Secure Collaborative Privacy in Cloud Data with Advanced Symmetric Key Block Algorithm

Twinkle Graf.F[1], Mrs.Prema.P[2]
[1](M.E- CSE, Dhanalakshmi College of Engineering, Chennai, India)
[2](Asst. Professor – CSE, Dhanalakshmi College of Engineering, Chennai, India)

ABSTRACT: *Developments in cloud computing has led uploading large amount of data in the cloud. Many users can be facilitated through this. But in such cases security issues are more vulnerable. Clients can download data from cloud and reuse them for their research purposes. They can collaborate actively online for learning purposes. When the data is arbitrarily partitioned between two parties, both parties want to grab the data but they do not want that the other party should learn anything about their own data. The final weights are learned by the parties. There lacks a solution that allows two or more parties, to collaboratively conduct the updation. In our proposed approach, the parties encrypt their arbitrarily partitioned data and upload the cipher texts to the cloud .The cloud can execute most operations pertaining with the symmetric key block algorithm implemented in proposed scheme. Each updation is independent to the number of parties that is multiparty collaboration can be performed. Access to stored information on computer databases has increased greatly. So the main idea of our proposed scheme is to implement a privacy preserving algorithm for the cloud database*.

Keywords:

Symmetric blocks, Trust Agent, Cloud Service provider, Cipher blocks.

1 INTRODUCTION:

With the development in cloud storage servers and internet technologies most of the data are stored in the cloud. As cloud computing is a pay for use model, it attracts large number of users. With cloud services providing and storing data, users can easily modify and share data as a group. To ensure data integrity, data can be audited publicly; users need to compute signatures on each and every byte in the shared data. Different blocks of data are signed by different users due modifications in the data performed by different users. For security reasons, once a user is revoked from the group, the blocks, which were previously signed by this removed user must be re-signed by an existing user. The normal method, in which an existing user will resign to download the data which allows an existing user to download the corresponding data content is somewhat inefficient when data size is large i.e. metadata cases.

The paper proposes a novel public verifying mechanism for the integrity of shared data with efficient user identification. By utilizing signatures, we allow the cloud to re-sign blocks on behalf of existing users during user revocation. In addition, a public verifier is always able to verify the integrity of shared data without retrieving the entire data from the cloud.

2 MODELS AND ASSUMPTIONS

2.1 SYSTEM MODEL

With data storage and sharing services, such as Google Drive, provided by the cloud, people can easily work together as a group by sharing data with each other. Moreover people can upload their own datasets or data and also other users can also make use of them. Some websites also allow appending or changing the information as known by the user. The user may be one or the participating parties in much number should verify whether the information is updated by the authenticated user or someone else. More specifically, once a user creates shared data in the cloud, every user in the group is able to not only access and modify shared data, but also share the latest version of the shared data with the rest of the group. Although cloud providers promise a more secure and reliable environment to the users, the integrity of data in the cloud may still be compromised, due to the existence of hardware/software failures and human errors .In these mechanisms, a signature is attached to each block in data, and the integrity of data relies on the correctness of these signatures. One of the most significant and common features of these mechanisms is their ability to allow not only the data owner, but also a public verifier, such as a third party auditor (TPA) also may be a Trust Agent (TA), to check data integrity in the cloud without downloading the entire data, referred to as *public auditing*. Different from these works, our recent work focuses on how to preserve identity privacy from the TPA or TA when auditing the integrity of *shared data*.

2.2 CONTRIBUTION TO THE STUDY:

The main contributions of this paper are :

We propose an efficient method for encoding the data or information to be stored in the cloud. We ensure that a trust agent TA will be responsible for key generation and verification. The use of symmetric block key based cryptographic algorithm is used.

