DATA SECURITY IN CLOUDS USING DECENTRALIZED ACCESS CONTROL, ANONYMOUS AUTHENTICATION AND RSA BASED ENCRYPTION

[1]VARALATCHOUMY.M, [2]PRATEEK. S. BHARADWAJ, [3]R. ROHITH, [4]NAGACHANDRA. K. P, [5]PRAVEEN. M

[1]Asst. Prof, Dept of ISE, DSCE, Bangalore
[2,3,4,5]Student, B.E, Dept of ISE, DSCE, Bangalore
E-mail: rohit.rr.30.rg@gmail.com

Abstract—A new scheme for securing the data stored in clouds that supports anonymous authentication using a decentralized access control method is proposed. In the proposed scheme, the cloud verifies the authenticity of the series without knowing the user's identity before storing data. Our scheme also has the added feature of RSA algorithm for encrypting data on cloud and access control in which only valid users are able to decrypt the stored information and deploying it to the cloud. The scheme prevents replay attacks and supports creation, modification, and reading data stored in the cloud and also addresses user revocation. Moreover, our authentication and access control scheme is decentralized unlike other access control schemes designed for clouds which are centralized. The communication, computation, and storage overheads are comparable to centralized approaches.

Index Terms—Decentralized Access Control, KDC, RSA algorithm

I. INTRODUCTION

Cloud computing is set of services offered through the internet. Cloud computing is receiving a lot of attention from both academic and industrial worlds. Cloud services are delivered from data centers located throughout the world. The boom in cloud computing has brought lots of security challenges for the consumers and service providers. In cloud computing, users can outsource storage and infrastructure to servers using Internet. Clouds can provide several types of services like applications (e.g., Google Apps, Microsoft online), infrastructures (e.g., Amazon's EC2, Eucalyptus, Nimbus), and platforms to help developers write applications.

Data stored in clouds is highly sensitive, for example, medical records and social networks. Providing security and privacy are important issues in cloud computing. Two main things are firstly, the user should authenticate itself before initiating any transaction, and on the second one is that, it must be ensured that the cloud does not tamper or interfere with the data that is outsourced or the data which is sent to the user. The wide acceptance of www has raised security risks along with the uncountable benefits so is the case with cloud computing. Also user privacy is required so that the cloud or other users do not know the identity of the user. The cloud can hold the user accountable for the data it outsources to the client, and likewise, the cloud is itself accountable for the services it provides to the client or the user who is accessing the cloud. The validity of the user who stores the data is also verified (by the admin). Apart from the technical solutions to ensure security and privacy in cloud, there is also a need for law enforcement such as access policies provided to the client or the users.

II. EXISTING SYSTEM

Existing work on access control in clouds are centralized in nature. All schemes use ABE or symmetric key approach and does not support authentication. Earlier work provides privacy preserving authenticated access control in cloud. However, the authors take a centralized approach where a single Key Distribution Center (KDC) distributes secret keys and attributes to all users. Unfortunately, a single KDC is not only a single point of failure but difficult to maintain because of the large number of users that are supported in a cloud environment.
Therefore, emphasize that clouds should take a decentralized approach while distributing secret keys and attributes to users. It is also quite natural for clouds to have many KDCs in different locations in the world. Although a decentralized approach is proposed in some of the existing papers, their technique does not authenticate users, who want to remain anonymous while accessing the cloud. In an earlier work, a distributed access control mechanism in clouds was proposed. However, the scheme did not provide user authentication. The other draw back was that a user can create and store a file and other users can only read the file. Write access was not permitted to users other than the creator.
Cloud servers are prone to Byzantine failure, where a storage server can fail in arbitrary ways. The cloud is also prone to data modification and server colluding attacks. In server colluding attack, the adversary can compromise storage servers, so that it can modify data files as long as they are internally consistent.

