ISSN 2229-5518

Data Security Algorithms for Cloud Storage System using Cryptographic Method

Prakash G L, Dr. Manish Prateek, and Dr. Inder Singh

Abstract—Cloud computing paradigm enables the users to access the outsourced data from the cloud server without the hardware and software management. For the effective utilization of sensitive data from CSP, the data owner encrypts before outsourcing to the cloud server. To protect data in cloud, data privacy is the challenging task. In order to address this problem, we proposed an efficient data security method using cryptographic techniques. Thus, the proposed method not only encrypts the sensitive data, but also detects the dishonest party to access the data using combined hash functions. We have analyzed the proposed method in terms of storage, communication and computational overheads. The result shows that the proposed security method is more efficient than the existing security system.

Index Terms— Hash Function, Data Storage, Data Control, Verification, Encryption, Data Privacy,

1 INTRODUCTION

In cloud computing, data is stored in remote massively scalable data centers where compute resources can be dynamically shared to achieve significant economies of scale. The storage capacity needs to scale with compute resources to effectively manage and gain maximum cloud benefits.

Armbrust et al.[8] explained cloud as the data Center hard-ware and software that provide services on-demand network access to a shared pool of consumable computing resources. It has the following common characteristics; (i)pay-per-use (ii)elastic capacity (iii)self-service interface and (iv)resources that are abstracted or virtualized.

For organizations into cloud computing, storage manage-ment is extremely important. To avoid data loss, the cloud system must provide data protection and resiliency. If loss does occur, the environment must be able to recover the data quickly in order to restore access to the cloud services. The storage management and information protection in cloud environments helps to deliver a workload-optimized approach.

Depending on the type of cloud used, the cloud provider's responsibilities could include providing infrastructure, physical security of the premises, operating system and network security. Sharing of cloud resources such as providing infrastructure, operating system, application and network security is controlled by the cloud service provider depending on the cloud deployment model.

On the other hand the actively processing cloud data is controlled by cloud users depending on the cloud service model is used in their application. An organization classify the information according to the sensitivity to its loss or disclosure. The level of information sensitivity classification defined by the data owner based on the security control.

The detailed functional components of cloud computing security architecture are explained in the figure 1. Based on three cloud computing service models such as; infrastructure as a service, platform as a service and software as a service, it contains; authorized users, data provider, communication Access Point(CAP), Security Access Point(SAP), Application Access Point(AAP), and Application servers, functional components [1].

In software as a service model, where software is represented by various application servers of same and/or different types. The main characteristics of cloud computing model is, the data provider and cloud users do not access servers directly. To access the various cloud services based on type of user re-quest and other processing parameters, the Application Access Point Server distribute the service request to the individual application server. The users may access cloud services on demand through internet using communication Access Point.

- *Research Scholar, Department of Computer Science and Engineering, University of Petroleum and Energy Studies, Dehradun, Email:glprakash78@gmail.com*
- *Center for Information Technology, University of Petroleum and Energy Studies, Dehradun,*

International Journal of Scientific & Engineering Research, Volume 5, Issue 3, March -2014

ISSN 2229-5518

The SAP server provides the front-end security service before accessing the cloud resources. Once the user has been authenticated, SAP verifies whether the user requests are authorized to access internal cloud resources or not. After authentication, enforcement the final security service is provided by the SAP server.

When a user request some application service via CAP to the cloud, that request will first reach SAP server. The server will forward it to the Policy Decision Point (PDP) Server for the authentication and authorization decision. If both are approved, users application request will be passed to the appropriate application server, where it will be served, and the response will be returned to the user. All described security actions, SAP server, then forwarding it to the PDP server, receiving reply back to the SAP server and, finally access to the application server providing the requested service, are performed instantaneously and transparently to the user [2]-[11]. Therefore, the user is not aware of any of these actions, except if some unauthorized action is attempted.

