International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.04,Issue.18,
June-2015,
Pages:3411-3414

www.ijsetr.com

Design and Implementing Secure Access Control with User Authentication Protocol in Distributed Clouds

MOHAMMAD MOHASIN[1], AKKANA VAMSI KRISHNA[2]

[1]PG Scholar, Dept of CSE, Narayana Engineering College, Nellore, AP, India.
[2]Assistant Professor, Dept of CSE, Narayana Engineering College, Nellore, AP, India.

Abstract: This paper introduces a new decentralized access control scheme for secure data storage in clouds that supports anonymous authentication. In this scheme, the cloud verifies the authenticity of the series without knowing the user's identity before storing data. This scheme also has the added feature of access control in which only valid users are able to decrypt the stored information. The scheme prevents replay attacks and supports creation, modification, and reading data stored in the cloud. This also addresses user revocation. Moreover, our authentication and access control scheme is decentralized and robust, unlike other access control schemes designed for clouds which are centralized. The communication, computation, and storage overheads are comparable to centralized approaches.

Keywords: Attribute Based Encryption, Attribute Based Signature, User Revocation.

I. INTRODUCTION

Research in cloud computing is receiving a lot of attention from both academic and industrial worlds. In cloud computing, users can outsource their computation and storage to servers (also called clouds) using Internet. Much of the data stored in clouds is highly sensitive, for example, medical records and social networks. Security and privacy are, thus, very important issues in cloud computing. In one hand, the user should authenticate itself before initiating any transaction, and on the other hand, it must be ensured that the cloud does not tamper with the data that is outsourced. User privacy is also 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, and likewise, the cloud is itself accountable for the services it provides. The validity of the user who stores the data is also verified. Apart from the technical solutions to ensure security and privacy, there is also a need for law enforcement.

A. Existing System

Existing work on access control in cloud are centralized in nature. The scheme uses a symmetric key approach and does not support authentication. The schemes do not support authentication as well. Earlier work by Zhao et al. 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. We, 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 Yang et al. proposed a decentralized approach, their technique does not authenticate users, who want to remain anonymous while accessing the cloud. In an earlier work, Ruj et al. proposed a distributed access control mechanism in clouds. However, the scheme did not provide user authentication. The other drawback 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.

B. Proposed System

In the preliminary version of this paper, we extend our previous work with added features that enables to authenticate the validity of the message without revealing the identity of the user who has stored information in the cloud. In this version we also address user revocation that was not addressed. We use ABS scheme to achieve authenticity and privacy. Unlike our scheme is resistant to replay attacks, in which a user can replace fresh data with stale data from a previous write, even if it no longer has valid claim policy. This is an important property because a user, revoked of its attributes, might no longer be able to write to the cloud. We, therefore, add this extra feature in our scheme and modify appropriately. Our scheme also allows writing multiple times which was not permitted in our earlier work.

C. Main Contributions

The main contributions of this paper are the following:

- Distributed access control of data stored in cloud so that only authorized users with valid attributes can access them.

- Authentication of users who store and modify their data on the cloud.
- The identity of the user is protected from the cloud during authentication.
- The architecture is decentralized, meaning that there can be several KDCs for key management.
- The access control and authentication are both collusion resistant, meaning that no two users can collude and access data or authenticate themselves, if they are individually not authorized.
- Revoked users cannot access data after they have been revoked.
- The proposed scheme is resilient to replay attacks. A writer whose attributes and keys have been revoked cannot write back stale information.
- The protocol supports multiple read and write on the data stored in the cloud.

II. BACKGROUND WORK

In this section, we present our cloud storage model, adversary model and the assumptions we have made in the paper.

A. Assumptions

We make the following assumptions in our work:

- The cloud is honest-but-curious, which means that the cloud administrators can be interested in viewing user's content, but cannot modify it. This is a valid assumption that has been made. Honest-but-curious model of adversaries do not tamper with data so that they can keep the system functioning normally and remain undetected.
- Users can have either read or write or both accesses to a file stored in the cloud.
- All communications between users/clouds are secured by secure shell protocol, SSH.

B. Access Policies

Access policies can be in any of the following formats: 1) Boolean functions of attributes, 2) linear secret sharing scheme (LSSS) matrix, or 3) monotone span programs. Any access structure can be converted into a Boolean function.

C. Attribute-Based Encryption

ABE with multiple authorities as proposed by Lewko and Waters.