To provide secure data storage, the data needs to be encrypted. However, the data is often modified and this dynamic property needs to be taken into account while designing efficient secure storage techniques. Efficient search on encrypted data is also an important concern in clouds. The clouds should not know the query but should be able to return the records that satisfy the query. This is achieved by means of searchable encryption. The keywords are sent to the cloud encrypted, and the cloud returns the result without knowing the actual keyword for the search. The problem here is that the data records should have keywords associated with them to enable the search. The correct records are returned only when searched with the exact keywords. Security and privacy protection in clouds are being explored by many researchers.

Many homomorphic encryption techniques have been suggested to ensure that the cloud is not able to read the data while performing computations on them. Using homomorphic encryption, the cloud receives cipher text of the data and performs computations on the cipher text and returns the encoded value of the result. The user is able to decode the result, but the cloud does not know what data is has operated on. In such circumstances, it must be possible for the user to verify that the cloud returns correct results.

Disadvantages of Existing system

- The identity of the user is not protected from the cloud during authentication.
- There can be only one KDC for key management.
- Access control of data stored in cloud is centralized.
- Two users can collude and access data or authenticate themselves, if they are individually not authorized. This is called collusion attack.
- Revoked users can also access data even after they have been revoked.
- Prone to replay attacks.
- Single read and writes on the data stored in the cloud.

III. PROPOSED SYSTEM

According to the proposed system user can create a file and store it securely in the cloud. There are multiple KDCs (for example, here 2), which can be scattered. For example, these can be servers in different parts of the world. A creator on presenting the token to one or more KDCs receives keys for encryption/decryption and signing. We use R.S.A algorithm for encrypting the data since it is universally accepted. The example for operating environment for this project is the hospital data.

A. *System Architecture*

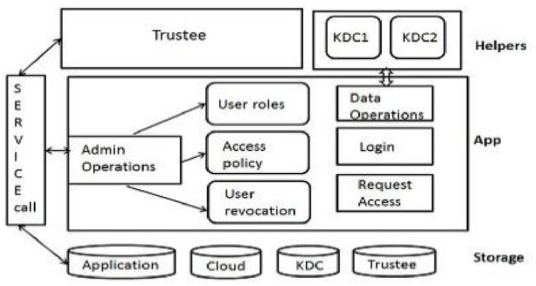

Fig. 1 System Architecture

The general block diagram describing the activities performed is as shown in Fig. 1. There are 3 layers in this architecture are Helpers, App and Storage.

Helper layer has Trustee and Key Distribution Center (KDC). Helper can have any number of KDC for security. Trustee helps to generate tokens for the users. KDC will generate Keys for encryption and decryption.

App layer has the following services: Request access will allow user to request access to data in cloud Admin operation are to accept or reject user request. User roles are nurse, doctor and lab technician. Access policies are read, write or read-write or nothing.Data operations are encryption and decryption. Service calls are calls made to trustee and KDC for Token and Key generation. User revocation is to prevent the access to data in cloud after user moves out of company.

Storage layer will contain the databases required for the proposed system.

IV. MODULES IMPLEMENTED

B. *Trustee*

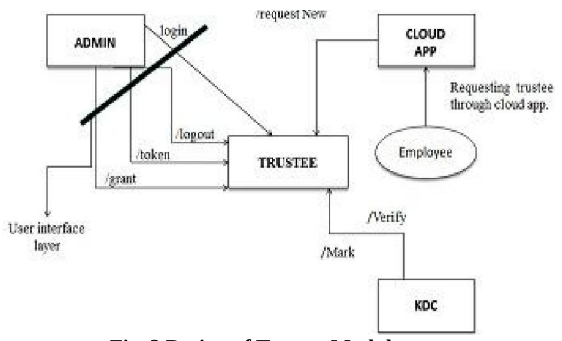

Fig. 2 Design of Trustee Module

Trustee will be maintaining the list of tokens. This is permission level security. As shown in Fig. 2, there are seven services provided by this module. */login* service will the used by the admin through an user interface layer for logging into the trustee application. */logout* service will be used by the admin through an user interface layer for logging out from the trustee application. */token* service will communicate with the data access object (DAO) layer to fetch the list of all tokens along with their status generated earlier. It will be used by the admin through the user interface layer.

