Privacy Proven Public Auditing Scheme in Cloud
Public Auditing For Multiple Tasks

[1]Inbarasi.K, [2]Dr.Arivazhagu.U.V.
[1]Student, [2]Associate Professor
Kingston Engineering College, Vellore, India

***Abstract* - Cloud offers services that can not only be stored but can also be shared among multiple users. Data integrity is the major problem that could be assured by applying auditing mechanism. Several mechanisms has been introduced to preserve the data and identity of the users during public auditing which also leaves the traces to the auditor about the information of the users and their data. In this paper, we propose a new mechanism that solves the traceability problem while auditing and also a new batch auditing system that uses SOAP is introduced to handle multiple requests concurrently by improving the efficiency of the auditing.**

***Keywords:* public auditing, privacy, traceability, batch auditing**

I. INTRODUCTION:

Data stored in cloud cannot be assured for integrity due to the occurrence of threats like security and privacy threats. In order to verify the integrity of data, a third person is appointed to publicly verify the data of the user. Auditing is performed on the request given by the user i.e. Audit request on which the cloud sends audit challenge to the auditor that is verified by the auditor. In public auditing, several techniques were adopted to preserve the data privacy and identity privacy of the user. The latest and successful mechanism used to preserve data and identity from the auditor was Homomorphic Authenticable Ring Signatures (HARS) [2].

In the traditional mechanisms, the data is encrypted by the user to preserve from the auditor called as homomorphic encryption and identity is preserved by using ring signatures. However, the metadata is not unknown to the auditor since every updates to the data in the cloud is also updated to the auditor. This is called traceability problem.

In the mechanism designed in Oruta [2], the user gets the keys from the TPA for the data, encrypts it and then stores it in the cloud. During auditing, if the cloud generated signatures and the signatures generated by the users are equal then the data is said to be uncorrupted, if not the data is said to be corrupted. This mechanism achieves data and identity privacy but still leaves traces to the auditor about the updates of the user to the data.

Therefore in order to avoid traceability problem, instead of sending homomorphically encrypted data to the auditor, we just get only the keys for the signatures from auditor. So the auditor is unaware of the data content or any updates to the data preserving the data and identity of the user.

Handling multiple audit requests concurrently improves the efficiency of auditing because handling requests individually one by one consume more time thereby reducing the efficiency of auditing. Various protocols and algorithms like batch auditing protocol [5] and Homomorphic Proxy Signatures (HAPS) [3] were used in the traditional systems to concurrently handle multiple requests. In this paper we use SOAP based protocol to verify multiple audit requests simultaneously thereby preserving the data and identity of the user.

II. EXISTING TECHNIQUES:

Several mechanisms were proposed to perform multiple auditing simultaneously but those techniques do not meet the privacy terms to preserve both data and identity of the user.

Batch auditing Protocol:

Batch auditing protocol first verifies the tag generated for the user, generates the random challenge for which the responses were generated. The challenge and the response are compared and if they are equal the data is said to be verified. This protocol achieves batch auditing but preserves only data of the user, however the identity of the user is revealed.

Homomorphic Authenticable Proxy Signatures:

When verifier receives too many audit requests at a time, performance of multiple auditing tasks is increased. Homomorphic authenticable Proxy Signatures (HAPS) in batch auditing are used to reduce the number of paring operations to minimize the time taken for auditing. Though HAPS achieve batch auditing, it preserves only data nut reveals the identity information of the user to the auditor.

Pairing based batch verifier:

Pairing based verifying techniques [1] uses Σ-protocol, a three step protocol (commit, challenge, response) for batch verification. If it fails, it uses divide and conquer method to identify the bad signatures.

III. PROPOSED TECHNIQUE:

Our proposed technique is to eliminate traceability problem and a new batch auditing system to handle multiple requests simultaneously. By adopting this new technique, we can preserve the information that can be revealed to the auditor while updating the user files and data.

Traceability problem and solution:

In order to preserve the data from the auditor, we generate signatures for the data and get keys from the auditor to homomorphically encrypt the data and then store the data in the cloud. Since the user gets the keys from the auditor every time when the data is updated, auditor is aware that the data is being updated. Therefore to make unsure about the metadata of the user data, in this mechanism we only get the keys for the signatures and store the data, signatures for the data along with the code in the cloud. During auditing, cloud sends challenge and auditor verifies the challenge with the signatures thereby solving the traceability problem.

