Journal Homepage: -www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)
Article DOI:10.21474/IJAR01/7170
DOI URL: http://dx.doi.org/10.21474/IJAR01/7170

RESEARCH ARTICLE

SECURE DATA SHARING IN CLOUDS USING USER REVOCATIONS.

Mrs. Priyanka V. Surnar and Mr. S. G. Swami.
M.S.Bidve Engineering College SRTM University, Nanded.

Manuscript Info

Manuscript History

Received: 22 March 2018
Final Accepted: 24 April 2018
Published: May 2018

Keywords:-
Confidentiality, Integrity, Forward-Backward Access Control, Thread Security, Asymmetric Key.

Abstract

Cloud computing is technique that provides data privacy and reliability, access control, data forwarding(sharing) without using compute-intensive re-encryption, insider threat security, and forward and backward access control. The SeDaSC technique encrypts a file with a single encryption key. Two different key shares for each of the users are generated, with the user only receiving one share. The control of a single share of a key allows the SeDaSC attitude to counter the insider threats. The other key share is stored by a trusted third party, which is called the cryptographic server. This method is applicable to usual and mobile cloud computing environments. We implement a working prototype of the SeDaSC method and assess its performance based on the time consumed during a choice of operations. We officially confirm the working of SeDaSC by using high-level Petri nets, a Z3 solver.

Introduction:-

Cloud computing is elastic, flexible, and on-demand luggage compartment and computing services for customers. Organizations with a little budget can now make use of high computing and storage services not including heavily investing in infrastructure and maintenance The data are regularly encrypted before storing to the cloud. The access control, key management, encryption, and decryption processes are handled by the customers to ensure data security. When the data are to be shared among a group, the cryptographic system provide good services to different users, access control, and manage the keys in an effective manner to protect data confidentiality. A single key shared between all groups members will result in the access of past data to a newly joining member. A separate key for every user is a cumbersome solution. The data must be separately encrypted for every user. The changes in the data involve the decryption of all of the copy of the users and encryption once more with the customized contents, a attitude named Secure Data Sharing in Clouds (SeDaSC).The SeDaSC methodology works with three entities as follows: Users, A cryptographic server (CS), Cloud. The data holder submit the data, the list of the users, and the parameters necessary for generate an access control list (ACL) to the CS. The CS is a trusted third party and is answerable for key management, encryption, decryption, and access control. The CS generates the symmetric key and encrypts the data with the generated key.

Major contributions of in this paper, are as follows.

1. The planned method ensures the confidentiality of the data on the cloud by using symmetric encryption.
2. The protected data sharing over the cloud among the group of users is ensured without the elliptic curve or bilinear Diffie–Hellman problem (BDH) cryptographic reencryption.
3. The control of a portion of the key secures the data

Corresponding Author:- Priyanka V. Surnar.
Address:- M.S.Bidve Engineering College SRTM University, Nanded.

4. against malicious insiders within the group.
5. The proposed SeDaSC methodology secures the data
6. against issues of forward and backward access control that arise due to insider threats.
7. We perform formal modeling and verification of the
8. SeDaSC methodology by using high-level Petri nets
9. (HLPNs), the Satisfiability Modulo Theory Library (SMTLib),and a Z3 solver.

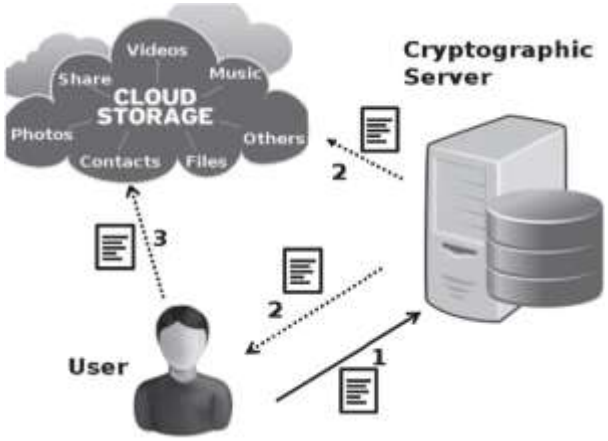

Fig. 1:-Basic idea for the SeDaSC methodology.

Literature Survey:-

Khan *et al.* [1] also utilize the El-Gamal cryptosystem and bilinear pairing for the sharing of sensitive in sequence in the cloud. The proposed scheme in [1] utilize the concept of incremental cryptography that divides the data into blocks and incrementally encrypts the blocks. The proposed scheme uses a trusted third party as a proxy that performs the compute-intensive operations of key generation, reencryption, and managing access to the data. However, the computational complexities of bilinear pairing still exist in the system.