/grant service is used by the admin either to approve or reject a new token. Upon accepting or rejecting, this service will send out an email to the one who requested the token. */request*New service will be used by the CloudApp to request a new token on behalf of the user. This will be implemented as an API service without user interface which means admin cannot use the service from user interface layer. */mark* service will be used by the KDC for marking a specific token as EXPIRED. This will also be implemented as an API service without any user interface layer. */verify* service will be used by KDC for checking if the token is valid one or not. This also will be implemented as an API service without any user interface layer.

C. Key Distribution Center (KDC)

KDC is a key distribution center which responsible for managing the generation and Distribution of keys and also to perform data operations like encryption and decryption. It is implemented as a dynamic web project without any user interface i.eno user including the admin cannot access the application.

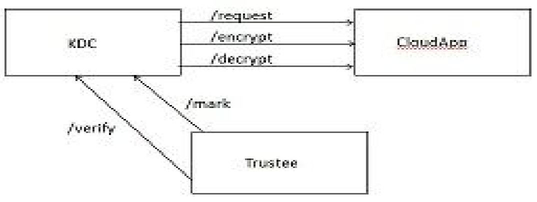

Fig. 3 Design of KDC module.

As shown in Fig. 3, thereare three services provided by KDC. */request* service will be consumed by the CloudApp to raise a new request for generating the key on behalf of organization employees and it will be implemented as an API service. */encrypt* service will be used by CloudApp to encrypt the organizational data. The key identifier received by an employee through an email should be provided as an input for the service along with the plain text that needs to be encrypted. */decrypt* service is used by the CloudApp to decrypt the cipher text when encryption key id is used for encryption along with the cipher text should be provided as an input for this service.

D. CloudApp

CloudApp is a module which acts as an intermediator between the user and other modules. User has access only to the CloudApp and other functions are being carried out through forwarding the requests to appropriate modules.

Table. 1 Work Flow of Cloud App

Inputs	Outputs
1) User request for a token.	1) Forwards request for a token to trustee.
2) User request for keys(via token)	2) Forwards request for keys to KDC.
3) Data + key id for encryption.	3) Forwards Data + key id to KDC for encryption.
4) Encrypted data from KDC	4) Writes the encrypted data to cloud.

Inputs:

1. User request for a token.
   When an employee wants to write data to a cloud, he has to request for a token first. Token is like an identifier. This is the way in which anonymous authentication is carried out.
2. User request for keys.
   Keys can be requested only if the token is obtained and is a valid one. Keys are used for encryption of data. Hence after token is generated and sent to the employee, he/she requests for a keys by passing tokens along with the request.
3. Data +key pointers for encryption.
   Once the request for keys is validated and if the token happens to be a valid one, keys are generated and corresponding key pointers will be sent as an email to the employee. He sends this key id along with the data to CloudApp for data encryption.
4. Encrypted data from KDC.
   After the key id +data encrypted by the KDC, the encrypted data is sent to the CloudApp for uploading it to cloud.

Corresponding Outputs:

1. Forwards request for token to trustee.
   Once the employee requests for a token, this request is forwarded to the trustee module where the token is generated and admin logs in from the trustee module to approve or reject the token.
2. Forwards request for keys to KDC.
   Once the token is approved and is sent to employee, he requests for keys for encryption purpose. This request is forwarded to KDC by the CloudApp. This is done by passing the token for KDC to verify the authenticity of the employee.
3. Forwards Data+key id to KDC for encryption.
   After user gets key pointers, he sends the data along with the key pointers to CloudApp for encrypting the data. CloudApp forwards this block to the KDC for encryption. Encryption is carried out in KDC.
4. Writes the encrypted data to Cloud.
   KDC encrypts the data and sends this encrypted data to CloudApp. It is the responsibility of the CloudApp to write this encrypted data to the cloud.

