ISSN:0975-9646

Improving Data Integrity for Data Storage Security in Cloud Computing

Poonam M. Pardeshi, Prof. Bharat Tidke
Department of Computer Engineering, University of Pune
Flora Institute of Technology, Pune, Maharashtra, India

***Abstract*- With the provision of innumerable benefits, cloud has become an emerging standard that brings about various technologies and computing ideas for internet. Massive storage centers are provided by the cloud which can be accessed easily from any corner of the world and at any time. The on-demand service provision with utilization of fewer resources of client system benefits the client. However, data outsourcing paradigm in cloud is one of the biggest security concerns. Frequent integrity checking is needed to keep an eye on data. The proposed scheme makes use of Merkle Hash Tree (MHT) and AES algorithm to maintain data integrity at the untrusted server. In most of the previously proposed schemes, RSA algorithm was used for storage security. AES being faster in encryption-decryption and the buffer-space requirement being less as compared to RSA, we try to improve the performance by making use of AES algorithm. The cloud must not impose on user the responsibility to verify his/ her stored data. Taking this into consideration and relieve client from the overhead of data integrity verification, we introduce an entity called the Third Party Auditor (TPA), which acts on behalf of client for data integrity checking and send an alert to notify the status of the stored data. The proposed storage security scheme also assures recovery of data, in case of data loss or corruption, by providing a recovery system. Thus the proposed scheme aims at keeping the user data integrated and support data restore. The system also reduces the server computation time when compared with previous systems.**

Keywords— Advanced Encryption Standard; Cloud Computing; Merkle Hash Tree; Public Auditability; Recovery System; Third Party Auditor

I. INTRODUCTION

Cloud Computing is has gained popularity in recent years. Cloud facilitates the storage of various sorts of data. Cloud is highly scalable when it comes to huge data and can provide infinite computing resources on demand. Clients can use cloud services without any installation and the data uploaded on cloud is accessible from any corner of the world, all it needs to be accessed is a computer with active internet connection on it. The users can subscribe high quality services of data and software which resides solely on the remote servers and enjoy the provision of on-demand provision of services. As a customizable computing resources and a huge amount of storage space are provided by internet based online services, the shift to online storage has contributed greatly in eliminating the overhead of local machines in storage and maintenance of data. The cloud provides a number of benefits such as flexibility, disaster recovery, pay-per-use and easy to access and use model which contribute to the reason of moving into cloud. A large number of clients store their important data in the cloud without keeping a single copy of this data in their local computers. Thus, cloud helps free up the space on the local disk, hence also called as 'A Hard-disc in the sky'.

Even though immense advantages are offered by cloud, a lot of security concerns still exist in it. The most worrisome concern is its storage security [8,10,11,12,13]. Most of the times, the user does not maintain any copy of outsourced data in their local system. The question regarding data security becomes crucial when it comes to confidential data. The integrity of the data has to be looked upon seriously in order to gain user trust and satisfaction. However, maintaining security is a challenging task. What if the storage server itself is not trustworthy? For example, the server or the Cloud Service Provider (CSP) may delete some less frequently accessed data to save the storage space. It may also try to hide errors in case of Byzantine errors to maintain their reputation. Therefore, although outsourcing data into the cloud may look economically attractive, the data integrity and availability factor may impede its adoption by users. The user must have the knowledge whether his/ her data is secured. The user needs to be convinced regarding the safety of remotely stored data. However, it is not feasible for the user himself to verify his data.

There exist many systems that have tried to solve the problem of data integrity. The auditing can be performed in two ways viz. Private and Public [10]. In Private Auditability, the client is responsible to verify the data. No one else except the client can question the server regarding the data integrity, whereas, Public Auditability is more convenient and preferred over Private Auditability because it allows a third party to perform integrity verification on behalf of client. The client is not solely responsible for it and so it largely reduces client's burden. We refer this third party as the Third Party Auditor (TPA).

The other important piece in maintaining user data in cloud is the restore system. If under some unpleasant situation, the integrity of data is lost, ultimately CSP is responsible for it and there should be some provision to heal the situation. This is because what a user needs is his/ her data in original its form irrespective of what problem occurred at the server. Considering this fact, the proposed system is equipped with a recovery system which stores a backup of the user data. This contributes to availability of data anytime.

II. BACKGROUND THEORY

A. Auditing

The verification of user data can be carried out in two ways, either by the user himself (data owner) or by a third party auditor. The verifier's role fall under two categories:

- *Private Auditability*

  Only data owner is allowed to check the integrity of the stored data. No one else can question the server regarding the data. This kind of auditability increases verification overhead of the user.