Chen and Tzeng [2] planned a method based on the shared key derivation method for securing data sharing among a group. This technique uses a binary tree for the computation of keys. The computational cost of the proposed scheme is high as the rekeying mechanism is heavily employed in the proposed scheme. This scheme is not modified for public cloud systems because certain operations require centralized mediations.

Xu *et al.* [3] proposed a certificate less proxy reencryption (CL-PRE) scheme for securely sharing the data within a group in the public cloud. In the CL-PRE scheme, the data owner encrypts the data with the symmetric key. Consequently, the symmetric key is encrypted with the public key of the data owner. Both the encrypted data and the key are uploaded to the cloud. The encrypted key is reencrypted by the cloud that acts as a proxy reencryption agent that becomes decryptable by the user's private key. The public–private keys generated in the proposed scheme are not based on the certificates. The user's identity is used to generate the public–private key pair. The proxy reencryption is based on bilinear pairing and the BDH that makes the CL-PRE scheme computationally intensive. The computational cost of the bilinear pairing is high as compared with the standard operations in finite fields.

To reduce the computational overhead of bilinear pairing,

Seo *et al.* [4] introduced a mediated certificate less encryption approach for data sharing in the public cloud that avoids bilinear pairing. In the proposed scheme, the cloud generates the public–private key pairs for all of the users and transmits the public keys to all of the participating users. Partial decryption is performed at the cloud. Due to the fact that key management and partial decryption are handled by the cloud, user revocation is easier to handle. However, the proposed scheme treats the public cloud both as a trusted and untrusted entity at the same time. From a security perspective, it is not recommended to shift the key generation process to the shared multitenant public cloud environment. Moreover, the decryption is performed twice in the system that reduces the advantage of not pairing to some extent.

Mazhar Ali [5] proposed The SeDaSC methodology, which is proposed in this paper, securely shares the data among a group without using the El-Gamal cryptosystem, the BDH, and bilinear pairing. The SeDaSC methodology is based on symmetric cryptography without reencryption. The aforesaid properties avoid computationally intensive operations and make the SeDaSC methodology lightweight methodology. Moreover, the forward and backward access control is ensured by only allowing user access to a portion of the key that prohibits insiders to launch individual or coordinated attacks on the data.

Proposed System:-
In this section, we proposed method of SeDaSC that secures the sharing and forwarding of data among a group without involving reencryption in the cloud environment.

Entities:-
The SeDaSC method has the following entities.

Cloud:-
The cloud provides storage services to the user. The data on the cloud necessitate to be protected against privacy breach. The confidentiality of the data is ensured by storing encrypted data over the cloud. The cloud in the SeDaSC methodology only involves basic cloud operations of file upload and download. so, no changes at the protocol or implementation level on the cloud are required.

CS:-
The CS is a trusted party and is dependable for security operations, such as key management, encryption, decryption, the management of the ACL for providing confidentiality, and secure data forwarding among the group. The users of SeDaSC are necessary to be registered with the CS to obtain the security services.

Users:-
The users are the clients of the storage cloud. For each data file, one user will be the owner of the file, whereas the others in the group will be the data consumers. The owner of the file decides the access rights of the other group members.

Cryptographic Keys:-
SeDaSC methodology maintains a single cryptographic key for each of the data files. After encryption / decryption, the whole key is not stored and possessed by any of the involved parties.
The key is partitioned into two constituent parts and are possessed by different entities.
The following are the keys that are used within SeDaSC.

Symmetric Key K:-
K is a random secret generated by the CS for each of the data files. The length of *K* in SeDaSC is 256 bits, as is recommended by most of the standards regarding key length for symmetric key algorithms (SKAs). However, the length of the key can be altered according to the requirements of the underlying SKA. *K* is obtained in a two-step process.

In the first step, a random number *R* of length 256 bits is generated such that $R = \{0, 1\}256$.

In the next step, *R* is passed through a hash function that could be any hash function with a 256-bit output. In our case, we used secure hash algorithm 256 (SHA-256).

The second step completely randomizes the initial user-derived random number *R*. The output of the hash function is termed as *K* and is used in symmetric key encryption

User Key Share K'$_i$: K'_i is computed for each of the users in the group as follows:
$K'_i = K \oplus K_i$.

