Ciphertext-Policy Attribute-Based Threshold Decryption with Flexible Delegation and Revocation of User Attributes

Luan Ibraimi[1,2], Milan Petkovic[2], Svetla Nikova[1], Pieter Hartel[1], Willem Jonker[1,2]

[1] Faculty of EEMCS, University of Twente, the Netherlands
[2] Philips Research, the Netherlands

Abstract. In Ciphertext-Policy Attribute-Based Encryption (CP-ABE), a user secret key is associated with a set of attributes, and the ciphertext is associated with an access structure or decryption policy over attributes. The user can decrypt the ciphertext if and only if the attribute set of his secret key satisfies the decryption policy specified in the ciphertext. Several CP-ABE schemes have been proposed, however, to become practical the problem of revocation and delegation should be addressed. In this paper, we propose Ciphertext-Policy Attribute-Based Threshold Decryption (CP-ABTD) which extends CP-ABE with flexible attribute delegation and instantaneous attribute revocation. CP-ABTD has three advantages over CP-ABE. First, Alice (delegator), who has a secret key associated with a set of attributes, can delegate her authorization to Bob (delegatee). Second, Alice can decide whether to allow Bob to be able to delegate her authorization further. Third, the proposed scheme achieves instantaneous attribute revocation, that is, once the attribute is revoked the user cannot use it in the decryption phase. We demonstrate how to apply the proposed CP-ABTD scheme to securely manage Personal Health Records (PHRs).

1 Introduction

Modern distributed information systems require flexible access control models which go beyond discretionary, mandatory and role-based access control. Recently proposed models, such as attribute-based access control, define access control policies based on different attributes of the requester, environment, or the data object. On the other hand, the current trend of service-based information systems and storage outsourcing require increased protection of data including access control methods that are cryptographically enforced. The concept of Attribute-Based Encryption(ABE) fulfills the aforementioned requirements. It provides an elegant way of encrypting data such that the encryptor defines the attribute set that the decryptor needs to posses in order to decrypt the ciphertext. Since Sahai and Waters [1] proposed the basic ABE scheme, a number of more advanced schemes have been developed, such as most notably Ciphertext-Policy ABE schemes (CP-ABE) [2,3]. In these schemes, a ciphertext is associated with an access structure and the user secret key is associated with a set of attributes. A secret key holder can decrypt the ciphertext if the attributes associated with his secret key satisfy the access structure associated with the ciphertext. For example, consider a situation when two organizations, a Hospital and a University, conduct research in the field of neurological disorders. The Hospital wants to allow access to their research results to all staff from the University who have the role Professor and belong to the Department of Neurology (DN). To enforce the policy, the Hospital encrypts the data according to the access structure $\tau_{Results}$=(University Professor $\wedge$ Member of DN). To decrypt the ciphertext the user must have a secret key associated with a set of attributes ω=(University Professor, Member of DN) to satisfy the access structure $\tau_{Results}$.

The state-of-the-art CP-ABE schemes provide limited support for authorization delegation and revocation of attributes, two features, which are becoming increasingly important in modern access control systems. In particular, delegation and attribute revocation are important requirements in the domain of access control to health data, which is our application field for attribute-based encryption. Therefore, in a CP-ABE scheme, it will be necessary for the user (delegator) holding a secret key associated with a number of attributes, to generate for another user (delegatee) a new secret key associated with more restrictive number of attributes. Suppose Bob, who has a secret key sk_{Bob} associated with an attribute set ω_{Bob} =(Head of DN, University Professor, Member of DN), needs to delegate the authority to his PHD student (Charlie) to read research results at the time when Bob is away from the University. Using his secret key sk_{Bob}, Bob could simply generate a new secret key for Charlie $sk_{Charlie}$ associated with the attribute $\omega_{Charlie}$ =(University Professor, Member of DN). A straightforward approach to achieve this functionality would be that Bob gives his key sk_{Bob} to Charlie. However, this approach has the following disadvantages: a) Bob might want to delegate only a subset of his attributes to Charlie, i.e. Bob does not want to delegate the attribute Head of the DN to Charlie; b) Bob might want to prevent two or more delegatee's to collude. Suppose Bob delegates to Charlie the attributes (University Professor, Member of DN) and to Alice the attribute (Head of DN). By generating two different keys $sk_{Charlie}$ and sk_{Alice}, Bob wants to prevent Charlie and Alice to combine their attributes and extend their delegated authorizations. The second feature, attribute revocation is of the utmost importance for any attribute-based access control system, because the management of attributes is dynamic. Following the above mentioned example, suppose Charlie has defended his PhD thesis and starts to work for a pharmaceutical company. Ideally, Charlie's attributes should be revoked immediately, so that he cannot obtain new research results and pass them to the pharmaceutical company. In general, attribute revocation may happen for the following reasons: 1) an attribute is not valid because it has expired, or 2) change of affiliation - a list of attributes identifying the user has been updated.

Contribution. In this paper we study attribute delegation and attribute revocation in CP-ABE. Previous CP-ABE systems can either support only uncontrolled delegation [2] (the delegator cannot prevent the delegatee to delegate further his

authority), or use a system where attributes are valid within a specific time frame [4] (there is no way to revoke an attribute before the expiration date). We propose a new scheme called Ciphertext-Policy Attribute-Based Threshold Decryption (CP-ABTD), where the power to decrypt is divided between the user and an online semi-trusted mediator. Our scheme has two important features: instant attribute revocation and flexible (controlled) attribute delegation.
Attribute revocation is achieved immediately. If a user attribute is revoked, the user cannot use the secret key associated with the revoked attribute. The idea is that the user does not hold the full secret key. The secret key is divided into two shares, one share for the mediator and the other for the user. To decrypt the data, the user must contact the mediator to receive a decryption token. The mediator keeps an attribute revocation list (ARL) and refuses to issue the decryption token for revoked attributes. Without the token, the user cannot decrypt the ciphertext, therefore the attribute is implicitly revoked.
The delegation is flexible in the sense that a delegator can decide whether the delegatee can delegate the delegator's attributes further. The mediator keeps an attribute delegation list (ADL) which has information about users who are allowed to delegate their attributes further. The intuition behind this is that when the delegator computes the delegatees secret key, the mediator has to compute the share of the delegatee's key as well. When the delegation is allowed, the delegator can delegate to the delegatee all or subset of her attributes, and the mediator will compute the share of the delegatee's key as well. When the delegation is not allowed, the delegator cannot delegate any attribute to the delegatee since the mediator will refuse to compute the share of the delegatee's key.
We define a security model for the scheme which formalizes the security attacks and provide a security proof under generic group model. Finally, we demonstrate applicability of the proposed CP-ABTD scheme to the management of Personal Health Records (PHRs).

1.1 Related Work

Mediated Cryptography. Boneh et al.[5,6] introduce a method for fast revocation of public key certificates and security capabilities in a RSA cryptosystem called mediated RSA (mRSA). The method uses an online semi-trusted mediator (SEM) which has a share of each users secret key, while users have the remaining share of the secret key. To decrypt or sign a message, a user must first contact the SEM and receive a message-specific token. Without the token, the user cannot decrypt or sign a message. Instantaneous user revocation is obtained by instructing the SEM to stop issuing tokens for future decrypt/sign requests. Thus, in mediated cryptography the Trusted Authority (TA) responsible to generate user key pair, does not deliver the full decryption key to users, but it delivers only a share of it. This method achieves faster revocation of user's security capabilities compared to previous certification techniques such as Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP). Libert and Quisquater [7] show that the architecture for revoking security capabilities can be applied

to several existing public key encryption schemes including the Boneh-Franklin scheme, and several signature schemes including the GDH scheme. Nali et al. [8] present a mediated hierarchical identity-based encryption and signature scheme. The hierarchical nature of the schemes and the instant revocation capability offered by the SEM architecture allow to enforce cryptographically access control in hierarchically structured communities of users whose access privileges change dynamically. Nali et al. [9] also show how to extend the Libert and Quisquater mediated identity-based cryptographic scheme to allow the enforcement of role-based access control (RBAC).
Attribute-Based Encryption. Sahai and Waters in their seminal paper [1] introduce the concept of Attribute-Based Encryption(ABE). There are two types of ABE schemes: Key-Policy ABE schemes (KP-ABE) [10] and Ciphertext Policy ABE schemes (CP-ABE) [2,3]. In KP-ABE, a ciphertext is associated with a set of attributes and a user secret key is associated with an access structure. A secret key holder can decrypt the ciphertext if the attributes associated with the ciphertext satisfy the access structure associated with the secret key. A related work to KP-ABE is a technique of searching on encrypted data [11,12,13,14]. In CP-ABE the idea is reversed. A ciphertext is associated with the access structure and the user secret key is associated with a set of attributes. A secret key holder can decrypt the ciphertext if the attributes associated with the secret key satisfy the access structure associated with the ciphertext.
Delegation. Hierarchy Identity-Based Encryption (HIBE) [15,16] schemes cover the notion of delegation. In HIBE, a holder of a secret key which is generated from the holder identity can create and delegate a new secret key for which his identity is a prefix. For example, a user with a secret key for the identity Department:Education can delegate to Alice a secret key for Department:Education.Alice. Shi and Waters [17] propose a scheme which allows users to delegate their capabilities in predicate encryption systems. Bethencourt et al. [2] show how to extend CP-ABE scheme to support delegation. The disadvantage of their scheme is that once the user (delegator) delegates her attributes to another user (delegatee), there is no mechanism which prevents the delegatee to delegate further their attributes, which is required in some scenarios.
Revocation. Credential revocation is a critical issue for access control systems. For ABE systems, Pirretti et al. [4] propose to use user attributes for a limited time period. After a specific time frame the attribute would become invalid. However, this approach would require the list of attributes to be updated regularly, which might be not a desired solution for the trusted authority.

1.2 Organization

The rest of this paper is organized as follows. Section 2 provides background information. In Section 3 we give a formal definition of the CP-ABTD scheme. Section 4 describes the construction for CP-ABTD scheme. In Section 5 we apply the proposed scheme and describe a general architecture for secure management of Personal Health Records (PHRs). The last section concludes the paper.

2 Background

In this section, we briefly review the basis of bilinear pairing, security in the generic group model and give a formal definition of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme.

2.1 Bilinear Pairing

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative groups of prime order p, and let g be a generator of $\mathbb{G}_0$ and $\hat{e}(g, g)$ be a generator of $\mathbb{G}_1$. A pairing (or bilinear map) satisfies the following properties [18]:

1. Bilinear: for all $g, g_1 \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p^*$, we have $\hat{e}(g^a, g_1^b) = \hat{e}(g, g_1)^{ab}$.
2. Non-degenerate: $\hat{e}(g, g_1) \neq 1$.
3. Computable: for any $g, g_1 \in \mathbb{G}_0$, there exist an efficient algorithm to compute $\hat{e}(g, g_1)$.

2.2 Security in the Generic Group Model

We prove the security of the scheme based on the generic group model, introduced by Shoup [19]. The proof in the generic group model is based on the fact that the discrete logarithm and the Diffie-Hellman problem are hard to solve as long as the order of the group is a large prime number. The same applies to a group with bilinear pairing where finding the discrete logarithm is a hard problem. In the generic group model, group elements are encoded as unique random strings, in such a way that the adversary can not test any property other than equality.
We prove the security of our scheme based on the argument that no adversary that acts generally on the groups can break the security of our scheme. This means that if there is an efficient adversary who can discover vulnerabilities in our scheme, then these vulnerabilities can be used to exploit mathematical properties of groups used in the scheme. In the generic model, the adversary has access to the oracles that compute group operations in $\mathbb{G}_0$, $\mathbb{G}_1$, and to the oracle that performs non-degenerate paring $\hat{e}$, while the adversary can test the equality by itself. While it is preferred to prove the security of the scheme by reducing the problem of breaking the scheme to a well studied mathematical problem, a proof in the generic model gives high confidence in the security of the scheme. We leave it as a theoretical future problem to construct a scheme which can be proven secure in the standard assumptions.

2.3 Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

The building block of our construction is a CP-ABE scheme. In CP-ABE scheme a message is encrypted under an access structure τ over the set of possible attributes, and a user secret key sk_ω is associated with an attribute set ω. A secret key sk_ω can decrypt the message encrypted under the access structure τ, if and only if the user attribute set ω satisfies the access structure τ. CP-ABE scheme consists of two entities: a trusted authority (TA) and users. The four algorithms: Setup, Keygen, Encrypt and Decrypt, are defined as follows [2]:

- Setup(k): run by the TA, this algorithm takes as input a security parameter k and outputs the public key pk and a master key mk.
- Keygen(ω, mk): run by the TA, this algorithm takes as input the master key mk and a set of attributes ω. The algorithm outputs a secret key sk_ω associated with ω.
- Encrypt(m, τ, pk): run by the encryptor, this algorithm takes as input a public key pk, a message m, and an access structure represented by an access tree τ. The algorithm returns the ciphertext c_τ such that only users who have the secret key shares associated with attributes that satisfy the access tree τ will be able to decrypt the message.
- Decrypt(c_τ, sk_ω): run by the decryptor, this algorithm takes as input a ciphertext c_τ, a secret key sk_ω associated with ω, and it outputs a message m, or an error symbol $\perp$ when the attribute set ω does not satisfy the access tree τ.

3 The concept of CP-ABTD

CP-ABTD adds an extra entity to the original CP-ABE scheme. Basically, CP-ABTD scheme involves three parties: a trusted authority (TA), a mediator and users. The TA uses the master key to generate a user secret key, which is then divided into two shares such that the first share of the user secret key is sent to the mediator and the second share of the user secret key is sent to a user. The user, holding a share of the secret key associated with a set of attributes can create a new share for a secret key associated with the same or more restricting attributes than she currently holds. This is done independently, that is, the delegator does not have to contact the trusted authority to generate the share of the delegatee's secret key. On the other hand, the mediator can not compute the share of the delegatee's secret key, without the involvement of the delegator. This is because the delegator has to specify the number of attributes he is willing to delegate to the delegatee. Therefore, the delegator sends to the mediator the transformation key which is used by the mediator to compute the first share of the delegatee's secret key. In the encryption phase, same as in CP-ABE scheme, the encryptor encrypts the data according to the access policy, while in the decryption phase, a user must contact the mediator to get the appropriate decryption token.
The CP-ABTD scheme consists of seven algorithms: Setup, Keygen, Encrypt, Delegate, m-Delegate, m-Decrypt, and Decrypt (the Setup and Encrypt algorithms are same as in CP-ABE scheme):

- Keygen(mk, ω, I_u): run by the TA, this algorithm takes as input the master key mk, the user attribute set ω, and the user identifier I_u. The algorithm outputs two secret key shares associated with ω and I_u : $sk_{\omega I_u,1}$ and $sk_{\omega I_u,2}$. The first share of the secret key $sk_{\omega I_u,1}$ is delivered to the mediator, and the second share of the secret key $sk_{\omega I_u,2}$ is delivered to the user. The secret key shares are delivered through a secure channel to the mediator and to the user.