SOAP based batch auditing:

When the user wants his/her data to be verified for integrity, the user sends an audit request. The cloud on receiving audit signal from the auditor sends the lists of files under the user along with the signatures generated for it, which are then converted into XML format using SOAP protocol. SOAP encodes any type of messages into XML format. On receiving the batch files, the auditor parse the xml structure into batches and verifies the batches simultaneously. XML parser converts an xml document into XML-DOM object which are then verified in batches simultaneously.

Figure 1 demonstrates the solution for traceability problem and SOAP based batch auditing. The data owner encrypts his/her data using symmetric or asymmetric encryption techniques. The data owner then generates signatures for the encrypted data content and keys are received from the public auditor without sending the encrypted data. Since the signatures are generated without the sending the file content to the auditor, the auditor is unaware of the modifications done to the data. This solves the traceability problem. Whenever, the user wants his/her data to be verified for data integrity, he sends audit request to the auditor. The auditor requests for the audit challenge from the cloud service provider. On receiving the audit request from the auditor, the cloud server sends the audit challenge which contains signatures of the data content and files in batch manner. The signatures in batch manner are converted into XML structure using SOAP. The auditor on receiving the XML structured content parses it and extracts signatures for each signature in a batch. Comparing the signatures generated by the cloud and user generated signatures, the auditor sends the audit verify to the cloud server.

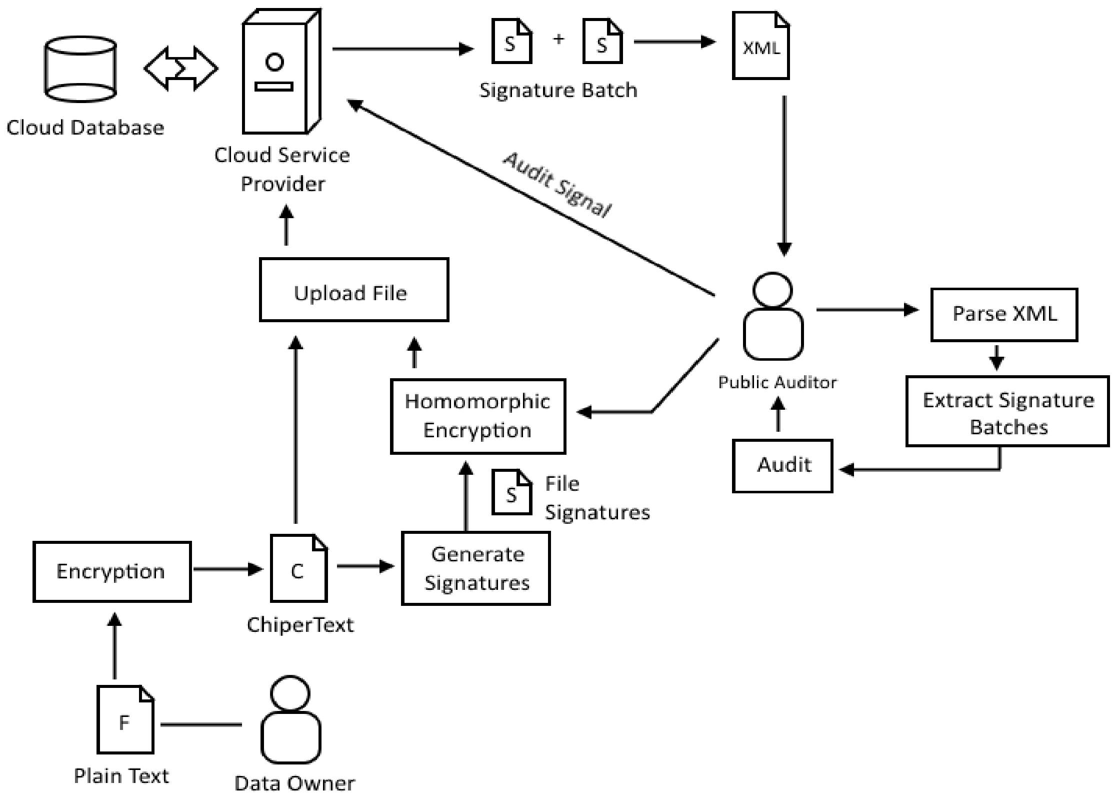

Figure.1 Traceability and SOAP based batch auditing

IV. IMPLEMENTATION DETAILS:

The implementation of the SOAP based batch auditing is as follows:

4.1 Key Generation:

Key generation is the initiation module of the file storing operation. A key is used to encrypt and decrypt data. Symmetric key algorithm uses a single shared key. Keeping data secret requires keeping this key secret. Public key algorithms use a public key and a private key. The public key is made available to everyone. A sender encrypts data with the public key. Only the holder of the private key can decrypt this data.

For shared data in the cloud, the user uses his/her private key to store data and other users uses public key to retrieve the data. The keys for the data to be stored are generated using the private and public key pairs. Each user in the group generates his/her public key and private key. Key generation module uses the KeyGen algorithm by the users to generate their own public/private key pairs. A user randomly picks a value generated from a group hash function and computes key for the block. The content is encrypted using an asymmetric scheme while the ring signatures are encrypted using a homomorphic scheme.

4.2 *Signature Generation:*

SigGen algorithm is used to compute ring signatures on blocks by using its user's private key and all the group members' public keys, a user can be the original user or group members. In SigGen, a user (either the original user or a group user) is able to compute ring signatures on blocks in shared data by using its own private key and all the group members' public keys. A user in the group is able to generate a signature on a block and its block identifier with his/her private key and all the group members' public keys. A block identifier is a string that can distinguish the corresponding block from others. A user generates a signature on a block and its block identifier with his/her private key and all the group members' public keys. A block identifier is a string that can distinguish the corresponding block from others.

With ring signatures, a verifier is convinced that a signature is computed using one of group members' private keys, but the verifier is not able to determine which one. Given a ring signature and a group of d users, a verifier cannot distinguish the signer's identity with a probability more than 1/d. This property can be used to preserve the identity of the signer from a verifier. Ring signatures cannot be directly used into public auditing mechanisms, because these ring signatures do not support blockless verifiability. Therefore ring signatures are combined with homomorphic authenticators to support blockless verifiability. Only a user with a private key can generate valid signature.

4.3 *Modify*

Each user in the group is able to perform an insert, delete or update operation on a block, and compute the new ring signature on this new block in Modify.

Figure 2 demonstrates the modifications that can be implemented to the data stored in the cloud. Insert, delete, update are the modifications that can be done to the user data. Whenever the user modifies the data content, new signatures are generated for the modified content and then stored into the database. Thus the old signatures are removed and new signatures are updated to the cloud.

A user in the group modifies the block in shared data by performing one of the following operations:

Insert: the user inserts the new block into the shared data and computes the new identifier of the inserted block. This user outputs the new signature of the inserted block with SigGen and uploads to the cloud server. Consider the number of blocks is *n* and after insertion the total number of blocks increases to *n*+1.

Delete: the user deletes the block, its identifier and ring signature from the cloud server. The identifiers and contents of other blocks in shared data remain same. Consider the number of blocks is *n* and after deletion the total number of blocks in shared data decreases to *n*-1.

Update: the user updates the shared data with new block. The virtual index of the block remains same. The identifier of the updated block changes and others remain same. The identifier of other blocks remains same. The number of blocks in shared data remains same, *n*.

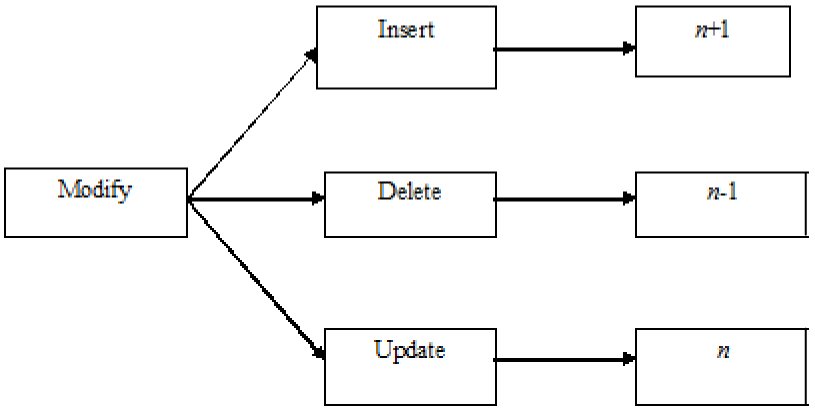

Figure.2 Modifying the cloud data

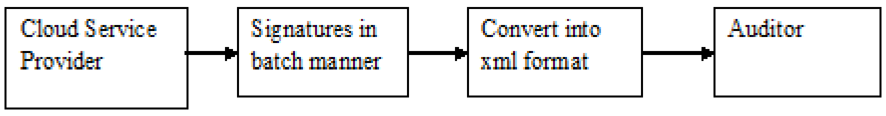

Figure 3 generating proofs for batches

4.4 *Batch Proof Generation*

Proof generation algorithm is operated by a public verifier and the cloud server together to interactively generate a proof of possession of shared data. If a user wants to protect the content of private data in the cloud, this user can also encrypt data before outsourcing it into the cloud server with encryption techniques, such as the combination of symmetric key encryption and attribute-based encryption The public verifier sends an audit challenge to the cloud server with initiate the proof generation algorithm to generate an auditing proof for each shared data. The cloud server runs the proof generation algorithm and sends the audit proof to the public verifier.

