AN ENCRYPTION SCHEME FOR A SECURE POLICY UPDATING

Luan Ibraimi
Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands
ibraimi@ewi.utwente.nl

Muhammad Asim
Philips Research Eindhoven, Eindhoven, The Netherlands
muhammad.asim@philips.com

Milan Petković
Philips Research Eindhoven and Faculty of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
milan.petkovic@philips.com

Keywords: Proxy re-encryption, Attribute-based encryption, Access policy, Attribute-based proxy re-encryption.

Abstract: Ciphertext policy attribute based encryption is an encryption technique where the data is encrypted according to an access policy over attributes. Users who have a secret key associated with a set of attributes which satisfy the access policy can decrypt the encrypted data.
However, one of the drawbacks of the CP-ABE is that it does not support updating access control policies without decrypting the encrypted data.We present a new variant of the CP-ABE scheme called ciphertext policy attribute based proxy re-encryption (CP-ABPRE). The proposed scheme allows to update the access control policy of the encrypted data without decrypting the ciphertext. The scheme uses a semitrusted entity called proxy to re-encrypt the encrypted data according to a new access control policy such that only users who satisfy the new policy can decrypt the data. The construction of our scheme is based on prime order bilinear groups. We give a formal definition for semantic security and provide a security proof in the generic group model.

1 INTRODUCTION

Recent studies explore the use of cryptographic techniques to enforce access control policies. Ciphertext policy attribute based encryption (CP-ABE) schemes allow the data to be encrypted according to an access control policy over a set of descriptive attributes (e.g. doctor and nurse). Once the data is encrypted, it can be safely stored in an un-trusted server such that everyone can download the encrypted data (even a malicious user), but only users who have the right secret key associated with a set of attributes which satisfy the access policy can decrypt. Therefore, when the data is encrypted using a CP-ABE, access policy moves with the data and there is no need for the use of other entities, such as access-control managers, to enforce access control policy. For instance, Bob can encrypt his health data according to the access policy $p_1 = [Bob\ OR\ (GP\ AND\ Hospital\ 1)]$, and upload encrypted data to an un-trusted Personal Health Record (PHR) server. Only users who have attributes *Bob* or *GP* and $Hospital\ 1$ can decrypt the ciphertext, so neither the server itself nor an unauthorized person can decrypt the ciphertext.

Despite numerous advantageous features of the CP-ABE schemes compared to the traditional access control technologies, CP-ABE schemes does not support updating access control policies. The only way is to decrypt the data and then re-encrypt it according to a new access control policy. Following the above example, if Bob wants to change the access control policy from p_1 to $p_2 = [Bob\ OR\ (GP\ AND\ (Hospital\ 1\ OR\ Hospital\ 2))]$ (in order to hear a second opinion from a *GP* from $Hospital\ 2$), Bob has to re-encrypt his data according to p_2. A naive solution for Bob to re-encrypt his data would be to send to the PHR server his secret key. Once the PHR server receives the secret

key, it decrypts the data and then use the CP-ABE scheme to re-encrypt the data according to the new policy p_2. However , the drawback of this approach is that the server accesses sensitive plain data. To avoid this drawback Bob might perform by himself the re-encryption process. Therefore, Bob has to download the encrypted data from the PHR server, decrypt the data locally using his secret key, and then re-encrypt the data using the CP-ABE scheme. The drawback of this approach is that Bob has to be online during each re-encryption process which is not very efficient both from the communication and processing point of view.

Our Contribution. To overcome the aforementioned drawbacks of the CP-ABE schemes, we propose a ciphertext policy attribute based proxy re-encryption (CP-ABPRE) scheme. In the proposed scheme Bob has to compute only once the re-encryption key $\mathsf{rk}_{p_1 \to p_2}$ which is used by a semitrusted entity called proxy (i.e. PHR server) to update all ciphertexts encrypted according to policy p_1 into ciphertexts encrypted according to policy p_2. The proxy is a semitrusted entity in the sense that it does not have access to the plain data. However it needs to perform re-encryption computations, and also has to stop performing these computations when Bob (the delegator) who generated the re-encryption key $\mathsf{rk}_{p_1 \to p_2}$ does not want to re-encrypt future ciphertexts associated with the access policy p_1. One of the distinctive features of the proposed scheme is that it is collision resistance, the feature which is lacking in almost all the proxy re-encryption schemes in the conventional public key cryptography. The collision resistance feature implies that even if the proxy and delegate collude they cannot generate a new secret key. In general, the scheme is useful for dynamic environments where the access policy which controls access to the data changes frequently (e.g. personal health record systems).

The construction of our scheme is based on prime order bilinear groups. The size of the ciphertext depends on the size of the access policy and the size of the user secret key depends on the number of attributes that the user possesses. We give a formal definition for semantic security and provide a security proof in the generic group model.

1.1 Related Work

Proxy Re-encryption. In a proxy re-encryption scheme, introduced by Mambo and Okamoto (**?**), a proxy is a semitrusted entity which can transform an encryption computed under Bobs' (delegator) public key to an encryption computed under Alices'(delegatee) public key. The proxy is a semitrusted entity i.e. it is trusted to perform only the ciphertext re-encryption, without knowing the secret keys of Bob and Alice, and without having access to the plain data. Blaze, Bleumer and Strauss (**?**) introduced the notion of "atomic proxy functions" - functions that transform ciphertext corresponding to one key into ciphertext corresponding to another key without revealing any information about the secret decryption keys or plain data. However the scheme presented in (**?**) is bidirectional where one re-encryption key can be used to transform ciphertext from the delegator to the delegatee and vice versa, and is useful only for the scenarios where the trust relationship between involved parties is mutual. To overcome this situation Jakobsson (**?**) and Zhou et al. (**?**) proposed a quorum-controlled protocol where a proxy is divided into many components. Dodis and Ivan (**?**) propose a number of unidirectional proxy re-encryption for El-Gamal, RSA and IBE scheme, where the delegator's secret key is divided into two shares: one share for the proxy and one share for the delegatee. The drawback of the proposed schemes is that they are collusion-unsafe, i.e. if the proxy and the delegatee collude then they can recover the delegator's secret key. Matsuo (**?**) and Green and Atteniese (**?**) propose identity-based proxy re-encryption scheme, where the encrypted data under the public key generated by delegators' identity is re-encrypted to an encrypted data under the public key generated by delegatees' identity.

Attribute-based Encryption. Sahai and Waters (**?**) introduce the concept of Attribute-Based Encryption (ABE) where a ciphertext and user secret key are associated with a set of attributes. ABE relies on the presence of a trusted authority (TA) who is in possession of a master key which is used to generate secret keys of users. A user can decrypt the ciphertext if the user secret key has the list of attributes specified in the ciphertext. In CP-ABE (**?**; **?**; **?**) the user secret key is associated with a set of attributes and a ciphertext is associated with an access control policy over a list of attributes. The decryptor can decrypt the ciphertext if the list of attributes associated with the secret key satisfies the access policy. In Key-Policy Attribute-Based Encryption (KP-ABE)

(**?**) the idea is reversed and the secret key is associated with an access control policy over a list of attributes and the ciphertext is associated with a list of attributes. The decryptor can decrypt the ciphertext if the list of attributes associated with the ciphertext satisfy the access policy associated with the secret key.

Attribute-based Encryption and Proxy Re-encryption. Guo et al. (**?**) propose a proxy re-encryption scheme based on the Goyal et al. (**?**) KP-ABE scheme. The proposed scheme can transform a ciphertext associated with a set of attributes into a new ciphertext associated with another set of attributes. Generally, adapting CP-ABE to proxy re-encryption is more suitable than adapting KP-ABE to proxy re-encryption since CP-ABE allows the encryptor to express her policies in the encryption phase, while in KP-ABE the access policy is associated with the secret key and is defined in the key generation phase.

Lliang et al.(**?**) proposed an attribute-based proxy re-encryption scheme. The Lliang et al. scheme is based on the Cheung and Newport CP-ABE scheme (**?**) and it inherits the same limitations that (**?**) has: it supports only access policies with AND boolean operator, and the size of the ciphertext increases linearly with the number of attributes in the system.

1.2 Organization

The remainder of this paper is organized as follows. Section **??** provides background information. In Section **??** we give a formal definition of the Ciphertext-Policy Attribute-Based Proxy Re-Encryption scheme (CP-ABPRE) and its security model. Section **??** describes the construction of the CP-ABPRE scheme. The last section concludes the paper.

2 BACKGROUND - BILINEAR GROUPS

The scheme presented in section **??** is based on pairings over groups of prime order. Let $\mathbb{G}_0$ and $\mathbb{G}_T$ be two multiplicative groups of prime order p, and let g be a generator of $\mathbb{G}_0$. A pairing (or bilinear map) $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ satisfies the following properties (**?**):

1. Bilinear: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p^*$, we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
2. Non-degenerate: $\hat{e}(g, g) \neq 1$.

$\mathbb{G}_0$ is said to be a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ can be computed efficiently. Note that the map is symmetric since $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab} = \hat{e}(g^b, g^a)$.

3 CIPHERTEXT-POLICY ATTRIBUTE-BASED PROXY RE-ENCRYPTION (CP-ABPRE)

A CP-ABPRE scheme extends CP-ABE scheme by adding a proxy component to the existing components: the trusted authority (TA) and users. Another extension has been made to the number of algorithms. CP-ABPRE uses the RKGen algorithm to generate a re-encryption key and $\mathsf{Re-Encrypt}$ algorithm to re-encrypt the ciphertext, in addition to the four algorithms of CP-ABE scheme: Setup, KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$.

Definition 1. *A CP-ABPRE scheme is a tuple of six algorithms* (Setup, KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$, RKGen, $\mathsf{Re-Encrypt}$):

- $\mathsf{Setup}(\lambda)$ run by the trusted authority (TA), the algorithm on input of the security parameter λ outputs the master secret key MK which is kept private, and the master public key PK which is distributed to users.
- $\mathsf{KeyGen}(\mathsf{MK}, \omega)$ run by the trusted authority (TA), the algorithm takes as input a set of attributes ω identifying the user, and the master secret key MK, and it outputs a user secret key SK_ω associated with the set of attributes ω.
- $\mathsf{Encrypt}(m, p_1, \mathsf{PK})$ run by the encryptor, the algorithm takes as input a message to be encrypted m, an access policy p_1 over a list of attribute which specifies which combination of attribute the decryptor needs to posses in order to obtain m, and the master public key PK. The algorithm outputs the ciphertext CT_{p_1} associated with the access policy p_1.
- $\mathsf{RKGen}(\mathsf{SK}_\omega, p_1, p_2, \mathsf{PK})$ run by the delegator, this algorithm takes as input the secret key SK_ω, the access policies p_1 and p_2, and the master public key PK. The algorithm outputs a unidirectional re-encryption key $\mathsf{rk}_{p_1 \to p_2}$ if SK_ω satisfies p_1, or an error symbol $\perp$ if ω does not satisfy p_1.
- $\mathsf{Re-Encrypt}(\mathsf{CT}_{p_1}, \mathsf{rk}_{p_1 \to p_2})$ run by the proxy, this algorithm takes as input the ciphertext CT_{p_1} and the re-encryption key $\mathsf{rk}_{p_1 \to p_2}$, and outputs the ciphertext CT_{p_2} associated with the access policy p_2.

- $\mathsf{Decrypt}(\mathsf{CT}_{p_i}, \mathsf{SK}_\omega)$ run by the decryptor, the algorithm takes as input the ciphertext C_{p_i} and the secret key SK_ω, and output a message m if ω satisfies p_i, or an error symbol $\perp$ if ω does not satisfy p_i .

Security Model. In the following we present the game-based security definition (security model) of the CP-ABPRE scheme. Informally, the security model guarantees that: a) an user (adversary) who does not have enough attributes to satisfy the access policy p^* of the ciphertext cannot learn any information about the plaintext being encrypted, b) two users cannot combine their attributes to extend their decryption power, for instance two users cannot combine their secret keys and decrypt a ciphertext associated with p^* if none of users secret keys satisfy p^*, and c) the proxy and an user cannot combine the re-encryption key and the secret key in order to compute a new secret key. Therefore in the security game, played between the adversary $\mathcal{A}$ and the challenger (the challenger simulates the game and answers $\mathcal{A}$'s queries) we allow $\mathcal{A}$ to compromise users secret key except the secret keys which satisfy the challenge access policy p^*. In addition, $\mathcal{A}$ is allowed also to compromise proxy keys or re-encryption keys with the following restriction:

- $\mathcal{A}$ is not allowed to ask secret key queries for the attribute set ω which satisfies p_2 if $\mathcal{A}$ has a re-encryption key $\mathsf{rk}_{p^* \to p_2}$. The reason for this restriction is that $\mathcal{A}$ can use the re-encryption key to re-encrypt the challenge ciphertext associated with p^* to a ciphertext associated with p_2 and decrypt the re-encrypted ciphertext using his secret key which satisfies p_2. In the sequel we will refer to p_2 as a challenge derivative access policy if $\mathcal{A}$ has the re-encryption key $\mathsf{rk}_{p^* \to p_2}$.

At one point of the security game $\mathcal{A}$ gives to the challenger two messages and the challenge access policy p^*, and the challenger return to $\mathcal{A}$ a ciphertext of one of the two messages encrypted under p^*. $\mathcal{A}$ has to guess which of the messages was encrypted. If the guess is correct, then $\mathcal{A}$ wins the game. Formally the security game is defined as follows:

1. Setup. The challenger run $\mathsf{Setup}(\lambda)$ to generate $(\mathsf{PK}, \mathsf{MK})$, and gives PK to $\mathcal{A}$.
2. Phase1. $\mathcal{A}$ performs a polynomially bounded number of queries:
   - $\mathsf{Keygen}(\omega_j)$. $\mathcal{A}$ asks for a user secret key for any attribute set ω_j. The challenger returns SK_{ω_j} to $\mathcal{A}$.
   - $\mathsf{RKGen}(p_1, p_2)$. $\mathcal{A}$ asks for a re-encryption key for $\mathsf{rk}_{p_1 \to p_2}$, where $p_1 \neq p_2$. The challenger runs $\mathsf{SK}_\omega = \mathsf{Keygen}(\omega_j)$ such that SK_ω satisfies p_1, and returns $\mathsf{rk}_{p_1 \to p_2}$ to $\mathcal{A}$.
3. Challenge. $\mathcal{A}$ sends to the challenger two messages m_0, m_1 and the challenge access policy p^* . $\mathcal{A}$ is not allowed to chose a challenge access structure p^* if it has made the following queries in Phase1:
   - $\mathsf{Keygen}(\omega_j)$ queries such that SK_{ω_j} satisfies a challenge access structure p^* .
   - $\mathsf{Keygen}(\omega_j)$ queries such that SK_{ω_j} satisfies any challenge derivative access policies.
   - $\mathsf{RKGen}(p_1, p_2)$ queries if $\mathcal{A}$ previously has issued $\mathsf{Keygen}(\omega_j)$ such that SK_{ω_j} satisfies p_2 and p_1 is a challenge derivative access policy.

   The challenger selects $b \in_R (0,1)$ and returns $\mathsf{CT}_{p^*} = \mathsf{Encrypt}(m_b, p^*, \mathsf{PK})$.
4. Phase2. $\mathcal{A}$ can continue querying Keygen and RKGen. $\mathcal{A}$ is not allowed to make queries specified in the Challenge phase.
5. Guess. $\mathcal{A}$ outputs a guess b', where $b' \in (0,1)$.

Definition 2. *A CP-ABPRE scheme is said to be secure against adaptive chosen plaintext attack (IND-CPA) if any polynomial-time adversary $\mathcal{A}$ has only a negligible advantage in the CP-ABPRE game, where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

4 CONSTRUCTION OF CP-ABPRE SCHEME

Before introducing the scheme, we briefly explain the structure of the access policy associated with the ciphertext. In our scheme an access control policy is a monotonic boolean formula of conjunction and disjunctions of attributes. The TA in the Setup phase defines the universe of all attributes Ω. An example of the universe of all attribute can be $\Omega = \{A, B, C, D, F\}$, and an example of an access policy can be $p_1 =$(A $\wedge$ B) $\vee$ (C $\wedge$ D) where $\{A, B, C, D\} \in \Omega$.

Assigning Values to Attributes in the Access Policy. To enforce the access policy in such a way that only users who satisfy the access policy can decrypt the ciphertext, in the encryption phase, the encryptor encrypts the data according to the access policy. Therefore, the encryptor in the encryption phase picks a secret value s and shares it according to the access policy under which the data is encrypted. We use Benaloh and Leichter (**?**) se-

cret sharing scheme to share s. The scheme (**?**) works as follows:

- Transforms an access policy p_1 into an access tree τ and set the value of the root node of τ to be s. Then, recursively for each non-leaf node do the following:
  - If the symbol is $\vee$, set the values of each child node to be s.
  - If the symbol is $\wedge$, for each child node, except the last one, assign a random value s_i where $1 \leq s_i \leq p-1$, and to the last child node assigns $s_t = s - \sum_{i=1}^{t-1} s_i \mod p$.

For example, to share s according to the access policy p_1 =(A $\wedge$ B) $\vee$ (C $\wedge$ D), the Benaloh and Leichter (**?**) secret sharing scheme works as follows: a) assign s to OR ($\vee$) operator, b) assign s to two AND ($\wedge$) operators and c) assign shares s_A to A, s_B to B, s_C to C and s_D to D, such that $s = s_A + s_B$ and $s = s_C + s_D$.

Policy Evaluation. To decrypt a ciphertext, a user secret key SK_ω associated with a set of attributes ω has to satisfy the policy p_1 =(A $\wedge$ B) $\vee$ (C $\wedge$ D) associated with the ciphertext. In the example, if $\omega = \{A, B\}$ then the policy is satisfied since $s = s_A + s_B$. This can be verified by substituting the attributes in $\omega \cap p_1 = \{A, B\}$ (attributes which appear in ω and p_1) by true, and attributes in $p_1 \setminus \omega = \{C, D\}$ (attributes which appear in p_1 but not appear in ω) by false. We say that the user satisfies the policy if p_1 =(true $\wedge$ true) $\vee$ (false $\wedge$ false) evaluates to true.

4.1 The Scheme

In this section we describe the construction of the proposed CP-ABPRE scheme. The scheme consists of the following algorithms:

1. $\mathsf{Setup}(\lambda)$. The setup algorithm selects a bilinear group $\mathbb{G}_0$ of prime order p and generator g, and the bilinear map $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$. Next to this, the algorithm generates the list of attributes in the system $\Omega = \{a_1, a_2, ..., a_k\}$, picks randomly $\alpha, \beta, f, x_1, x_2, \cdots, x_k \in \mathbb{Z}_p^*$, and sets $T_j = g^{x_j}$ $(1 \leq j \leq k)$. Note that for each $a_j \in \Omega$ $(1 \leq j \leq k)$ there is an $x_j \in \mathbb{Z}_p^* (1 \leq j \leq k)$. The algorithm also defines the function $H_1 : \mathbb{G}_T \to \mathbb{G}_0$. The public key is published as:

   $$\mathsf{PK} = (g, \hat{e}(g,g)^{(\alpha+\beta)}, g^f, \{T_j\}_{j=1}^k, H_1).$$

   The master secret key consists of the following components:

   $$\mathsf{MK} = (\alpha, \beta, f, \{x_j\}_{j=1}^k).$$

2. $\mathsf{KeyGeneration}(\mathsf{MK}, \omega)$. The key generation algorithm takes as input the attribute set ω which characterize the user. For each user the algorithm picks at random $r \in \mathbb{Z}_p^*$ and computes the secret key SK_ω which consists of the following components:

   $$\begin{aligned} \mathsf{SK}_\omega = (D^{(1)} &= g^{\alpha - r}, \\ \{D_j^{(2)} &= g^{\frac{r+\beta}{x_j}}\}_{a_j \in \omega}). \end{aligned}$$

3. $\mathsf{Encryption}(m, p_1, \mathsf{PK})$. To encrypt a message $m \in \mathbb{G}_T$, under the access policy p_1 over the set of attributes from Ω, the encryption algorithm picks at random $s \in \mathbb{Z}_p^*$ and assigns s_i values to attributes in p_1 (s_i values are shares of s and are generated using the Benaloh and Leichter (**?**) secret sharing scheme). The resulted ciphertext consists of the following components:

   $$\begin{aligned} \mathsf{CT}_{p_1} = (C^{(1)} &= g^s \\ C^{(2)} &= m \cdot \hat{e}(g,g)^{(\alpha+\beta)s}, C^{(3)} = g^{fs}, \\ \{C_{j,i}^{(4)} &= g^{x_j s_i}\}_{a_j \in p_1}). \end{aligned}$$

4. $\mathsf{RKGen}(\mathsf{SK}_\omega, p_1, p_2, \mathsf{PK})$: The algorithm outputs a re-encryption key which is used by the proxy to update the ciphertext associated with p_1 to a ciphertext associated with p_2. Let $\omega' \subseteq \omega$ be the smallest set which satisfies the access policy p_1. The algorithm first parses SK_ω as $(\mathsf{D}^{(1)}, \{\mathsf{D}_j^{(2)}\}_{a_j \in \omega})$, picks at random $l, x' \in \mathbb{Z}_p^*$, it sets $(g^f)^{x'} = g^x$ and computes the re-ecnryption key $\mathsf{rk}_{p_1 \to p_2}$ which consists of the following components:

   $$\begin{aligned} \mathsf{rk}_{p_1 \to p_2} = (\hat{D^{(1)}} &= D^{(1)} \cdot g^l, \\ \hat{D^{(2)}} &= \mathsf{Encryption}(g^{x-l}, p_2, \mathsf{PK}), \\ \hat{D^{(3)}} &= g^{x'} = g^{\frac{x}{f}}, \\ \hat{D_j^{(4)}} &= \{D_j^{(2)}\}_{a_j \in \omega'}. \end{aligned}$$

   Note. Note that the message g^{x-l} encrypted in this phase belongs to the group $\mathbb{G}_0$, while the message m encrypted in the Encryption phase belongs to the group $\mathbb{G}_T$. The encryption of g^{x-l} is done in the same way as the encryption of m with a small change on the computation of $C^{(2)}$. The only purpose for this change is to keep g^{x-l} in group $\mathbb{G}_0$. So, in encrypting m in the Encryption phase the $C^{(2)}$ had the form:

   $$C^{(2)} = m \cdot \hat{e}(g,g)^{(\alpha+\beta)s}$$

for a random $s \in \mathbb{Z}_p^*$. In encrypting g^{x-l} in the RKGen phase the $C^{(2)}$ has the form:

$$C^{(2)} = g^{x-l} \cdot H_1(\hat{e}(g,g)^{(\alpha+\beta)z})$$

where z is a random element in $\mathbb{Z}_p^*$. All the other components are computed in the same way as in the Encryption phase.

5. $\mathsf{Re-Encrypt}(\mathsf{CT}_{p_1}, \mathsf{RK}_{p_1 \to p_2})$. The algorithm parses CT_{p_1} as $(\mathsf{C}^{(1)}, \mathsf{C}^{(2)}, \mathsf{C}^{(3)}, \{\mathsf{C}_{j,i}^{(4)}\}_{a_{j,i} \in p_1})$, and $\mathsf{RK}_{p_1 \to p_2}$ as $(\mathsf{D}^{\hat{(1)}}, \mathsf{D}^{\hat{(2)}}, \mathsf{D}^{\hat{(3)}}, \{\mathsf{D}_j^{\hat{(4)}}\}_{a_j \in \omega'})$, and computes the following:

   (a) In the first step, for every attribute $a_j \in \omega'$, it computes the following:

$$\begin{aligned} I^{(1)} &= \prod_{a_j \in \omega'} \hat{e}(\hat{D}_j^{(4)}, C_{j,i}^{(4)}) = \prod_{a_j \in \omega'} \hat{e}(g^{\frac{r+\beta}{x_j}}, g^{x_j s_i}) \\ &= \hat{e}(g^{r+\beta}, g^s) \end{aligned}$$

   (b) In the second step it computes the following:

$$\begin{aligned} I^{(2)} &= \hat{e}(C^{(1)}, D^{\hat{(1)}}) \cdot I^{(1)} \\ &= \hat{e}(g^s, g^{\alpha-r} \cdot g^l) \cdot \hat{e}(g,g)^{(r+\beta)s} \\ &= \hat{e}(g^s, g^{\alpha+\beta} \cdot g^l) \end{aligned}$$

   (c) In the third step it computes the following:

$$\begin{aligned} I^{(3)} &= \frac{C^{(2)}}{I^{(2)}} = \frac{m \cdot \hat{e}(g^s, g^{\alpha+\beta})}{\hat{e}(g^s, g^{\alpha+\beta} \cdot g^l)} \\ &= \frac{m}{\hat{e}(g^s, g^l)} \end{aligned}$$

$$\begin{aligned} \hat{C}^{(2)} &= \hat{e}(C^{(3)}, D^{\hat{(3)}}) \cdot I^{(3)} \\ &= \hat{e}(g^{sf}, g^{\frac{x}{f}}) \cdot \frac{m}{\hat{e}(g^s, g^l)} \\ &= m \cdot \hat{e}(g^s, g^{x-l}) \end{aligned}$$

   (d) In the fourth step it sets:

$$C^{\hat{(1)}} = C^{(1)}.$$
$$C^{\hat{(3)}} = D^{\hat{(2)}}.$$

   The algorithm outputs the re-encrypted ciphertext, which consists of the following components:

$$\mathsf{CT}_{p_2} = (C^{\hat{(1)}}, C^{\hat{(2)}}, C^{\hat{(3)}}).$$

6. $\mathsf{Decrypt}(\mathsf{CT}_{p_i}, \mathsf{SK}_\omega)$: The decryption algorithm takes as input the ciphertext C_{p_i} and secret key SK_ω. It checks if the secret key SK_ω related to the attribute set ω satisfies the access policy p_i. If not, then it outputs $\perp$.

   (a) If ω satisfies the access policy p_i and C_{p_i} is a regular ciphertext, then the decryption algorithm performs the following:

      i. In the first step, the algorithm chooses the smallest set $\omega' \subseteq \omega$ which satisfies the access policy p_i and parses C_{p_i} as $(C^{(1)}, C^{(2)}, \{C_{j,i}^{(4)}\}_{a_j \in p_i})$, and SK_ω as $(D^{(1)}, \{\mathsf{D}_j^{(2)}\}_{a_j \in \omega})$.

      ii. In the second step, for every attribute $a_j \in \omega'$, it computes

$$\begin{aligned} Z^{(1)} &= \prod_{a_j \in \omega'} \hat{e}(D_j^{(2)}, C_{j,i}^{(4)}) \\ &= \prod_{a_j \in \omega'} \hat{e}(g^{\frac{r+\beta}{x_j}}, g^{x_j s_i}) \\ &= \hat{e}(g^{r+\beta}, g^s) \end{aligned}$$

      iii. In the third step, it computes

$$\begin{aligned} Z^{(2)} &= \hat{e}(D^{(1)}, C^{(1)}) \cdot Z^{(1)} \\ &= \hat{e}(g^{\alpha-r}, g^s) \cdot \hat{e}(g^{r+\beta}, g^s) \\ &= \hat{e}(g,g)^{(\alpha+\beta)s} \end{aligned}$$

      iv. In the final step, the message is obtained by computing

$$m = \frac{C^{(2)}}{Z^{(2)}}$$

   (b) If ω satisfies the access policy p_i and C_{p_i} is a re-encrypted ciphertext, then the decryption algorithm performs the following:

      i. In the first step it parses C_{p_i} as $(C^{\hat{(1)}}, C^{\hat{(2)}}, C^{\hat{(3)}})$

      ii. In the second step it recovers the message in the following way:

$$m = \frac{C^{\hat{(2)}}}{\hat{e}(C^{\hat{(1)}}, \mathsf{Decrypt}(C^{\hat{(3)}}, \mathsf{SK}_\omega))}$$

      Note. The operation $\mathsf{Decrypt}(C^{\hat{(3)}}, \mathsf{SK}_\omega) = g^{x-l}$ (where g^{x-l} is part of the group $\mathbb{G}_0$) is done in similar way as $\mathsf{Decrypt}(C_{p_i}, \mathsf{SK}_\omega) = m$ (where m is part of the group $\mathbb{G}_T$) explained under (a). The only change is under (iv) where g^{x-l} is computed as:

$$g^{x-l} = \frac{C^{(2)}}{H_1(Z^{(2)})}$$

      while m was computed as:

$$m = \frac{C^{(2)}}{Z^{(2)}}$$

In the following, we presents the properties of our proposed scheme:

- **Uni-directional.** The re-encryption key $\mathsf{rk}_{p_1 \to p_2}$ only allows the proxy to re-encrypt ciphertexts encrypted under the policy p_1 into ciphertexts encrypted under policy p_2, and not the other way around. For instance, the re-encryption key $\mathsf{rk}_{p_1 \to p_2}$ can be used to re-encrypt ciphertexts associated with a policy $p_1 = [Patient\ AND\ Bob]$ into ciphertext associated with a policy $p_2 = [General\ Practitioner\ (GP)]$. The idea is that a GP should access his patients' health data, however individual patients should not be able to access GPs' data since GP possess data from different patients.
- **Non-interactive.** The re-encryption key $\mathsf{rk}_{p_1 \to p_2}$ is computed by the delegator without any interaction with the delegatee, the TA authority or the proxy. To compute $\mathsf{rk}_{p_1 \to p_2}$, the delegator uses his secret key and the master public key. Therefore the delegator remains off-line while computing the re-encryption key and the proxy perform re-encryption process to update (or re-encrypt) ciphertext without any interaction with the delegator.
- **Key Optimal.** The delegator and the delegatee don't need to store extra secrets in addition to their original secret keys associated with a set of attributes, regardless of how many delegations he/she gives (or accepts).
- **Non-transitivity.** The proxy cannot re-delegate the decryption rights. Alternatively it can be said that the proxy cannot combine re-encryption keys to create new delegations. For example, proxy cannot construct a re-encryption key $\mathsf{rk}_{p_1 \to p_3}$ from other two re-encryption keys $\mathsf{rk}_{p_1 \to p_2}$ and $\mathsf{rk}_{p_2 \to p_3}$ under it possession.
- **Collusion Safe.** The proxy and a cannot combine their secrets in order to derive a new secret key. For example, the proxy should not be able to combine the re-encryption key $\mathsf{rk}_{p_1 \to p_2}$ where $p_1 = [GP\ AND\ Hospital\ 1]$ and $p_2 = [GP\ AND\ (Hospital\ 1\ OR\ Hospital\ 2)]$ with delegatee's who has a secret key associated with attributes $\{GP, Hospital\ 2\}$ in order to compute a delegator's secret key which is associated with the attributes $\{GP, Hospital\ 1\}$. Collusion safeness also implies that two users cannot combine their secret keys in order to extend their decryption power. For instance, a user, Alice who has a secret key associated with attributes $\{\mathsf{Nurse}, \mathsf{Hospital}\ 1\}$ should not be able to combine her secret key with a user, Charlie who has a secret key associated with the attributes $\{GP, Hospital\ 2\}$ and be able to decrypt a ciphertext encrypted under the policy $p = [Nurse\ AND\ Hospital\ 2]$ which cannot be satisfied neither by Alice nor by Charlie.
- **Multi-user Decryption.** In existing proxy re-encryption, once the proxy performs the re-encryption, the delegator losses the decryption power, thus the delegator cannot use his secret key to decrypt the re-encrypted data. The reason is that the mapping ciphertext-public key is one-to-one, which implies that one ciphertext can be decrypted only by one secret key, thus after the re-encryption is performed only the delegatee has a power to decrypt the ciphertext. One can argue that the proxy can keep a copy of the original ciphertext and enable the delegator to decrypt the original ciphertext. However, this solution requires for the proxy to keep the original ciphertext for each re-encrypted data.

  CP-ABPRE scheme has a property which allows the delegator to generate a re-encryption key in such a way that that the delegator does not loose his decryption power after the proxy performs the re-encryption, and the re-encrypted ciphertext can be decrypted by many users whose secret key satisfies the access policy. As an example, suppose there is an encrypted data according to the policy $p_1 = [(A\ AND\ B)\ OR\ (C\ AND\ D)]$. Bob has a secret key $\mathsf{SK}_{\omega_{Bob}}$ associated with a set of attributes $\omega_{Bob} = \{A, B, F\}$. Since Bob satisfy the access policy p_1, Bob is capable to compute a re-encryption key that can update the access policy p_1 into another policy p_2. If Bob updates the access policy p_1 into p_2, where $p_2 = [C\ AND\ F]$ then Bob looses his decryption power because Bob does not satisfy the access policy p_2. However, Bob can retain his decryption power by creating a policy $\widetilde{p} = p_1\ \mathsf{OR}\ p_2$.
- **Multi-user & Single-user Delegation.** In CP-ABE schemes many users may have a secret key with an attribute sets that may satisfy access policy associated with ciphertext. Hence many users can compute the re-encryption key as they atisfy the access policy. However, this property may not always be of potential interest and might become a security threat in some scenarios. In practice this threat can be overcomed by defining attributes that are unique to an individual, in addition to the attributes that may be possessed by multiple users. For example, consider Alice who has a secret key $\mathsf{SK}_{\mathsf{Alice}_\omega}$ associated with a set of attributes $\omega = \{Alice, Patient\}$ (Alice is an individual attribute which can be possessed solely by Alice and Patient is an attribute which can be possessed by

many users), and a ciphertext encrypted under an access policy $p_1 = [Alice\ AND\ Patient]$. It is obvious that only Alice satisfies the access policy p_1 and only Alice can compute the re-encryption key $\mathsf{rk}_{p_1 \to p_2}$, for any p_2.

4.2 Efficiency

The size of the secret key SK_ω depends on the number of attributes the user possess and consists of $|\omega| + 1$ group elements in $\mathbb{G}_0$, where $|\omega|$ is the cardinality of ω. The size of the ciphertext C_p depends on the size of the access policy p_1 and has $|p| + 1$ group elements in $\mathbb{G}_0$, and 1 group element in $\mathbb{G}_T$. The size of the re-encryption key $\mathsf{rk}_{p_1 \to p_2}$ depends on ω' which is the smallest set which satisfies p_1 and has $|\omega'| + 1$ group elements in $\mathbb{G}_0$.

5 CONCLUSIONS AND FUTURE WORK

In this work we present a new proxy re-encryption scheme in the CP-ABE setting. The scheme is unidirectional and allows a user (the delegator) to change dynamically the access policy associated with the ciphertext, without necessarily decrypting the ciphertext. To reduce computations performed at the delegators' side and to avoid the need for the delegator to be online all the time, the delegator computes a re-encryption key and delegates the power to the proxy to update the access control policy associated with ciphertext.

There are two interesting open problems. First, it would be interesting to hide the access control policy from the semi-trusted proxy and from the user who decrypts the data since in our scheme the access policy has to be in clear in order for the user who decrypts the data to apply the right attributes to satisfy the access policy associated with the ciphertext. Second, we leave as an open problem to provide a security proof in the standard model where the problem of breaking the scheme is reduced to a well-studied complexity-theoretic problem.

REFERENCES

Benaloh, J. and Leichter, J. (1995). Generalized secret sharing and monotone functions. In S.Goldwasser , editor, *Proceedings of Eurocrypt 1998*, volume 403 of *LNCS*, pages 27–35. Springer-Verlag, 1995.

Bethencourt, J. and Sahai, A. and Waters, B. Ciphertext-policy attribute-based encryption. In D. Shands, editor, *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Washington, DC, USA, 2007.

Blaze, M. and Bleumer, G., and Strauss, M. Divertible Protocols and Atomic Proxy Cryptography. In K Nyberg, editor, *Proceedings of Eurocrypt 1998*, volume 1403 of *LNCS*, pages 127–144. Springer-Verlag, 1998.

Boneh, D. and Franklin, M. Identity-based encryption from the weil pairing. In J. Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Heidelberg, 2001.

Cheung, L. and Newport, C. Provably secure ciphertext policy ABE. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 456–465. ACM, 2007.

ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

Goyal, V. and Pandey, O. and Sahai, A. and Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

Green, M. and Ateniese, G. Identity-based proxy re-encryption. In J. Katz and M. Yung, editors, *Proceedings of Applied Cryptography and Network Security*, volume 4521 of *LNCS*, pages 288–306. Springer-Heidelberg, 2007.

Guo, S. and Zeng, Y. and Wei, J. and Xu, Q. Attribute-based re-encryption scheme in the standard model. *Wuhan University Journal of Natural Sciences*, 13(5):621–625, 2008.

Ibraimi, L. and Tang, Q. and Hartel, P. and Jonker, W. Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In F. Bao, H. Li, and G. Wang, editors, *Proceedings of Information Security Practice and Experience*, volume 5451 of *LNCS*, pages 1–12. Springer-Heidelberg, 2009.

Ivan, A. and Dodis, Y. Proxy Cryptography Revisited. In *Proceedings of the Network and Distributed System Security Symposium*. The Internet Society, 2003.

Jakobsson, M. On quorum controlled asymmetric proxy re-encryption. In H. Imai and Y. Zheng, editors, *Proceedings of Public Key Cryptography*, volume 1560 of *LNCS*, pages 112–121. Springer-Heidelberg, 1999.

Liang, X. and Cao, Z. and Lin, H. and Shao, J. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 276–286. ACM, 2009.

Mambo, M. and Okamoto, E. Proxy cryptosystems: delegation of the power to decrypt ciphertexts. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 80(1):54–63, 1997.

Matsuo, T. Proxy Re-encryption Systems for Identity-Based Encryption. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Proceedings of*

Pairing 2007, volume 4575 of *LNCS*, pages 247–267. Springer-Heidelberg, 2007.

Rivest, R. L. and Shamir, A. and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):126, 1978.

Sahai, A. and Waters, B. Fuzzy identity-based encryption. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 457–473. Springer-Heidelberg, 2005.

Shoup, V. Lower Bounds for Discrete Logarithms and Related Problems. In F. Walter, editor, *Proceedings of Eurocrypt 1997*.

Zhou, L. and Marsh, M. A. and Schneider, F. B. and Redz, A. Distributed blinding for ElGamal re-encryption. In *Proceedings of 25th IEEE International Conference on Distributed Computing Systems*, pages 815–824. IEEE Computer Society, 2005.

APPENDIX

Security Proof in Generic Group Model. We provide a security proof in the generic group model, introduced by Shoup (**?**). The model relies on the fact that it is hard to find the discrete logarithm in a group (including a group with bilinear pairing) when the order of the group is a large prime number. In this model group elements are encoded as unique random strings, in such a way that the adversary $\mathcal{A}$ can manipulate group elements using canonical group operations in $\mathbb{G}_0$ and $\mathbb{G}_T$ and cannot test any property other than equality. Thus a cryptographically secure group provides no mathematical properties of its group other than its group structure.

Theorem 1. *The advantage of any adversary $\mathcal{A}$ in the security game receiving at most q group elements from queries it makes to the oracles for computing group operation in $\mathbb{G}_0$ and $\mathbb{G}_T$, pairing operation $\hat{e}$ and from the interaction with the CP-ABPRE security game is bounded by $O(\frac{q^2}{p})$.*

Proof. Following the arguments from the proof in (**?**), we bound the advantage of $\mathcal{A}$ in a modified game in which the challenge ciphertext is either $C^{(1)} = \hat{e}(g,g)^{(\alpha+\beta)s}$ or $C^{(1)} = \hat{e}(g,g)^{\theta}$, instead of giving a challenge ciphertext as defined in the security game of Section **??** as $C^{(1)} = m_b \cdot \hat{e}(g,g)^{(\alpha+\beta)s}$ where $b \in (0,1)$. We show that $\mathcal{A}$ cannot distinguish which game is playing. Then we show that there is no $\mathcal{A}$ which has a non-negligible advantage in a modified game, so there is no $\mathcal{A}$ with has a non-negligible advantage in the security game of Section **??**, either. Note that if there is an $\mathcal{A}$ that has advantage ε in the security game of Section **??** then there can be another adversary which has advantage $\frac{\varepsilon}{2}$ in the modified security game.

We will write $\gamma_0(x) : \mathbb{Z}_p^* \to \{0,1\}^{\lceil \log p \rceil}$ as a random encoding for the group element $g^x \in \mathbb{G}_0$, and $\gamma_1(x) : \mathbb{Z}_p^* \to \{0,1\}^{\lceil \log p \rceil}$ as a random encoding for group element $\hat{e}(g,g)^x \in \mathbb{G}_T$. Each random encoding is associated with a rational function (a function written as a division of two polynomial functions). Let f be a rational function over the variables $\{\alpha, \beta, \theta, s, s_{\hat{i}}, \{x_j\}(1 \le j \le k), r, f, l\}$, where each variable is an element picked at random in the scheme. $\mathcal{A}$ receives the following encodings from the interaction with the simulator in the security game:

- Components generated by the Setup algorithm:
  1. $\gamma_0(1)$ representing the group generator g.
  2. $\gamma_0(f)$ representing the group element g^f.
  3. $\{\gamma_0(x_j)\}(1 \le j \le k)$ representing $\{T_j = g^{x_j}\}_{j=1}^{k}$.
  4. $\gamma_1(\alpha+\beta)$ representing $\hat{e}(g,g)^{\alpha+\beta}$.
- Components generated by the KeyGen oracle in Phase1 and Phase2 of the security game. Let ω be the attribute set for which $\mathcal{A}$ asks for e secret key.
  1. $\gamma_0(\alpha - r)$ representing $D^{(1)} = g^{\alpha - r}$.
  2. $\{\gamma_0(\frac{r+\beta}{x_j})\}_{a_j \in \omega}$ representing $\{D_j^{(2)} = g^{\frac{r+\beta}{x_j}}\}_{a_j \in \omega}$.
- Components generated by the RKGen oracle in Phase1 and Phase2 of the security game. Let $\mathsf{RKGen}(p_1, p_2)$ be the re-encryption query used to re-encrypt messages encrypted under the access policy p_1 into messages encrypted under the access policy p_2. Let ω' be the set of attributes that satisfy the access policy p_1.
  1. $\gamma_0(\alpha - r + l)$ representing $\hat{D^{(1)}} = g^{\alpha - r + l}$.
  2. $\gamma_0(z)$, $\gamma_0(R)$, $\gamma_0(fz)$ and $\{\gamma_0(x_j z_{\hat{i}})\}_{a_{j,\hat{i}} \in p_2}$ representing $\hat{D_j^{(2)}} = \mathsf{Encryption}(g^{x-l}, p_2, \mathsf{PK})$.
  3. $\gamma_0(x')$ representing $\hat{D^{(3)}} = g^{x'} = g^{\frac{x}{f}}$.
  4. $\{\gamma_0(\frac{r+\beta}{x_j})\}_{a_j \in \omega}$ representing $\{\hat{D_j^{(4)}} = g^{\frac{r+\beta}{x_j}}\}_{a_j \in \omega'}$.
- Components generated by the Encryption oracle in the Challenge phase of the security game. Let $\mathcal{A}$ asks for a challenge for messages $m_0, m_1 \in \mathbb{G}_T$ and the access policy p^*.
  1. $\gamma_0(s)$ representing $C^{(1)} = g^s$.
  2. $\gamma_1(\theta)$ representing $C^{(2)} = \hat{e}(g,g)^{\theta}$.
  3. $\gamma_0(fs)$ representing $C^{(3)} = g^{fs}$.

4. $\{\gamma_0(x_j s_{\hat{i}})\}_{a_{j,\hat{i}} \in p^*}$ representing $\{C^{(4)}_{j,\hat{i}} = g^{x_j s_{\hat{i}}}\}_{a_{j,\hat{i}} \in p^*}$.

$\mathcal{A}$ uses the group elements received from the interaction with the simulator to perform generic group operations and equality tests.

- Queries to the oracles for group operation in $\mathbb{G}_0$ and $\mathbb{G}_T$. $\mathcal{A}$ asks for multiplying or dividing group elements represented with their random encodings, and associated with a rational function. The oracle returns $f + f'$ when $\mathcal{A}$ asks for multiplying f and f', or $f - f'$ when $\mathcal{A}$ asks for dividing f and f' (Note that $\mathcal{A}$ knows only the encodings of f and f').
- Queries to the oracle for computing pairing operation $\hat{e}$. $\mathcal{A}$ asks for pairing of group elements represented with their random encoding and associated with a rational function. The oracle returns ff' when $\mathcal{A}$ asks for pairing f and f'.

We show that $\mathcal{A}$ cannot distinguish with non-negligible advantage the simulation of the modified game where the challenge ciphertext is set $C^{(2)} = \hat{e}(g,g)^{\theta}$, with the simulation of the real game where the challenge ciphertext would have been set $C^{(2)} = \hat{e}(g,g)^{(\alpha+\beta)s}$.

First, we show the $\mathcal{A}$'s view when the challenge ciphertext is $\gamma_1(\theta)$. Following the standard approach for security in generic group model, $\mathcal{A}$'s view can change when an unexpected collision happen due to the random choice of the formal variables $\{\alpha, \beta, \theta, s, s_{\hat{i}}, \{x_j\}_{1 \le j \le k}, r, f, l\}$ chosen uniformly from $\mathbb{Z}_p^*$. A collusion happen when two queries evaluate to the same value. For any two distinct queries the probability of such collusion happen is at most $O(q^2/p)$. Since for large p the probability of such collusion is negligible we ignore this case.

Second, we show what the adversaries view would have been if the challenge ciphertext had been set $\gamma_1((\alpha+\beta)s)$. Again, $\mathcal{A}$ view can change when a collusion happen, such that the values of two different rational functions coincide. We show that $\mathcal{A}$ cannot make a polynomial query which would be equal to $(\alpha+\beta)s$, and therefore a collusion cannot happen. In table **??** we list possible queries that $\mathcal{A}$ can make into $\mathbb{G}_T$ using the group elements received from interaction with the simulator in the security game.

As is shown in table **??** (the highlighted cell), $\mathcal{A}$ can pair s with $\alpha - r$, and $\frac{r+\beta}{x_j}$ with $s_i x_j$, and then sum the results to get $s(\alpha - r) + \sum_{a_i \in \omega} r s_i + \sum_{a_i \in \omega} \beta s_i$. In order to get only $(\alpha+\beta)s$, $\mathcal{A}$ has to create polynomial requests to cancel sr and to compute βs. We observe that $\mathcal{A}$ to obtain βs and sr has to pair $\frac{r+\beta}{x_j}$ with $s_{\hat{i}} x_j$. From the table **??** we can see that $\mathcal{A}$ can construct a

Table 1: Possible queries into $\mathbb{G}_T$.

1	$\alpha+\beta$	t_j
$(\alpha-r)s$	$(r+\beta)s_i$	$\frac{r+\beta}{x_j}s$
fz	xs	x
$s(\alpha-r)+(r+\beta)s_i$	$r+\beta$	$(r+\beta)s_i$
$x_j s_i$	$(\alpha-r)(x_j s_i)$	z
$\alpha - r \pm (r+\beta)s_i$	$s(\alpha-r+l)$	R
$(\alpha+\beta) \pm s$	$(\alpha-r+l)$	

query polynomial of the form:

$$\underbrace{s\alpha}_{A} - \underbrace{sr}_{B} + \sum_{a_i \in \omega} \underbrace{r s_i}_{C} + \sum_{a_i \in \omega} \underbrace{\beta s_i}_{D}$$

However $\mathcal{A}$ cannot construct a query polynomial of the form $(\alpha+\beta)s = \alpha s + \beta s$ if $\mathcal{A}$ does not have a secret key which satisfies the access policy. First, there must be at least one rs_i missing (there must be one ciphertext component $g^{x_j s_i}$ for which $\mathcal{A}$ does not have a secret key component $g^{\frac{\beta+r}{x_j}}$ to pair, therefore $\mathcal{A}$ cannot cancel x_j), therefore $\mathcal{A}$ cannot reconstruct rs under the term C, and as a sequence cannot cancel term B and C. Second, there must be at least one βs_i missing, hence $\mathcal{A}$ cannot reconstruct βs under the term D. As a result of the above analysis, we conclude that $\mathcal{A}$ cannot make a polynomial query which has the form $(\alpha+\beta)s$.