The rest of the paper is organized as follows. In Section II and III, we have defined the related work and background for the proposed security system respectively. Our proposed system security algorithms and performance analysis are explained in Section IV and V respectively. Finally we conclude in Section VI.

2 RELATED WORK

To ensure the data integrity of a file consisting of a finite ordered set of data blocks in cloud server several solutions are defined by Qian Wang et al, in [3]. The first and straight forward solution to ensure the data integrity is, the data owner pre-compute the MACs for the entire file with a set of secrete keys, before our sourcing data to cloud server. During auditing process, for each time the data owner reveals the secret key to the cloud server and ask for new MAC for verification. In this method the number of verification is restricted to the number of secrete keys. Once the keys are exhausted, the data owner has to retrieve the entire file from the cloud server to compute the new MACs for the remaining blocks. This method takes the huge number of communication overhead for verification of entire file, which effect the system efficiency.

The another solution to overcome the drawback of previous method, is to generate the signatures for every block instead of MACs to obtain the public audit-ability. This solution can provide probabilistic assurance of data correctness and public audit-ability, which again results in large communication overhead and effect the system efficiency. The above solutions are supports only static data and none of them can deal with the dynamic data updates.

Qian Wang et al, in [4] designed an efficient solution to support the public audit-ability without retrieving the data blocks from server. The design of dynamic data operations is a challenging task for cloud storage system. They proposed a RSA signature authenticator for verification with data dynamic support. To support the efficient handling for multiple auditing task, they extended the technique of bilinear aggregate signature and then they introduced a third party auditor to perform the multiple auditing task simultaneously. In the recent resource sharing paradigm in distributed system such as cloud computing, the most challenging task.

In data sharing system is defining of access policies and dynamic data updation. In [5], Junbeom Hur, explains the cryptographic based solution for data sharing using cipher-text policy attribute-based encryption(CP-ABF) to improve the security of the data. In this method the data owners defines the access policies on the data to be distributed. The major drawback of this method is the unauthorized users can access the key to decrypt the encrypted data.

In cloud computing, both data and applications are controlled by the data owner and cloud service provider. To access the cloud data and applications as a cloud service more securely a data security model has been defined Mohamed, E.M. in [6]. In this security model, it provides a single default gateway as a platform to secure user data across public cloud applications. The default gateway encrypts only sensitive data using encryption algorithm, before sending in to the cloud server. In this method the data is accessed by only authorized users but the cloud service provider can grant the access permission for unauthorized users while cheating to the data owner. Therefore, this method degrades the security as proper key management is not implemented in the system.

To increase the revenue and degree of connectivity from cloud computing model while accessing and updating data from data center to the cloud user, Dubey et al. in [7] developed a system using RSA and MD5 algorithms for avoiding unauthorized users to access data from cloud server. The main drawback of this method is that the cloud service provider has also an equal control of data as the data owner and the computation load for cloud service provider is proportional to the degree of connectivity so that the performance of the system can degrade.

International Journal of Scientific & Engineering Research, Volume 5, Issue 3, March -2014

ISSN 2229-5518

Fig. 1. Functional Components of Cloud Computing Security Architecture

3 MODEL/ARCHITECTURE

3.1 Background

The block diagram of cloud computing system architecture is defined in the figure 2. It contains there are four functional blocks for data storage and accessing data from data centers in public cloud servers such as, data owner, Cloud Service Provider(CSP), authorized users, and Trusted Third Party [2]. The functions of these functional blocks are as follows;

Data owner: The data owner can be any organization for generating outsourcing data to store in data center of public cloud model for the external use on the demand of the authorized users on the basis of pay per usage.

Cloud Service Provider: Manage the cloud servers and data centers in the public cloud and provide the storage infrastructure to the data owner for storage of outsourced data in data center on the payment based on the requested storage capacity. It coordinates the trusted third party to verify the authorized users and to retrieve the data from cloud server to make them available for authorized user on demand.

Users: the set of authorized users to access the remote data stored in cloud server through trusted third party and cloud service provider. All the users are the clients of the data owner.