Figure 3 explains the process of generating proofs for the signatures from the cloud in batch manner. The cloud service provider generates for the data content, converts the content into xml format and then sends it to the auditor for auditing.

To audit the integrity of shared data, a public verifier

(i) Randomly picks an element of set [1,n] where n is the total number of blocks.
(ii) Generates signatures for the data and converts into xml format using SOAP.
(iii) Sends auditing challenge to the cloud server.

After receiving an auditing challenge from the public verifier, the cloud server generates proofs of possession of selected blocks, chooses a random element, computes identifier, aggregates signatures and returns an auditing proof to the verifier.

4.5 Batch Proof Verify

In ProofVerify, the public verifier audits the integrity of shared data by verifying the proof the proof. The public verifier audits the integrity of shared data by verifying the proof. With the auditing proof, an auditing challenge, public aggregate key and all group members public keys the public verifier checks the correctness of the proof by checking the following.

User Generated Block Signatures = Cloud Generated Block Signatures

If the above equation holds then the public verifier believes that the blocks in shared data are all correct. Otherwise, the integrity of shared data is incorrect.

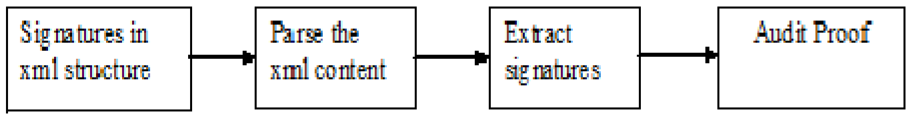

Figure 4 verifying proofs in batch manner

V. RESULTS AND DISCUSSIONS:

As discussed earlier public auditing designed in the existing systems preserves the data and the identity of the user from the auditor but still leaves the traces about the user's modification to the data since the user gets the keys by sending the encrypted content to the auditor. Since the keys are generated for each function of the user, the auditor may know that the data content is being modified. This traceability problem is resolved in this paper by just getting the keys without sending the content to the auditor, thereby preserving the details about the data modifications from the auditor ensuring both data and identity privacy f the user. Also to improve the efficiency of the auditing, multiple audit requests are handled simultaneously using SOAP which is referred to as SOAP based batch auditing. The techniques and ideas proposed in this paper are studied theoretically and is to be proved experimentally.

V. CONCLUSION:

We proposed a solution for the traceability problem discussed in the existing systems that was likely to occur during auditing by eliminating the traces that can be identified by the auditor. Also a SOAP based batch auditing is introduced to verify multiple audit requests simultaneously preserving the data and identity privacy of the users and their data and also minimizing the auditing time.

REFERENCES:

[1] A. L. Ferrara, M. Green, S. Hohenberger, and M. Pedersen,"Practical short signature batch verification," in Proc. of CT-RSA, volume 5473 of LNCS. Springer-Verlag, 2009, pp. 309–324.
[2] B. Wang, B. Li, and H. Li, (2012) "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," Proc. IEEE Fifth Int'l Conf. Cloud Computing, pp. 295-302.
[3] B. Wang, B. Li, and H. Li, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," Proc. IEEE Transactions, Cloud Computing, vol.2, no.1, pp. 43-56 2014.\
[4] B. Wang, B. Li, and H. Li, "Panda: Public Auditing for Shared data with Efficient User Revocation in the Cloud," IEEE Trans. Services Computing, 20 Dec. 2013, DOI: 10.1109/TSC.2013.2295611.
[5] C. Wang, S.S. Chow, Q. Wang, K. Ren, and W. Lou, (Feb. 2013) "Privacy-Preserving public Auditing for Secure Cloud Storage," IEEE Trans.Computers, vol. 62, no. 2, pp. 362-375.
[6] Kenneth G. Paterson and Jacob C.N. Schuldt, "Efficient identity-based signatures in the standard model,"
[7] Md.Tajuddin and K.China Busi, "An enhanced dynamic auditing protocol in cloud computing," International journal of Engineering trends and technology (IJETT)-vol.4 issue7-july 2013.