var_4, Var_5$ represent temporary processing variables and d_{max} represents the maximum depth.

The cloud service composition function denoted by $Comp_{Sws}$ (Sws_{Wr}) is realized using the following algorithm

Step 01: START
Step 02: Initialization $d_{max} = 2$
Step 03: *DO*
Step 04: $Sws_{tmp} = f_{SWS-DS}\left(ss_{Q_{Wr_n}} , ss_{R_{Wr_n}} , \{\}, 1 \right)$
Step 05: $d_{max} = d_{max} + 1$
Step 06: *While* $Sws_{tmp} \neq \{\}$
Step 07: *END*

provided by the worker offering the cloud search services to support the search application.

The CSA architecture considered for the B+tree search algorithm is described in this section. The B+tree search algorithm is designed to provide appropriate search responses. The B+tree search algorithm relies on the encrypted tree data and the encrypted keyword contents housed as the encrypted data component of the cloud service provided by the worker for provisioning of the search responses. The cloud search services offered by the worker are composed using the B+tree search algorithm.

The encrypted keyword contents of r encrypted tree data records is defined as

$$O_{KB} = \{ okb_1, okb_2, okb_3, okb_4, \ldots\ldots\ldots okb_r \}$$

Let the cache of a keyword s_{k_a} which represents the a^{th} search keyword be represented as $Cache_{s_{k_a}} = < s_{k_a}, r_{k_a}, e_{k_a} >$, where r_{k_a} is the number of relations of the keyword and e_{k_a} represents the number of edge keywords.

It is evident that greater the number of keywords and greater the relations that exist, larger is the data size and increasing the number of disk operation for the search operation. The number of occurrences of a keyword in an encrypted data is directly proportional or equivalent to the number of relations r_{k_a} of a keyword. Also it can be stated that for a constant m is equivalent to a function of the number of relations $(f_{num\_r})\, r_{k_a}$ of a keyword s_{k_a} and a function of the tree depth $(f_{edg\_dpt\,h})$ of a keyword s_{k_a} .

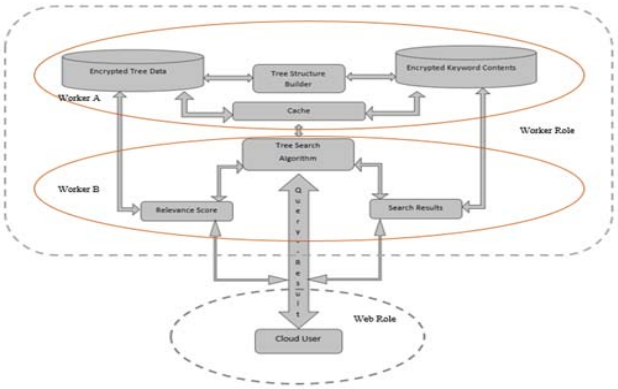

Figure 3 : System Architecture of Azure Cloud search over encrypted data

$$f_{num\_r}(r_{k_a}) \times f_{edg\_dpt\,h}(r_{k_a}) \approx m$$

Also

$$r_{k_a} \approx \sum_{x=1}^{x=m} m/x \approx m \int_{x=1}^{x=m} 1/x\, dx = m \ln m$$

From the above equation it is clear that even if the number of relations r_{k_a} of a keyword s_{k_a} increases, the cache size does not increase by a great extent. Generally the keywords require $2S_{Unit}$ cloud storage space per keyword (s_{k_a}). The space utilized in storing the cache defined above is given by

$$\sum_{r_{k_a}} (\, 2 + f_{num\_r}(r_{k_a})\,) \approx S_{Util}\,(2 + \ln S_{Util})$$

where S_{Util} is the cloud space required to store the same keyword s_{k_a}.

It is considered that only one entry of a s_{k_a} keyword is allowed in the cache. In order to compare the normal caching strategy with the caching strategy used in B^+ tree search, the comparison ratio is defined as

$$\frac{2S_{Util}\left(1 + \ln S_{Util}/2\right)}{2S_{Util} \ln S_{Util}} = 1/\ln S_{Util} + 1/2$$

Hence the proposed caching strategy improves the cloud storage space utilization by approximately 50% .

