Integrity Checking for Cloud Environment Using Encryption Algorithm

P.Varalakshmi[#1]
Department of Information Technology
Madras Institute of Technology
Chennai – 44
varanip@gmail.com

Hamsavardhini Deventhiran[#2]
Department of Information Technology
Madras Institute of Technology
Chennai -- 44
hamsdev.cse@gmail.com

***Abstract*-- Integrity checking becomes imperative to secure data in a cloud environment. It is important to ensure that the stored data is neither compromised nor corrupted. Many of existing protocols reveals client's sensitive data by sharing the encryption and decryption keys with the cloud server. Also, entrusting service provider at cloud side is of no use. The proposed work draws a conceptual cloud architecture by adopting an encryption algorithm with dynamic small size key to ensure the security and doesn't compromise any information with the cloud server. It involves a third party broker who encrypts the clients' information for ensuring the security, partitioned into multiple segments based on the remaining dynamic storage capacity presents in the VMs of cloud storage servers and store these encrypted segments of the clients' file on the corresponding VMs of the cloud storage providers. It then generate the hash value of the encrypted segments, store and manage these details for verification purpose. The performance measures like better encryption time and a quicker detection of compromise of the clients' files in cloud environment is possible with the proposed system.**

Keyterms: Data integrity, encryption, cloud storage server, broker.

I. INTRODUCTION

Nowadays storing the data in remote servers (cloud environment) becomes natural and also essential. Cloud is a large shared resource pool and the users move towards them with respect to their needs. It offers amenities for data storage, data access and other computational capabilities in a reliable manner. But, security becomes the major concern for both cloud service provider and the clients who are using the cloud resources.

Meanwhile, it reduces the overhead of cloud users in maintaining their crucial data in a secured manner. But in those cases, the provider must ensure that their infrastructure is secure enough to protect clients' data and its applications and the customer on the other hand, must be able to attest that the provider has taken proper security measures to protect their crucial information.

Although cloud architecture has its own security, Integrity is the core process to ensure storage security in cloud. In case of compromise by service provider at any cost, entrusting cloud side is of no use. Also at the same time, data must not be tampered by cloud server. Many works has been proposed by keeping the decryption algorithm in server side. Since there exist an environment where the remote server can't be trusted, it is necessary to assure that the data has not been tampered with.

The proposed system ensures the integrity of the client's information at the cloud sites with a trusted third party called a cloud broker.

II. RELATED WORK

Somani et al. [5] assess cloud storage and ensure security by implementing digital signature with RSA algorithm. Digital signature is used to ensure both sender and recipient that the message was only from trusted party, and that it was not altered in transient. The task of allowing a third party auditor (TPA) on behalf of the cloud client is considered in [3], to verify the integrity of the data stored in the cloud. It ensures remote data integrity by constructing Merkle hash tree for block tag authentication. To reduce heavy computational overhead at both client and the provider side, Sunil et al [6] proposed a approach that combines Diffie-Hellman key exchange protocol with cryptography to protect user's outsourced data. Here, data owner encrypts data files along with capability list before storing in cloud. To be noted, Service provider is unable to see the original file because the symmetric key is shared only between data owner and users in capability list. Sravan kumar et al. [17] modelled a method of proof for checking the correctness of user's sensitive data by adopting the use of Meta data. Meta data is created using randomly selected bits from original data file and is appended in an encrypted form to be stored on cloud. Whenever verifier wants to check integrity, he throws a challenge by specifying block number and its corresponding Meta data and finally decrypted for proof of correctness. In addition of providing threshold for retrieving meaningful data among distributed cloud servers, yashaswi singh et al.[4] defines another threshold which states that at least 'q' number of providers among 'p' providers (where p<q) must take part to retrieve a meaningful information. Cost also minimized for storing data among distributed servers. Main idea behind the

proposed design of cong wang et al. [10] is that the user would not only able to audit the storage but also detects the locality of the error i.e., identity of misbehaving servers. Comparison of pre-computed tokens with received response either guarantees the integrity or identifies the malicious server.

Qin Liu [18] gave a scheme for Hierarchical Identity based encryption algorithm which enables higher level users to share the storage data efficiently with lower level users. The file is encrypted only using public keys of intended recipients and stored in server. Number of authorized public keys is also taken as input for encryption algorithm. Syam kumar et al. [7] rely on pre-computation of verification token using sobol sequence for ensuring cloud data security. For a random set of blocks, token is generated and stored locally. Whenever user challenges cloud service provider, it computes a signature which is verified over the tokens stored locally by users. Zhang lianhong et al. [8] designed a scheme for data integrity check based on RSA security assumption by combining Identity based cryptography and Digital signature. Here, data is divided into blocks of variable size and those block tags are authenticated for integrity check instead of original data blocks.

III. PROPOSED SYSTEM

The proposed system comprised of three entities, a trusted third party called a cloud broker, clients and cloud storage service providers as shown in fig 1.

Clients of the cloud may be individual customers and any organization that have data to be stored in the cloud and rely on the cloud for data computation and its security maintenance. Mainly they store encrypted data in the remote server and to ensure security it queries the data through the broker for integrity checking.

The *storage servers* provide significant resources and expertise in managing cloud storage servers. It stores client's sensitive data in encrypted form and is considered to be un-trusted entities.

The *broker* acts as an interface between cloud storage server and client. Request Handling and Integrity Checking is done at this side. It is considered to be a trusted entity to assess and expose risk of cloud storage services on behalf of the clients upon request.

The Broker entity comprised of a partitioner, encryptor, decryptor, hash key generator, verifier and local database manager. The client's storage requests of files are sent to the request handler at the broker side and are queued at the encryptor. The files are then encrypted using encryption algorithm and sent to the partitioner.

Partitioner monitors the current remaining virtual storages present at various VMs at the cloud storage servers. Based on this remaining storage in VMs, the client files are taken and partitioned into multiple segments.

The segments are then given to the hash key generator to generate the hash key for those segments and returned to partitioner. Hash value of the encrypted segment is generates with SHA algorithm as follows.

Hash value=SHA(encrypted segment)

The details like the identity of client's file, encrypted segments, hash key generated for those encrypted segments, corresponding VM's IP address in which the encrypted segments are stored, and the sequence number of those encrypted segments are stored in the database, which is managed by the database manager. These details are required for the integrity checking of the clients' information which is managed by the database manager at the broker site.

When the clients want to retrieve the file from the cloud storage provider, it will send the retrieval request to the request handler in the broker site. The request handler will send this retrieval request to the verifier module of broker.

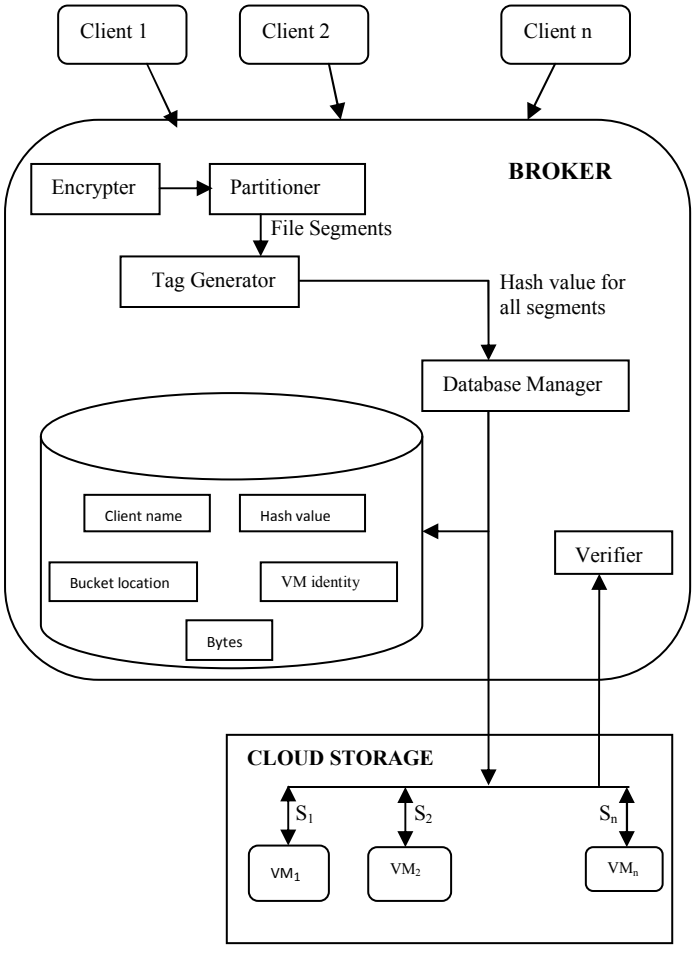
Verifier in turn with the identity of the client's file, retrieve the corresponding details from the database and send the retrieval request to the respective VMs. It will generate the hash key of that encrypted segments which is retrieved from VMs once again and compare it with already stored hash value. If all segments match, the verifier concludes that the file is intact. Then it combines all the encrypted segments of that file based on the sequence number.

Then, it decrypts that combined file using reverse of encryption algorithm and send it to the client. If there is any mismatch for the newly calculated hash value with originally stored hash value, the verifier concludes that the file got compromised by the corresponding VM and the same information is sent to the client.

