International Journal of Engineering Applied Sciences and Technology, 2019
Vol. 4, Issue 8, ISSN No. 2455-2143, Pages 290-294
Published Online December 2019 in IJEAST (http://www.ijeast.com)

INTEGRITY AUDITING OF CLOUD DATA WITH HOMOMORPHIC AUTHENTICATORS

Ms. Adlin Femil S
Department of CSE
Marthandam College of Engineering And Technology, Marthandam

.Ms. Bindhu A,
Department of CSE
Marthandam College of Engineering And Technology, Marthandam

Ms. Sreeja S S
Department of CSE
Marthandam College of Engineering And Technology, Marthandam

Abstract - An access control mechanism for the integrity of shared data with user resciding in cloud. By utilizing the idea of signatures, once a user entered in the group, the user can able to upload the file in cloud, with the use of secret key. The user needs to store only secret keys and signatures. It allows verification without the need for the challenger to compare against the original data.A public verifier is always able to audit the integrity of shared data without retrieving the entire data from the cloud.To securely introduce an effective auditor, the auditing process should bring no new vulnerabilities towards user data privacy. It enables an external auditor to examine the user's cloud data without knowing about the data script. As a result, the efficiency of user resciding can be significantly improved by the third party auditing mechanism.

Keywords- *Revocation, Auditing, Integrity of data*

I. INTRODUCTION

Cloud computing is a type of computing that implies the sharing of resources in computing through existing servers or personal devices to handle applications. Auditing through cloud is to provide a interface and a class of elements that companies who are interested in design or provide their audit processes (cloud or otherwise) as well as cloud computing providers to automate the Audit, financial valuation, liabilities for their infrastructure (IaaS), platform (PaaS), and application (SaaS) environments and allow authorized consumers of their services to do likewise via an open, extensible and secure interface and methodology "Lanxiang Chen et al.(2012) Algebraic signature" is a type of hash function that has algebraic properties: taking the signature of the sum of some file blocks gives the same result as taking the sum of the signatures of the corresponding blocks. A remote data possession checking scheme generally has five stages, namely Setup, TagBlock, Challenge, ProofGen and ProofVerify. In Setup stage, it generates some initialization parameters, such as the master key k, the tag encryption key k_t and some random numbers. In TagBlock stage, it selects c file blocks randomly to compute an algebraic signature of the sum as the verifiable tag, and then encrypts the tag using the tag encryption key k_t. The number of verification is t, and it needs to compute t tags. In the Challenge stage, user computes k_i for the i^{th} verification using the master key k, and then sends the (r_2, k_i) to storage server. In the ProofGen stage, the storage server computes the locations of the requested blocks using k_i, and computes their sum, F'_i, and then returns the (F'_i, T'_i) to the user. Note that T'_i is the verifiable tag stored on storage server. In the ProofVerify stage, the user decrypts T'_i using the tag decryption key k_t and computes the algebraic signature of F'_i , and then checks whether they are equal. If yes, it indicates that the file is not destroyed. "Ateniese et al. (2008) present a scheme with somewhat limited dynamism". They have developed a dynamic PDP solution called Scalable PDP. This scheme is based entirely on symmetric-key cryptography. The main idea is that, before outsourcing, data owner pre-computes a certain number of short possession verification tokens, each token covering some set of data blocks. The data is handed over to the server to check the integrity of data.

A Cloud Auditor can evaluate the services provided by a cloud provider in terms of security controls, privacy impact, and performance. The trusted party authority can evaluate the cloud services for the confirmation of integrity and security in cloud data.

For surety auditing, a Cloud Auditor can make an assessment of the security govern in the notice system to determine the extent towhichthe controls areinstru ment correctly, operating as designed, and make the desired event with venerate to congregation the shelter requirements for the system. "Erway, A. Kupcu et al. 2009" provide a definitional framework and efficient constructions for dynamic provable data possession (DPDP), which extends the PDP model to support provable updates on the stored data.