The azure cloud access cost for the caching strategy is defined as

$$ACost_{Cache} = \sum_{\{r_{k_a} : f_{num\_r}(r_{k_a}) \le t\}} f_{num\_r}(r_{k_a}) \approx \int_{S_{Util}/t}^{S_{Util}} \frac{S_{Util}}{f_{edg\_dpt\,h}}\, df_{edg\_dpt\,h} = S_{Util} \ln t$$

where $f_{num\_r}(r_{k_a}) \le t \leftrightarrow f_{edg\_dpt\,h}(r_{k_a}) \ge S_{Util}/t$

The probability of $AProb$ finding the keyword s_{k_a} in the encrypted data is defined as

The access time of the cache to search for a keyword s_{k_a} within the encrypted data with a probability $AProb_{Cache}$ is defined as

$$AProb_{Cache} = \ln t / \ln S_{Util}$$

$$ATime_{Cache} = AProb_{Cache} \log_b ACost_{Cache} + (1 - AProb_{Cache})(\log_b ACost_{Cache} + 1) = \log_b ACost_{Cache} + (1 - AProb_{Cache})$$

where b represents the branching factor of the encrypted tree.

The cache created based on the encrypted tree data and encrypted keyword content is encoded in a binary format for faster access.

The encrypted relevance score is a ratio between the query keyword and the response keyword based on the encryptions constructed. The encrypted relevance score is used by the Search Application in ranking the search responses received by the n search service provided by workers considered in the B^+tree search.

The search query SS_Q could be defined as a set of keywords and relational operators. The search encrypted service offered supports queries containing Boolean operators like $AND, OR, NOT, +, -, ""$ commonly available with the major search operation provided by the worker.

$$ss_Q = < s_{k\,ss_Q}, R_{K\,ss_Q} >$$

The search query ss_Q could be represented as a $p \times q$ matrix where p represents the number of keywords queried for and q represents the number of relations, logical operators and special characters defined for querying amongst the p keywords.

The search response ss_R is a set of responses and the corresponding relevance score defined as

$$ss_R = < s_{R\,ss_R}, ORS_{R\,ss_R} >$$

The search response ss_R could also be represented as a $r \times r$ matrix where r the number of responses obtained for the search query in ss_Q. The encrypted relevance score is defined as

$$ors_{R\,ss_R}(ss_Q, ss_R) = \frac{\sum_r s_{r\,ss_R}, ss_Q}{\| ss_Q \| \| s_r \|}$$

To represent the encrypted relevance score to a scale of $0\ to\ 1$, Normalization is considered in the B^+tree search hence the encrypted relevance score could be defined as

$ors_{R\,ss_R}(ss_Q, ss_R) = s_{r\,ss_R}', ss_Q$, where $s_{r\,ss_R}' = \frac{s_{r\,ss_R}}{s_r}$

The Azure cloud search provided by worker could be considered as the core of the B^+tree search architecture. The worker discussed in this section not only rely on the encrypted tree data to provide effective search queries but also rely heavily on the encrypted keyword contents to provide effective and accurate search responses. The cloud search worker not only incorporates effective hierarchical caching strategies enhancing query response time but also provide relevant query responses. In addition to the query responses, the search worker also provide encrypted relevance scores associated with each query responses enabling effective ranking when multiple cloud search response are composed.

V. Performance Analysis

The security of the designed system is provided by using CRSA. As long as private key (encrypted) is kept secret the cloud provider cannot deduce index tree or documents set. Since trapdoor is also encrypted using CRSA, the provider cannot make out the keywords inside the trapdoor maintaining the confidentiality at index and query level. The documents in cloud storage are also protected, since documents are encrypted using CRSA. Without having the decryption key it is highly hard to decrypt the documents thus provides security at storage level.

To be useful and usable, databases must support operations, such as search, deletion and insertion of data. For large organizations the databases are huge in size and cannot be maintained entirely in memory. By using balanced B+ trees to construct the index for the data we can improve the search efficiency. B+ tree minimizes the disk I/O (disk read and disk write) by copying a block of data (page) containing many records at a time into memory. This in turn improves the search efficiency. Asymptotically, Searching an unsorted database without indexing will have a worst case running time of O(n), where n represents the number of keywords. If the same data is indexed with a B+ Tree, the same search operation will run in logarithmic time i.e O(log n).

VI. Result Analysis

The privacy preserved multi-keyword search based on the encrypted cloud data is been implemented. The system model presented has been developed on Visual Studio 2010 framework 4.0 with C#. The overall system has been developed and implemented with Microsoft Azure platform.

Different parameters like computation overhead, computation time, and bytes overhead have been considered to study and compare the performance of our proposed scheme with existing scheme.

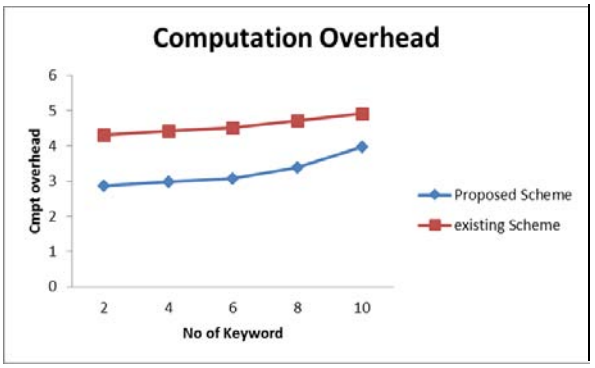

Figure 4 : Computation Overhead

Figure 4 depicts the computation overhead in seconds based on the number of keywords. In this study, we compared the performance of our proposed system with the existing system proposed in [15]. Results clearly show that even for 10 keywords, the overhead computation using CRSA is low as compared to the existing system [15]. For example, existing system takes approximately 4.5 seconds for searching 2 keywords, whereas our proposed CRSA based scheme takes only 3 seconds. The computation cost for search increases linearly in both schemes. But from Figure 4 it is evident that our proposed CRSA based scheme performs better even under increased number of keywords.

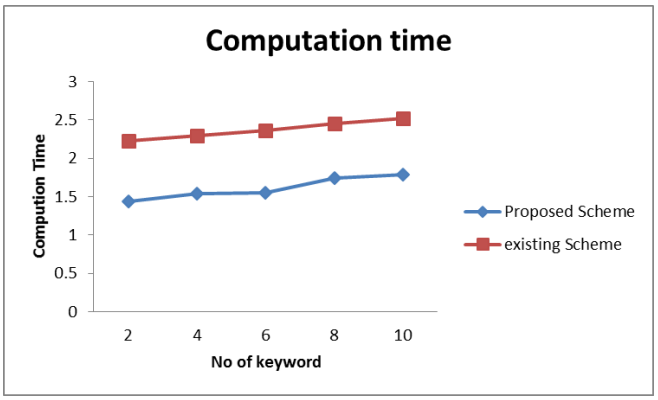

Figure 5 : Time Computation

The graph in Figure 5 plotted above makes the comparison of the search computation time in seconds of our proposed system against the existing system. For two keywords search, the time taken by the existing [15] scheme is approximately 2.5 seconds, whereas our proposed system takes approximately 1 seconds less. As the number of keywords increased for search, the computation time for search also increases linearly in both schemes. But CRSA based scheme is found to perform better. Thus it is evident that encryption algorithm CRSA with B+ tree as index tree performs better than RSA and B tree combination.