– $\mathsf{Delegate}(sk_{\omega I_u,2}, \hat{\omega}, I_j)$: run by the user (the delegator). This algorithm takes as input the secret key $sk_{\omega I_u,2}$, the delegatee's attribute set $\hat{\omega} \subseteq \omega$, and delegatee's identifier I_j. The output of the algorithm is the second share of the delegatee's secret key $sk_{\hat{\omega} I_j,2}$, and the transformation key $sk_{\omega \to \hat{\omega}}$.
– $\mathsf{m\text{-}Delegate}(sk_{\omega I_u,1}, \hat{\omega}, sk_{\omega \to \hat{\omega}})$: run by the mediator. This algorithm takes as input the secret key $sk_{\omega I_u,1}$, the delegatee's attribute set $\hat{\omega} \subseteq \omega$, and the transformation key $sk_{\omega \to \hat{\omega}}$. The output of the algorithm is the first share of the delegatee's secret key $sk_{\hat{\omega} I_j,1}$.
– $\mathsf{m\text{-}Decrypt}(c_\tau, I_i, sk_{\omega I_i,1})$: run by the mediator, this algorithm takes as input a ciphertext c_τ, the identifier I_i and the secret key $sk_{\omega I_i,1}$, and outputs a message $\hat{c}_\tau$, or an error symbol $\perp$ when the non-revoked attributes from the set ω does not satisfy the access tree τ.
– $\mathsf{Decrypt}(\hat{c}_\tau, sk_{I_i \omega,2})$: run by the message receiver, this algorithm takes as input a ciphertext $\hat{c}_\tau$, and a secret key $sk_{I_i \omega,2}$, and outputs a message m, or an error symbol $\perp$ when the non-revoked attributes from the set ω does not satisfy the access tree τ.

In practice, there might be multiple entities acting as mediators, and a global entity acting as TA. For example, a healthcare organization may choose Proxy_1 as its mediator and a government organization may choose Proxy_2 as its mediator, where each mediator has the first share of the secret key for registered users in the hospital organization, respectively in the government organization. Vanrenen et al. [20] propose the use of peer-to-peer networking (P2P) which would allow users to require a decryption token from every mediator, such as the mediator either tries to compute a decryption token by itself, or forwards the request to its neighbors.

3.1 Security Model

In our scheme the TA is a fully trusted entity which stores securely the master key. We skip discussions about the key escrow problem, since different existing threshold schemes [21,22] can be applied to solve this problem. A mediator is a semi-trusted entity, namely, it should honestly issue decryption tokens to the users, and honestly create a share of delegatee's secret key, but it is not trusted in the sense that it should not obtain information about the plaintaixt.
We define semantic security of the CP-ABTD scheme using a security game between a challenger and an adversary. For a scheme to be semantically secure the adversary must not learn anything about the plaintext when the ciphertext and the public key used to create the ciphertext are given. The security game formally captures the following security requirements:

– Resistance against collusion, where different users cannot combine their attribute sets to extend their decryption power.

– Resistance against malicious cooperation between the mediator and some users to decrypt the ciphertext associated with an access policy, when the users secret key does not satisfy the access policy associated with the ciphertext.
– The delegated secret key should not compromise the security of the scheme.

The challenger simulates the game and answers adversary $\mathcal{A}$ queries as follows:

1. Setup. The challenger runs the Setup algorithm to generate (pk, mk) and gives the public key pk to the adversary $\mathcal{A}$.
2. $\mathsf{Phase1}$. $\mathcal{A}$ performs a polynomially bounded number of queries:
   – $\mathsf{Keygen}^1(\omega, I_u)$. $\mathcal{A}$ asks for a secret key for the attribute set ω and identifier I_u, and receives the first share of the secret key $sk_{\omega I_u,1}$.
   – $\mathsf{Keygen}^2(\omega, I_u)$. $\mathcal{A}$ asks for a secret key for the attribute set ω and identifier I_u, and receives the second share of the secret key $sk_{\omega I_u,2}$.
   – $\mathsf{Delegate}(\omega, I_u, \hat{\omega}, I_j)$. $\mathcal{A}$ asks for the second share of the delegated secret key $sk_{\hat{\omega} I_j,2}$, for $\hat{\omega} \subseteq \omega$, which is a secret key created from $sk_{\omega I_u,2}$. $\mathcal{A}$ receives the second share of the secret key $sk_{\hat{\omega} I_j,2}$.
   – $\mathsf{m} - \mathsf{Delegate}(\omega, I_u, \hat{\omega}, I_j)$. $\mathcal{A}$ asks for the first share of delegated secret key $sk_{\hat{\omega} I_j,1}$, for $\hat{\omega} \subseteq \omega$, which is a secret key created from $sk_{\omega I_u,1}$. $\mathcal{A}$ receives the first share of the secret key $sk_{\hat{\omega} I_j,1}$.
3. $\mathsf{Challenge}$. $\mathcal{A}$ sends to the challenger two messages m_0, m_1, and the challenge access structure τ^*, such that non of the full secret keys generated from the interaction with Keygen^1, Keygen^2, $\mathsf{Delegate}$ and $\mathsf{m} - \mathsf{Delegate}$ oracles satisfy τ^*. The challenger picks a random bit $b \in (0, 1)$ and returns $c_{\tau^*} = \mathsf{Encrypt}(m_b, \tau^*, pk)$.
4. $\mathsf{Phase2}$. $\mathcal{A}$ can continue querying with the restriction that non of the full secret keys generated from the interaction with Keygen^1, Keygen^2, $\mathsf{Delegate}$ and $\mathsf{m} - \mathsf{Delegate}$ oracles satisfy τ^*.
5. Guess. $\mathcal{A}$ outputs a guess $b' \in (0, 1)$.

Definition 1. *A CP-ABTD scheme is said to be semantically secure if any polynomial-time adversary has only a negligible advantage in the security game, where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

4 CP-ABTD scheme

In this section, firstly, we give an informal description of the scheme and techniques we use, secondly we give a description of the access structure specified in the ciphertext, and then we give the construction of the scheme.

4.1 Description of the scheme and Techniques

Our scheme is similar to the work of Cheung and Newport [3] on ciphertext-policy attribute-based encryption, however we make major changes in the Key

Generation phase, Encryption phase and Decryption phase in order to improve the expressivity of the scheme (the scheme in [3] supports only access policies with logical conjunction), and we improve the efficiency of the scheme (in [3] the size of the ciphertext and secret key increases linearly with the total number of attributes in the system).
In our scheme, the access policy, in addition to the logical conjuction, supports logical disjunction, and the size of the ciphertext depends on the size of the access policy, and the size of the secret key depends on the number of attributes the user has. For each attribute, there is one component in the secret key. We split the attribute component of the secret key into two shares: user share and mediator share. An attribute is revoked when the mediator refuses to use his share of the attribute component in the decryption phase. In our scheme we assume that each user has a unique identifier I_u (in CP-ABE the user is identified only with a set of attributes) and may have many attributes. The identifier is used by the mediator to check if there are revoked attribute related to I_u. Different users having different identifiers, may have the same attribute set. For example, Alice with an identifier I_{Alice}, and Bob with an identifier I_{Bob}, may have the same attribute set $\omega = (att_1, att_2)$. The technique of splitting the attribute components of the secret key into two shares, and the technique of using an identifier I_u for each user, helps us to achieve the following attribute revocations: i) revoking an attribute from a single user without affecting other users, and ii) revoking an attribute from the system where all users are affected. For instance, when an attribute is revoked from the system, say att_1, then no user can use att_1 in the decryption phase, since the mediator will refuse to use the share of the attribute component associated with att_1, and when att_1 is revoked from Alice's attribute set, then only Alice cannot use att_1 since the mediator will refuse only requests coming from Alice.
Another important feature that we want to achieve in constructing the scheme is what we call *controlled delegation*. The technique that we use to achieve this feature is as follows: the delegator randomizes the share of the secret key using a random element, such as the secret key generated by the delegator is indistinguishable from the secret key generated by the trusted authority. The delegator sends to the mediator the transformation key which contains the random element and help the mediator to compute the other share of the user secret key. The mediator uses the user identifier I_u to check whether the user has the permission to delegate his/her authority. An user is not allowed to delegate when the mediator refuses to compute the other share of the secret key, on the contrary, the user is allowed to delegate his secret key.