- Setup(1_k) → {s_k, p_k} is a probabilistic algorithm run by the client. It takes as input a security parameter, and outputs a secret key sk and a public key pk. The user is provided with the private and shared keys, and register the shared key to the server.
- Modifyblock(s_k, p_k, F, info,datablock) → {e(F), e(info), e(M)} is an algorithm run by the client to prepare(a part of) the file for untrusted storage. The system takes secret and shared keys as input (a part of) the file F with the definition info of the update to be performed (e.g., full re-write, modify block i, remove a block i, insert a block after block i, etc.), and the previous metadata M_c. The output is an "encoded" version of (a part of) the file e(F) (e.g., by adding randomness, adding security, encrypting for confidentiality, etc.), along with the information e(info) about the update (changed to fit the encoded version), and the new metadata e(M). The client sends e(F), e(info), e(M) to the server;
- VerifyUpdate(s_k, p_k, F, info, M_c, M'_c, $P_{M'c}$) → {allow, deny} is run by the client to verify the server'sbehavior during the update. It takes all inputs of the PrepareUpdate plus the M'_c, $P_{M'c}$ sent by the server. It results with acceptance (F can be deleted in that case) or rejection signals;
- VerifyGen(sk, pk, Mc) → {c} is a probabilistic procedure executed by the user to create a verify the server. It capture the secret and public keys, along with the latest client metadata Mc as input, and outputs a challenge c that is then sent to the server; VerifyProof(pk, Fi, Mi, c) → {P} is the method run by the server upon reception of a challenge from the client. It takes as input the shared key, the current version of the file and the metadata, and the challenge c. It results with an evidence P that is sent to the client;
- Eval(s_k, p_k, M_c, c, P) → {allow, deny} is the method run by the client upon receipt of the proof P from the server. It takes as input the secret and shared keys, the client metadata M_c, the challenge c, and the proof P sent by the server. If the output was accept it means that the server still has the file with correctness.

"Zhu. et al.(2011) "Consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The TPA get rid of the involvement of the client through the auditing of whether his data stored in the cloud are indeed intact, which can be important in achieving economies of scale for Cloud Computing. The support for data manipulations through the most common forms of data operations, such as block updation, adding, and removing, is also a significant step toward practicality, since services in Cloud Computing are not limited to backup data only. "Wang, H. Li, et al (2013) This scheme support public auditability and dynamic data operations". To support efficient handling of multiple auditing tasks, the technique of bilinear aggregate signature to extend is explored as well, where TPA can perform multiple auditing tasks simultaneously. To effectively support public auditability without having to retrieve the data blocks themselves, the homomorphic authenticator technique is used. Homomorphic authenticators are unforgeable metadata generated from individual data blocks, which can be securely aggregated in such a way to assure a verifier that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. This scheme used PKC-based homomorphic authenticator to equipment the verification protocol with public auditability. The limitation is that it is a single server mode "Wang et al.2012". propose to uniquely integrate the homomorphic linear authenticator with random masking technique to achieve privacy-preserving public auditing. In random marking technique the server's response is masked with randomness generated by the server. With random masking, the TPA no longer has all the necessary information to build up a correct group of linear equations and therefore cannot derive the user's data content, any number of linear combinations of the same set of file blocks can be collected. Executing the public auditing system consists of two phases, Setup and Audit:

- KeyGen: The user initializes the shared and secret parameters of the system by executing KeyGen, and pre-processes the data file F by using SignGen to generate the verification metadata. The user then stores the data file F and the verification metadata at the cloud server, and deletes its local copy. During pre-processing, the user may alter the data file F by expanding it or including additional metadata to be stored at server.

- Verification: The TPA generates an audit message or challenge to the cloud server to make sure that the cloud server has retained the data file F properly at the time of the audit. The cloud server will derive a response message by executing GenProof using F and its verification metadata as inputs. The TPA then verifies the response via VerifyProof.
- To improve the security of cloud data with the support of masking technique and HLA the TPA would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process. Also it allows TPA to perform multiple auditing tasks. Yet it does not support distributed server model.

II. SYSTEM OVERVIEW

System Model

The system model in this paper includes three entities: Security Mediator, Blind Signature and Key generation

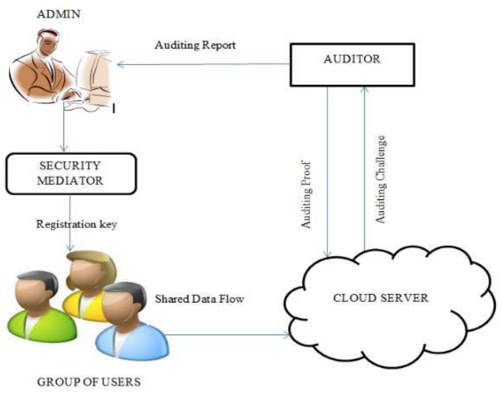


Fig1: Architecture Diagram

SECURITY MEDIATOR

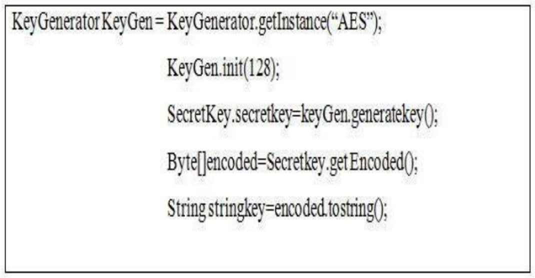
Blind Signatures - Blind signatures, form a special type of signatures where the message owner and the signer are different parties. More specifically, the message owner choose a blinding factor to blind the content of her message and sends the blinded message to the signer. After received the blinded message, the signer generates an identity on the blinded message and returns it to the message owner.