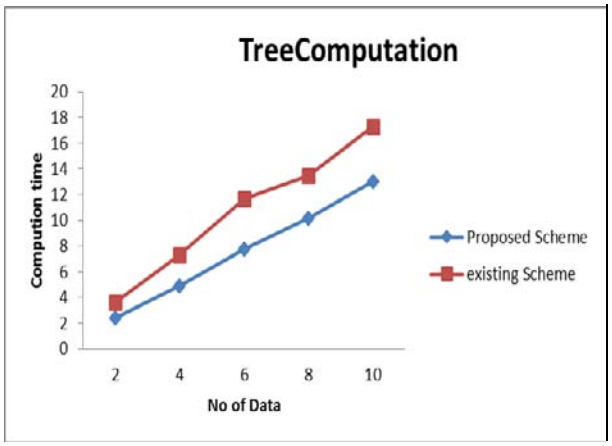

Figure 6 : Computation of Tree Structure

The graph in Figure 5 plotted above makes the comparison of the search computation time in seconds of our proposed system against the existing system [15]. For two files search, the time taken by the existing scheme is approximately 2.5 seconds, whereas our proposed system takes approximately 0.5 seconds less. As the number of data files increases, the computation time for search also increases linearly in both schemes. But B+ tree index based scheme is found to perform better. Thus it is evident that encryption algorithm CRSA with B+ tree as index tree performs better than RSA and B tree combination.

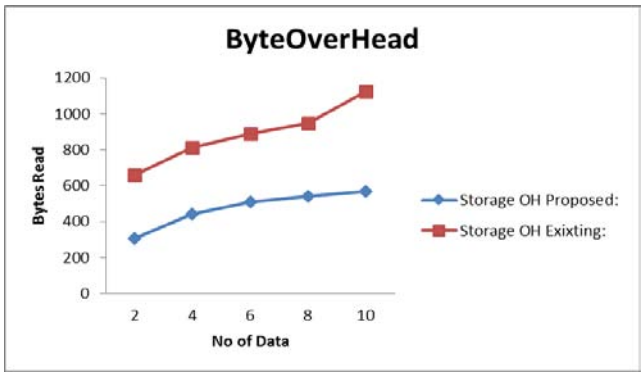

Figure 7 : Computation of Byte Overhead

The graph in Figure 7 portrays the overhead computation in bytes. For two data files the number of bytes read is around 200 bytes compared 600 bytes from existing system. As the number of data files increases the bytes read for search also increases linearly in both schemes. But CRSA/B+ tree based scheme is found to perform better.

Therefore from these results, we have established that the proposed model can be an effective, robust and optimum adaptable approach for privacy preserving multi-keyword search of encrypted data in cloud environment.

VII. Conclusion and Future Work

The insights of privacy-assured searchable cloud data storage services are discussed. Despite the popularity of cloud services and their wide adoption by enterprises and governments, cloud providers still lack services that guarantee both data privacy and privacy preserving search operation on encrypted. Here we tried to address the security issues considering a large set of cloud data and users based on preserving the privacy of multi keyword search over an encrypted data. We have designed a cryptographic scheme using C-RSA and B+ tree. The B^+tree search algorithm is adopted for the ranked search technique to fetch the relevance score so as to retrieve the similarity between the query keyword search performed by the cloud user and the documents which are outsourced on cloud. Detailed analysis which examines the privacy and search efficiency of our proposed model is given. The experimental results proves our proposed model induces low overhead on the overall system. Using the C-RSA, the computation overhead is much reduced which means, if any changes have to be made to the already encrypted documents, it can be easily done with the C-RSA technique which allows dual encryption hence proves it is dynamic, thus reducing the computation overhead compared to other cryptographic methods. Therefore, specifying the computation overheads and comparatively proving efficiency. Finally, we conduct comprehensive performance analysis, which shows that our scheme is more secure, efficient and practical than existing schemes.

The C-RSA cryptographic technique induces low computation overhead with the asymmetric key. This can further be improved with the use of ECC technique which proves much reduced computation overhead with the symmetric keys without compromising security.