4.2 Access Structure

In our scheme, an access structure is represented by an access tree τ, in which inner nodes are either $\wedge$ (and) or $\vee$ (or) boolean operators, and leave nodes are attributes. The access tree τ specifies which combination of attributes the decryptor needs to posses in order to decrypt the ciphertext. Figure 1 presents an example of an access tree τ representing an access structure: $(a_1 \wedge a_4) \vee (a_3 \vee a_5)$.

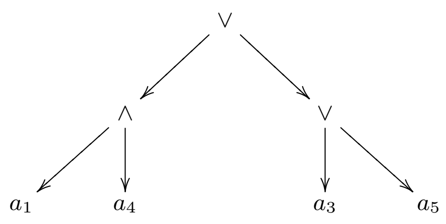

Fig.1. Access tree $\tau = (a_1 \wedge a_4) \vee (a_3 \vee a_5)$

To decrypt an encrypted message under the access tree τ, the decryptor must possess a secret key which is associated with the attribute set which satisfies τ. Attributes are interpreted as logic variables, and possessing a secret key associated with an attribute makes the corresponding logical variable *true*. There are several different sets of attributes that can satisfy the access tree τ presented in Figure 1, such as the attribute set (a_1, a_4), the attribute (a_3), or the attribute (a_5). In our scheme, we assume that attributes are ordered in the access tree e.g index(a_1)=1, index(a_4)=2, index(a_3)=3 and index(a_5)=4.

4.3 Construction of the Scheme

1. $\mathsf{Setup}(k)$: On input of the security parameter k, the algorithm generates a group $\mathbb{G}_0$ of prime order p with a generator g and a bilinear map $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. The algorithm generates the system attribute set $\Omega = (a_1, a_2, \dots a_n)$, for some integer n, and for each $a_j \in \Omega$ chooses a random elements $t_j \in \mathbb{Z}_p^*$.
   Let $y = \hat{e}(g, g)^{\alpha}$ where $\alpha \in_R \mathbb{Z}_p^*$, and $T_j = g^{t_j}$ $(1 \le j \le n)$. The public key is $pk = (\hat{e}, g, y, T_j\,(1 \le j \le n))$, and the master key is $mk = (\alpha, t_j\,(1 \le j \le n))$.
2. $\mathsf{Keygen}(mk, \omega, I_u)$: To generate a secret key for the user with an attribute set ω and an identifier I_u, the Keygen algorithm performs as follows:
   (a) Compute the base component of the secret key: $d_0 = g^{\alpha - u_{id}}$ where $u_{id} \in_R \mathbb{Z}_p^*$ (for each user with an identifier I_u a unique random value u_{id} is generated).
   (b) Compute the attribute component of the secret key. For each attribute $a_j \in \omega$, choose $u_j \in_R \mathbb{Z}_p$ and compute $d_{j,1} = g^{u_j t_j^{-1}}$ and $d_{j,2} = g^{(u_{id} - u_j) t_j^{-1}}$.
   The secret key of the form: $sk_{\omega I_u,1} = (\forall a_j \in \omega : d_{j,1})$ is delivered to the mediator, and the secret key of the form: $sk_{\omega I_u,2} = (d_0, \forall a_j \in \omega : d_{j,2})$ is delivered to the user.
3. $\mathsf{Encrypt}(m, \tau, pk)$: To encrypt a message $m \in \mathbb{G}_1$ the algorithm proceeds as follows:
   – Select a random element $s \in \mathbb{Z}_p^*$ and compute:

$$c_0 = g^s$$
$$c_1 = m \cdot y^s = m \cdot \hat{e}(g, g)^{\alpha s}$$

- Set the value of the root node of τ to be s, mark all nodes as un-assigned, and mark the root node assigned. Recursively, for each assigned non-leaf node, suppose its value is s, do the following.
  - If the symbol is $\wedge$ and its child nodes are marked un-assigned, let n be the number of child nodes, set the value of each child node, except the last one, to be $s_i \in_R \mathbb{Z}_p^*$, and the value of the last node to be $s_n = s - \Sigma_{i=1}^{n-1} s_i \mod p$ (i represents the index of an attribute in the access tree). Mark this node assigned.
  - If the symbol is $\vee$, set the values of its child nodes to be s. Mark this node assigned.
- For each leaf attribute $a_{j,i} \in \tau$, compute $c_{j,i} = T_j^{s_i}$.

Return the ciphertext $c_\tau = (\tau, c_0, c_1, \forall a_{j,i} \in \tau : c_{j,i})$.

4. Delegate$(sk_{\omega I_u,2}, \hat{\omega}, I_j)$ Assume that the set $\hat{\omega} \subseteq \omega$ is the attribute set that the delegator wants to delegate to the delegatee with an identifier I_j. In the delegation phase, the delegator plays the role of the TA for the delegatee. The delegatee's secret key is randomized with a random value, therefore delegatee's secret key is equivalent to the one received by the TA. To compute delegatee's private key, the delegator chooses a random element $r' \in \mathbb{Z}_p$, for each $a_j \in \hat{\omega}$ sets $g^{t_j \cdot r'} = g^{r''_j}$, and sets the transformation key: $sk_{\omega \to \hat{\omega}} = g^{r'}$. For each $a_j \in \hat{\omega}$ computes $\hat{d_{j,2}}$ as:

$$\begin{aligned} \hat{d_{j,2}} &= g^{(u_{id} - u_j)t_j^{-1} - r'} \\ &= g^{(u_{id} - u_j)t_j^{-1} - r''_j t_j^{-1}} \\ &= g^{(u_{id} - \hat{u_j})t_j^{-1}} \end{aligned}$$

where $\hat{u_j} = u_j + r''_j$.

The secret key of the form $sk_{\hat{\omega} I_j,2} = (d_0, \forall a_j \in \hat{\omega} : \hat{d_{j,2}})$ is sent to the delegatee, and $(\hat{\omega}, sk_{\omega \to \hat{\omega}})$ to the mediator.

5. m-Delegate$(sk_{\omega I_u,1}, \hat{\omega}, sk_{\omega \to \hat{\omega}})$: The mediator receives from the delegator I_u a list of attributes $\hat{\omega}$ that the delegator wants to delegate to the delegatee, and the transformation key $sk_{\omega \to \hat{\omega}}$. The mediator checks the Attribute Delegation List (ADL) if the delegator with an identifier I_u is allowed to delegate his secret key.

   (a) If the delegation is not allowed, the mediator will refuse to compute the first share of the delegatee's secret key, thus, an unauthorized delegation of the secret key from delegator to the delegatee will fail. The mediator returns an error symbol $\perp$.

   (b) If the delegation is allowed, the mediator computes $sk_{\hat{\omega} I_j,1}$ as follows. For each $a_j \in \hat{\omega}$ compute:

$$\begin{aligned} \hat{d_{j,1}} &= g^{u_j t_j^{-1} + r'} \\ &= g^{\hat{u_j} t_j^{-1}} \end{aligned}$$

The mediator computes the first share of the delegatee's secret key: $sk_{I_j\hat{\omega},1} = (\forall a_j \in \hat{\omega} : \widehat{d_{j,1}})$.
Note: In this paper, we do not explain how the mediator authenticates the delegator. For user authentication, different existing techniques can be applied, hence, we skip a formal description of this problem.