- *Public Auditability*

This kind of auditability allows anyone, not just the client, to challenge the server and perform data verification check. This is where a Third Party Auditor (TPA) comes into play.

B. Third Party Auditor(TPA)

The TPA is an entity that acts in behalf of the client. It has the expertise, capabilities, knowledge and professional skills that client does not have. It handles the work of integrity verification and reduces the overhead of the client sue to which, client no longer needs to verify the integrity of the data at the server on its own. Cloud Storage Architecture:

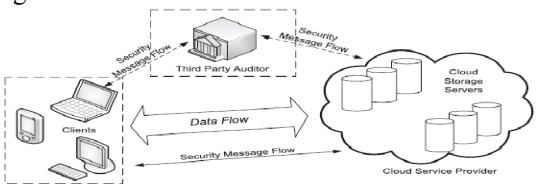

Fig 1: Cloud Storage Architecture [1]

Fig.1 shows the storage architecture of the cloud. The three network entities viz. the client, cloud CSP and TPA are present in the cloud environment. The client stores data on the storage server provided by the CSP. TPA keeps a check on client's data by periodically verifying integrity of data on-demand and notifies client if any variation of fault is found in client data.

C. Merkle Hash Tree (MHT)

A Merkle Hash Tree is a well-studied authentication structure [7]. It is used to efficiently prove that a set of elements are undamaged and unaltered. It helps greatly in reduction of server time [9]. It is used by cryptographic methods to authenticate the file blocks. The leaf nodes of the MHT are the hash values of the original file blocks. The idea behind generating MHT is to break the file into a number of blocks. Apply hashes to the authentic data values i.e. the original file blocks and combine iteratively. Now, rehash the result hash nodes and combine in a tree-like fashion and repeat this procedure till we get a tree with a single root. The MHT is generated by the client and is stored at both the client and the server side. Fig 2 depicts an example of MHT. The tree has four leaf nodes viz. m_1, m_2, m_3 and m_4. Initially, we apply hash on each of these file blocks and obtain $h(m_1)$, $h(m_2)$, $h(m_3)$ and $h(m_4)$. Then, $h(m_1)$ and $h(m_2)$ are hashed and combined together to get ha. Similar process happens with blocks m3 and m4 and here, we get h_b. Here, h is a secure hash function.

This can be expressed as

$$h_a = h(h(m_1)|| h(m_2)) \quad \text{and} \quad h_b = h(h(m_3)|| h(m_4))$$

Further, ha and h_b are combined and rehashed to obtain the root as hr. This can be expressed as

$$h_r = h(h_a|| h_b)$$

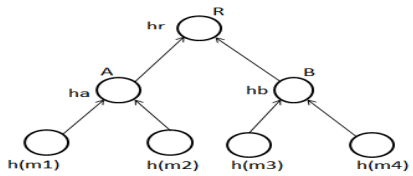

Fig 2. Merkle Hash Tree

Road Map

Section 3 gives the survey on various systems developed for storage security in cloud. Section 4 describes the proposed security model. Section 5 presents the performance analysis, and then section 6 gives the concluding remark of the whole paper and discusses the future work.

III. LITERATURE REVIEW

Recently, much work has been done in the area of cloud security. Majority of them focus on the integrity verification of data stored in the cloud. Deswarte et al. in [1], use RSA based hash function for verification of the file stored at the remote server. Using this scheme, it is possible for the client to perform multiple challenges using the same metadata.

Disadvantage: The limitation of this scheme lies in the computational complexity at the server which must exponentiate all the blocks in the file.

Miller and Schwarz [2] proposed a technique using which the data stored remotely across multiple sites can be ensured. The scheme makes use of algebraic signature. In this, a function is used to fingerprint the file block and then verifies if the signature of the parity block is same as the signature of block.

Disadvantages: 1) The main disadvantage of this scheme is that the computation complexity at client side and server side takes place at the cost of linear combination of file blocks. 2) Also, the security of this scheme remains unclear.

Ateniese et al. [3] were the first in considering the concept of Public Auditing for ensuring possession of files at untrusted servers. For auditing of outsourced data, the scheme utilizes RSA based homomorphic tags, thus achieving public auditing. In this protocol, the client need to verify if the server has retained file data without actually retrieving the data from server and without having the server access the entire file.