Algorithm 1:-
Key Generation and Encryption

Input:-
F, the ACL, the SKA, the 256-bit
hash function H_f

Compute:-
$R = \{0, 1\}^{256}$
$K = Hf(R)$
$C = SKA(F, K)$
for each user *i* in **the ACL, do**
$Ki = \{0, 1\}^{256}$
$Ki = K \oplus Ki$
Add *K'i* for user *i* in the ACL
Send *K i* for user *i*
end for
delete (*K*)
delete (K_i)
return *C* to the owner or upload to the cloud.

C SeDaSC Design:-
In this session , present the design of SeDaSC.

File Upload:-
Whenever a call for share data among the group arises, the owner of the file sends the encryption request to the CS. The request is accompanied by the file (*F*) and a list (*L*) of users that are to be granted access to the file. *L* also contains the access rights for each of the users. The users may have READ-only and/or READ–WRITE access to the file.

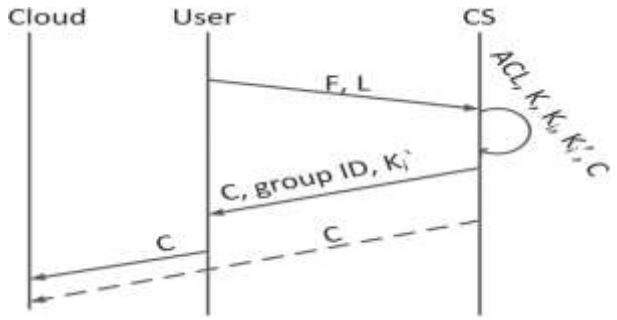

Fig. 2:-File upload.

File Download:-
The authorized user sends a download request to the CS or downloads the encrypted file (*C*) from the cloud and sends the decryption request to the CS. The cloud verifies the authorization of the user through a locally maintained ACL.

Algorithm 2 Decryption Algorithm

Input:-
C, the ACL, the SKA

Compute:-
Get *K*_
i from the requesting user
Get *C* from the requesting user or download from the cloud
Retrieve *Ki* from the ACL
If *Ki* does not exist in the ACL, then
return the access denied message to the user

else
$K = Ki \oplus K\_$
i
F = SKA(C, K)
send F to the user
end if
delete (K)
delete (Ki).

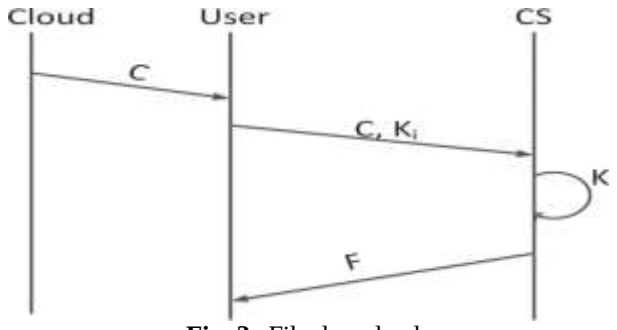

Fig. 3:-File download.

File Update:-
Updating the file has a similar procedure to that of uploading the file. The difference is that, while updating, all of the activities related to the creation of the ACL and key generation are not carried out. The user, who has downloaded the file and made any changes, sends an update request to the CS.

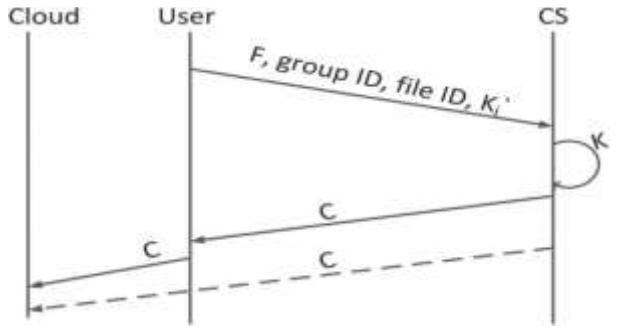

Fig. 4:-File Update

New Group User Inclusion:-
If a new user joins the group, the adding of the user is made on the demand of the file owner. The demand contains the user ID of the joining user, along with the access power parameters to be included in the ACL, and the group ID. The parameters include the IDs of the files for which the user has been approved access rights. It also includes the details indicating the READ and/or WRITE rights granted to the user. on the other hand, the date can be mentioned from which the access rights are valid for the user. This ensures the backward access control for the joining member. The CS, after receiving the joining request, updates the ACLs related to the files for which the access is granted. The key shares are generated, and the user shares are sent to the user along with the corresponding file IDs.

Departing Group User:-
The CS is notify about a departing member by the group owner. The CS removes all of the records for the departing user from the ACLs of the related files. As the whole key is not overcome by the group members, the departing member (even being malicious) will be unable to decrypt any of the group data files. Even the presence of encrypted files with a mean departing member will not affect the privacy of the data.