V. REAL-TIME APPLICATION

A real time application developed based on proposed system is presented here. A Hospital Management System over the Cloud has been developed. Since the hospital data is extremely sensitive, a scenario of how the medical records can be uploaded to cloud is shown below. There are two main actors in this case.

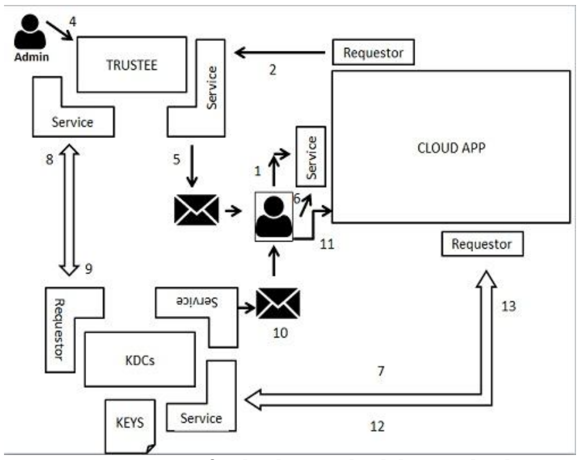

Fig. 4 Process of uploading medical data to cloud

One is the admin who is responsible for authorizing the users of the hospital. The other actor is the user who can upload the medical records. The users are assigned roles such as doctor, nurse or lab technician. The various steps involved in Hospital Management System is as depicted in Fig. 4. The various steps involved are discussed using the follow algorithm:

- Step 1: Employee requests the token to *CloudApp.*
- Step 2: *CloudApp* forwards the request from the employee to the *trustee*.
- Step 3: *trustee* will generate a token right away and marks it as "new" status.
- Step 4: Admin logs in to the *trustee* and approves the token.
- Step 5: *trustee* sends the token to the employee through an email.
- Step 6: Employee requests the *CloudApp* for the key. He/she passes the token along with this request.
- Step 7: *CloudApp* forwards this request along with the token to the *KDC.*
- Step 8: *KDC* requests the trustee to check if the token is valid or not.
- Step 9: *trustee* responds back by comparing the keys sent by the employee and the keys generated. If both are equal, it means the token is valid. Otherwise it is invalid.
- Step 10: If the token is valid, *KDC* will generate the keys and share the pointer to the keys with the employee through email.
- Step 11: Employee provides the data along with the key pointers to the *CloudApp* for encryption.
- Step 12: *CloudApp* forwards that key pointers and data to the *KDC.*
- Step 13: *KDC* will encrypt the data with the actual key generated earlier and returns to the CloudApp.
- Step 14: *CloudApp* writes the encrypted data to the cloud.

VI. EXPERIMENTAL RESULTS

RSA algorithm for encryption of data unlike other existing systems which uses attribute based encryption. The disadvantages of using attribute based encryption are that, there aren't a lot of deployments as it is still new, and the full benefit can be obtained only after deploying sufficient infrastructure. Vendors are still playing with the right implementation of the right protocols.
The major advantage of using RSA algorithm is thatit uses Public Key encryption. This means that your text will be encrypted with someone's Public Key (which everyone knows about). However, only the person it is intended for can read it, by using their private key (which only they know about). Attempting to use the Public Key to decrypt the message would not work. RSA can also be used to "sign" a message, meaning that the recipient can verify that it was sent by the person they think it was sent by.
One of the advantage of cloud computing is cost. Backing up your data isn't always cheap, especially when you factor in the cost of any equipment needed to do so – think external hard drives or backup tapes. Additionally, there is the cost of the time it takes to manually complete routine backups. Online storage services reduce much of the cost associated with traditional backup methods, providing ample storage space in the cloud for a low monthly fee.

CONCLUSION

We have presented a decentralized access control technique with anonymous authentication, which provides user revocation and prevents replay attacks. The cloud does not know the identity of the user who stores information, but only verifies the user's credentials. Key distribution is done in a decentralized way. One limitation is that the cloud knows the access policy for each record stored in the cloud. In future, we would like to hide the attributes and access policy of a user.

REFERENCES

[1] S. Ruj, M. Stojmenovic, and A. Nayak, "Privacy Preserving Access Control with Authentication for Securing Data in Clouds," Proc. IEEE/ACM Int'l Symp. Cluster, Cloud and Grid Computing, pp.556563, 2012.
[2] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward Secure and Dependable Storage Services in Cloud Computing," IEEE Trans. Services Computing, vol. 5, no. 2, pp. 220-232, Apr.June 2012.
[3] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy Keyword Search Over Encrypted Data in Cloud Computing," Proc. IEEE INFOCOM, pp. 441-445, 2010.
[4] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," Proc. 14th Int'l Conf. Financial Cryptography and Data Security, pp. 136149, 2010.
[5] H. Li, Y. Dai, L. Tian, and H. Yang, "Identity-Based Authentication for Cloud Computing," Proc. First Int'l Conf. Cloud Computing (CloudCom), pp. 157-166, 2009.

[6] C. Gentry, “A Fully Homomorphic Encryption Scheme,” PhD dissertation, Stanford Univ., http://www.crypto.stanford.edu/ craig, 2009.

[7] A.-R. Sadeghi, T. Schneider, and M. Winandy, “Token-Based Cloud Computing,” Proc. Third Int’l Conf. Trust and Trustworthy Computing (TRUST), pp. 417-429, 2010.

[8] R.K.L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B.S. Lee, “Trustcloud: A Framework for Accountability and Trust in Cloud Computing,” HP Technical Report HPL-2011-38, http://www.hpl.hp.com/techreports/ 2011/HPL-2011-38.html, 2013.

[9] R. Lu, X. Lin, X. Liang, and X. Shen, “Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing,” Proc. Fifth ACM Symp. Information, Computer and Comm. Security (ASIACCS), pp. 282-292, 2010.

[10] D.F. Ferraiolo and D.R. Kuhn, “Role-Based Access Controls,” Proc. 15th Nat’l Computer Security Conf., 1992.

[11] D.R. Kuhn, E.J. Coyne, and T.R. Weil, “Adding Attributes to RoleBased Access Control,” IEEE Computer, vol. 43, no. 6, pp. 79-81, June 2010.

[12] M. Li, S. Yu, K. Ren, and W. Lou, “Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-Owner Settings,” Proc. Sixth Int’l ICST Conf. Security and Privacy in Comm. Networks (SecureComm), pp. 89-106, 2010.

[13] S. Yu, C. Wang, K. Ren, and W. Lou, “Attribute Based Data Sharing with Attribute Revocation,” Proc. ACM Symp.Information, Computer and Comm. Security (ASIACCS), pp. 261-270, 2010.

[14] G. Wang, Q. Liu, and J. Wu, “Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Storage Services,” Proc. 17th ACM Conf. Computer and Comm. Security (CCS), pp. 735-737, 2010.

[15] F. Zhao, T. Nishide, and K. Sakurai, “Realizing Fine-Grained and Flexible Access Control to Outsourced Data with Attribute-Based Cryptosystems,” Proc. Seventh Int’l Conf. Information Security Practice and Experience (ISPEC), pp. 83-97, 2011.

[16] S. Ruj, A. Nayak, and I. Stojmenovic, “DACC: Distributed Access Control in Clouds,” Proc. IEEE 10th Int’l Conf. Trust, Security and Privacy in Computing and Communications (TrustCom), 2011.

[17] http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs01-en.pdf, 2013.

[18] http://securesoftwaredev.com/2012/08/20/xacml-in-the-cloud, 2013.

[19] S. Jahid, P. Mittal, and N. Borisov, “EASiER: Encryption-Based Access Control in Social Networks with Efficient Revocation,” Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS), 2011.

[20] R.L. Rivest, A. Shamir, and Y. Tauman, “How to Leak a Secret,” Proc. Seventh Int’l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT), pp. 552-565, 2001.

★★★