Privacy Preserving Access Control with Authentication for Securing Data in Clouds

Sushmita Ruj*, Milos Stojmenovic†, Amiya Nayak*
*SEECS, University of Ottawa, Canada – {sruj,anayak}@site.uottawa.ca
†Singidunum University, Belgrade, Serbia – mstojmenovic@singidunum.ac.rs

***Abstract*—In this paper, we propose a new privacy preserving authenticated access control scheme for securing data in clouds. In the proposed scheme, the cloud verifies the authenticity of the user without knowing the user's identity before storing information. Our 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. 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: Access control, Authentication, Attribute-based signatures, Attribute-based encryption, Cloud storage.

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. This frees users from the hassles of maintaining resources on-site. 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 (e.g., Amazon's S3, Windows Azure).

Since services are outsourced to a remote server, security and privacy are of immense concern 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.

Efficient search 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 [1], [2]. 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. Authentication of users using public key cryptographic techniques has been studied in [3]. Many homomorphic encryption techniques have been suggested [4], [5] to ensure that the cloud is not able to read the data while performing computations on them. Using homomorphic encryption, the cloud receives ciphertext of the data and performs computations on the ciphertext and returns the encoded value of the result. The user is able to decode the result, but the cloud does not know what data it has operated on. In such circumstances, it must be possible for the user to verify that the cloud returns correct results. The other closely related problem is that of code obfuscation [6], in which the user sends the code in an encoded form, and the cloud can execute the code and returns the result without knowing the actual code.

Accountability of clouds is a very challenging task and involves technical issues and law enforcement. Neither clouds nor users should deny any operations performed or requested. It is important to have log of the transactions performed; however, it is an important concern to decide how much of information to keep in the log. Accountability has been addressed in TrustCloud [7].

Let us now consider the following situation. Suppose a Law student, Alice, wants to send a series of reports about some malpractices by authorities of University X to all the professors of University X, Research chairs of universities in the country, and students belonging to Law department in all universities in the province. She wants to remain anonymous while publishing all evidence of malpractice. She stores the information in the cloud. Access control is important in such case, so that only authorized users can access the data. It is also important to verify that the information comes from a reliable source. The problems of access control, authentication, and privacy protection should be solved simultaneously. We address this problem in entirety in this paper.

Access control in clouds is gaining attention because it is important that only authorized users have access to valid service. A huge amount of information is being stored in the cloud, and much of this is sensitive information. Care should be taken to ensure access control of this sensitive information which can often be related to health, important documents (as in Google Docs or Dropbox) or even personal information (as in social networking). There are broadly three types of access control: *User Based Access Control* (UBAC), *Role Based Access Control* (RBAC), and *Attribute Based Access Control* (ABAC). In UBAC, the access control list (ACL) contains the list of users who are authorized to access data. This is not feasible in clouds where there are many users. In RBAC (introduced by [8]), users are classified based on their individual roles. Data can be accessed by users who have matching roles. The roles are defined by the system. For example, only faculty members and senior secretaries might have access to data but not the junior secretaries. The ABAC is more extended in scope, in which users are given attributes, and the data has

978-0-7695-4691-9/12 $26.00
DOI 10.1109/CCGrid.2012.92

attached access policy. Only users with valid set of attributes, satisfying the access policy, can access the data. For instance, in the above example certain records might be accessible by faculty members with more than 10 years of research experience or by senior secretaries with more than 8 years experience. The pros and cons of RBAC and ABAC are discussed in [9]. There has been some work on ABAC in clouds (for example, [10], [11], [12], [13], [14]). All these work use a cryptographic primitive known as Attribute Based Encryption (ABE).

An area where access control is widely being used is health care. Clouds are being used to store sensitive information about patients to enable access to medical professionals, hospital staff, researchers, and policy makers. It is important to control the access of data so that only authorized users can access the data. Using ABE, the records are encrypted under some access policy and stored in the cloud. Users are given sets of attributes and corresponding keys. Only when the users have matching set of attributes, can they decrypt the information stored in the cloud. Access control in health care has been studied in [10], [11].

Access control is also gaining importance in online social networking where users (members) store their personal information, pictures, videos and share them with selected groups of users or communities they belong to. Access control in online social networking has been studied in [15]. Such data are being stored in clouds. It is very important that only the authorized users are given access to those information. A similar situation arises when data is stored in clouds, for example in Dropbox, and shared with certain groups of people.

It is just not enough to store the contents securely in the cloud but it might also be necessary to ensure anonymity of the user. For example, a user would like to store some sensitive information but does not want to be recognized. However, the user should be able to prove to the other users that he/she is a valid user who stored the information without revealing the identity. There are cryptographic protocols like ring signatures [16], mesh signatures [17], group signatures [18], which can be used in these situations. Ring signature is not a feasible option for clouds where there are a large number of users. Group signatures assume the pre-existence of a group which might not be possible in clouds. Mesh signatures do not ensure if the message is from a single user or many users colluding together. For these reasons, a new protocol known as Attribute Based Signature (ABS) has been applied. ABS was proposed by Maji *et al.* [19]. In ABS, users have a claim predicate associated with a message. The claim predicate helps to identify the user as an authorized one, without revealing its identity. Other users or the cloud can verify the user and the validity of the message stored. ABS can be combined with ABE to achieve authenticated access control without disclosing the identity of the user to the cloud.

Existing work [29], [10], [11], [12], [13], [14] on access control in cloud are centralized in nature. Except [29], all other schemes use attribute based encryption (ABE). The scheme in [29] uses a symmetric key approach and does not support authentication. The schemes [10], [11], [14] do not support authentication as well. Earlier work by Zhao *et al.* [13] 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. In an earlier work, Ruj *et al.* [14] proposed a distributed access control mechanism in clouds. However, the scheme did not provide user authentication. The other drawback was that an 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. In this paper, we extend their work with added features which enable to authenticate the validity of the message without revealing the identity of the user who has stored information in the cloud. We use attribute based signature scheme [20] to achieve authenticity and privacy. Unlike [20], our scheme is resistant to replay attacks, in which an 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 an 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 [20] appropriately. Our scheme also allows multiple writes which was not permitted by [14].

A. Our Contributions

The main contributions of this paper are the following:

1) Distributed access control of data stored in cloud so that only authorized users with valid attributes can access them.
2) Authentication of users who store and modify their information of the cloud.
3) The identity of the user is protected from the cloud during authentication.
4) The architecture is decentralized, meaning that there can be several KDCs for key management.
5) The access control and authentication are both collision resistant, meaning that no two users can collude and access data or authenticate themselves, if they are individually not authorized.
6) The proposed scheme is resilient to replay attacks. A writer whose attributes and keys have been revoked cannot write back stale information.
7) The protocol supports multiple read and write on the data stored in the cloud.
8) The costs are comparable to the existing centralized approaches, and the expensive operations are mostly done by the cloud.

B. Organization

The paper is organized as follows. Related work is presented in Section II. The mathematical background and assumptions are detailed in Section III. We present our privacy preserving access control scheme in Section IV followed by a real life example in Section V. The security is analyzed in Section VI. Computation complexity is discussed in Section VII, and comparison with other work is presented in Section VIII. We conclude in Section IX.

II. RELATED WORK

ABE was proposed by Sahai and Waters [21]. In ABE, an user has a set of attributes in addition to its unique ID. There are two classes of ABEs. In Key-policy ABE or KP-ABE (Goyal *et al*[22]), the sender has an access policy to encrypt data. A writer whose attributes and keys have been revoked cannot write back stale information. The receiver receives attributes and secret keys from the attribute authority and is able to decrypt information if it has matching attributes. In Ciphertext-policy, CP-ABE (Bethencourt et al. [23]), the receiver has the access policy in the form of a tree, with attributes as leaves and monotonic access structure with AND, OR and other threshold gates.

All the approaches take a centralized approach and allow only one KDC, which is a single point of failure. Chase [24] proposed a multi-authority ABE, in which there are several KDC authorities (coordinated by a trusted authority) which distribute attributes and secret keys to users. Chase and Chow [25] devised a multi-authority ABE protocol which required no trusted authority which requires every user to have attributes from at all the KDCs. Recently, Lewko and Waters [26] proposed a fully decentralized ABE where users could have zero or more attributes from each authority and did not require a trusted server.

All these approaches had no way to authenticate users. To ensure user authentication Attribute Based Signatures were introduced by Maji *et al.* [19]. This was also a centralized approach. A recent scheme by the same authors [20] takes a decentralized approach and provides authentication without disclosing the identity of the users. However, as mentioned earlier in the previous section it is prone to replay attack.

III. BACKGROUND

In this section, we present our cloud storage model and the assumptions we have made in the paper. Table I presents the notations used throughout the paper. We also describe mathematical background used in our proposed solution.

TABLE I
NOTATIONS

Symbols	Meanings
U_u	u-th User/Owner
$\mathcal{A}_j$	j-th KDC
$\mathbb{A}$	Set of KDCs
L_j	Set of attributes that KDC $\mathcal{A}_j$ possesses
$l_j = \|L_j\|$	Number of attributes that KDC $\mathcal{A}_j$ possesses
$I[j,u]$	Set of attributes that $\mathcal{A}_j$ gives to user U_u for encryption/decryption
I_u	Set of attributes that user U_u possesses
$J[j,u]$	Set of attributes that $\mathcal{A}_j$ gives to user U_u for claim attributes
J_u	Set of attributes that user U_u possesses as claim attributes
$AT[j]$	KDC which has attribute j
$PK[j]/SK[j]$	Public key/secret key of KDC $\mathcal{A}_j$ for encryption/decryption
$sk_{i,u}$	Secret key given by $\mathcal{A}_j$ corresponding to attribute i given to user U_u
TPK/PSK	Trustee public key/secret key
$APK[j]/ASK[j]$	Public key/secret key of KDC $\mathcal{A}_j$ for verifying claim
$\mathcal{X}$	Boolean access structure
$\mathcal{Y}$	Claim policy
τ	Time instant
R	Access matrix of dimension $m \times h$
M	Matrix of dimension $l \times t$ corresponding to the claim predicate
MSG	Message
$\|MSG\|$	Size of message MSG
C	Ciphertext
$H, \mathcal{H}$	Hash functions, example SHA-1
E_x	Exponentiation in group G_x
τ_H	Time to hash using function H
$\tau_{\mathcal{H}}$	Time to hash using function $\mathcal{H}$
$\tau_P/\tau_{\hat{P}}$	Time taken to perform 1 pairing operation in $e/\hat{e}$
$\|G\|$	Size of group G

A. Assumptions

We make the following assumptions in our work.

1) The cloud is honest and 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 in [10], [11].
2) Users can have either read or write or both accesses to a file stored in the cloud.
3) All communications between users/clouds are secured by Secure Shell Protocol, SSH.

B. Formats of 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 [26]. An example of a Boolean function is $((a_1 \wedge a_2 \wedge a_3) \vee (a_4 \wedge a_5)) \wedge (a_6 \vee a_7))$, where $a_1, a_2, \ldots, a_7$ are attributes. Boolean functions can also be represented by access tree, with attributes at the leaves and $AND(\wedge)$ and $OR(\vee)$ as the intermediate nodes and root. Boolean functions can be converted to LSSS matrix as presented in [14]. We do not present it here due to lack of space.

Let $\mathcal{Y} : \{0,1\}^n \rightarrow \{0,1\}$ be a monotone Boolean function [20]. A monotone span program for $\mathcal{Y}$ over a field $\mathbb{F}$ is an $l \times t$ matrix M with entries in $\mathbb{F}$, along with a labeling function $a : [l] \rightarrow [n]$ that associates each row of M with an input variable of , that, for every $(x_1, x_2 \ldots, x_n) \in \{0,1\}^n$, satisfies the following:

$$\mathcal{Y}(x_1, x_2, \ldots, x_n) = 1 \Leftrightarrow \exists v \in \mathbb{F}^{1\times l} : vM = [1, 0, 0, \ldots, 0]$$
$$\text{and } (\forall i : x_{a(i)} = 0 \Rightarrow v_i = 0)$$

In other words, $\mathcal{Y}(x_1, x_2, \ldots, x_n) = 1$ if and only if the rows of M indexed by $\{i | x_{a(i)} = 1\}$ span the vector $[1, 0, 0, \ldots, 0]$. Span programs can be constructed from Boolean functions in a similar way as shown later in Section V.

C. Mathematical background

We will use bilinear pairings on elliptic curves. Let G be a cyclic group of prime order q generated by g. Let G_T be a group of order q. We can define the map $e : G \times G \rightarrow G_T$. The map satisfies the following properties:

1) $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G$ and $a, b \in \mathbb{Z}_q$, $\mathbb{Z}_q = \{0, 1, 2, \ldots, q-1\}$.
2) Non-degenerate: $e(g, g) \neq 1$.

Bilinear pairing on elliptic curves groups are used. We do not discuss the pairing functions which mainly use Weil and Tate pairings [27] and computed using Miller's algorithm. The choice of curve is an important consideration because it determines the complexity of pairing operations.

PBC library (Pairing Based Cryptography) [27] is a C library which is built above GNU GMP (GNU Math Precision) library and contains functions to implement elliptic curves and pairing operations. The curves chosen are either MNT curves or supersingular curves. Considering the requirements, elliptic curve group of size 159, with an embedding degree 6 (type d curves of PBC [27]) can be used. Pairing takes 14 ms on Intel Pentium D, 3.0GHz CPU [14]. Such operations are very suitable for a cloud computing environment. A new library for attribute based encryption is also available at [28].

D. Attribute based encryption

Attribute based encryption with multiple authorities as proposed by Lewko and Waters [26] proceeds as follows [14]:

1) 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 $H : \{0,1\}^* \rightarrow G_0$ which maps the identities of users to G_0. The hash function used here is SHA-1. Each KDC $A_j \in \mathcal{A}$ has a set of attributes L_j. The attributes disjoint ($L_i \bigcap L_j = \phi$ for $i \neq j$). Each KDC also chooses two random exponents $\alpha_i, y_i \in \mathbb{Z}_q$. The secret key of KDC $\mathcal{A}_j$ is

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

The public key of KDC $\mathcal{A}_j$ is published:

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

2) Key generation and distribution by KDCs: User U_u receives a set of attributes $I[j,u]$ from KDC $\mathcal{A}_j$, and corresponding secret key $sk_{i,u}$ for each $i \in I[j,u]$

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

where $\alpha_i, y_i \in SK[j]$. 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.

3) Encryption by sender: The encryption function is $ABE.Encrypt(MSG, \mathcal{X})$. Sender decides about the access tree $\mathcal{X}$. LSSS matrix R can be derived as described in III-B. Sender encrypts message MSG as follows:

1) Choose a random seed $s \in \mathbb{Z}_q$ and a random vector $v \in \mathbb{Z}_q^h$, with s as its first entry; h is the number of leaves in the access tree (equal to the number of rows in the corresponding matrix R).
2) Calculate $\lambda_x = R_x \cdot v$, where R_x is a row of R.
3) Choose a random vector $w \in \mathbb{Z}_q^h$ with 0 as the first entry.
4) Calculate $\omega_x = R_x \cdot w$.
5) For each row R_x of R, choose a random $\rho_x \in \mathbb{Z}_q$.
6) The following parameters are calculated:

$$\begin{array}{l} C_0 = MSGe(g,g)^s \\ C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\pi(x)}\rho_x}, \forall x \\ C_{2,x} = g^{\rho_x} \forall x \\ C_{3,x} = g^{y_{\pi(x)}\rho_x} g^{\omega_x} \forall x, \end{array} \quad (4)$$

where $\pi(x)$ is mapping from R_x to the attribute i that is located at the corresponding leaf of the access tree.

7) The ciphertext C is sent by the sender (it also includes the access tree via R matrix):

$$C = \langle R, \pi, C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, \forall x\}\rangle. \quad (5)$$

4) Decryption by receiver: The decryption function is $ABE.Decrypt(C, \{sk_{i,u}\})$, where C is given by equation (13). Receiver U_u takes as input ciphertext C, secret keys $\{sk_{i,u}\}$, group G_0, and outputs message msg. It obtains the access matrix R and mapping π from C. It then executes the following steps:

1) U_u calculates the set of attributes $\{\pi(x) : x \in X\} \bigcap I_u$ that are common to itself and the access matrix. X is the set of rows of R.
2) For each of these attributes, it checks if there is a subset X' of rows of R, such that the vector $(1, 0 \ldots, 0)$ is their linear combination. If not, decryption is impossible. If yes, it calculates constants $c_x \in \mathbb{Z}_q$, such that $\sum_{x \in X'} c_x R_x = (1, 0, \ldots, 0)$.
3) Decryption proceeds as follows:
   a) For each $x \in X'$, $dec(x) = \frac{C_{1,x} e(H(u), C_{3,x})}{e(sk_{\pi(x),u}, C_{2,x})}$
   b) U_u computes $MSG = C_0 / \Pi_{x \in X'} dec(x)$.

E. Attribute based signature scheme

Attribute based signature scheme [20] has the following steps.

1) 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 $\mathcal{H}$ be a hash function. Let $A_0 = h_0^{a_0}$, where $a_0 \in \mathbb{Z}_q^*$ is chosen at random. $(TSig, TVer)$ mean $TSig$ is the private key with which a message is signed and $TVer$ is the public key used for verification. The secret key for the trustee is $TSK = (a_0, TSig)$ and public key is $TPK = (G_1, G_2, \mathcal{H}, g_1, A_0, h_0, h_1, \ldots, h_{t_{max}}, g_2, TVer)$.

2) User registration: For an user with identity U_u the KDC draws at random $K_{base} \in G$. Let $K_0 = K_{base}^{1/a_0}$. The following token γ is output

$$\gamma = (u, K_{base}, K_0, \rho), \quad (6)$$

where ρ is signature on $u || K_{base}$ using the signing key $TSig$.

3) KDC setup: Choose $a, b \in \mathbb{Z}_q^*$ randomly and compute: $A_{ij} = h_j^a$, $B_{ij} = h_j^b$, for $\mathcal{A}_i \in \mathbb{A}$, $j \in [t_{max}]$. The private key of i-th KDC is $ASK[i] = (a, b)$ and public key $APK[i] = (A_{ij}, B_{ij} | j \in [t_{max}])$.

4) Attribute generation: The token verification algorithm verifies the signature contained in γ using the signature verification key $TVer$ in TPK. This algorithm extracts K_{base} from γ using (a, b) from $ASK[i]$ and computes $K_x = K_{base}^{1/(a+bx)}$, $x \in J[i,u]$. The key K_x can be checked for consistency using algorithm $ABS.KeyCheck(TPK, APK[i], \gamma, K_x)$, which checks $\hat{e}(K_x, A_{ij} B_{ij}^x) = \hat{e}(K_{base}, h_j)$, for all $x \in J[i,u]$ and $j \in [t_{max}]$.

5) Sign: The algorithm $ABS.Sign(TPK, \{APK[i] : i \in AT[u]\}, \gamma, \{K_x : x \in J_u\}, MSG, \mathcal{Y})$, has input the public key of the trustee, the secret key of the signer, the message to be signed and the policy claim $\mathcal{Y}$. The policy claim is first converted into the span program $M \in \mathbb{Z}_q^{l \times t}$, with rows labeled with attributes. M_x denotes row x of M. Let π' denote the mapping from rows to the attributes. So, $\pi'(x)$ is the mapping from M_x to attribute x. A vector v is computed which satisfies the assignment $\{x : x \in J[i,u]\}$. Compute $\mu = \mathcal{H}(MSG||\mathcal{Y})$. Choose $r_0 \in \mathbb{Z}_q^*$ and $r_i \in \mathbb{Z}_q, i \in J_u$, and compute:

$$Y = K_{base}^{r_0}, S_i = (K_i^{v_i})^{r_0}.(g_2 g_1^{\mu})^{r_i} (\forall i \in J_u) \quad (7)$$

$$W = K_0^{r_0}, P_j = \Pi_{i \in AT[u]} (A_{ij} B_{ij}^{\pi'(i)})^{M_{ij} r_i} (\forall j \in [t]) \quad (8)$$

The signature is calculated as

$$\sigma = (Y, W, S_1, S_2, \ldots, S_t, P_1, P_2, \ldots, P_t). \tag{9}$$

6) Verify: Algorithm $ABS.Verify(TPK,$ $\sigma = (Y, W, S_1, S_2, \ldots, S_t, P_1, P_2, \ldots, P_t), MSG, \mathcal{Y})$, converts $\mathcal{Y}$ to the corresponding monotone program $M \in \mathbb{Z}_q^{l \times t}$, with rows labeled with attributes. Compute $\mu = \mathcal{H}(MSG||\mathcal{Y})$. If $Y = 1$, $ABS.Verify = 0$ meaning false. Otherwise, the following constraints are checked.

$$\hat{e}(W, A_0) \stackrel{?}{=} \hat{e}(Y, h_0) \tag{10}$$

$$\Pi_{i \in l} \hat{e}(S_i, A_{i'j} B_{i'j}^{\pi'(i)})^{M_{ij}}) \stackrel{?}{=} \begin{cases} \hat{e}(Y, h_1)\hat{e}(g_2 g_1^{\mu}, P_1), j = 1 \\ \hat{e}(g_2 g_1^{\mu}, P_j), j > 1, \end{cases} \tag{11}$$

where $i' = AT[i]$.

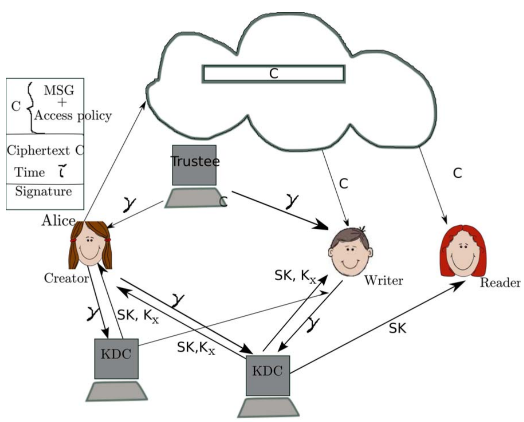

Fig. 1. Our secure cloud storage model

IV. Proposed privacy preserving authenticated access control scheme

In this section we propose our privacy preserving authenticated access control scheme. According to our scheme an 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 Section III-D and III-E respectively. We will first discuss our scheme in details and then provide a concrete example to demonstrate how it works. We refer to the Fig. 1. There are three users, a creator, a reader and writer. Creator Alice receives a token γ 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 γ. 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 $\mathcal{X}$. The access policy decides who can access the data stored in the cloud. The creator decides on a claim policy $\mathcal{Y}$, 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

An user U_u first registers itself with one or more trustees. For simplicity we assume there is one trustee. The trustee gives it a token $\gamma = (u, K_{base}, K_0, \rho)$, where ρ is the signature on $u||K_{base}$ signed with the trustees private key $TSig$ (By Equation 6). The KDCs are given keys $PK[i], SK[i]$ for encryption/decryption and $ASK[i], APK[i]$ for signing/verifying. The user on presenting this token obtains attributes and secret keys from one or more KDCs. A key for an attribute x belonging to KDC $\mathcal{A}_i$ is calculated as $K_x = K_{base}^{1/(a+bx)}$, where $(a, b) \in ASK[i]$. The user also receives secret keys $sk_{x,u}$ for encrypting messages. The user then creates an access policy $\mathcal{X}$ which is a monotone Boolean function. The message is then encrypted under the access policy as

$$C = ABE.Encrypt(MSG, \mathcal{X}) \tag{12}$$

The user also constructs a claim policy $\mathcal{Y}$ to enable the cloud to authenticate the user. The creator does not send the message MSG as is, but uses the time stamp τ and creates $\mathcal{H}(C)||\tau$. This is done to prevent replay 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. The original work by Maji *et al.* [20] suffers from replay attacks. 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 as

$$\sigma = ABS.Sign(\text{Public key of trustee, Public key of KDCs, token, signing key, message, access claim})$$

The following information is then sent in the cloud.

$$c = (C, \tau, \sigma, \mathcal{Y}). \tag{13}$$

The cloud on receiving the information verifies the access claim using the algorithm $ABS.verify$. The creator checks the value of $V = ABS.Verify(TPK, \sigma, c, \mathcal{Y})$. If $V = 0$, then authentication has failed and the message is discarded. Else, the message (C, τ) is stored in the cloud.

B. Reading from the cloud

When an user requests data from the cloud, the cloud sends the ciphertext C using SSH protocol. Decryption proceeds using algorithm $ABE.Decrypt(C, \{sk_{i,u}\})$ and the message MSG is calculated as given in Section III-D4.

C. Writing to the cloud

To write to an already existing file, the user must send 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.

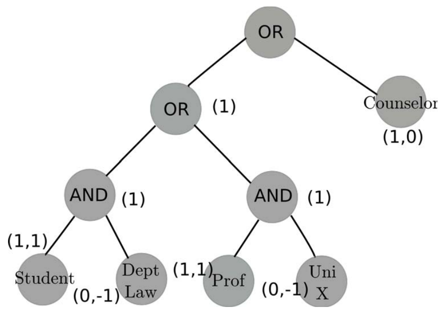

Fig. 2. Example of claim policy

V. Real life example

We now revisit the problem we stated in the introduction. We will use a relaxed setting. Suppose Alice is a Law student and wants to send a series of reports about malpractices by authorities of University X to all the professors of University X, Research chairs of universities X, Y, Z and students belonging to Law department in university X. She wants to remain anonymous, while publishing all evidence. All information is stored in the cloud. It is important that users should not be able to know her identity, but must trust that the information is from a valid source. For this reason she also sends a claim message which states that she "Is a law student" or "Is a student counselor" or "Professor at university X". The tree corresponding to the claim policy is shown in Figure 2.

The leaves of the tree consists of attributes and the intermediary nodes consists of Boolean operators. In this example the attributes are "Student", "Prof", "Dept Law", "Uni X", "Counselor". The above claim policy can be written as a Boolean function of attributes as

((Student AND Dept Law) OR (Prof AND Uni X)) OR (Student Counselor).

Using the algorithm given in [14], the span program for this policy is

$$M = \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 1 & 1 \\ 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

An assignment $v = (v_1, v_2, v_3, v_4, v_5)$ satisfies this span program if $vM = (1, 0)$.

The cloud should verify that Alice indeed satisfies this claim. Since she is a Law student, $v = (1, 1, 0, 0, 0)$ and is a valid assignment. As a valid user she can then store all the encrypted records under the set of access policy that she has decided. The access policy in case of Alice is

((Prof AND Uni. X) OR
(Research Chair AND ((Uni X OR Uni Y) OR Uni Z)) OR
((Student AND Dept Law)AND Uni X)

Later when a valid user, say Bob wants to modify any of these reports he also attaches a set of claims which the cloud verifies. For example, Bob is a research chair and might send a claim "Research chair" or "Department head" which is then verified by the cloud. It then sends the encrypted data to the Bob. Since Bob is a valid user and has matching attributes, he can decrypt and get back the information.

If Bob wants to read the contents without modifying them, then there is no need to attach a claim. He will be able to decrypt only if he is a Professor in University X or a Research chair in one of the universities X, Y, Z or a student belonging to Department of Law in university X.

Here it is to be noted that the attributes can belong to several KDCs. For example the Professors belonging to university X have credentials given by the university X, and a Ph. D. degree from a University P, the student counselor might be a psychologist authorized by the Canadian Psychological Association and assigned an employee number by a university, the research chairs can be jointly appointed by the universities X, Y, Z and the government. The students can have credentials from the university and also a department.

Initially Alice goes to a trustee for example the Canadian health service and presents her a health insurance number or federal agency presents her a social insurance number. Either or both of these trustees can give her token(s) $\gamma = (u, K_{base}, K_0, \rho)$. With the token she approaches the KDCs in the university X and department D and obtains the secret keys for decryption and for keys K_x and K_y for signing the assess policy. She can also access the public keys $APK[i]$ of other KDCs. The entire process is carried on in the following way:

A. Data Storage in clouds

Let the data be denoted by MSG, $\mathcal{X}$ is the access policy-

((Prof AND Uni. X) OR
(Research Chair AND ((Uni X OR Uni Y) OR Uni Z)) OR
((Student AND Dept Law)AND Uni X)

Alice encrypts the data and obtains the ciphertext

$$C = Enc(MSG, \mathcal{X}).$$

Alice also decides on a claim policy $\mathcal{Y}$ which is shown in Figure 2. From the matrix, $v = (1, 1, 0, 0, 0)$ can be calculated. The values of $Y, W, S_1, S_2, S_3, S_4, S_5, P_1, P_2$ can be calculated. $\mu = \mathcal{H}(MSG||\mathcal{Y})$. The current time stamp τ is attached to the ciphertext to prevent replay attacks. The signature σ is calculated as $ABS.Sign$. The ciphertext

$$c = (C, \tau, \sigma, \mathcal{Y})$$

is then send to the cloud. The cloud verifies the signature using the function $ABS.Verify$ as given in Equation (11). If Alice has valid credentials then the ciphertext (C, τ) is stored, else it is discarded.

B. Reading from the cloud and modifying data

Suppose Bob wants to access the records stored by Alice. Bob then decrypts the message MSG using his secret keys using function $ABE.Decrypt$. Writing proceeds like file creation. It is to be noted that the time τ is added to the data so that even if Bob's credentials are revoked, he cannot write stale data in the cloud.

VI. Security of the protocol

In this section we will prove the security of the protocol. We will show that our scheme authenticates an user who wants to write to the cloud. An user can only write provided the cloud is able to validate its access claim. An invalid user cannot receive attributes from a KDC, if it does not have the credentials from the trustee. If an user's credentials are revoked, then it cannot replace data with previous stale data, thus preventing replay attacks.

Theorem 1: Our authentication scheme is correct, collusion secure, resistant to replay attacks, and protects privacy of the user.

Proof: We first note that only valid users registered with the trustee(s) receive attributes and keys from the KDCs. An user's token is $\gamma = (u, K_{base}, K_0, \rho)$, where ρ is signature on $u||K_{base}$ with $TSig$ belonging to the trustee. An invalid user with a different user-id cannot create the same signature because it does not know $TSig$.

We next show that only a valid user with valid access claim is only able to store the message in the cloud. This follows from the functions $ABS.Sign$ and $ABS.Verify$ given in Section III-E. An user who wants to create a file and tries to make a false access claim, cannot do so, because it will not have attribute keys K_x from the related KDCs. At the same time since the message is encrypted, an user without valid access policy cannot decrypt and change the information.

Two users cannot collude and create an access policy consisting of attributes shared between them. Suppose, there are two users A and B who have attributes x_A and x_B respectively. They have the following information K_{base_A}, K_{x_A} and K_{base_B}, K_{x_B}, respectively. A new value of $K_{x_B} = K_{base_A}^{1/(a+bx')}$ cannot be calculated by B, because it does not know the values of (a, b). Thus the authentication is collusion secure.

Our scheme is resistant to replay attacks. If a writer's access claims are revoked, it cannot replace a data with stale information from previous writes. This is because it has to attach a new time stamp τ and sign the message $\mathcal{H}(C)||\tau$ again. Since it does not have attributes, it cannot have a valid signature.

■

VII. Computation Complexity

In this section we present the computation complexity of the privacy preserving access control protocol. We will calculate the computations required by users (creator, reader, writer) and that by the cloud.

The creator needs to encrypt the message and sign it. Creator needs to calculate one pairing $e(g, g)$. Encryption takes 2 exponentiations to calculate each of $C_{1,x}$. So this requires $2mE_T$ time, where m is the number of attributes. User needs to calculate 3 exponentiation to calculate $C_{2,x}$ and $C_{3,x}$. So time taken for encryption is $(3m + 1)E_0 + 2mE_T + \tau_P$. To sign the message, $Y, W, S_i's$ and P_js have to be calculated as well as $\mathcal{H}(C)$. So, time taken to sign is $(2l + 2)E_1 + 2tE_2 + \tau_{\mathcal{H}}$.

The cloud needs to verify the signature. This requires checking for equation (11). Time taken to verify is $(l + 2t)\tau_{\hat{P}} + l(E_1 + E_2) + \tau_{\mathcal{H}}$. To read, an user needs only to decrypt the ciphertext. This requires $2m$ pairings to calculate $e(H(u), C_{3,x})$ and $e(sk_{\pi(x),u}, C_{2,x})$ and $O(mh)$ to find the vector c. Decryption takes $2m\tau_P + \tau_H + O(mh)$. Writing is similar to creating a record. The size of ciphertext with signature is $2m|G_0| + m|G_T| + m^2 + |MSG| + (l + t + 2)|G_1|$.

We will compare our computation costs with existing schemes like [11], [10], [13] in next Section VIII.

VIII. Comparison with other access control schemes in cloud

We compare our scheme with other access control schemes (in Table II) and show that our scheme supports many features that the other schemes did not support. 1-W-M-R means that only one user can write while many users can read. M-W-M-R means that many users can write and read. We see that most schemes do not support many writes which is supported by our scheme. Our scheme is robust and decentralized, most of the others are centralized. Our scheme also supports privacy preserving authentication, which is not supported by others. In Tables III and IV we compare the computation and communication costs incurred by the users and clouds and show that our distributed approach has comparable costs to centralized approaches. The most expensive operations involving pairings and is done by the cloud. If we compare the computation load of user during read we see that our scheme has comparable costs. Our scheme also compares well with the other authenticated scheme of [13].

IX. Conclusion

We present a privacy preserving access control scheme for clouds. Our scheme not only provides fine-grained access control but also authenticates users who store information in the cloud. The cloud however does not know the identity of the user who stores information, but only verify 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 protect the privacy of user attributes as well.

Acknowledgement

This work is partially supported by NSERC Grant CRDPJ386874-09 and This work was partially supported by the following grant: "Digital signal processing, and the synthesis of an information security system", TR32054, Serbian Ministry of Science and Education.

References

[1] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *IEEE INFOCOM.* 2010, pp. 441–445.

[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography Workshops*, ser. Lecture Notes in Computer Science, vol. 6054. Springer, 2010, pp. 136–149.

[3] H. Li, Y. Dai, L. Tian, and H. Yang, "Identity-based authentication for cloud computing," in *CloudCom*, ser. Lecture Notes in Computer Science, vol. 5931. Springer, 2009, pp. 157–166.

[4] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009, http://www.crypto.stanford.edu/craig.

[5] A.-R. Sadeghi, T. Schneider, and M. Winandy, "Token-based cloud computing," in *TRUST*, ser. Lecture Notes in Computer Science, vol. 6101. Springer, 2010, pp. 417–429.

TABLE II
COMPARISON OF OUR SCHEME WITH EXISTING ACCESS CONTROL SCHEMES

Schemes	Fine-grained access control	Centralized/ Decentralized	Write/read access	Type of access control	Privacy preserving authentication
[29]	Yes	Centralized	1-W-M-R	Symmetric key cryptography	No authentication
[10]	Yes	Centralized	1-W-M-R	ABE	No authentication
[11]	Yes	Centralized	1-W-M-R	ABE	No authentication
[14]	Yes	Decentralized	1-W-M-R	ABE	No authentication
[13]	Yes	Centralized	M-W-M-R	ABE	Authentication
Ours	Yes	Decentralized	M-W-M-R	ABE	Authentication

TABLE III
COMPARISON OF COMPUTATION AND SIZE OF CIPHERTEXT WHILE CREATING A FILE

Schemes	Computation by creator	Computation by cloud	Size of ciphertext
[10]	$(m+2)E_0$	0	$m \log \lvert G_0 \rvert + \lvert G_T \rvert + m \log m + \lvert MSG \rvert$
[11]	$(m+2)E_0$	0	$m \log \lvert G_0 \rvert + \lvert G_1 \rvert + \lvert MSG \rvert$
[14]	$(3m+1)E_0 + 2mE_T + \tau_P$ (encrypt)	0	$2m \lvert G_0 \rvert + m \lvert G_T \rvert + m^2 + \lvert MSG \rvert$
[13]	$E_1 + (2m+1)E_0 + m\tau_H$ (encrypt) $(2l+2)E_1 + 2tE_2 + \tau_{\mathcal{H}}$ (sign)	$(l+2t)\tau_{\hat{P}} + l(E_1+E_2) + \tau_{\mathcal{H}}$ (verify)	$\lvert G_2 \rvert + (2m+1)\lvert G_1 \rvert + \lvert MSG \rvert$ $+(l+t+2)\lvert G_1 \rvert + m^2$
Our approach	$(3m+1)E_0 + 2mE_T + \tau_P$ (encrypt) $(2l+2)E_1 + 2tE_2 + \tau_{\mathcal{H}}$ (sign)	$(l+2t)\tau_{\hat{P}} + l(E_1+E_2) + \tau_{\mathcal{H}}$ (verify)	$2m \lvert G_0 \rvert + m \lvert G_T \rvert + m^2 + \lvert MSG \rvert$ $+(l+t+2)\lvert G_1 \rvert$

TABLE IV
COMPARISON OF COMPUTATION DURING READ AND WRITE BY USER AND CLOUD

Schemes	Computation by user while write	Computation by user while read	Computation by cloud while write
[10]	No write access	$m\tau_P$	No write access
[11]	No write access	$m\tau_P$	No write access
[14]	No write access	$2m\tau_P + \tau_H + O(mh)$	No write access
[13]	$E_1 + (2m+1)E_0 + m\tau_H$ (encrypt) $(2l+2)E_1 + 2tE_2 + \tau_{\mathcal{H}}$ (sign)	$(2m+1)\tau_P$ (decrypt)	$(l+2t)\tau_{\hat{P}} + l(E_1+E_2) + \tau_{\mathcal{H}}$ (verify)
Our approach	$(3m+1)E_0 + 2mE_T + \tau_P$ (encrypt) $(2l+2)E_1 + 2tE_2 + \tau_{\mathcal{H}}$ (sign)	$2m\tau_P + \tau_H + O(mh)$ (decrypt)	$(l+2t)\tau_{\hat{P}} + l(E_1+E_2) + \tau_{\mathcal{H}}$ (verify)

[6] G. Wroblewski, "General method of program code obfuscation," Ph.D. dissertation, Wroclaw University of Technology, 2002, http://www.ouah.org/wobfuscation.pdf.

[7] 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. Available at http://www.hpl.hp.com/techreports/2011/HPL-2011-38.html.

[8] D. F. Ferraiolo and D. R. Kuhn, "Role-based access controls," in *15th National Computer Security Conference*, 1992.

[9] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding attributes to role-based access control," *IEEE Computer*, vol. 43, no. 6, pp. 79–81, 2010.

[10] 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," in *SecureComm*, 2010, pp. 89–106.

[11] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *ACM ASIACCS*, 2010, pp. 261–270.

[12] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *ACM CCS*, 2010, pp. 735–737.

[13] F. Zhao, T. Nishide, and K. Sakurai, "Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems," in *ISPEC*, ser. Lecture Notes in Computer Science, vol. 6672. Springer, 2011, pp. 83–97.

[14] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: Distributed access control in clouds," in *IEEE TrustCom*, 2011.

[15] S. Jahid, P. Mittal, and N. Borisov, "EASiER: Encryption-based access control in social networks with efficient revocation," in *ACM ASIACCS*, 2011.

[16] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 2248. Springer, 2001, pp. 552–565.

[17] X. Boyen, "Mesh signatures," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 4515. Springer, 2007, pp. 210–227.

[18] D. Chaum and E. van Heyst, "Group signatures," in *EUROCRYPT*, 1991, pp. 257–265.

[19] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures: Achieving attribute-privacy and collusion-resistance," *IACR Cryptology ePrint Archive*, 2008.

[20] ——, "Attribute-based signatures," in *CT-RSA*, ser. Lecture Notes in Computer Science, vol. 6558. Springer, 2011, pp. 376–392.

[21] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 3494. Springer, 2005, pp. 457–473.

[22] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM Conference on Computer and Communications Security*, 2006, pp. 89–98.

[23] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy*. 2007, pp. 321–334.

[24] M. Chase, "Multi-authority attribute based encryption," in *TCC*, ser. Lecture Notes in Computer Science, vol. 4392. Springer, 2007, pp. 515–534.

[25] M. Chase and S. S. M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *ACM Conference on Computer and Communications Security*, 2009, pp. 121–130.

[26] A. B. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 6632. Springer, 2011, pp. 568–588.

[27] "http://crypto.stanford.edu/pbc/."

[28] "libfenc: The functional encryption library. http://code.google.com/p/libfenc/."

[29] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *ACM Cloud Computing Security Workshop (CCSW)*, 2009.