System Initialization: Select a prime q, generator g of G_0, groups G_0 and G_T of order q, a map $e: G_0 \times G_0 \rightarrow G_T$, and a hash function $H: \{0,1\}^* \rightarrow G_0$ that maps the identities of users to G_0. The hash function used here is SHA-1. Each KDC has a set of attributes. The attributes disjoint. Each KDC also chooses two random exponents. The secret key of KDC A_j is

$$SK[j] = \{\alpha_i, y_i, i \in L_j\}. \quad (1)$$

The public key of KDC is published

$$PK[j] = \{e(g,g)^{\alpha_i}, g^{y_i}, i \in L_j\}. \quad (2)$$

Key Generation: User receives a set of attributes from KDC and corresponding secret key for each attribute

$$sk_{i,u} = g^{\alpha_i} H(u)^{y_i}, \quad (3)$$

Note that all keys are delivered to the user securely using the user's public key, such that only that user can decrypt it using its secret key.

Encryption by Sender: The encryption function is $ABE.Encrypt(MSG, \chi)$. Sender decides about the access tree χ. LSSS matrix R. sender encrypts message MSG as follows:

- Choose a random seed and a random vector, with s as its first entry; h is the number of leaves in the access tree.
- Calculate $\lambda_x = R_x.v$.
- Choose a random vector with) as the first entry.
- Calculate$\omega_x = R_x.\omega$.
- For each row R_x of R, choose a random.
- The ciphertext is calculated with different parameters.
- The ciphertext C is sent by the sender.

Decryption by Receiver: The decryption function is $ABE.Decrypt(C, \{sk_{i,u}\})$. The receiver takes as input ciphertext, secret keys, group, and outputs message. It obtains the access matrix and mapping. It then executes the following steps:

- User calculates the set of attributes that are common to itself and the access matrix.
- For each of these attributes, it checks if there is a subset of rows of matrix, such that the vector (1,0, . . . ,0) is their linear combination. If not, decryption is not possible. If yes, it calculates constants.
- Decryption proceeds as follows:
  - Decrypt the each attribute.
  - User computes original message.
- Attribute-Based signature Scheme
  - ABS scheme has the following steps.

System Initialization: Select a prime q, and groups G_1 and G_2, which are of order q. we define the mapping $\hat{e} = G_1 \times G_1 \rightarrow G_2$. Let g_1, g_2 be generators of G_1 and h_j be generators of G_2, for $j \in [t_{max}]$, for arbitrary t_{max}. Let H be hash function. (T Sig, T Ver) mean T Sig is the public key used for verification. The secret ket fro the trustee is TSK = (a_0, T Sig) and public key is T PK.

User Registration: For a user with identity the KDC draws at random. It generates the token output.

KDC Setup: Chooses random integers and attributes. And compute the private key ASK and public key APK.

Attribute Generation: The token verification algorithm verifies the signature contained in token using the signature verification key T Ver in T PK. This algorithm extracts K_{base} from token using attributes from ASK [i] and computes key. The key can be checked for consistency using algorithm KeyCheck function.

Sign: This algorithm has input the public key of the trustee, the secret key the signer, the message to be signed and policy claim. The policy claim is first converted into the span program, with rows labeled with attributes. Then the signature is calculated σ.

Verify: This algorithm converts y to the corresponding monotone program with rows labeled with attributes. computes μ. If Y = 1, ABS. Verify = 0 meaning false. Otherwise, the other constraints are checked.

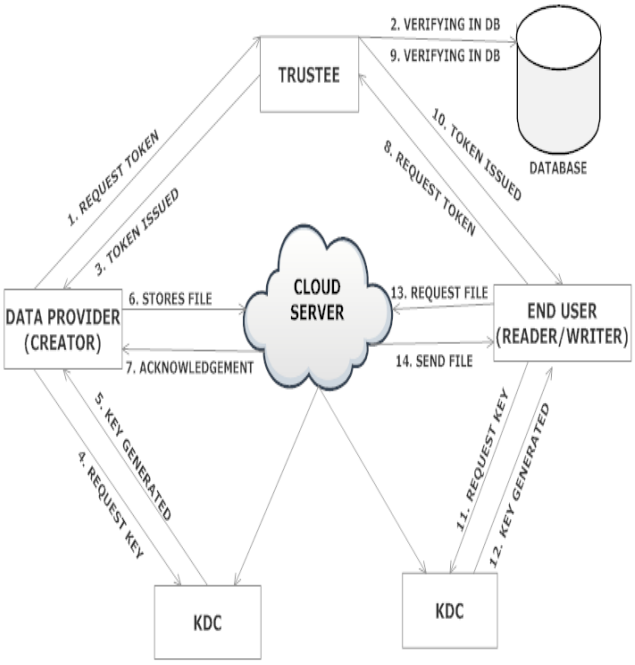

Fig.1. Proposed System Overview.