6. $\mathsf{m\text{-}Decrypt}(c_\mathcal{T}, sk_{\omega I_i,1}, I_i)$: when receiving the ciphertext $c_\mathcal{T}$, the recipient I_i firstly chooses the smallest set $\omega' \subseteq \omega$ that satisfies τ and forwards to the mediator $(c_\mathcal{T}, \omega', I_i)$. The mediator checks the Attribute Revocation List (ARL) if any $a_j \in \omega'$ is revoked either from system attribute set Ω or from the user attribute set ω.
   (a) If there is any attribute revoked, the mediator returns an error symbol $\perp$ and does not perform further computations.
   (b) If there is no attribute revoked, the mediator computes $\hat{c}_\mathcal{T}$ as follows. For every attribute $a_j \in \omega'$ compute:
$$\begin{aligned}\hat{c}_\mathcal{T} &= \prod_{a_j \in \omega'} \hat{e}(T_j^{s_i}, g^{u_j t_j^{-1}}) \\ &= \hat{e}(g,g)^{\sum_{a_j \in \omega'} u_j s_i}\end{aligned}$$
   Sends $\hat{c}_\mathcal{T}$ to the recipient.
7. $\mathsf{Decrypt}(\hat{c}_\mathcal{T}, sk_{\omega I_i,2})$: To decrypt the ciphertext the recipient proceeds as follows:
   (a) For every attribute $a_j \in \omega'$ compute:
$$\begin{aligned}c''_\mathcal{T} &= \prod_{a_j \in \omega'} \hat{e}(T_j^{s_i}, g^{(u_{id}-u_j)t_j^{-1}}) \\ &= \prod_{a_j \in \omega'} \hat{e}(g^{t_j s_i}, g^{(u_{id}-u_j)t_j^{-1}}) \\ &= \hat{e}(g,g)^{\sum_{a_j \in \omega'}(u_{id}-u_j)s_i}\end{aligned}$$
   (b) compute:
$$\begin{aligned}\hat{e}(c_0, d_0) \cdot \hat{c}_\mathcal{T} \cdot c''_\mathcal{T} &= \hat{e}(g^s, g^{\alpha-u_{id}}) \cdot \hat{e}(g,g)^{\sum_{a_j \in \omega'} u_j s_i} \cdot \hat{e}(g,g)^{\sum_{a_j \in \omega'}(u_{id}-u_j)s_i} \\ &= \hat{e}(g^s, g^{\alpha-u_{id}}) \cdot \hat{e}(g,g)^{u_{id}s} \\ &= \hat{e}(g^s, g^\alpha)\end{aligned}$$
   (c) return m, where
$$\begin{aligned}m &= \frac{c_1}{\hat{e}(g^s, g^\alpha)} \\ &= \frac{m \cdot \hat{e}(g,g)^{\alpha s}}{\hat{e}(g^s, g^\alpha)}\end{aligned}$$

Efficiency. The size of the shares of the secret key $sk_{\omega I_u,1}$ and $sk_{\omega I_u,2}$ depend on the number of attributes the user has and consists of $|\omega| + 1$ group elements in $\mathbb{G}_0$ ($|\omega|$ is the cardinality of of a set ω). The size of the ciphertext c_τ depends on the size of the access policy τ and has $|\tau| + 1$ group elements in $\mathbb{G}_0$, and one group element in $\mathbb{G}_1$. The size of the transformation key $sk_{\omega \to \hat{\omega}}$ is one group element in $\mathbb{G}_0$. In the m-Decrypt phase, the mediator has to compute ω' pairing operations, where $\omega' \subseteq \omega$ is the attribute set which satisfy the access policy τ. In the decryption phase, to reveal the message, the user has to compute $\omega' + 1$ pairing operations.

Security Analysis. The very important security property of an Attribute-Based Encryption (ABE) scheme is the collusion resistance - it should be not possible for different users to combine their attribute sets in order to extend their decryption power. For example, suppose there is a message encrypted under the access tree $\tau = (a_1 \wedge a_2 \wedge a_3)$. Suppose Alice has a secret key $sk_{\omega_A I_A}$ associated with an attribute set $\omega_A = (a_1, a_2)$, and Bob has a secret key $sk_{\omega_B I_B}$ associated with an attribute set $\omega_B = (a_3, a_4)$. Neither Alice's secret key, nor Bob'secret key satisfies the access tree τ. But, if Alice and Bob combine their attribute sets $\omega_A \cup \omega_B = (a_1, a_2, a_3, a_4)$, then the combined attribute sets satisfies the access tree τ. In a collusion resistent scheme, there should be not possible for Alice and Bob to combine their secret keys, therefore to prevent collusion, the Keygen algorithm of our scheme generates a random value u_{id} for each user, which is embedded in each component of the user secret key. Users cannot combine components of the secret key since different users have different random value in their secret keys. A full formal security proof is given in Appendix A.

Attribute Revocation. As already mentioned in section 1, there can be many reasons why an attribute can be revoked. We assume that the mediator maintains an Attribute Revocation List (ARL) which simply has information about attributes revoked from system attribute set Ω, and attributes revoked from user attribute set ω.

The basic idea is that, when an attribute a_j is revoked from the system attribute set Ω, the trusted authority (TA) removes the a_j from the system attribute set, and notifies the mediator to stop performing decryption tasks for all users whose attribute secret key involves a_j. When an attribute is revoked from a specific user I_u, the trusted authority notifies the mediator to stop helping the user I_u to perform decryption tasks for the attribute a_j.

The revocation of user attribute implies the revocation of the same attribute for underlying users. For instance, suppose the attribute a_1 is revoked from Alice who has an identifier I_A and a secret key associated with the attribute set $\omega_A = (a_1, a_2, a_3)$. Lets assume that Alice has delegated her authority to Bob by creating a new secret key associated with an attribute set $\omega_B = (a_1, a_2)$. An attribute revocation from the set ω_A, will imply the revocation of the same attribute from the set ω_B, since Bob has received the secret key associated with the set ω_B from Alice.

We assume that there is a policy of revocation authorization maintained by the mediator that describes who is responsible to revoke system or user attributes. At least, the TA should be able to revoke the system and user attributes, the delegator should be able to revoke the delegatee attributes, and the owner of the attribute should be able to revoke its attribute because the owner may be the first to notice the comprise of her secret key.

Attribute Delegation. For ABE systems, which identify users with a set of attributes, delegation means that a user playing the role of the delegator gives all or a subset of her attributes to other users playing the role of the delegatee. Suppose Alice (the delegator) who has a secret key $sk_{\omega_A I_A}$ associated with a list of attributes $\omega_A = (a_1, a_2, a_3)$, has given to Bob (the delegatee) a secret key $sk_{\omega_B I_B}$ associated with the list of attributes $\omega_B = (a_1, a_3)$. Bob using $sk_{\omega_B I_B}$ can decrypt every ciphertext which access tree is satisfied by ω_B. Now Bob wishes to delegate to Charlie a secret key $sk_{\omega_C I_C}$ associated with the list of attributes ω_C. In our scheme, in case the delegation is allowed, Bob can delegate to Charlie a secret key which attribute set does not exceed the number of attributes that Bob has in his secret key, otherwise the security of CP-ABTD scheme would be compromised. In case the delegation is not allowed, Bob should not be able to delegate any of his attributes.
Similar to the revocation, we assume that the mediator maintains an Attribute Delegation List (ADL) which has information about users who are allowed to delegate their attributes. The TA in the Keygen phase, or the delegator in the Delegate phase, notify the mediator whether the user (delegatee) has the permission to delegate further his secret key.