The SeDaSC methodology is proposed to provide the following services to the outsourced data:

1. confidentiality;
2. secure data sharing among the group;
3. secure data from unauthorized access of valid insiders
4. within the group; and
5. forward and backward access control to counter insiders and departing group users.

This paper, we propose a secure multiowner data sharing scheme, named Mona, for dynamic groups in the cloud. By leveraging group signature and dynamic broadcast encryption techniques, any cloud user can anonymously share data with others. Meanwhile, the storage overhead and encryption computation cost of our scheme are independent with the number of revoked users. In addition, we analyze the security of our scheme with rigorous proofs, and demonstrate the efficiency of our scheme in experiments.

Advantages:-
We propose a secure multi-owner data sharing scheme. It implies that any user in the group can securely share data with others by the untrusted cloud. We provide secure and privacy-preserving access control to users, which guarantees any member in a group to anonymously utilize the cloud resource.

Formal Analysis:-
we provide a brief introduction to HLPNs, the SMT-Lib,and a Z3 solver .

HLPNs;-
Petri nets are used for the graphical and mathematical demonstration of the system. Petri nets can model a range of systems, such as distributed, parallel, concurrent, nondeterministic, stochastic, or asynchronous systems . We have used a variation of a conventional Petri net called an HLPN.

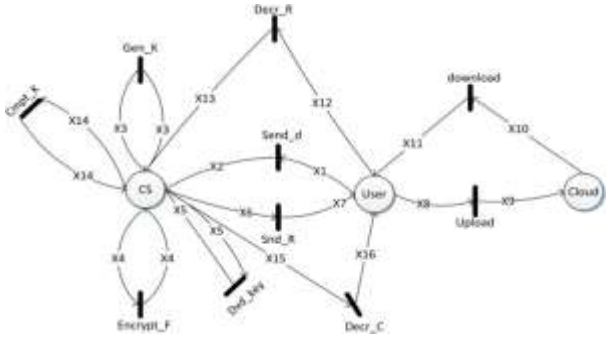

Fig.5:-HLPN model for SeDaSC.

SMT-Lib and Z3 Solver:-
The SMT is used for validating the satisfiability of rules
over the theories under consideration. We use a Z3 solver with the SMT-Lib that is not only theorem prover developed at Microsoft Research but is also an automated satisfiability checker.

Performance Evaluation:-
Experimental Setup:-
To evaluate the performance of the proposed methodology, we implemented the SeDaSC methodology in Netbeans IDE 8.0.2. The proposed methodology consists of three entities, i.e., the cloud, the CS, and the users. The CS is implemented as a third party. The functionality required by the user is implemented as a client application that connects with the CS to receive the services.

The classes TcpClient and TcpListener have been used to implement the Transmission Control Protocol (TCP). The communication was then secured using the SSLStream class. The scheme uses the SHA-256 hash function for generating keys and the AES for encryption and decryption.

Results:-

The SeDaSC methodology has been evaluate for the following three different cases.

Key Generation:-

Here is only one symmetric key generated for each file. Though, the keyshares are separately computed for every user in the group. The shares are computed at the time of file submission. We evaluate SeDaSC for time consumption in key generation. The time is computed for different numbers of users. We set the number of users to be 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. The results are shown in Fig. The time consumption for key generation increases with the increase in the number of users. Yet, it may be noted that the increase in the time consumption is not uniformly proportional to the increase in the number of users. For example, key generation takes 0.004 s for 10 users, and the time increases to 0.00512 s in the case of 50 users. The time has not increased in the same proportion as the number of users.

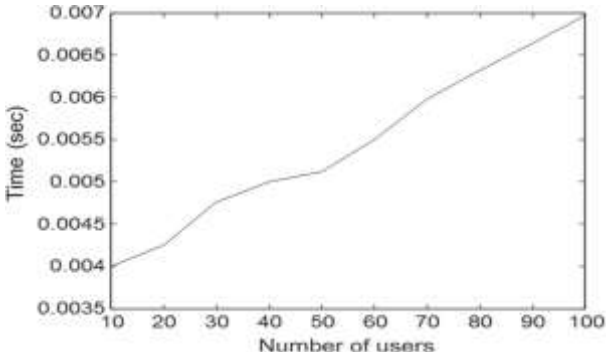

Fig.6:-Time consumption for key generation.

Fig.7:-key generation.

Encryption and Decryption:-

We evaluated the time consumption during the encryption and decryption of the file with varying file sizes. The file sizes used were 0.1, 0.5, 1, 10, 50,100, and 500 MB.