III. PROPOSED WORK

In this section, we propose our privacy preserving authenticated access control scheme. According to our scheme a user can create a file and store it securely in the cloud. This scheme consists of use of the two protocols ABE and ABS, as discussed in above sections. We refer to the Fig. 1. There are three users, a creator, a reader, and writer. Creator Alice receives a token from the trustee, who is assumed to be honest. A trustee can be someone like the federal government who manages social insurance numbers etc. On presenting her id (like health/social insurance number), the trustee gives her a token. There are multiple KDCs (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. In the Fig. 1, SKs are secret keys given for decryption, K_x are keys for signing. The message MSG is encrypted under the access policy X. The access policy decides who can access the data stored in the cloud. The creator decides on a claim policy Y, to prove her authenticity and signs the message under this claim. The ciphertext C with signature is c, and is sent to the cloud. The cloud verifies the signature and stores the ciphertext C. When a reader wants to read, the cloud sends C. If the user has attributes matching with access policy, it can decrypt and get back original message. Write proceeds in the same way as file creation. By designating the verification process to the cloud, it relieves the individual users from time consuming verifications. When a reader wants to read some data stored in the cloud, it tries to decrypt it using the secret keys it receives from the KDCs. If it has enough attributes matching with the access policy, then it decrypts the information stored in the cloud.

A. Data Storage in Clouds

A user first registers itself with one or more trustees. For simplicity we assume there is one trustee. The trustee gives it a token. The KDCs are given keys such as PK and SK for encryption/ decryption and ASK, APK for signing/verifying. The user on presenting this token obtains attributes and secret keys from one or more KDCs. A key for an attributes belonging to KDCs the user also receives secrete keys for encrypting messages. The user the creates an access policy which is a monotone Boolean function. The message is then encrypted under the access policy. The user also constructs a claim policy to enable the cloud to authenticate the user. The creator does not send the message, but uses the time stamp and creates. This is done to prevent reply attacks. If the time stamp is not sent, then the user can write previous stale message back to the cloud with a valid signature, even when its claim policy and attributes have been revoked. In their scheme, a writer can send its message and correct signature even when it no longer has access rights. In our scheme a writer whose rights have been revoked cannot create a new signature with new time stamp and, thus cannot write back stale information. It then signs the message and calculates the message signature. Then the cloud on receiving the information verifies the access claim using the algorithm. The creator checks claim using algorithm. The creator checks the value of V. If V = 0, then authentication has failed and the message is discarded. Else the message is stored in the cloud.

B. Reading from The Cloud

When a user requests data from the cloud, the cloud sends the ciphertext using SSH protocol. Decryption proceeds using algorithm ABE. Decrypt and the message is calculated.

C. Writing to the Cloud

To write to an already existing file, the user must sends its message with the claim policy as done during file creation. The cloud verifies the claim policy, and only if the user is authentic, is allowed to write on the file.

D. User Revocation

We will now discuss how to handle user revocation. It should be ensured that users must not have the ability to access data, even if they possess matching set of attributes. For this reason, the owners should change the stored data and send updated information to other users. Once the attributes are identified, all data that possess the attributes are collected. For each such data record, the following steps are then carried out:

- A new value is selected from the integer set.
- The first entry of vector is changed to new value.
- λ is calculated for each row corresponding to leaf attributes.
- Ciphertext is recalculated for row.
- New value of ciphertext is securely transmitted to the cloud.
- New ciphertext is calculated and stored in the cloud.
- New value of cipher is stored with the data, but is transmitted to the users, who wish to decrypt the data.

IV. 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.

V. 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. 556- 563, 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. 136- 149, 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.

Author's Profile:

Mohammad Mohasin has received her B.Tech in Computer Science and Engineering from Narayana Engineering College, Nellore affiliated to JNTU, Anantapur in 2013 and pursuing M.Tech degree in Computer Science in Narayana Engineering College (NEC), Nellore, A.P affiliated to JNTU, Anatapur in (2013-2015).

Akkana Vamsi Krishna has received his B.Tech degree in Computer Science and Engineering from Saraswathi Velu College of Engineering, affiliated to Anna University in 2011 and M.Tech degree in Computer Science & Engineering from Nova Institute of Technology, affiliated to JNTUK in 2013. He is dedicated to teaching field from last 2+ Years. He has guided 2 P.G and 10 U.G students. His research area included Computer Networks. At present he is working as Assistant Professor in Narayana Engineering College, Nellore, Andhra Pradesh, India.