In encryption algorithm [2], dynamic 4-bit key size is used and different 4-bit keys are used for different characters in the plain text. Algorithm can encrypt and decrypt the large data since there will be no increase in key size. Even though any one key is identified in encryption, other keys cannot be identified because of using different 4-bit keys for different characters rather than choosing only one 4-bit key and adding one to initial key. Security is also enhanced by encrypting secret key by using 4-bit shared key. In order to extract the secret plain text from the cipher text, first the secret key has to be decrypted by using shared key and then the character should be decrypted by using that secret key. In encryption after decrypting all the characters, plain text will be found.

Choosing the optimal bit size for keys in encryption and decryption algorithm is necessary for the efficient computation of the algorithm. If the bit size of the key is very large, the complexity of the algorithm is increased. This also increases the cost of the encryption and decryption algorithm. In many encryption and decryption algorithms the key size is very large. In [1], the dynamic 4-bit key is used for encryption and decryption for small amount of data. In encryption algorithm, 4-bit key size is used for encryption as well as decryption. 4-bit Randomizer is used in this algorithm which returns the random value from 10 to 15 i.e. from 1000 to 1111 for encrypting the characters and same key has to be used for decryption since it is a symmetric algorithm. So only 4-bit key size is used throughout the algorithm but different keys are used for characters in the plain text by using 4-bit Randomizer. Hence, the algorithm is used to encrypt and decrypt the large amount of data with small key size in efficient and secured manner.

Figure 1. Proposed System Architecture



Encryption algorithm is given below:

A key value that varies from 10 to 15 is chosen.

1) *The generated value is converted into binary value in 4 bits which is the secret key for encrypting a character.*
2) *The ASCII value for the character is computed in the plain text and the ASCII value is converted to binary representation in 8 bits.*
3) *Left circular shift operation is performed for n (the value generated in Step1) times on the binary representation of ASCII value*
4) *Divide the result from step 4 by the secret key.*
5) *The remainder is represented in first four bits and quotient in the last 5-bits which forms 9-bit cipher text for a character.*
6) *The 4-bit key is represented in 8 bits.*
7) *Again left circular shift operation is performed on these bits.*
8) *The resulting bit is divided with another 4-bit secured key.*
9) *Now the remainder is represented in first four bits and quotient in the last five bits which gives 9-Bit cipher text for the secret key.*
10) *18-Bit cipher text is formed by using 9-Bit cipher text for secret key and 9- bit cipher text for the character*

The above algorithm is repeated for encrypting the whole file. And this algorithm is reversed to decrypt characters from the cipher text.

IV. IMPLEMENTATION AND RESULTS

Eucalyptus users interacts via client tools with the Eucalyptus cloud have a variety of features for implementing, managing and maintaining collection of virtual resources such as network, storage and machines. It is an open source framework and is used in order to set up a cloud environment. It can be installed on any Open Source OS from source or by using binary packages. To initiate the cloud setup we have to start eucalyptus services on both front end and back end.

Xen is a hypervisor which allows several guest operating systems to execute on the same computer hardware concurrently. Xen systems have a structure with the Xen hypervisor as the lowest privileged layer. Above this layer, the hypervisor schedules one or more guest OS, across the physical CPUs. The first guest operating system, called "domain 0"(dom0), boots automatically when the hypervisor boots and receives special management privileges and direct access to all physical hardware by default.

Installation of xen in node controller allows controlling the hypervisor via HTTP from local host. It needs to be synchronized while running the eucalyptus. To enable a VM image as an executable entity, user must add a root disk image, a kernel/ramdisk pair to Walrus and register the uploaded data with Eucalyptus.

The proposed system is tested with 50 files varying the size of the files from 10K to 5MB. Fig 2 shows the encryption time of our encryption algorithm varies with increase in file size from 10KB to 5MB compared with already existing RSA algorithm. The graph shows that the encryption algorithm gives better encryption time compared to RSA even for large size files.

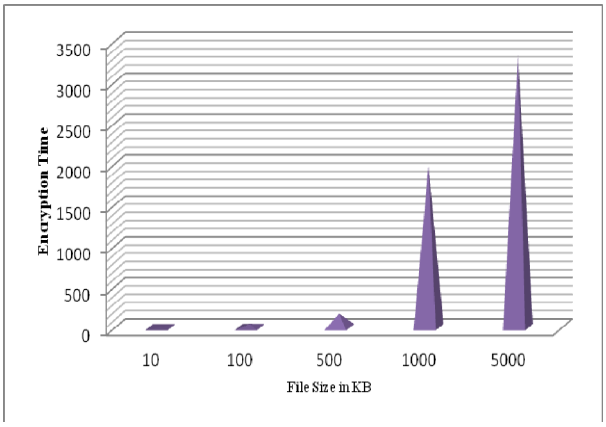


Figure 2. Encryption Time Taken By encryption Algorithm Compared With RSA

The content of 10 files are modified(corrupted) with 10%, 20%, 50% and 100% in the proposed system. Fig 3 shows the time taken by the existing algorithm and the proposed algorithm to detect the compromise of the clients' files.