By sampling random sets of blocks from the server, the model generates probabilistic proofs of possession by sampling random sets of blocks. This reduces I/O cost drastically. The Provable Data Possession [PDP] model for remote data checking supports large data sets in widely-distributed storage systems. It is provably-secure scheme for remote data checking.

Disadvantages: 1) An overhead of generating metadata is imposed on client. 2) No support provided for dynamic auditing. 3) Requires more than 1kilo-byte of data for a single verification.

A scheme called, "Proofs of Retrievability" (POR) [4], proposed by Juels and Kalisiki focuses on static archival of large files. To ensure data possession and retrievability, it makes use of spot checking and error correcting codes. Some special blocks called as "sentinels" are randomly embedded into the file F for detection. Further, the file is encrypted out in order to protect the position of these sentinel blocks. POR scheme cannot be used for public databases; it is suitable only for confidential data. Disadvantages: 1) Dynamic updation is prevented due to

the introduction of sentinel nodes. 2) Number of queries clients used is fixed priori. 3) Preprocessing of each file is needed prior to storage at the server. 4) The scheme cannot be used for public databases and can only be used for confidential data. 5) Does not support Public Auditability, i.e., it supports only two-party auditing, which is not efficient because neither the client nor the cloud service provider can give assurance to provide balance auditing.
Shacham and Waters design an improved PoR scheme with full proofs of security in the security model defined in [4]. They use publicly verifiable homomorphic authenticators built from BLS signatures [18], based on which the proofs can be aggregated into a small authenticator value, and public retrievability is achieved. Still, the authors only consider static Data files.

Disadvantage:
The scheme works only with static data files
Scalable and Efficient Provable Data Possession (S-PDP and E-PDP) protocols contribute to the work of Ateniese et al. [5]. The paper presents the dynamic version of prior PDP scheme and relies, in both the setup and verification phases, only on efficient symmetric-key operations. It makes use of less storage space (size of challenge and response is significantly less, less than a single data block), and uses less bandwidth. As no bulk encryption of outsourced data is required, the scheme delivers better performance on client side.

Disadvantage: 1) The number of queries which can be answered is fixed priori. 2) Not applicable for dynamic data operations, supports only basic block operation with limited functionality. 3) It is a partially dynamic scheme, not fully dynamic because it does not support block insertion.

The scheme proposed by C.Erway el at [6] is a dynamic auditing protocol that can support the dynamic operations of the data on the cloud servers. This scheme requires the linear combination of data blocks to be sent to the auditor for verification. The scheme makes use of a TPA for integrity verification. It also supports data dynamics via the most general forms of data operation, such as block modification, insertion and deletion.
Disadvantages: 1) The scheme may leak data content to the auditor because it requires the server to send linear combinations of data blocks to the auditor for verification. 2) The efficiency of this scheme is not clear.
Table 1 describes the comparison of existing literature reviewed system with proposed system.

IV. PROPOSED SCHEME

Data security in cloud is one of the serious issues with cloud storage facility. Client store their data at the cloud, delete the local copy of that data and rely completely on the cloud server for data safety and maintenance. For this, auditing of the data is necessary to assure client safety of his data. To overcome this problem of data security, we introduce an AES based Storage integrated.

Table 1: Comparison between different systems

Scheme/ Ref.No / Attributes	[3] G. Ateniese et al	[4] A. Juels et al	[5] G. Ateniese et al	[7] C. Wang et al	[17] S. Zhong et al	Proposed System [AESSS]
Privacy Preserving	No	Yes	No	No	Yes	Yes
Unbound no. of queries	Yes	No	No	Yes	Yes	Yes
Public Verifiability	Yes	No	No	Yes	Yes	Yes
Use of TPA	No	No	No	Yes	No	Yes
Recoverability	No	Yes	No	No	No	Yes
Untrusted Server	Yes	Yes	Yes	Yes	Yes	Yes

Design
Fig 3 gives a block representation of AES based Storage Security. It has three network entities, viz. the client (client system), the CSP and the TPA.

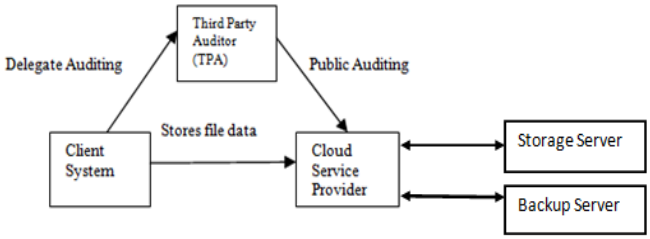

Fig.3. Block Diagram of AES based Storage Security System

a) *Client (User):* It is a network entity that stores data on the cloud server and relies on it for the maintenances and storage of the data.
b) *Cloud Service Provider (CSP):* It is the cloud server that provides significant storage space, resources and maintenance for user data.
In the block diagram, two more blocks are present, Storage Server and the Backup server. The storage server is where the original *files* of the client are stored and the backup server is the one where the backup copies of the file are stored for recovery purpose.
c) *Third Party Auditor (TPA)*: TPA is an entity that has knowledge and expertise that client does not possess. It is responsible for data integrity verification and works on behalf of the client.