Trusted Third Party: an entity who is trusted by all other entities of the system such as CSP, data owner and users. The functions of TTP is to verify whether the requested user in authorized or not, updating the block status table of file and calculate the hash value for file and block status table [10].

3.2. Assumptions

The following assumptions are considered to evaluate the data privacy of the proposed system

1) The data owner and users have mutual distrust relation with cloud service provider.

2) Trusted third party helps to make the indirect mutual trust between authorized user and data owner with cloud service provider.

3) Public cloud model is considered for storage of outsourced data in data centers

The data owner has a file *(F)* consisting of *m* blocks of equal size. Since the data is storing on remote data center, for confidentiality all the blocks are encrypted using symmetric data encryption algorithm before sending to the cloud server.

3.3. Objectives

The objectives of our proposed security system are summarized as follows.

1) Design an efficient data privacy algorithm using cryptographic techniques.

2) Detect the dishonest party using combined hash values verification.

3) Reduce the computational overheads of CSP, while introducing TTP.

4) Access the out sourced data, even if data owner is in off-line.

International Journal of Scientific & Engineering Research, Volume 5, Issue 3, March-2014

ISSN 2229-5518

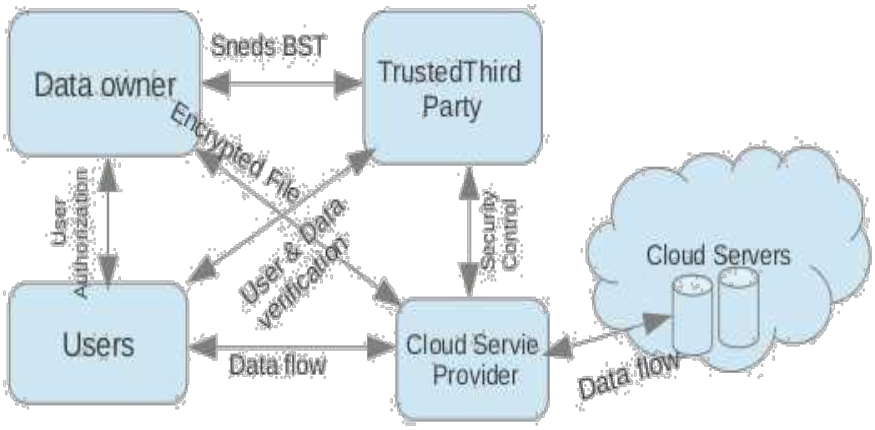

Fig. 2. Block diagram of Cloud Computing System Architecture

4 PROPOSED SECURITY SYSTEM

4.1. Block Status Table

The Block Status Table(BST) is a small data structure used to access the outsourced encrypted file from the cloud service provider. It consists of two column such as SN_j and BN_j , where SN_j is the sequence number of physical storage of data block j in the file and BN_j is the data block number. Initially the data owner stores table entries as $SN_j = BN_j = j$. For insertion of data blocks, the BST is implemented using linked list. The structure of BST for insertion of data blocks as shown in the Table 1.

TABLE 1: STRUCTURE OF BLOCK STATUS TABLE

Sequence Number	Block Number
1	1
2	2
3	3
4	4

4.2. Proposed System

The proposed system consists of a four functional role; data owner, user, trusted third party and cloud service provider.

Owner: the data owner generates the initial symmetric key for data encryption and decryption. The key is rotated forward direction to generate the new key for next block data encryption and rotate backward direction for previous block encryption. For a file where b_j is the j^{th} block and m is the number of data blocks, initially the owner generate a BST with $SN_j = BN_j = j$ for the input file and character map table explained in the Table 2. To protect the data from the unauthorized users, before F, BST, key to the TTP. The BN_j and b_j are concatenated to helps to reconstruct the original file in the correct order while file retrieving.