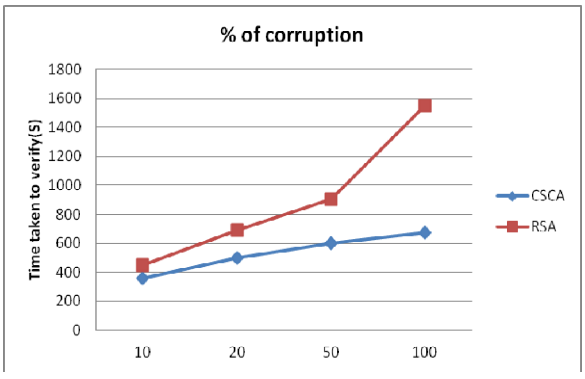


Figure 3. Time Taken To Detect conciliation With encryption and RSA

It is obvious from the above graph that encryption algorithm vary linearly with increase in percentage of corruption compared to the existing algorithm.

V. CONCLUSION

The proposed system provides a better integrity checking with the help of the trusted third party, a cloud broker with the encryption algorithm. Storage security is highly enhanced since only the encrypted data is stored at cloud sites and the client even doesn't get the whole file without the knowledge of the Database manager which is at broker side.

The performances measures such as encryption time and time taken to detect corruption are reduced and shown with encryption algorithm for a cloud environment.

REFERENCES

[1] R.Satheesh Kumar, E.Pradeep, K.Naveen and R.Gunasekaran, "*A Novel Approach for Enciphering Data of Smaller Bytes*", International Journal of Computer Theory and Engineering, Volume 2, No. 4, 2010.

[2] Y.RA. Kannan, S. Aravind Prasad and P. Varalakshmi, "*Cognitive Symmetric Key Cryptographic Algorithm*", The Second International Conference on Computer Science and Information Technology (CCSIT- 2012).

[3] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou and Jin Li, "*Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing*", IEEE transactions on parallel and distributed systems, volume 22, no. 5, 2011.

[4] Yashaswi Singh, Farah Kandah, Weiyi Zhang, "*A Secured Cost-effective Multi-Cloud Storage in Cloud Computing*",IEEE Infocom 2011 Workshop on Cloud Computing.

[5] Uma Somani, Kanika Lakhani and Manish Mundra, *"Implementing Digital Signature with RSA Encryption Algorithm to Enhance the Data Security of Cloud in Cloud Computing"*, 1st International Conference on Parallel, Distributed and Grid Computing (PDGC - 2010).

[6] Sunil Sanka, Chittaranjan Hota, Muttukrishnan Rajarajan, "*Secure Data Access in Cloud Computing*",978-1-4244-7932-0/10,IEEE 2010.

[7] P.Syam Kumar, R.Subramanian and D. Thamizh Selvam, *"Ensuring Data Storage Security in Cloud Computing using Sobol Sequence*", 2010 1st International Conference on Parallel, Distributed and Grid Computing (PDGC - 2010).

[8] Zhang lianhong, Chen Hua," *Secuirty Storage in the Cloud Computing: A RSA-based Assumption Data Integrity Check without Original Data",* 2010 International Conference on Educational and Information Technology (ICEIT 2010).

[9] Patrick Naughton and Herbert Schildt, *"The Complete Reference of Java",* Osborne/McGraw-Hill publications © 1999.

[10] Cong Wang, Qian Wang, Kui Re, Ning Cao and Wenjing Lou, "*Towards Secure and Dependable Storage Services in Cloud Computing*", IEEE transactions on services computing 2011.

[11] http://www.roseindia.net/tutorial/java/core/search.html

[12] http://open.eucalyptus.com/wiki/EucalyptusHypervisorGuide_v2.0

[13] http://open.eucalyptus.com/wiki/EucalyptusGettingStarted_v2.0

[14] Zhuo Hao, Sheng Zhong and Nenghai Yu, "*A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability*", IEEE transactions on knowledge and data engineering, volume 23, no. 9, 2011.

[15] Francesc Sebe, Josep Domingo-Ferrer, Antoni, Yves Deswarte, and Jean-Jacques Quisquate, "*Efficient Remote Data Possession Checking in Critical Information Infrastructures*", IEEE Transactions on Knowledge and Data Engineering, volume 20, no. 8, 2008.

[16] Dalia Attas and Omar Batrafi, "*Efficient Integrity Checking technique for Securing Client Data in Cloud Computing*", International Journal of Electrical & Computer Sciences IJECS-IJENS Volume 11, No: 05, 2011.

[17] Qin Liu, Guojun Wang, Jie Wu, *"Efficient Sharing of Cloud Storage Services",* 2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010).

[18] Sravan Kumar and Ashutosh Saxena, "*Data Integrity Proofs in Cloud Storage",* 978-1-4244-8953-4/11/$26.00©,2011 IEEE.