General Idea
In proposed system, server is considered as untrusted entity. After a check is performed, a notification is sent to the client about the status of his data; indicating whether the data is in its actual form or if its integrity is lost. Also, as the server is considered to be untrusted, instead of storing data directly to the server, we encrypt it using AES-128 algorithm before storing it so that the server cannot read the content in the files. According to a performance evaluation, if we go from AES-128 to 192 bits key, the power and time consumption increases by 8% and 256 bits key causes an increase of 16 % [15,16]. So we propose use of industry-

standard high grade Advanced Encryption Standard (AES) symmetric encryption algorithm with key length of 128-bits for this purpose. Merkle Hash Tree is used for authentication of file and integrity verification.
Secondly, a Recovery System is provided, which is useful in case of data loss or when the file stored at the server side is corrupted.

Data Uploading and Downloading

When the file is uploaded to the cloud server, before storing it, AES algorithm is used to encrypt the data to protect the content from being displayed to the server. Similarly, at the time of download, the data is decrypted to plain text form

Uploading and Downloading Process

The user data is encrypted using AES and then stored at the cloud server. This is done as shown in the fig. 4. At the time of download, the user files are decrypted using AES. This can be seen in Fig. 5.

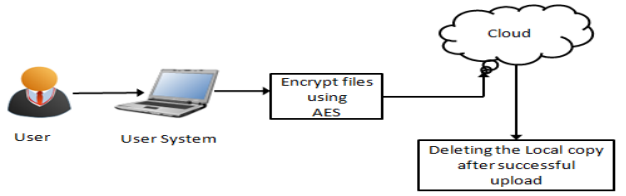

Fig. 4: Data Uploading

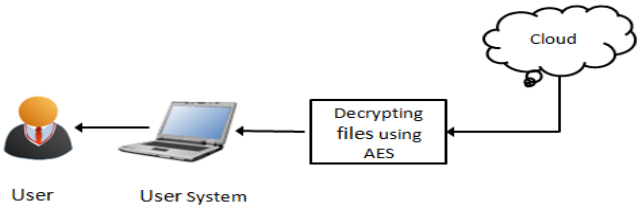

Fig. 5: Data Downloading

Notations

E_{sk} - Encryption using Secret key
F- File stored at the untrusted server
m - File block
T- Tag (signature)
ϕ- Set of tags

Storage Security Model

AES being better than RSA in many ways, the proposed system makes use of AES algorithm instead of RSA. The proposed security model consists of two phases, viz. the setup phase and the integrity verification phase.

- *The Setup Phase*

In the setup phase, the file F= $\{m_1, m_2...m_n\}$ is generated by the client, which is a finite collection on n blocks. Using the key generation algorithm, the secret key is generated. The overall flow of this process is depicted in Fig.6.
The setup phase has five steps. In the first step, a signature is generated for each file block using the secret key and SHA1 hash algorithm. This is done as $Ti = E_{sk}(H(m_i))$, where mi is the ith block of the file. In second step, a set of signatures of file blocks ϕ= $\{T_i\}$ is generated, also known as the set of Tags. Then Merkle Hash Tree is constructed and in fourth step, the root of the tree is signed using the secret key as sigsk (H(R)). In the last step, the client advertises $\{F, \phi, sig_{sk}(H(R))\}$ to the server and deletes F and $sig_{sk}(H(R))$ from its local storage.

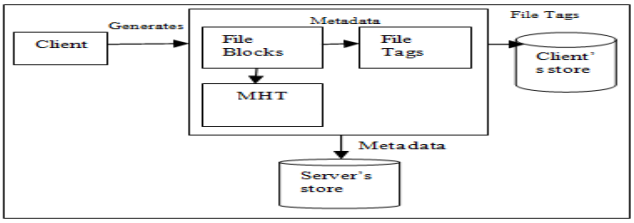

Fig 6: Pre-processing File Blocks [14]

- *Integrity Verification Phase*

The integrity verification process, in Fig.7, is where client initiates by sending a request to TPA for auditing the desired file or data. This is done by sending some metadata such as FileId and ClientId. The TPA generates a challenge, sends it to the CSP and in response, the server generates a proof for the corresponding challenge. In the proof, the server generates the proof. The proof contains the signature of the root and the root of the MHT generated for the respective file. The verification process is done in two stages. First is file authentication and second is integrity checking. For authentication of the file, the signature of the root is checked. If it matches with signature stored during file upload, the output is given as True otherwise emits False. If the output is True, the integrity is checked by checking the value of the root with previously stored root. Any changes made to the file blocks are reflected in the value of the root. If the root does not match, it means that some changes are made to the file and the file has lost its integrity. In both the cases, a notification is sent to the client. In case of data loss or if the file is corrupted, the client can recover the file from the recovery system if he has previously taken a backup of the file. Integrity verification is done by checking the value of only Tags; TPA does not need to access the actual data for it. Due to this, TPA cannot view client's data and it makes the process Privacy-Preserving.

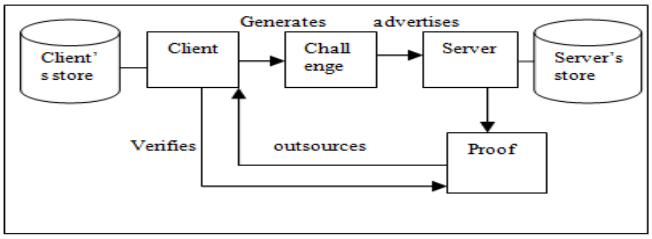

Fig. 7: Integrity checking process flow [14]

To take the auditing process to a deeper level, after a file is not found to be in the integrated state, further checking at the block level is done to find out particularly which block is corrupted or modified.

- *The Recovery System*

The user has the right to decide whether to store his/ her files in the recovery system or not. The files stored in this

backup system can be recovered easily in case of link failure or storage server crash, loss or corruption of original file and in similar unpleasant circumstances.

In the verification process, if it is found that the file has lost its integrity, then the TPA checks the file at the block level, i.e. the leaf nodes are checked to see which block is infected. After detection of the infected block, instead of fetching the entire file, the TPA fetches only the infected block from the recovery server. This greatly reduces the communication bandwidth required for recovery.

The Recovery system adds to the plus points as it contributes to the availability of data which is a very important parameter to be observed.

V. Performance Analysis:

Encryption and Decryption Time

Figures 8 and 9 graphically represent the time required for encryption and decryption respectively on different file sizes. The behavior of the graphs shows that for file size up to 1000 kb, the required is less and it gradually rises when the file size is increased. If the encryption and decryption time is compared with similar systems, it shows that time required by AESS System is significantly less.

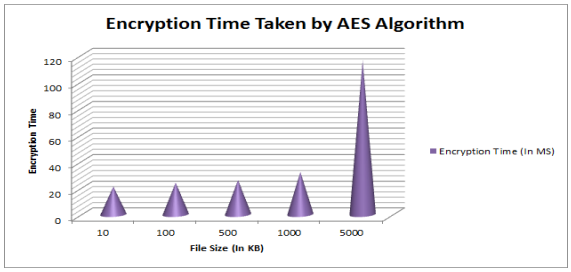

Fig 8: Encryption time by AES

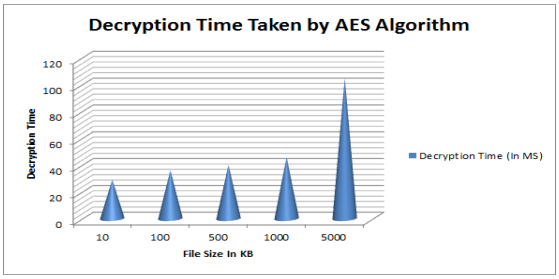

Fig 9: Decryption time by AES

Server Computation Time:

The server computation time of this system is compared with the RSASS system and the S-PDP scheme. The graph in fig. 10 indicates that for the file blocks of any size, the server computation time for the AES based system remains less. For example, if a file of size 120 kb is considered, then the time needed by RSASS system is between 4 to 5 seconds. For similar file size, the time needed by S-PDP system is around 6.3 second whereas for AES based Storage Security System, the server time lies between 1 to 2 seconds which is much less as compared to both the other systems.

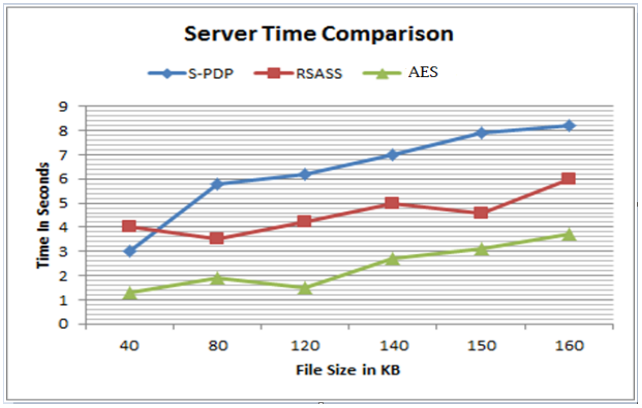

Fig. 10: Server Time Comparison

VI. Conclusion and Future Scope:

In this paper, a secured and efficient AES based system has been proposed for auditing user data stored at untrusted server. The system guarantees data the achievement of data integrity and availability. The system supports Public Auditing by making use of TPA and Privacy Preserving by not leaking the data to TPA during integrity verification process. By frequent integrity checking, the system assures data possession at remote server.

In future, the AES bases Storage Security System can be further extended to support dynamic operations on data. Also, the system can further be enhanced to support dynamic auditing, by which, the auditor can periodically perform check on the data and maintain it even when the client does not request for it. This will completely remove the burden of client and help keep data safe.

References

[1] Y. Deswarte, J. Quisquater, and A. Saidane, *Remote integrity checking,* In Proc. of Conference on Integrity and Internal Control in Information Systems (IICIS'03), November 2003.

[2] T. Schwarz and E.L. Miller, *Store, forget, and check: Using algebraic signatures to check remotely administered storage*, In Proceedings of ICDCS '06. IEEE Computer Society, 2006.

[3] G.Ateniese, *Provable Data Possession at Untrusted Stores*, Proc. 14th ACM Conf. Computer and Comm. Security (CCS' 07), 2007.

[4] A. Juels, *Pors: Proofs of Retrievability for Large Files*, Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, 2007.

[5] G.Ateniese, *Scalable and Efficient Provable Data Possession*, Proc. Fourth Int'l Conf. Security and Privacy in Comm. Networks (SecureComm '08), 2008.

[6] C,Erway, A.Kuocu, C. Pamanthou, R.Tamassia, *Dynamic Provable Data Possession*, Proc. 16th ACM Conf. Computer and Comm. Security (CCS'09),2009.

[7] Cong Wang, *Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing*, IEEE Transactions on Parallel and Distributed Systems, May 2011.

[8] C.Wang, Q.Wang, Kui Ren, Wenjing Lou, *Ensuring Dynamic Data Storage Security in Cloud Computing*, Proc. 17th Int'l Workshop Quality of Service (IWQos'09),2009.

[9] P. Golle, S. Jarecki, and I. Mironov, *Cryptographic primitives enforcing communication and storage complexity.* In Financial Cryptography, pages 120-135, 2002.

[10] L. Chen and H. Chen, *Ensuring Dyanmic Data Integrity with Public Auditing for Cloud Storage*", In Proc. Of International Conference on Computer Science and Service System (ICSSS' 2012), 2012.

[11] D.G.Feng, M. Zang, Y. Zang and Z. Xu,"*Study on cloud computing security"*, Journal of Software, vol.22 (1), pp. 71-83, 2011.

[12] L.M. Kunfam, "*Data Security in the world of cloud computing*", IEEE Security and Privacy, vol.7 (4),pp.61-64,2009.

[13] B. Waters and H.Shacham, *Compact proofs of Retrievability*, Proc.14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT' 08), pp.90-107, 2008.

[14] M. Venkatesh, *Improving Public Auditability, Data Possession in Data Storage Security for Cloud Computing*, ICRTIT-IEEE 2012

[15] Elminaam, Diaa Salama Abdul, Hatem Mohamed Abdul Kader, and Mohie Mohamed Hadhoud. *Performance Evaluation of Symmetric Encryption Algorithms.* IJCSNS International Journal of Computer Science and Network Security 8.12 (2008): 280-286.

[16] Simar Preet Singh, and Raman Maini, "*COMPARISON OF DATA ENCRYPTION ALGORITHMS*", International Journal of Computer Science and Communication (IJCSC), Vol. 2, No. 1, January-June 2011, pp. 125-127

[17] Z. Hao, S. Zhong and N. Yu, *A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability* ,IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No. 9, September 2011