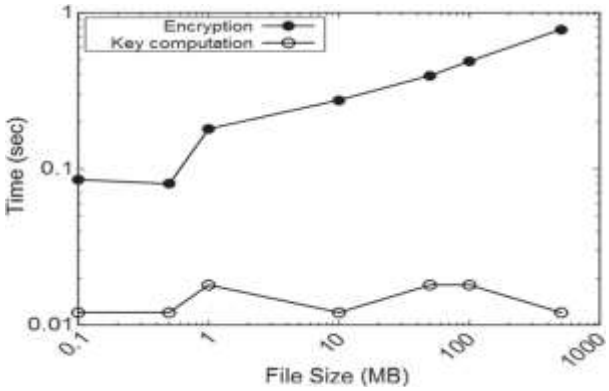

Fig. 8:-Performance of file encryption for SeDaSC.

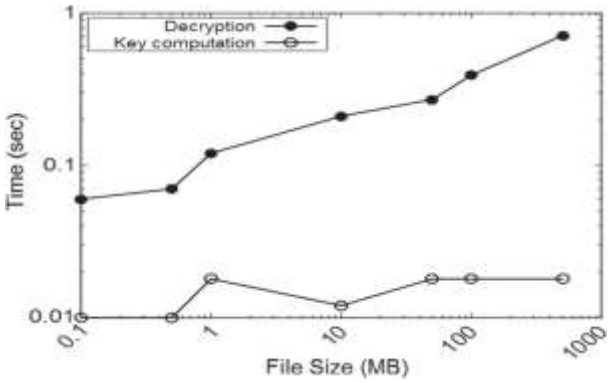

Fig. 9:-Performance of file decryption for SeDaSC.

File Upload/Download:-

We also evaluate the SeDaSC methodology on the basis of file upload & download.

Fig. 10:-File upload page for SeDaSC.

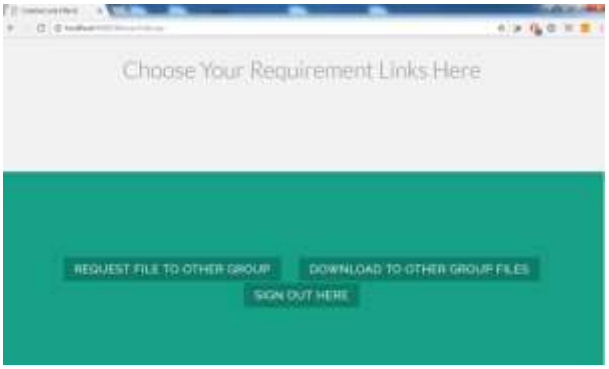

Fig. 11:-Download other group file in SeDaSC.

User Revocations:-
If admin found misbehavior in any group, then admin will cancel that group member.

Fig 12:-User Revocations for SeDaSC.

Fig. 13:-User deleted.

Conclusion:-

We proposed the SeDaSC attitude, which is a cloud storage security scheme for group data. The proposed technique provides data confidentiality, secure data sharing without reencryption, access control for malicious insiders, and forward and backward access control.

The encryption and decryption functionalities are performed at the CS that is a trusted third party in the SeDaSC methodology.

The proposed methodology can be also employed to mobile cloud computing due to the fact that compute-intensive tasks are performed at the CS.

The working of SeDaSC was formally analyzed using HLPNs, theSMT-Lib, and a Z3 solver.
If admin found misbehavior in any group, then admin will cancel that group member

References:-

1. A. N. Khan, M. M. Kiah, S. A. Madani, M. Ali, and S. Shamshir-band,"Incremental proxy re-encryption scheme for mobile cloud computing environment," *J. Supercomput.*, vol. 68, no. 2, pp. 624–651, May 2014.
2. Y. Chen and W. Tzeng, "Efficient and provably-secure group key management scheme using key derivation," in *Proc. IEEE 11th Int. Conf. TrustCom*, 2012, pp. 295–302.
3. L. Xu, X. Wu, and X. Zhang, "CL-PRE: A certificateless proxy reencryption scheme for secure data sharing with public cloud," in *Proc. 7th ACM Symp. Inf. , Comput. Commun. Security*, 2012, pp. 87–88.
4. S. Seo, M. Nabeel, X. Ding, and E. Bertino, "An Efficient Certificateless Encryption for Secure Data Sharing in Public Clouds," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2107–2119, Sep. 2013.
5. Mazhar Ali and Eraj Khan "SeDaSC: secure Data Sharing in clouds" in IEEE Trans,Nov 2014.