2.3 REQUIREMENT

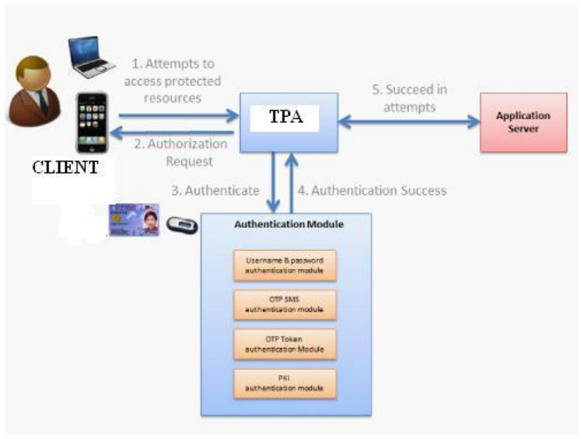
Now a days database management in the earlier period has become upgraded with data mining tools thus the vast amount of data available n the cloud are managed mechanically. All data are processed through machinated activities. The enormous amount of data stored in the cloud is easily being hacked and viewed. For instance, a bank transaction report which has been managed by a DB manager can be viewed to learn the tactics of DB management by any other DB manager. But in such a case the report should only be read by the other person. It should not allow write property to the other user. It will lead to security issues.

2.4 ASSUMPTIONS

In this paper, we assume that the scheme supports secure and efficient dynamic operations on data blocks, including: update, delete and append.

3 PROPOSED SCHEME

To enhance the security in cloud environment where data is shared and stored and transferred we propose a secure algorithm through which collaborative learning is performed. For instance PHI (protected Health Information) are all needed to be kept secure according to Health Insurance Probability and Accountability Act (HIPAA).For this purpose.


1. Attempts to access protected resources
CLIENT
2. Authorization Request
TPA
5. Succeed in attempts
Application Server
3. Authenticate
4. Authentication Success
Authentication Module
Username & password authentication module
OTP SMS authentication module
OTP Token authentication Module
PKI authentication module

3.1 ENTITIES

The proposed system has three important entities:

Fig 3.1.1 Entities

1) **(Participating) Users or Parties**: May be an individual or organization.

2) **Cloud Service Provider** (CSP): CSP contains resources and expertise in building and managing distributed cloud storage servers, owns and operates and leases the live Cloud computing systems.

3) **Trust Agent (TA),Third Party Authority (TPA)**
TA has expertise and capabilities that users may not have, is trusted to assess, audit and expose risk of cloud storage services on

behalf of the users upon request from the users.

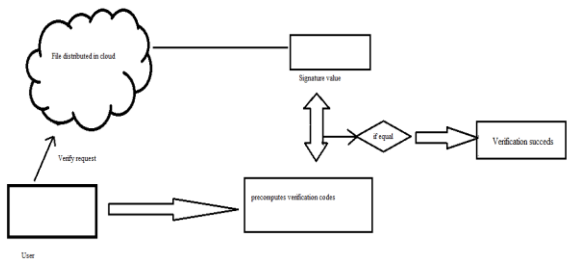


Fig 3.1.2 Trust Agent

Here we can adapt the symmetric key block encryption algorithm where the datasets or data is being separated as blocks or KEYBLOCKS(KB) as referred in this paper.

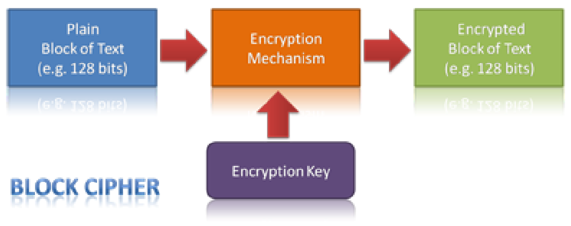


Fig 3.1.3 Cipher Block Encryption

First an encryption key or Encryption number is being generated which is the initial (init) key. In this block based method a proposed KEYBLOCK(KB) will contain all the possible words which can be denoted as n number of characters. This may include all characters of ASCII code from 0 to 255 (totally 256 characters). The key size could be 512 bit key size. After this key is generated by the TA , it stores this key for encoding and decoding. The key generation can be proposed as:

3.2 Proposed Key finding Steps:

1. Select any private key of Size 256 X 2 bits .
2. Size of selected key will be varying from 128 bits to 512 bits or 16 to 64 characters.
3. Any character from 0 to 255 ASCII code can be chosen.
4. Divide 64 bytes into 4 blocks of 16 bytes likes KEYBLOCKS KB1, KB 2, KB 3, and KB 4.
5. XOR operation between Block1 and Block3 can be stored in new KB 13.
6. XOR operation between Block2 and Block13 can be stored in new KB 213.
7. Apply XOR operation between KB213 and KB 4 can be stored in new KB4213.
8. Repeat Step 7, 8, 9 till we get random number / 4.
9. Exit