TABLE 2: SETUP FOR KEY AND FILE PRE-PROCESSING

Algorithm Setup(file, key)

Input: source file and 256 bits key

Output: Number of data blocks *(m)* and key rotation setup

step 1: Divide the source file in to blocks of equal size.

Number of blocks *(m)* = file size/block size

step 2: Create circular double linked list of 16 nodes for key rotation and store one character in each node.

Step 3: Create a simple data encoding map table for all the characters.

International Journal of Scientific & Engineering Research, Volume 5, Issue 3, March-2014

ISSN 2229-5518

TABLE 3: DATA ENCRYPTION ALGORITHM

Algorithm Data Encryption(Source file(F), key, Decrypted file)

Input: Source file and key

Output: Decrypted file

step 1: Split the characters of the file/string into chunks 128 characters

step 2: Get the binary equivalent of the current Character (process character by character of chunks)

step 3: Remove the first character from the binary value and store it into first Character variable and consider the rest of binary value for shift operations.

step 4: Store binary value into a circular doubly linked list for bit operations.

step 5: Select a key character from i^{th} position , such a way that if 1st chunk character is selected for encryption, then select the first character of the key, so $i = 1$.

step 6: if selected chunk character greater size of key(16), get the modulus of chunk character position.

step 7: add the stripped off first Character to the resultant binary value of circular array after bit operations.

step 8: add the selected i^{th} key character and $(i + 2)^{th}$ key character.

step 9: right shift the binary bits in circular array by (added value mod 5), if mod 5 is 0 then do by 2.

step 10: add the selected i^{th} key character value and its previous $(i - 1)^{th}$ character value, for instance if 1^{st} key character is selected then the previous character would the 16^{th} character (key is stored into circular array or double way linked list for this operations)

step 11: get the character equivalent of the binary value

step 12: Get the mapped value from encoding Map , which is the decrypted value of the character.

step 13: Repeat steps 2 through 12 for all the chunks with different Key

(by shifting the key character right using Circular double linked list)

Thrusted Third Party Role: The TTP receives the {F, BST,key} from data owner, then it computes the combined hash values for the encrypted file F and BST using the following formula.

$$FH_{TTP} = h(b_1[i] \oplus b_2[i] \oplus \ldots \oplus b_m[i]$$

$$= \oplus\{ h(b_j[i]\}_{i=1}^{m} \quad \forall i = 1 \ldots n \quad \ldots \quad (1)$$

$$TH_{TTP} = h(BN_1[i] \oplus BN_2[i] \oplus \ldots \oplus BN_m[i]$$

$$= \oplus\{ h(BN_j[i]\}_{i=1}^{m} \quad \forall i = 1 \ldots n \quad \ldots \quad (2)$$

where m is the number of data blocks, n is the number of bits in each block. The TTP store the computed hash values of file (FH_{ttp}) and BST (TH_{ttp}) in the local storage, then it sends only {F and BST } to the cloud service provider.

User Role: The authorized user sends a request to both the CSP and TTP to access the outsourced file from the cloud server. After receiving {F, BST} from the CSP, and

International Journal of Scientific & Engineering Research, Volume 5, Issue 3, March-2014

ISSN 2229-5518

{FH_{ttp}, TH_{ttp}, *key*} from the TTP, the user verifies the contents of BST_{csp} entries by computing the combined hash values of BST_{csp} using the following equation.

$$TH_{user} = h(BN1[i] \oplus BN2[i] \oplus .... \oplus BNm[i]),.....(3)$$

where $i=1,...n$ and $h(.)$ is the hash value i^{th} bit. The user compares the H_{ttp} received from TTP and TH_{user}. If(TH_{user} != TH_{ttp}), then issue a dishonest party report to the TTP and data owner. In case of (*THuser==THttp*) the user verifies the contents of the encrypted file F by calculating the FH_{user} using the following equation and comparing with FH_{ttp}.