SIGNING EFFICIENCY

The communication requirement between a SEM and a data owner during signature generation should be smaller than directly transferring the data to be signed.

KEY GENERATION

Key Generation is a probabilistic algorithm run by the client. It takes as input a security parameter, and outputs a secret key and a public key. The secret and

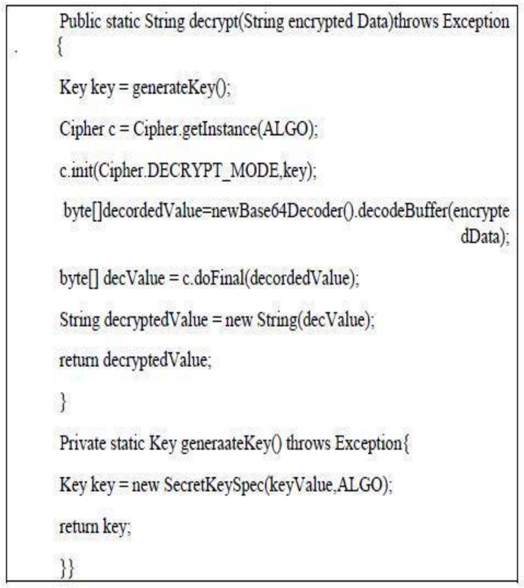


```
KeyGeneratorKeyGen=KeyGenerator.getInstance("AES");
                    KeyGen.init(128);
                    SecretKey.secretkey=keyGen.generatekey();
                    Byte[]encoded=Secretkey.getEncoded();
                    String stringkey=encoded.tostring();
```

Fig2: Key Generation Algorithm

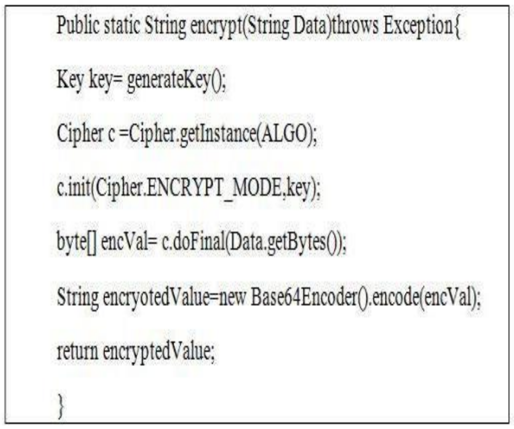
Advanced Encryption Standard Algorithm

The Advanced Encryption Standard (AES) is an encryption algorithm which encrypts the data to provide security and the encrypted data is placed on the cloud server. The AES algorithm is a n asymmetric algorithm which encrypt (encipher) data at sender side and decrypt (decipher) data at received side. Encrypion is the process of converting data in to an unkonwn form called cipher text; decrypting the cipher text is the process of conversion of the data back into its original form, called plaintext. The AES algorithm supports the key size of 128, 192, and 256 bits to encrypt and decrypt data in blocks.



```
Public static String decrypt(String encrypted Data)throws Exception
{
Key key = generateKey();
Cipher c = Cipher.getInstance(ALGO);
c.init(Cipher.DECRYPT_MODE,key);
 byte[]decordedValue=newBase64Decoder().decodeBuffer(encrypte
                                                        dData);
byte[] decValue = c.doFinal(decordedValue);
String decryptedValue = new String(decValue);
return decryptedValue;
}
Private static Key generaateKey() throws Exception{
Key key = new SecretKeySpec(keyValue,ALGO);
return key;
}}
```

Fig 3: Encryption



```
Public static String encrypt(String Data)throws Exception{

Key key= generateKey();

Cipher c =Cipher.getInstance(ALGO);

c.init(Cipher.ENCRYPT_MODE,key);

byte[] encVal= c.doFinal(Data.getBytes());

String encryotedValue=new Base64Encoder().encode(encVal);

return encryptedValue;

}
```

Fig 4: Decryption

III. PROPOSED SCHEME

Cloud Audit is the process of integrity verification over the cloud data by the third party authenticator. The proposed system specifies a moderized way to represent and share detailed, automated statistics about performance and security. The Users uploads their data to the cloud storage and no longer retainment of the data locally. Thus, the integrity and authentication of the data files. being stored on the distributed cloud servers must be guaranteed. One of the key issues is to effectively detect any unauthorized data modification and corruption.The technique of providing more security by using the third party auditor. The TPA allows the user to know the information about the data stored in the cloud. When anyone tries to steal or modify the data TPA informs the user by verifying the data. The third party doesn't even allow the CSP to read the data of the user. The main objective of this paper is to ensure the integrity of shared data.

IV. PERFORMANCE ANALYSIS

The elapsed time for AES encryption and decryption is calculated and compared by plotting a graph for input file of various sizes (10, 20,.., 50 bytes). It is analysed that, decryption consumes more time than encryption and also, as the size of input file increases, the execution time increases. Access Control mechanism are able to not only preserve the identities of the signers for dynamic groups, but also to efficiently audit the integrity of shared data for users. Our mechanism has the best performance on supporting groups. The total cost of adding a new user in our mechanism is only 0.13 second. When a user is revoked from the group, the cloud can re-sign all the blocks in shared data with a re-signing key, so that users do not need to download shared data and re-sign it by themselves

V. CONCLUSION

An access control mechanism, utilizing the AES algorithm and hashing code has been proposed and implemented to ensure data integrity and availability in cloud server by using key generation. When a user is revoked, the semi trusted cloud to resign the blocks that were signed by the revoked user with blind signature. The existing user in the group can have reduced computation time and communication resources during user revocation. To protect the integrity of the shared data, the uploaded file is attached with a signature or blind message. Once the user modifies the data, the file key will get change by using hash Code. By utilizing the hash code with uploaded data, this scheme achieves the integration and correctness of data, i.e., the key file will get change in each and every execution, this scheme can almost guarantee the simultaneous identification of the misbehaving users. In the proposed system, the key is generated using AES algorithm before uploading the data in cloud, then the public verifier audit the data uploaded by the user.

VI. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, (2010) "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, (2007) "Provable Data Possession at Untrusted Stores," in the Proceedings of ACM CCS 2007, pp. 598–610.

[3] H. Shacham and B. Waters, (2008) "Compact Proofs of Retrievability," in the Proceedings of ASIACRYPT 2008. Springer-Verlag, 2008, pp. 90–107.

[4] C. Wang, Q. Wang, K. Ren, and W. Lou, (2009) "Ensuring Data Storage Security in Cloud Computing," in the Proceedings of ACM/IEEE IWQoS 2009, 2009, pp. 1–9.

[5] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, (2009) "Enabling Public Veri- fiability and Data Dynamic for Storage Security in Cloud Computing," in the Proceedings of ESORICS 2009. Springer-Verlag, 2009, pp. 355– 370.

[6] C. Wang, Q. Wang, K. Ren, and W. Lou, (2010) "Privacy-Preserving Public Auditing for Data Storage

Security in Cloud Computing," in the Proceedings of IEEE INFOCOM 2010, 2010, pp. 525–533.

[7] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, (2011) "Dynamic Audit Services for Integrity Verification of Outsourced Storage in Clouds," in the Proceedings of ACM SAC 2011, 2011, pp. 1550–1557

[8] D. Cash, A. Kupcu, and D. Wichs, (2013)"Dynamic Proofs of Retrievability via Oblivious RAM," in Proceedings of EUROCRYPT 2013, 2013, pp. 279–295.

[9] C. Wang, Q. Wang, K. Ren, and W. Lou, (2011)"Towards Secure and Dependable Storage Services in Cloud Computing," IEEE Transactions on Services Computing, vol. 5, no. 2, pp. 220–232, 2011.

[10] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and S. Chen, "Dynamic Audit Services for Outsourced Storage in Clouds," IEEE Transactions on Services Computing, accepted.

[11] N. Cao, S. Yu, Z. Yang, W. Lou, and Y. T. Hou, (2012) "LT Codes-based Secure and Reliable Cloud Storage Service," in the Proceedings of IEEE INFOCOM 2012, 2012, pp. 693–701.

[12] J. Yuan and S. Yu, (2013)"Proofs of Retrievability with Public Verifiability and Constant Communication Cost in Cloud," in Proceedings of ACM ASIACCS-SCC'13, 2013.

[13] H. Wang, "Proxy Provable Data Possession in Public Clouds," IEEE Transactions on Services Computing, accepted.

[14] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, (2008) "Scalable and Efficient Provable Data Possession," in the Proceedings of ICST SecureComm 2008, 2008.

[15] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, (2009) "Dynamic Provable Data Possession," in the Proceedings of ACM CCS 2009, pp. 213–222.

[16] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, (2010)"Remote Data Checking for Network Coding-based Distributed Stroage Systems," in the Proceedings of ACM CCSW 2010, pp. 31–42.