References Références Referencias

1. M. Armbrust et al., 'Above the Clouds: A Berkeley View of Cloud Computing,' Feb 2009.
2. S. Kamara and K. Lauter, 'Cryptographic cloud storage,' in RLCPS, January 2010, LNCS. Springer, Heidelberg.
3. A. Singhal, 'Modern information retrieval: A brief overview,' IEEE Data Engineering Bulletin, vol. 24, no. 4, pp. 35–43, 2001.
4. Cloud Security Alliance, 'Security Guidance for Critical Areas of Focus in Cloud Computing,' http://www.cloudsecurityalliance.org, 2009.
5. R. Brinkman, 'Searching in encrypted data,' in University of Twente, PhD thesis, 2007.
6. Ning Cao; Cong Wang; Ming Li; Kui Ren; Wenjing Lou, 'Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data,' Parallel and Distributed Systems, IEEE Transactions on , vol.25, no.1, pp.222,233, Jan. 2014.
7. Dawn Xiaoding Song; Wagner, D.; Perrig, A., 'Practical techniques for searches on encrypted data,' Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, doi: 10.1109/SECPRI.2000.848445 vol., no., pp.44,55, 2000.
8. J. Li et al., 'Fuzzy Keyword Search Over Encrypted Data in Cloud Computing,' Proc. IEEE INFOCOM '10 Mini-Conf., San Diego, CA, Mar. 2010.
9. M. Li et al., 'Authorized Private Keyword Search over Encrypted Data in Cloud Computing,' 31st Int'l. Conf. Distributed Computing Systems, 2011, pp. 383–92.
10. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, 'Public key encryption with keyword search,' in Proc. of EUROCRYPT, 2004.
11. C. Wang et al., 'Secure Ranked Keyword Search Over Encrypted Cloud Data,' Proc. ICDCS '10, 2010
12. Wenjun Lu; Varna, A.L.; Min Wu, 'Confidentiality-Preserving Image Search: A Comparative Study Between Homomorphic Encryption and Distance-Preserving Randomization,' Access, IEEE , vol.2, no., pp.125,141, 2014.
13. W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, 'Secure knn computation on encrypted databases,' in Proc. of SIGMOD, 2009.
14. K. Ren, C. Wang, and Q. Wang, 'Security Challenges for the Public Cloud,' IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012.
15. Zhangjie Fu et al, 'Multikeyword Ranked Search Supporting Synonym Query over Encrypted Data in Cloud Computing', IEEE Conference, 2013.
16. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, 'Searchable symmetric encryption: improved definitions and efficient constructions,' in ACM CCS, 2006.
17. P. Naresh, K. Pavan kumar, and D. K. Shareef, 'Implementation of Secure Ranked Keyword Search by Using RSSE,' International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 2 Issue 3, March – 2013.
18. S. Buyrukbilen and S. Bairas, 'Privacy preserving ranked search on public key encrypted data,' in Proc. IEEE International Conference on High Performance Computing and Communications (HPCC), November 2013.
19. B. H. Bloom, 'Space/time trade-offs in hash coding with allowable errors,' Communications of the ACM, vol. 13, no. 7, 1970, pp. 422–426.
20. C. Gentry and Z. Ramzan, 'Single-database private information retrieval with constant communication rate,' in ICALP, pp. 803–815.2005.
21. Y. T. Hou, H. Li, W. Lou, and W. Sun. 'Privacy-preserving keyword search over encrypted data in cloud computing, Insecure Cloud computing,' edited by S. Jajodia et al., Springer, 2014.

22. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H., 'Privacy-preserving multikeyword text search in the cloud supporting similarity-based ranking,' Proceedings of the 8th ACMSIGSAC symposium on Information, computer and communications security, ACM, pp. 71–82.2013.
23. Prasanna B.T, C.B. Akki, 'A Survey on Homomorphic and Searchable Encryption Security Algorithms for Cloud Computing,' Communicated to International Journal of Information Technology and Computer Science, November, 2014.
24. Prasanna B.T, C.B. Akki, 'A Comparative Study of Homomorphic and Searchable Encryption Schemes for Cloud Computing,' Communicated to International Journal of Communication Networks and Distributed Systems, November, 2014.
25. Prasanna B.T, C.B. Akki, 'A Survey on Challenges and Security Issues in Cloud,' Presented in conference presented in Conference on Evolutionary Trends in Information Technology, May 20-22 2011, at Visvesvaraya Technological University, Belgaum, Karnataka.