4.4 Multi-Authority CP-ABTD

Ideally, we would like to have multiple independent authorities which would manage user attributes and distribute secret keys. Assume that the Attribute Authority (AA) from Hospital A manages the attribute set $\Omega_{HospitalA}$, and that the AA from Hospital B manages the attribute set $\Omega_{HospitalB}$. In the multi-authority setting, the encryptor has the flexibility to chose different attributes from different authorities in the access policy of the ciphertext, such that only users who have attributes from the given authority can decrypt the ciphertext. For instance, a patient may want to encrypt her health data, such that a user who has the attribute General Practitioner received from Hospital A or the attributes General Practitioner and Pediatrician received from Hospital B can decrypt the ciphertext.
Chase [23] gives the construction of the first multi-authority attribute-based encryption (ABE), which allows multiple independent authorities to monitor user attributes. We can apply the same idea to extend the scheme presented in section 4 to support multi-authority ciphertext-policy attribute-based encryption. The main requirement that we have is that each AA should use the same unique identifier I_u for each user, and then use a function which takes as input I_u to generate the same value u_{id} for each AA, where u_{id} is the same in all components of the secret key, and is used to connect the base component of the secret key with the attribute component of the secret key. The component of the master secret

key α is part of the base secret key which is not connected with attribute secret keys, therefore, there is no need for attribute authorities to know α. However, there should be an entity who will manage with α. Thus, in addition to AA, a central authority (CA) is needed. We extend the single-authority CP-ABTD scheme presented in section 4 to a multi-authority CP-ABTD as follows (only changes from the scheme in section 4 are presented):

1. Setup :
   (a) Central Authority (k): Generates a group $\mathbb{G}$ of prime order p with a generator g and a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$. Set the component of the master secret key $\alpha \in_R \mathbb{Z}_p^*$, and the component of the public key $\hat{e}(g, g)^{\alpha}$.
   (b) Attribute Authority(AA)$-l$: Generate the attribute set $\Omega_l = (a_{l,1}, a_{l,2}...a_{l,n})$. For each $a_{l,j} \in \Omega_l$ set the attribute secret key: $t_{l,1}...t_{l,n}$, and the attribute public key $T_{l,j} = g^{t_{l,j}}$ $(1 \leq j \leq n)$.
2. Keygen
   (a) Central Authority : Compute the base component of the secret key: $d_0 = g^{\alpha - u_{id}}$
   (b) Attribute Authority(AA)$-l$: Suppose the user with an identifier I_u applies for the set of attributes ω_k to the AA l. The AA l computes the attribute secret key as follows: for each $a_{l,j} \in \omega_l$, compute $d_{l,j,1} = g^{u_{id} t_{l,j}^{-1}}$ and $d_{l,j,2} = g^{(u_{id} - u_{l,j}) t_{l,j}^{-1}}$, where $u_{l,j} \in_R \mathbb{Z}_p$.

5 Secure management of PHRs using CP-ABTD

Issues around the confidentiality of health records are considered as the primary reason for the lack of the deployment of open interoperable health record systems. Health data is sensitive: inappropriate disclosure of a record can change a patient's life, and there may be no way to repair such harm financially or technically. Although, access to health data in the professional medical domain is tightly controlled by existing legislations, such as the U.S. Health Insurance Portability and Accountability Act (HIPAA) [24], private PHR repositories stay outside the scope of this legislation. Therefore, a number of patients might hesitate to upload their sensitive health records to web PHR systems such as the Microsoft Health Vault, Google Health or WebMD. The scheme presented in this paper helps PHR system designers to alleviate this issue and to protect their data from non-trusted parties that store their records. Figure 2 illustrates a general architecture of a PHR system that uses CP-ABTD. The architecture consists of a publishing server, a data repository that includes a security mediator (Proxy), a trusted authority and several data users. The publishing server can be implemented on a home PC of the data source (a patient) or as a trusted service. Its role is to protect and publish health records. The data repository stores encrypted health records, while the Proxy is used in the data consumption phase for revocation and delegation. The TA is used to set up the keys. Note that the TA and the publishing server do not have to be always online (the TA is needed only in the set-up phase while the publishing server can upload the protected

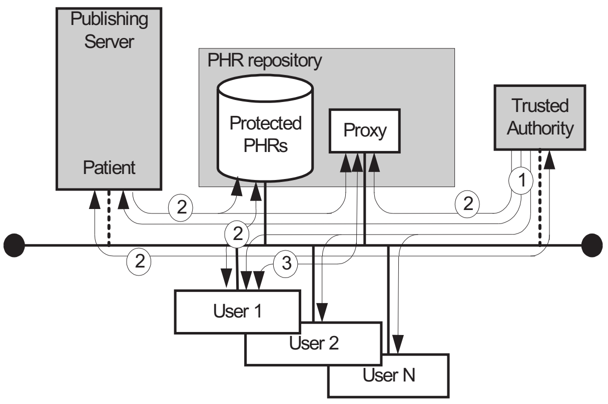

Fig. 2. Secure Management of PHR

data in an ad-hoc way). There are four basic processes in the management of PHRs:

1. Setup: The steps of this phase are depicted with number 1 in figure 2. In this phase, the TA distributes the keys to the patients, users and the Proxy.
2. Data protection (upload of data to the PHR): When a patient wants to upload protected data to the repository, she contacts the TA to check which attributes are allowed to be used as a policy. Then she creates her access control policy and encrypts the data with the keys corresponding to that policy. Then the data is uploaded to the repository. If she wants to change the policy she can re-encrypt the data and update the repository.
3. Data consumption (doctor's request response) and revocation: When a user wants to use patient data he contacts the PHR repository and downloads encrypted data. The user makes a request to the Proxy for a decryption token. The request contains the encrypted data and a set of user attributes which satisfy the access policy associated with the encrypted data. The Proxy checks if any attribute from the user request is not revoked, and, if so, the Proxy generates the decryption token and send it to the user. After receiving the decryption token the user reveals the patient data using the keys corresponding to the appropriate attributes which satisfy the access tree. The steps of this phase are depicted with number 3 in figure 2.
4. Delegation: when a user (delegator) wants to delegate the data to another user (delegetee) e.g. a doctor may want to delegate his decryption right to a nurse, the delegator calculates a share of the delegatee's key by himself, and distributes the transition key to the proxy. The Proxy uses the transition key to compute the other share of the delegatee's key.

The CP-ABTD scheme can also support the off-line use of data. Then the architecture is slightly changed in a way that the Proxy is distributed to the users or their domains within which the data will be used. As a consequence there will be a number of Proxies which will be coordinated by the central Proxy. The above defined process will not fundamentally change, except that in the central Proxy will update the local ones and that in the data consumption phase, the user will contact only the local Proxy.

6 Conclusion and Future Work

We propose Ciphertext-Policy Attribute-Based Threshold Decryption (CP-ABTD) scheme, which supports flexible (controlled) delegation and revocation of user attributes. In CP-ABTD, a delegator can delegate to the delegatee a secret key associated with all or a subset of his attributes. The delegatee's secret key received from the delegator is computationally indistinguishable from the secret key received directly from the Trusted Authority (TA). However, in some secure systems, secret key delegation is seen as a security threat and not a desirable property, therefore, our scheme allows the delegator to prevent the delegatee to delegate further his authority. Attribute revocation revokes the user's ability to use the revoked attribute in the decryption operation. Attribute revocation is controlled by the TA, the delegator or by the owner of the attribute. Finally, we demonstrate how to use the proposed CP-ABTD scheme to securely manage Personal Health Records (PHRs).

The work presented in this paper suggests an important challenge for our future work: to construct a ciphertext-policy attribute-based encryption scheme which would have both: the flexible delegation and attribute revocation properties, without involving a mediator in the system architecture.

References

1. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–Eurocrypt 2005*, volume 3494, pages 457–473. Springer, 2005.
2. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 321–334, 2007.
3. L. Cheung and C. Newport. Provably secure ciphertext policy ABE. *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 456–465, 2007.
4. M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 99–112, 2006.
5. D. Boneh, X. Ding, G. Tsudik, and C.M. Wong. A method for fast revocation of public key certificates and security capabilities. In *Proceedings of the 10th conference on USENIX Security Symposium-Volume 10 table of contents*, pages 22–22. USENIX Association Berkeley, CA, USA, 2001.

6. D. Boneh, X. Ding, and G. Tsudik. Fine-Grained Control of Security Capabilities. *ACM Transactions on Internet Technology*, 4(1):60–82, 2004.
7. B. Libert and J.J. Quisquater. Efficient revocation and threshold pairing based cryptosystems. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 163–171. ACM New York, NY, USA, 2003.
8. D. Nali, A. Miri, and C. Adams. Efficient Revocation of Dynamic Security Privileges in Hierarchically Structured Communities. In *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust (PST 2004), Fredericton, New Brunswick, Canada*, pages 219–223, 2004.
9. D. Nali, C. Adams, and A. Miri. Using Mediated Identity-Based Cryptography to Support Role-Based Access Control. In *Information Security*, volume 3225, pages 245–256. Springer, 2004.
10. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 89–98, 2006.
11. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology – EUROCRYPT 2008*, volume 4965, pages 146–162. Springer, 2008.
12. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. *Journal of Cryptology*, 21(3):350–391, 2008.
13. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027, pages 506–522. Springer, 2004.
14. X. Boyen and B. Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In *Advances in Cryptology – CRYPTO 2006*, volume 4117, pages 290–307. Springer, 2006.
15. J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332, pages 466–481. Springer, 2002.
16. D. Boneh, X. Boyen, and E.J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494, pages 440–456. Springer, 2005.
17. E. Shi and B. Waters. Delegating capabilities in predicate encryption systems. In *Proceedings of ICALP*, volume 5126, pages 560–578. Springer, 2008.
18. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
19. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. *LNCS*, pages 256–266, 1997.
20. G. Vanrenen and S. Smith. Distributing security-mediated PKI. In *Public Key Infrastructure*, volume 3093, pages 218–231. Springer, 2004.
21. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
22. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology – CRYPTO' 89 Proceedings*, volume 435, pages 307–315. Springer, 1990.
23. M. Chase. Multi-authority Attribute Based Encryption. In *Theory of Cryptography*, volume 4392, pages 515–534. Springer, 2007.
24. The US Department of Health and Human Services. Summary of the HIPAA Privacy Rule, 2003.

25. JT Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

A Security Proof

We prove the following theorem:

Theorem 1. *Let q be the total number of group elements the adversary it receives from the queries it makes to the oracles that perform group operations in $\mathbb{G}_0$, $\mathbb{G}_1$, and to the oracle that performs non-degenerate paring $\hat{e}$, and from its interaction with the security game. Let γ_0 and γ_1 be random encoding functions which map elements of additive group $\mathbb{Z}_n$ into a set $S \subset \{0,1\}^*$ of bit strings, where p is the largest prime divisor of n. Then the advantage of the adversary in the security game is $O(q^2/p)$.*

Proof. Consider groups $\mathbb{G}_0$ and $\mathbb{G}_1$, and generators g of group $\mathbb{G}_0$, and $\hat{e}(g,g)$ of group $\mathbb{G}_1$. In generic group model, group elements are encoded as unique random strings, in such a way that the adversary can not test any property other than equality. In our proof, we use γ_0 as a random encoding for group $\mathbb{G}_0$ (generic bilinear group), and γ_1 as a random encoding for group $\mathbb{G}_1$. Thus, for example, the group element $g^s \in \mathbb{G}_0$ will be encoded as $\gamma_0(s)$, and $\hat{e}(g,g)^\alpha \in \mathbb{G}_1$ will be encoded as $\gamma_1(\alpha)$.
We now give the simulation of the security game. Following the proof from [2], in the simulation we modify slightly the security game given in section 3.1, and simulate a game in which the c_1 component of the challenge phase is either $\gamma_1(\alpha s)$ or $\gamma_1(\theta)$, where $\theta \in_R \mathbb{Z}_p$, and the adversary has to decide whether $c_1 = \gamma_1(\alpha s)$ or $c_1 = \gamma_1(\theta)$. The advantage of the adversary to win the modified security game is at least $\frac{\epsilon}{2}$. When $c_1 = e(g,g)^{\alpha s}$, encoded as $\gamma_1(\alpha s)$, the adversary's advantage is ϵ. When $c_1 = \hat{e}(g,g)^\theta$, encoded as $\gamma_1(\theta)$, the advantage of the adversary is $\frac{1}{2}$. Therefore the overall advantage of the adversary in the modified security game is $\frac{\epsilon}{2}$.
In our security proof, the simulator simulates the modified security game and bounds the advantage of the adversary in the modified security game. The simulator maintains the table L_1 to store information about values generated from the interaction of the adversary with the Keygen^1 and Keygen^2 oracle, and the table L_2 contains information about values generated from the interaction of the adversary with the $\mathsf{Delegate}$ and $\mathsf{m-Delegate}$ oracles. The security game is simulated as follows:

1. Setup. For each attribute $a_j \in \Omega$, the simulation chooses random values t_j from $\mathbb{Z}_p$ and constructs $T_j = \gamma_0(t_j)$, representing $T_j = g^{t_j}$. The simulation sets the parameter $y = \gamma_1(\alpha)$, representing $Y = \hat{e}(g,g)^\alpha$, where α is a random value from $\mathbb{Z}_p$. The public parameters y and T_j $(1 \leq j \leq n)$ are sent to the adversary.
2. Phase1. $\mathcal{A}$ performs a polynomially bounded number of queries:

- $\mathsf{Keygen}^1(\omega, I_u)$. $\mathcal{A}$ makes request for the first share of the secret key for an attribute set ω and an identifier I_u. The simulator checks whether L_1 already contains a record for the attribute set ω and the identifier I_u. If such record exists, the simulator fetches $d_{j,1}$ from the record and sends $sk_{\omega I_u,1} = (\forall a_j \in \omega : d_{j,1})$ to the adversary $\mathcal{A}$. If such record does not exist, the simulator chooses a random variable $u_{id} \in \mathbb{Z}_p$, and computes $d_0 = \gamma_0(\alpha - u_{id})$, representing $d_0 = g^{\alpha - u_{id}}$. For each $a_j \in \omega$ the simulator chooses u_j and computes $d_{j,1} = \gamma_0(u_j t_j^{-1})$, representing $d_{j,1} = g^{u_j t_j^{-1}}$, and $d_{j,2} = \gamma_0((u_{id} - u_j)t_j^{-1})$, representing $d_{j,2} = g^{(u_{id} - u_j)t_j^{-1}}$. The simulation sends $sk_{\omega I_u,1} = (\forall a_j \in \omega : d_{j,1})$ to $\mathcal{A}$, and puts a new record $sk_{\omega I_u} = (\omega, I_u, d_0, \forall a_j \in \omega : \{d_{j,1}, d_{j,2}\})$ into the table L_1.
- $\mathsf{Keygen}^2(\omega, I_u)$. $\mathcal{A}$ makes request for the second share of the secret key for an attribute set ω and an identifier I_u. The simulator checks whether L_1 already contains a record for the attribute set ω and the identifier I_u. If such entry exists, the simulator sends $sk_{I_u \omega,2} = (d_0, \forall a_j \in \omega : d_{j,2})$ to the adversary $\mathcal{A}$. If such entry does not exist, the simulator calculates d_0, $d_{j,1,}$ and $d_{j,2}$ as explained under $\mathsf{Keygen}^1(I_u, \omega)$ and updates the table L_1 with the new record $sk_{\omega I_u} = (d_0, \forall a_j \in \omega : \{d_{j,1}, d_{j,2}\})$.
- $\mathsf{Delegate}(\omega, I_u, \hat{\omega}, I_j)$. $\mathcal{A}$ makes request for the second share of the secret key for an attribute set $\hat{\omega}$ and an identifier I_j. $\mathcal{A}$ also specifies the attribute set ω and identifier I_u from which the $sk_{\hat{\omega} I_j,2}$ has to be generated. The simulator checks whether L_2 already contains a record for the attribute set $\hat{\omega}$ and the identifier I_j derived from the attribute set ω and the identifier I_u. If such record does not exist, the simulator generates the record $sk_{\omega I_u} = (d_0, \forall a_j \in \omega : \{d_{j,1}, d_{j,2}\})$ as explained under $\mathsf{Keygen}^1(I_u, \omega)$ and update the table L_1 with the newly created record. The simulator uses $sk_{\omega I_u} = (d_0, \forall a_j \in \omega : \{d_{j,1}, d_{j,2}\})$to generate $sk_{\omega I_j,2}$ as follows: the simulator chooses a random value r' and computes $\hat{d_{j,2}} = \gamma_0((u_{id} - \hat{u_j})t_j^{-1})$, representing $\hat{d_{j,2}} = g^{(u_{id} - \hat{u_j})t_j^{-1}}$ where $\hat{u_j} = u_j + r_j^{"}$, and $\hat{d_{j,1}} = \gamma_0(\hat{u}_j t_j^{-1})$, representing $\hat{d_{j,1}} = g^{\hat{u_j} t_j^{-1}}$. The simulator sends to $\mathcal{A}$ the secret key $sk_{\hat{\omega} I_j,2} = (d_0, \forall a_j \in \hat{\omega} : \hat{d_{j,2}})$, and updates L_2 with the newly created record $sk_{\hat{\omega} I_j} = (d_0, \forall a_j \in \hat{\omega} : \{\hat{d_{j,1}}, \hat{d_{j,2}}\})$.
- $\mathsf{m} - \mathsf{Delegate}(\omega, I_u, \hat{\omega}, I_j)$. $\mathcal{A}$ makes request for the first share of the secret key for an attribute set $\hat{\omega}$ and an identifier I_j, derived from the attribute set ω and the identifier I_u. The simulator checks whether L_2 already contains a record for the attribute set $\hat{\omega}$ and the identifier I_j derived from the attribute set ω and the identifier I_u. If such entry exists, the simulator sends $sk_{\hat{\omega} I_j,1} = (d_0, \forall a_j \in \omega : d_{j,2})$ to the adversary $\mathcal{A}$. If such entry does not exist the simulator computes $sk_{\hat{\omega} I_j}$ as explained under $\mathsf{Delegate}(\omega, I_u, \hat{\omega}, I_j)$, it updates L_2 with the newly created record, and sends to the adversary $sk_{\hat{\omega} I_j,1} = (d_0, \forall a_j \in \omega : d_{j,2})$.

3. Challenge. The adversary submits two messages m_0, $m_1 \in \mathbb{G}_1$ and the challenge access policy τ^*. The adversary is not allowed to ask for challenge

access policy τ^* such that one of the full secret keys issued in Phase1 satisfies τ^*.
The simulation chooses a random $s \in \mathbb{Z}_p$, and for each $a_{j,i} \in \tau^*$ it constructs a value s_i as explained in section 4.3. Finally, the simulator sets $c_0 = \gamma_0(s)$ representing $c_0 = g^s$, and $c_1 = \gamma_1(\theta)$, representing $c_1 = \hat{e}(g,g)^\theta$. For each attribute $a_j \in \tau^*$ it sets $c_{ji} = \gamma_0(t_j s_i)$, representing $c_{ji} = g^{t_j s_i}$. The ciphertext $c_{\tau^*} = (c_0, c_1, \forall a_{j,i} \in \tau^* : c_{j,i})$ is sent to the adversary.

4. Phase2. $\mathcal{A}$ can continue querying with the restriction that non of the full secret keys generated from the interaction with Keygen^1, Keygen^2, Delegate and $\mathsf{m-Delegate}$ oracles satisfy τ^*.

The adversary performs queries to the generic group oracles: oracle for the group operation in $\mathbb{G}_0$ and oracle for the group operation in $\mathbb{G}_1$, and oracle that performs non-degenerate paring $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. The adversary communicates with the generic group oracles using the encodings of the group elements. Each oracle query made by the adversary is associated with a distinct rational function $f = \frac{\xi}{\varpi}$ in the indeterminate variables:

$$\hat{\Upsilon} = \{\alpha, s, s_i, u_{id}, r', t_j (1 \leq j \leq n), \theta\}$$

where ξ, ϖ are polynomial functions.
First we show what the adversaries behavior is when we set $c_1 = \gamma_1(\theta)$. When the adversary makes query to the oracles, the adversaries behavior can change when an unexpected collision happen. An unexpected collision may happen due to the random choice of the formal variables Υ which are chosen uniformly from $\mathbb{Z}_p$. An unexpected collision would be when two queries corresponding to two different rational functions $f = \frac{\xi}{\varpi}$ and $f' = \frac{\xi'}{\varpi'}$, such that $\frac{\xi}{\varpi} \neq \frac{\xi'}{\varpi'}$, and $\xi\varpi' - \xi'\varpi = 0$. If such collusion happens then the simulator quits and the adversary wins. Based on the Schwartz [25] lemma the probability of the collision is $O(1/p)$, thus, the advantage of the adversary is $O(q^2/p)$, and the probability of no such collision is $1 - O(q^2/p)$. Since this probability is negligible, we assume that no such collusion happen.
Now we show that the adversaries behavior is identically distributed even if we set $c_1 = \gamma_1(\alpha s)$. The adversaries behavior can differ when there are two different queries $f = \frac{\xi}{\varpi}$ and $f' = \frac{\xi'}{\varpi'}$ into $\mathbb{G}_1$, such that $f = f' = \alpha s$. We show that this is not possible since the adversary can not construct a query into $\mathbb{G}_1$ which coincide to $\hat{e}(g,g)^{\alpha s}$. To proof the theorem, we obtain the contradiction by showing that none of the adversaries queries can be equal to the form αs.
We proceed the proof by analyzing queries (see Table 1) that the adversary can make into generic group oracles using the group elements the adversary received from the interaction with the simulator in the security game. First we observe that the adversary can make query which contain the term αs, by pairing $g^{\alpha - u_{id}}$ and g^s to get $\alpha s - s u_{id}$. To get only αs, the adversary has to combine group elements received from the interaction with the simulator and from the generic group oracles to cancel the term $s u_{id}$. To construct $s u_{id}$, the adversary could try to pair $g^{t_j s_i}$ and g^{u_{id}/t_j} (g^{u_{id}/t_j} can be calculated by multiplying $g^{(u_{id} - u_j)/t_j}$

s	α	$\alpha \pm s$	$\alpha - u_{id}$
t_j	$\alpha t_j - u_{id} t_j$	$s \pm t_j$	$\alpha \pm t_j$
$t_j s_i$	$u_{id} s_i$	$u_{id} t_j^{-1}$	$\alpha s_i t_j - u_{id} s_i t_j$
0	$\alpha u_j t_j^{-1} - u_{id} u_j t_j^{-1}$	$\alpha s - s u_{id}$	

Table 1. Possible queries into $\mathbb{G}_1$ from the adversary

with g^{u_j/t_j}), and obtain $g^{u_{id} s_i}$. The adversary must have all necessary secret key components g^{u_{id}/t_j} to pair with $g^{t_j s_i}$ and obtain $g^{u_{id} s}$. This is not possible since in the Phase1 and Phase2, the adversary is not allowed to make queries to Keygen^1, Keygen^2, Delegate and $\mathsf{m} - \mathsf{Delegate}$ oracles such that the full secret key generated sk_{ω, I_u} does satisfy the challenge access policy τ^*. Thus, the attribute set ω cannot be used to reconstruct su_{id}, and then later cancel u_{id}. As a result of this the adversary query can not contain the term αs.