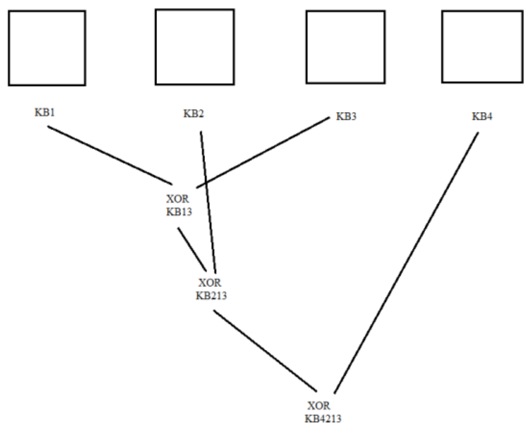


Fig 3.2 Keyblocks Division

A user or a participating party needs to verify his/her data content he/she has to find this key and keep. The initialization step is having the plain text value and hence the real owner of the data can only generate the key. If such a real owner of the data needs to ensure that his/her data is secure he/she needs to send a **verify** request to the CSP Cloud Service Provider. The signature value generated and the user's pre computed value, if both are same, then the data is disclosed to that user.

3.3 Steps of proposed Algorithm:

1. Initially select plane text of 16 bytes (or we can vary from 16 to 64 depend on requirement).
2. Initially insert key of size 16 bytes depend on the input plane text value.
3. XOR operation performed between key (KB4213) and plain text block (Text Block) will have the result stored in Cipher Block CB1.
4. Applying a right circular shift with 3 values will have the result stored in new CB2.

5. Applying XOR operation between CB2 and KB will have the stored value in new CB3.
6. Applying XOR operation between CB3 and KB4 will have the result stored in CB4.
7. CB4 is the output of the next round as a plane text block.
8. Repeat Step 1 to 7 till we get random number / 4.
9. Exit

3.4 Verify Token Creation:

The main idea is - when a file is distributed to the cloud, the user pre-computes a certain number of short verification tokens on individual vector F(j) (j {1, . . . , n}), each token covering a random subset of data blocks that would be distributed to the different cloud servers. Later, when the user wants to make sure the storage correctness for the data in the cloud, he sends the verify request to the cloud servers with a set of randomly generated block indices. Upon receiving verify, each cloud server (CS) computes a short "signature value" over the specified blocks and returns them to the user. The values of these signatures should match the corresponding tokens pre-computed by the user. Suppose if the user wants to verify the cloud server t times to ensure the correctness of data storage, the user must pre-compute x verification tokens for each F(j) (j {1, . . . , n}), a verification key *kverify* and a permutation key *Kperm*. To generate the *ith* token for server j, the user acts as follows,

1. Derive a verification key *kverify* and a permutation key *K(i) perm* based on *KPRP*.

2. Compute the set of r randomly-chosen indices.

3. Calculate the token *v(j)i* using the random verification value .

After token generation, the user has the choice of either keeping the pre-computed tokens locally or storing them in encrypted form on the cloud servers.

3.5 Correctness Verification

The response values from servers for each verification determines the correctness of the distributed storage. The procedure of the *ith* verify-response for verification over the d servers is described as follows:

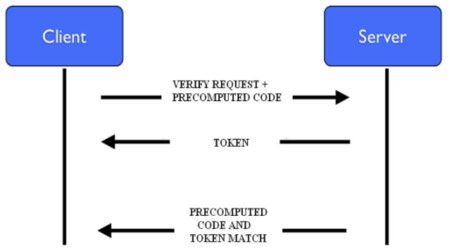

Client
Server
VERIFY REQUEST + PRECOMPUTED CODE
TOKEN
PRECOMPUTED CODE AND TOKEN MATCH

Fig 3.5.1 Token Verification

- The user shows the permutation key to each server.
- The server storage vector *F*(*j*) aggregates those k rows.
- Upon receiving linear combination from all the servers, the user verifies whether the received values remains as a valid codeword d**.**

4 SCHEME OVERVIEW:

To achieve the above goals, the main idea of our proposed scheme is to implement a privacy preserving equivalence for each step of the original algorithm. Our proposed scheme lets each party encrypt her/his input data set and upload the encrypted data to the cloud, allowing the cloud servers to perform most of the operations, i.e., additions and scalar products.