$$FH_{user} = h(b1[i] \oplus b2[i] \oplus .... \oplus bm[i]),.....(4)$$

If (FH_{user} ! = FH_{ttp}), then informs to the TTP to resolve such a conflict. The authorized user decrypt all the encrypted blocks (b_j) using the decryption algorithm explained in the Table 4, and symmetric key, then returns (BN_j ||b_j). The BN_j and BST_{csp} are utilized to reconstruct the original source file (*F*)

TABLE 4: DATA DECRYPTION ALGORITHM

Algorithm Data Decryption(Decrypted file), key, Source file(F))

Input: Encrypted file and KEY for encryption 16 characters

Output: Source file

step 1: Split the characters of the file/string into chunks 128 characters

step 2: Get the binary equivalent of the current Character (process character by character of chunks)

step 3: Remove the first character from the binary value and s ore it into first Character variable and consider the rest of binary value for shift operations.

step 4: Store binary value into a circular doubly linked list for bit operations.

step 5: select a key character from i^{th} position, such a way that if 1^{st} chunk character is selected for encryption then select the first character of the key, so *i = 1*.
if elected chunk character greater than size of key(16) get the modulus of chunk character position.

step 6: add the stripped off first Character to the resultant binary value of circular list after bit operations.

step 7: add the selected i^{th} key character and $(i + 2)^{th}$ key character.

step 8: right shift the binary bits in circular list by (added value mod 5), if mod 5 is 0 then do by 2.

step 9: add the selected i^{th} key character value and its previous $(i - 1)^{th}$ character value, for instance if 1^{st} key character is selected then the previous character would the 16^{th} character (key is stored into circular array or double way linked list for this operations)

step 10: get the character equivalent of the binary value

step 11: get the mapped value from encoded Map , which is the decrypted value of the character.

step 12: Repeat steps 2 through 11 for all the chunks with different Key
(by shifting the key character right using Circular double linked list)

International Journal of Scientific & Engineering Research, Volume 5, Issue 3, March-2014

ISSN 2229-5518

5. PERFORMANCE ANALYSIS

We evaluate the performance of the proposed cloud security method by analyzing the storage, communication and computation overhead in terms of data block storage and processing for the data security requirements.

5.1. Storage Overhead

It is the additional storage space required to store the necessary information for data security, other than the stored file. The data owner stores only the data encryption key and BST in its local storage, where the size of the key is 256 bits. The size of the BST is calculated based on the number of data blocks. The BST contains two columns, both are integer it occupies total 8 bytes of memory space for single data block, therefore the total storage space of BST is 8× m bytes, where m is the number of data blocks. While increasing the data block size still we can reduce the BST storage overhead.

The storage overhead for the combined hash values of the file and BST is 64 bytes, each of size 32 bytes. The total storage overhead at TTP is the sum of the storage space required for the key, FH_{ttp} and TH_{ttp} is 96 bytes. The total storage overhead of the proposed security system is the sum of the storage overhead at data owner, TTP and CSP side. The over all storage overhead is calculated using the following equation.

$$Overhead_{Storage} = overhead_{(Owner)} + overhead_{(TTP)} + overhead_{(CSP)}$$

$$= (8 \times m + 32) + 96 + (8 \times m) = (16 \times m + 128)\ bytes. \quad . \ . \ . \quad (5)$$

5.2. Communication Overhead

It is the additional information sent other than the out sourced data blocks, to access the data from the CSP. When the user request a file from the CSP, the CSP sends the encrypted file and BST_{csp} to the user and also TTP sends the TH_{ttp}, FH_{ttp} and key to the user. The total communication overhead for the system is the sum of the communication overheads between the owner, user, TTP and CSP. The over all communication overhead is calculated using the following equation.

$$Overhead_{commn} = overhead_{(Owner,TTP)} + overhead_{(TTP,CSP)} + overhead_{(TTP,CSP)} \quad . \ . \ . \quad (6)$$

5.3. Computation Overhead

For confidentiality requirement the static data in the cloud storage system has the computational cost for data access from the CSP. The computation cost is the cost of the time required to perform the data encryption, data decryption, BST generation, key generation and verification. Before access ing the data from the CSP, the authorized user verifies BST_{csp} and the data file. the cost required for these verification is $2 * m * h$, where h is the cost of one hash value computation [2]. The total computation overhead for data access is the sum of the cost of the verification, key rotation and data decryption. The total cost required to access the file from CSP is calculated using the following equation.

$$Cost_{computation} = 3 * m * h + keyrotation + decryption. . (7)$$

The maximum computation overhead required for detecting the unauthorized user on the TTP side is $2 * m * h$.

6. CONCLUSIONS AND FUTURE ENHANCEMENT

In this paper, we discuss the problem of data security in cloud storage system. To control the outsourced data and provide the quality of the cloud storage service for the users, we propose an efficient data encryption, data decryption, ke y rotation and cryptographic hash function techniques. To detect the dishonest party we implemented the verification techniques using hash function at TTP. We have investigated the computation overhead, communication overhead and storage overhead for the outsourced data. The simulation result for accessing the outsourced data from the CSP shows that the proposed cloud security system is highly secure than the existing security systems. To support, insertion, deletion and updation dynamic operations on encrypted data block, we can further extend this security system.

REFERENCES

[1] http://security.setecs.com, *Security Architecture for Cloud Computing Environments white Paper*, 2011

[2] Ayad Barsoum and Anwar Hasan, *Enabling Dynamic Data and Indirect Mutual Trust for Cloud Computing Storage Systems,* In IEEE Transac-tions on Parallel and Distributed Systems, 2012.

[3] Cong Wang, Kui Ren, and Jia Wang, *Secure and Practical Outsourcing of Linear Programming in cloud computing*, In IEEE Internationa Conference on INFOCOM, pages 820-826, 2011.

[4] Q. Wang, C. Wang, J. Li, K. Ren,and W.Lou, *Enabling Public Verifi-ability and Data Dynamics for Storage Security in Cloud Computing*, In Proceeding of 14th European Symposium, Research in Computer-Security (ESORICS 09), pages 355-370, 2009.

[5] Junbeom Hur, *Improving Security and Efficiency in Attribute-Based Data Sharing*, In IEEE Transactions on Knowledge and Data Engineering, Volume:25,Issue: 10, pages 2271-2282, Oct. 2013.

International Journal of Scientific & Engineering Research, Volume 5, Issue 3, March-2014

ISSN 2229-5518

[6] Mohamed E M,Abdelkader H S,El-Etriby S, *Enhanced Data Security Model for Cloud Computing*, Informatics and Systems (INFOS), 8th International Conference on, pages 12-17, May 2012.

[7] Dubey A K, Dubey A K,Namdev M, Shrivastava S S, *Cloud-user Security based on RSA and MD5 Algorithm for Resource Attestation and Sharing in Java Environment*, Software Engineering (CONSEG), CSI Sixth International Conference on, pages 18, September 2012

[8] M Armbrust, A Fox, R Grifth, A D Joseph, and R Katz, *Above the Clouds: A Berkeley View of Cloud Computing*, U C Berkeley Reliable Adaptive Distributed Systems Laboratory White Paper, 2009.

[9] Cong Wang, Ning Cao, Jin Li, Kui Ren and W Lou, *Secure Ranked Keyword Search over Encrypted Cloud Data*, In IEEE 30th International Conference on Distributed Computing Systems (ICDCS), pages 253-262, 2010.

[10] Kuyoro S O, Ibikunle F, Awodele O, *Cloud Computing Security Issues and Challenges*, In International Journal of Computer Networks, vol. 3, issue 5, 2011.

[11] Cong Wang, Kui Ren, W Lou, Jin Li, *Toward Publicly Auditable Secure Cloud Data Storage Services*, In IEEE Computer Networks, pages 19-24, 2010.

[12] Sumita Tyagi and R Singh Yadav, *Cryptography and Network Security*, Educational and Technical Publishers, 2010.