To support these operations over cipher texts, we adopt homomorphism encryption. It supports one step multiplication over cipher text, the intermediate results, for example, the intermediate products or scalar products, shall be first securely decrypted and then encrypted to support consecutive multiplication operations. For privacy preservation, however, the decrypted results known to each party cannot be the actual intermediate values. We design a secret sharing algorithm that allows the parties to decrypt only the random shares of the intermediate values. The random shares allow the parties to collaboratively execute the following steps without knowing the actual intermediate values. Data privacy is thus well protected. The overall algorithm is described in the parties jointly establish a neural network representing the whole data set without disclosing any private data to each other Secure Scalar Product and Addition.

4.1 Encryption:

Given a message M, encrypt it as:
The message being securely separated into blocks and applying XOR between the respective blocks and finding cipher blocks will encrypt the data.

4.2 Scalar product:

F(j) (j {1, . . . , n}) for each input file F being divided into blocks.

4.3 Secure Addition:

After obtaining the cipherblocks all the valuable codewords are added for every
M1;M2; . . .;Mn, the cloud computes their sum as:
Cosum C_i =CB1+CB2 . .CB(F(j(1. . .n))

4.4 Decryption:

We just demonstrate the decryption of Cosum as follows:

The cloud broadcasts Cosum to each party. On receiving the cipherblock , each party P computes Colums and returns the result to the cloud. With the results from all the parties, the cloud computes:
QCosum = decrypt (CB (F(j(1. . .n))) after decrypting the XOR blocks.

4.5 Key Verification:

Despite all the promises of cloud computing such as flexibility, pay as u go model etc., however, it has one unsolvable problem which is the security in cloud data. Data security in cloud data storage is very essential in a global distributed storage system. An effective and useful distributed block scheme is proposed to ensure the correctness of users' data in the cloud servers. If this correctness verification is too much resource consuming on the user's side, the task can be delegated to the third party auditor and the pre-computed tokens could be either in the user's local device or cloud server in encrypted format.

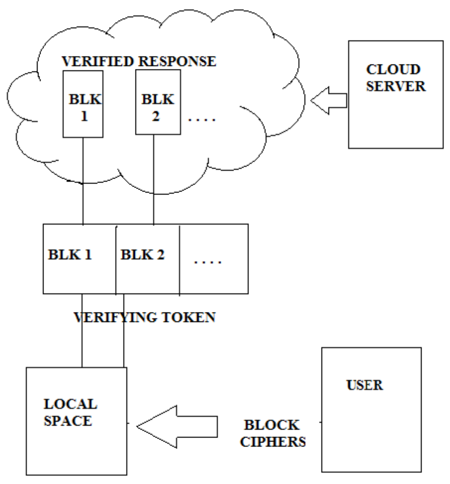


Fig 4.5 Distributed Block Scheme

5 CONCLUSION

Since the data in cloud can be viewed not only by authenticated users, there is a need to ensure security for all data. Privacy for such data in cloud can be preserved and write property is given to only those users whose pre computed code will be equal to the signature value generated by the server.

6 REFERENCES

[1] Sakshi Sanjay Deshmukh, Dr.G.R.Bamnote, Access to Encrypted Data in Cloud Database, *International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 3 Issue: 1 347 - 350*

[2] Cong Wang, Student Member, IEEE, Sherman S.M. Chow, Qian Wang, Student Member, IEEE,Kui Ren, Member, IEEE, and Wenjing Lou, Member, IEEE. Privacy-Preserving Public Auditing for Secure Cloud Storage, *IEEE TRANSACTIONS*

[3] Jiawei Yuan and Shucheng Yu, Member, IEEE "Privacy Preserving Back-Propagation Neural Network Learning Made
Practical with Cloud Computing". *IEEE Transactions on Parallel and Distributed Systems, (Volume:25 , Issue: 1)*

[4] Vishwa gupta, Gajendra Singh,Ravindra Gupta. "Advance cryptography algorithm for improving data security" *Volume 2, Issue 1, January 2012 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering*