Attribute-based Authenticated Key Exchange

M. Choudary Gorantla and Colin Boyd and Juan Manuel González Nieto

Information Security Institute, Faculty of IT, Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia.
Email: mc.gorantla@isi.qut.edu.au, {c.boyd,j.gonzaleznieto}@qut.edu.au

Abstract. We introduce the concept of attribute-based authenticated key exchange (AB-AKE) within the framework of ciphertext-policy attribute-based systems. A notion of AKE-security for AB-AKE is presented based on the security models for group key exchange protocols and also taking into account the security requirements generally considered in ciphertext-policy attribute-based cryptosystems. We also introduce a new primitive called encapsulation policy attribute-based key encapsulation mechanism (EP-AB-KEM) and then define a notion of chosen ciphertext security for EP-AB-KEMs. A generic one-round AB-AKE protocol that satisfies our AKE-security notion is then presented. The protocol is generically constructed from any EP-AB-KEM that satisfies chosen ciphertext security. Finally, we propose an EP-AB-KEM from an existing attribute-based encryption scheme and show that it achieves chosen ciphertext security in the generic group and random oracle models. Instantiating our AB-AKE protocol with this EP-AB-KEM will result in a concrete one round AB-AKE protocol also in the generic group and random oracle models.
Keywords. Attribute-based Key Exchange, Attribute-based KEM, Group Key Exchange

1 Introduction

In a distributed collaborative system, it is often convenient for the members to communicate with the others in the system using attributes that describe their roles or responsibilities. These attributes are highly desirable if the members join/leave the system dynamically. Consider an Internet forum where the members are organized into user groups based on the members' skills or privileges. It is a natural requirement that the members of a user group should be able to establish secure communication with the other members belonging to particular user groups. The communication in these forums is generally carried out through initiating a thread or by posting messages within an existing thread. To enable authentic and confidential communication, the forum administrator may specify an access policy with the user groups being attributes. Obviously, only the members of the forum whose attributes (i.e., membership to user groups) satisfy the policy should be able to have read and/or write access to the thread.

In the above scenario, the members do not necessarily have to know the identity of the other members with whom they want to communicate. In fact, the administrator may be requested not to disclose the identity of a member to the others for privacy reasons. Any member whose attributes satisfy the policy specified by the administrator should be able to participate in the communication. Note that the communication can naturally be among a group of more than two members, since the defined policy may be satisfied by attributes of more than two members. Hence, an authenticated group key exchange protocol that facilitates attributes usage can be employed in this setting. We call such a protocol, an attribute-based authenticated key exchange (AB-AKE) protocol. Once a session key among the willing participants has been established via the key exchange protocol, it can be used for establishing secure communication among the participants.

We can further envisage applications for AB-AKE in interactive chat rooms and also in organizations with strict hierarchy like the military. In interactive chat rooms, each room may be associated with a policy defined with a set of interests being the attributes. Any member whose attributes satisfy the policy of a chat room can have read and/or write access to it. Similarly, a policy over ranks as attributes can be specified for the units in the military by another unit at a higher level in the hierarchy. All the units whose attributes satisfy the policy can establish secure communication among themselves through an AB-AKE protocol.

Attribute-based Encryption. Sahai and Waters [22] introduced the concept of attribute based encryption (ABE) as an extension to ID-based encryption [4], in which a set of descriptive attributes is regarded as an identity. Goyal et al. [16] further extended the idea of ABE and introduced two variants: key policy attribute based encryption (KP-ABE) and ciphertext policy attribute based encryption (CP-ABE). In a KP-ABE system, the private key of a party is associated with an access policy defined over a set of attributes while the ciphertext is associated with a set of attributes. A ciphertext can be decrypted by a party if the attributes associated with the ciphertext satisfy the policy associated the user's private key. A CP-ABE system can be seen as a complementary form to KP-ABE system, wherein the private key is associated with a set of attributes, while a policy defined over a set of attributes is attached to the ciphertext. A ciphertext can be decrypted by a party if the attributes associated with its private key satisfy the ciphertext's policy. Goyal et al. [16] presented a construction of KP-ABE scheme, while the first CP-ABE scheme was proposed by Bethencourt et al. [3].

1.1 Contributions

In this paper, we introduce the concept of AB-AKE protocols. We assume that each member willing to participate in an AB-AKE protocol is issued a private key for a set of attributes that he/she possesses. Our modelling of AB-AKE follows the framework of CP-ABE in that the attributes are associated with the private keys. We assume that the members are given an access policy which their attributes have to satisfy for them to participate in the protocol. Alternatively, a common policy may be negotiated by the group members themselves. The protocol takes the access policy as input and computes the protocol messages for the other parties. Similar to the CP-ABE systems, we may assume that the policy is attached to the protocol messages in the AB-AKE protocol, although this assumption is not necessary since each member knows the policy at the outset of the protocol. A member whose attributes satisfy the given policy can compute the session key from the incoming messages and (if exists) its own contribution.

While a complementary flavour of AB-AKE can be conceptualized based on KP-ABE systems, we do not explore this direction in this work. For the type of applications that we have discussed earlier, AB-AKE protocols based on ciphertext-policy attribute based systems suit well. AB-AKE can be seen as an extension of group key exchange (GKE) [8, 20, 19] with the additional expressiveness provided by the ciphertext-policy attribute-based systems. We define a notion of authenticated key exchange security (AKE-security) for AB-AKE by adapting a corresponding notion for GKE to the attribute-based setting. The property of collusion resistance considered by attribute-based systems [16, 3, 24] is naturally embedded into our AKE-security notion.

We then propose a generic one-round AB-AKE protocol that satisfies our AKE-security notion. The protocol is based on a type of attribute-based key encapsulation mechanism (KEM) that we call *encapsulation-policy attribute based KEM* (EP-AB-KEM). In an EP-AB-KEM the attributes are associated to the private key of a party and access policy is attached to the encapsulation. We define a notion of chosen ciphertext security for EP-AB-KEM based on a corresponding notion considered for CP-ABE schemes.

Our AB-AKE protocol is generic in the sense that it can be instantiated using any EP-AB-KEM that satisfies chosen ciphertext security. We propose a chosen-ciphertext secure EP-AB-KEM based on the CP-ABE scheme of Bethencourt et al. [3] and using the generic technique of Boneh et al. [6]. While we apply the technique of Boneh et al. to the chosen plaintext secure EP-AB-KEM implicit in Bethencourt et al.'s scheme, we also make some non-trivial changes to adapt it to the attribute-based setting. The proposed EP-AB-KEM is then proven secure in the generic group and random oracle models. Incidentally, we are the first to model and construct EP-AB-KEMs, which are of independent interest.

Finally, an AB-AKE protocol satisfying our AKE-security provides implicit authentication that is similar to the corresponding notion considered for normal key exchange protocols. Particularly, our AKE-security notion ensures each protocol participant that no other party apart from parties who satisfy the given policy can possibly learn the value of the session key. Note that an EP-AB-KEM cannot achieve this property since it does not provide any sender authentication. Consequently, the receivers in EP-AB-KEM whose attributes satisfy the policy have no way of knowing whether the sender actually satisfies the same policy. For example, if we use an EP-AB-KEM in a user group any one can post a message that is encrypted with the symmetric of the EP-AB-KEM. Alternatively, if the message is encrypted with a session key derived from an AB-AKE protocol the readers will get the assurance that only someone with valid attribute set has posted the message.

Our generic construction of AB-AKE can be seen as an extension of Boyd et al.'s [7] technique for two-party key exchange to the attribute-based setting. One disadvantage of our protocol is that it cannot provide forward secrecy. However, for some of the applications that we have discussed forward secrecy may not be necessary. For example, in an Internet forum the administrator may like to moderate the content posted in the user groups or in the military a unit at a higher rank would like monitor the communication among the units at the same/lower rank. In such scenarios, an AB-AKE protocol without forward secrecy will be useful since any party with the right attribute set will be able to recover the session key and consequently the messages encrypted with it. However, note that forward secrecy is generally a highly desirable attribute for key exchange protocols. Hence, we also sketch constructions of AB-AKE protocols that can achieve forward secrecy.

1.2 Related Work

The concept of fuzzy secret handshake proposed by Ateniese et al. [1] seems closely related to our modelling of AB-AKE. However, there are a few important differences: In AB-AKE, we allow policies specified by the members to be very expressive consisting of several threshold gates, while fuzzy secret handshake only considers a single threshold gate. In a (fuzzy) secret handshake protocol, if a member do not satisfy the attributes specified by another member, the attributes of none of the members can be learned by the other member. On the other hand, in an AB-AKE protocol, if a member does not satisfy the policy specified by the other members, the members do not know anything about the attributes of the other members except what can be inferred by the policies attached to the protocol messages. Although both the properties look similar, we emphasize that an AB-AKE protocol would not hide the affiliation of the members even if the protocol was not successful [17]. Note that this property of "affiliation hiding" is the main requirement for (fuzzy) secret handshakes. Finally, the fuzzy secret handshake protocol of Ateniese et al. considers only two party setting, while our protocol naturally operates in a group setting.

1.3 Organization

Section 2 presents a security model for EP-AB-KEM and also proposes a chosen ciphertext secure EP-AB-KEM. We first define a security model for AB-AKE in Section 3 and then present a generic one round AB-AKE protocol based on EP-AB-KEM. In Appendix A, we outline how to construct AB-AKE protocols with forward secrecy. Appendices B, C and D contain preliminaries, proof of the proposed EP-AB-KEM and proof of the generic AB-AKE protocol respectively.

2 Encapsulation Policy Attribute-based KEM

We first give a formal definition of security for EP-AB-KEM. As in the earlier attribute-based systems [16, 3], we review the definition of an access structure and use it in the security model. Later, we present a concrete EP-AB-KEM based on the CP-ABE scheme of Bethencourt et al. [3].

Definition 1 (Access Structure [2])**.** Let $\{U_1, \cdots, U_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{U_1,\cdots,U_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{U_1, \cdots, U_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{U_1,\cdots,U_n\}} \setminus \{\phi\}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

In our EP-AB-KEM and later in the protocol, each party possesses a set of attributes. A policy over a set of attributes is specified through an access structure $\mathbb{A}$. Hence, $\mathbb{A}$ contains the authorized sets of attributes. As in the CP-ABE of Bethencourt et al., we consider only monotonic access structures. In the rest of the paper, by an access structure we mean a monotonic one.

A EP-AB-KEM consists of five polynomial-time algorithms:

Setup: This algorithm takes the security parameter k and the attribute universe description $\mathbb{U}$ as inputs. The public parameters PK and the master key MK are the outputs.

Encapsulation: This algorithm takes as input the public parameters PK and an access structure $\mathbb{A}$ over the attribute universe $\mathbb{U}$. It outputs an encapsulation C and a symmetric key K such that only a user who possesses attributes satisfying $\mathbb{A}$ can recover K from C. Similar to the CP-ABE schemes, we assume that the encapsulation implicitly contains $\mathbb{A}$.

KeyGen: This algorithm takes as input the master key MK and a set of attributes S of a user that give a description of the user's private key. The output is the user's private key SK.

Decapsulation: takes as input the public parameters PK, an encapsulation C which contains an access structure $\mathbb{A}$ and a private key SK corresponding to a set of attributes S. If S satisfies $\mathbb{A}$, the algorithm outputs a symmetric key K, otherwise it outputs $\perp$.

Delegate: This algorithm takes as input a secret key SK corresponding to a set of attributes S and a set $\tilde{S} \subseteq S$. It output a secret key $\tilde{SK}$ for $\tilde{S}$.

2.1 Security Model

Bethencourt et al. [3] defined the notion of indistinguishability under chosen plaintext attack (IND-CPA) for CP-ABE schemes. In this section, we adapt their notion and extend it to define a notion of indistinguishability under chosen ciphertext attacks (IND-CCA) for EP-AB-KEM. An adversary $\mathcal{A}^{CCA}$ against the IND-CCA notion is allowed to ask Extract and Decap queries. The security notion is formally defined as follows.

Definition 2. An EP-AB-KEM is IND-CCA secure if the advantage of any probabilistic polynomial time adversary $\mathcal{A}^{CCA}$ in the following game is negligible in the security parameter k.

Setup: The challenger runs the Setup algorithm and returns the public parameters PK to $\mathcal{A}^{CCA}$.

Phase 1: $\mathcal{A}^{CCA}$ issues Extract and Decap queries as follows:

Extract: This query can be issued multiple times with sets of attributes $S_1, \cdots, S_{q_1}$ as input. The challenger returns private keys corresponding to the input attribute sets.

Decap: This query is issued with an encapsulation C and an attribute set S as inputs. Note that C implicitly contains an access structure $\mathbb{A}$ defined over the attribute universe $\mathbb{U}$. If S satisfies $\mathbb{A}$, the challenger executes the Decap algorithm on C using a private key corresponding to S and returns a symmetric key K. Otherwise, the $\perp$ is returned.

Challenge: At the end of **Phase 1**, $\mathcal{A}^{CCA}$ gives an access structure $\mathbb{A}^*$ defined over $\mathbb{U}$ to the challenger. The challenger first chooses a bit b. It then runs the Encapsulation algorithm with $\mathbb{A}^*$ as input and generates a symmetric key–encapsulation pair (K_1, C^*). It then sets K_0 to be a random key drawn from the probability distribution of the symmetric key. The tuple (K_b, C^*) is returned to $\mathcal{A}^{CCA}$ as the challenge. A trivial restriction on the adversary's choice of $\mathbb{A}^*$ is that none of the attributes sets $S_1, \cdots, S_{q_1}$ passed as input to Extract queries in **Phase 1** should satisfy $\mathbb{A}^*$.

Phase 2: $\mathcal{A}^{CCA}$ is allowed to execute in the same way as in **Phase 1** with the following restrictions: (1) none of the attribute sets $S_{q_1+1}, \cdots, S_q$ passed as input to Extract queries satisfy $\mathbb{A}^*$ and (2) a Decap query with C^* as input in combination with an attribute set S^* that satisfies $\mathbb{A}^*$ is not allowed.

Guess: The goal of $\mathcal{A}^{CCA}$ is to guess whether the key K_b is encapsulated within C^* or not. $\mathcal{A}^{CCA}$ finally outputs a guess bit b'. It wins the game if $b' = b$. The advantage of $\mathcal{A}^{CCA}$ is given as $Adv_{\mathcal{A}^{CCA}} = |2 \cdot \Pr[b' = b] - 1|$

Existing security notions for CP-ABE schemes also consider the *selective model* where $\mathcal{A}^{CCA}$ declares the challenge access structure $\mathbb{A}^*$ before the **Setup** phase.

Remark 1. In the above definition, $\mathcal{A}^{CCA}$ is allowed to issue multiple Extract queries with attribute sets as input such that none of the individual sets S_i satisfy the challenge access structure $\mathbb{A}^*$. Hence, similar to earlier definitions of attribute based encryption schemes, our definition also takes care of collusion resistance. An EP-AB-KEM satisfying the above definition ensures that from the private keys of S_i's, $\mathcal{A}^{CCA}$ cannot construct a private key corresponding to another attribute set S^* such that S^* satisfies $\mathbb{A}^*$.

Hybrid CP-ABE. An EP-AB-KEM satisfying the above IND-CCA security notion can be combined with an IND-CCA secure data encapsulation mechanism (DEM) [11, 12] to construct an IND-CCA secure CP-ABE scheme. We do not formally analyze this hybrid construction as it is out of scope of this paper.

2.2 A Chosen Ciphertext Secure EP-AB-KEM

Bethencourt et al. [3] first proposed a construction of a CP-ABE scheme. Their scheme was shown IND-CPA secure assuming generic group and random oracle models. More recently, many CP-ABE schemes [15, 10, 24] were proposed and were shown IND-CPA secure without assuming generic group or random oracle models, but analyzed only in the selective model of security. Waters [24] gave a brief survey comparing existing CP-ABE schemes.

We now construct a chosen ciphertext secure EP-AB-KEM based on the CP-ABE scheme of Bethencourt et al. The idea is to enhance the security of the chosen plaintext secure EP-AB-KEM that is implicit in Bethencourt et al.'s CP-ABE scheme. For this purpose, the techniques of Fujisaki and Okamoto [14, 13] and Canetti et al. (CHK) [9] can be applied in the random oracle and standard models respectively. As remarked by Bethencourt et al., chosen ciphertext security for CP-ABE and correspondingly for EP-AB-KEM schemes can be achieved by a straightforward application of the Fujisaki-Okamoto technique.

Bethencourt et al. also suggested that the delegation mechanism of their CP-ABE scheme can be leveraged to achieve IND-CCA security using the CHK transform. However, we observe that applying the CHK transform to CP-ABE schemes (similarly to EP-AB-KEMs) is slightly more involved. Specifically, contrary to the approach followed by KP-ABE schemes, IND-CCA security for CP-ABE schemes cannot be achieved by directly leveraging the delegation mechanism. We later discuss why this is so and then present an IND-CCA secure EP-AB-KEM by making a few changes to the Setup and Encapsulation algorithms derived from Bethencourt et al.'s CP-ABE scheme. Although the CHK technique can be used to achieve IND-CCA security in the standard model, our EP-AB-KEM will only be secure assuming generic groups and random oracles since the base CP-ABE scheme also assumes the same. Finally, we choose the scheme of Bethencourt et al. because it is secure in the fully adaptive model (i.e., non-selective model). We later (in Appendix D) discuss the necessity of an EP-AB-KEM to be secure in the adaptive model for constructing AB-AKE protocols.

The scheme first generates a one-time key pair (sk, vk) for a signature scheme with the condition that the verification key is of the same length as the length of an attribute in the system. Let $\mathbb{A}$ be the access structure given as input to the EP-AB-KEM. We now construct a more restrictive access structure $\mathbb{A}' = \mathbb{A}$ AND vk and execute the CPA secure EP-AB-KEM under $\mathbb{A}'$. The resulting encapsulation is then signed using the one-time signing key sk. The encapsulation of the CCA secure EP-AB-KEM contains the encapsulation generated by the underlying CPA-secure KEM, the signature generated on it and the verification key vk. The recipient first checks the signature using vk and then executes the CPA-secure KEM's decapsulation algorithm under $\mathbb{A}'$ to extract the symmetric key.

While the above informal description of our construction directly follows the CHK technique, the tricky part in the context of EP-AB-KEM (or CP-ABE) is to empower the recipient with a private key corresponding to the attributes that satisfy the modified access structure $\mathbb{A}'$. The recipient may already possess attributes that satisfy $\mathbb{A}$. However, since the verification key vk is one-time and chosen randomly for each execution of EP-AB-KEM, the recipient cannot be issued with a private key that can decrypt messages encrypted under $\mathbb{A}' = \mathbb{A}$ AND vk. This problem cannot be addressed by the delegation mechanism in a EP-AB-KEM (or CP-ABE) scheme since it can be used to derive private key corresponding to an attribute set S' from the one corresponding to S only if $S' \subseteq S$. But, we have an additional attribute in the form of vk. Note that this is not a problem in the KP-ABE system since it naturally allows a party with a private key corresponding to an access structure $\mathbb{A}$ to derive private keys corresponding to access structures that are more restrictive than $\mathbb{A}$.

To address the above problem, we make modifications to the Setup and Encapsulation algorithms derived from the CP-ABE scheme of Bethencourt et al. [3]. Our EP-AB-KEM now enables a recipient with private key for attributes that satisfy $\mathbb{A}$ to decapsulate an encapsulation created under $\mathbb{A}'$, irrespective of the choice of vk by the sender. As in the CP-ABE scheme of Bethencourt et al., an access structure $\mathbb{A}$ is represented in the form of a an access tree $\mathcal{T}$.

Access Tree. Let $\mathcal{T}$ be a tree representing an access structure. Each interior node of $\mathcal{T}$ represents a threshold gate, while each leaf node is described by an attribute. Let num_x be the number of children of a node x and let k_x be its threshold value. We have $0 \leq k_x \leq num_x$. If the threshold gate represented by an interior node is an AND gate then $k_x = num_x$ and if the gate is OR, $k_x = 1$. The threshold value for each leaf node x is defined to be $k_x = 1$. The parent of a node x in the tree $\mathcal{T}$ is denoted by the function $\mathsf{parent}(x)$, while the attribute of a leaf node x is denoted by $\mathsf{att}(x)$. The children of each interior node are numbered from 1 to num_x. The function $\mathsf{index}(x)$ returns such a number associated with a node x. We assume that the index values are uniquely assigned in an arbitrary manner for a given access structure.

Satisfying an access tree. Let r be the root of an access tree $\mathcal{T}$. The subtree of $\mathcal{T}$ rooted at a node x is denoted by $\mathcal{T}_x$. If a set of attributes γ satisfy the access tree $\mathcal{T}_x$, it is denoted as $\mathcal{T}_x(\gamma) = 1$. The function $\mathcal{T}_x(\gamma)$ is computed recursively as follows: If x is an interior node, for each children x' of x, $\mathcal{T}_{x'}(\gamma)$ is evaluated. $\mathcal{T}_x(\gamma)$ returns 1 if and only if at least k_x children of x return 1. If x is a leaf node, $\mathcal{T}_x(\gamma)$ returns 1 if and only if $\mathsf{att}(x) \in \gamma$.

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative groups of prime order p and g be an arbitrary generator of $\mathbb{G}_0$. Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ be a bilinear map as defined in Section B.1. The Lagrange's coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$ is defined as: $\Delta_{i,S} = \Pi_{j \in S, j \neq i} \frac{x-j}{i-j}$.

$\mathsf{Setup}(1^k)$. It chooses the groups $\mathbb{G}_0$, $\mathbb{G}_1$ and defines a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. It also selects $\alpha, \beta_1, \beta_2 \in \mathbb{Z}_p$ such that $\beta_1 \neq \beta_2$, $\beta_1 \neq 0$ and $\beta_2 \neq 0$. The public key is

$$PK = \Big(\mathbb{G}_0, \mathbb{G}_1, e, g, h_1 = g^{\beta_1}, f_1 = g^{1/\beta_1}, h_2 = g^{\beta_2}, f_2 = g^{1/\beta_2}, e(g,g)^{\alpha}\Big)$$

The master key MK is $(\beta_1, \beta_2, g^{\alpha})$.

$\mathsf{Encapsulation}(PK, \mathcal{T})$. This algorithm generates an encapsulation and a symmetric key under the access tree $\mathcal{T}$ using the public key PK. It first executes the KeyGen algorithm of the signature scheme (ref. Section B.2) and obtains a one-time key pair (sk, vk). Let $\mathbb{A}$ be the access structure represented by $\mathcal{T}$. The algorithm now constructs a new access tree $\mathcal{T}'$ for the access structure ($\mathbb{A}$ AND vk) as follows: Let R be the root node of $\mathcal{T}$. The root node R' of the new tree $\mathcal{T}'$ is set as the AND gate with $\mathcal{T}$ as its subtree and the verification key vk as a leaf node attached to R'.

The algorithm now generates a polynomial q_x for each node x in the tree $\mathcal{T}'$ in a top-down approach as follows: Starting from the root node R', for each node x in the tree set the degree d_x of the polynomial associated with x to be $k_x - 1$ i.e., the degree of the polynomial is one less than the threshold value associated with the node x. The algorithm starts from the root node and first chooses a random $s \in \mathbb{Z}_p$. Then it chooses $d_{R'}$ other points randomly to define the polynomial $q(R')$. For any node x other than the root, it sets $q_x(0) = q_{\mathsf{parent}(x)}(\mathsf{index}(x))$ and chooses d_x other points randomly to define the polynomial $q(x)$.

Let Y be the set of leaf nodes in the subtree $\mathcal{T}$ rooted at R. The only other leaf node in the tree $\mathcal{T}'$ is the one that describes the verification key vk. The algorithm proceeds as follows:

1. $K = e(g,g)^{\alpha s}$

2. $C_1 = h_1^s$
3. $\forall y \in Y: \;\; C_y = g^{q_y(0)}, C'_y = H(\mathsf{att}(y))^{q_y(0)}$
4. $C_{vk} = h_2^{q_{vk}(0)}, \; C'_{vk} = H(vk)^{q_{vk}(0)}$
5. Let $\mathcal{C} = (\mathcal{T}', C_1, C_y, C'_y, C_{vk}, C'_{vk})$. Compute a signature $\sigma = \mathsf{Sig}_{sk}(\mathcal{C})$.

The final encapsulation $C = (\mathcal{C}, vk, \sigma)$

KeyGen(MK,S). The key generation algorithm takes as input the master key MK and a set of attributes S and outputs a private key corresponding to S. It chooses $r, r_{vk} \in \mathbb{Z}_p$ and $r_j \in \mathbb{Z}_p$ for each $j \in S$. The private key is computed as:

$$SK = (D = g^{(\alpha+r)/\beta_1}, \; E = g^{r/\beta_2}, \; \forall j \in S: \;\; D_j = g^r \cdot H(j)^{r_j}, \; D'_j = g^{r_j})$$

Delegate($SK, \tilde{S}$). It takes as input a secret key SK corresponding to a set of attributes S and another set $\tilde{S} \subseteq S$. The key SK is of the form $SK = (D, E, \forall j \in S: D_j, D'_j)$. The algorithm chooses $\tilde{r}$ and $\tilde{r}_k \forall k \in \tilde{S}$. The new key for $\tilde{S}$ is created as:

$$\tilde{SK} = (\tilde{D} = Df_1^{\tilde{r}}, \; \tilde{E} = Ef_2^{\tilde{r}}, \; \forall k \in \tilde{S}: \; \tilde{D}_k = D_k g^{\tilde{r}} H(k)^{\tilde{r}_k}, \; \tilde{D}'_k = D'_k g^{\tilde{r}_k})$$

Decapsulation(SK, C). Upon receiving an encapsulation C, the decryptor first parses the access tree $\mathcal{T}'$. It then extracts the subtree $\mathcal{T}$ rooted at R from $\mathcal{T}'$. Note that this can be easily done since the node that describes the verification key as an attribute can be identified with the help of the verification key vk sent in the encapsulation. The algorithm first verifies the signature σ on C using the verification key vk. If the verification succeeds, it proceeds as follows:

$$\begin{aligned} F_{vk} = \frac{e(C_{vk}, H(vk) \cdot g^{r/\beta_2})}{e(C'_{vk}, h_2)} &= \frac{e(C_{vk}, g^{r/\beta_2}) \cdot e(C_{vk}, H(vk))}{e(C'_{vk}, h_2)} \\ &= \frac{e(h_2^{q_{vk}(0)}, g^{r/\beta_2}) \cdot e(h_2^{q_{vk}(0)}, H(vk))}{e(H(vk)^{q_{vk}(0)}, h_2)} \\ &= e(g^{\beta_2 \cdot q_{vk}(0)}, g^{r/\beta_2}) = e(g,g)^{rq_{vk}(0)} \end{aligned} \tag{1}$$

A recursive algorithm DecryptNode($\mathcal{C}, SK, x$) that takes as input $\mathcal{C}$, a private key SK associated with a set of attributes S and a node x from the subtree $\mathcal{T}$ is then executed as below:

If x is a leaf node, then let $i = \mathsf{att(x)}$. If $i \notin S$, then DecryptNode($\mathcal{C}, SK, x$) $= \perp$. Otherwise it is defined as follows:

DecryptNode($\mathcal{C}, SK, x$) $= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} = \frac{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} = e(g,g)^{rq_x(0)}$

If x is an interior node then DecryptNode($\mathcal{C}, SK, x$) proceeds as follows: For all nodes z that are children of x, the algorithm DecryptNode($\mathcal{C}, sk, z$) is called. The output is stored as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists, the function returns $\perp$. Otherwise, the decapsulation algorithm proceeds as follows:

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}, \text{ where } i = \mathsf{index}(z), S'_x = \{\mathsf{index}(z) : z \in S_x\} \\ &= \prod_{z \in S_x} (e(g,g)^{r \cdot q_z(0)})^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{r \cdot q_{\mathsf{parent}(z)}(\mathsf{index}(z))})^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{r \cdot q_x(i) \cdot \Delta_{i,S'_x}(0)} \\ &= (e(g,g)^{r \cdot q_x(0)} \end{aligned} \tag{2}$$

Finally, the decapsulation algorithm calls the $\mathsf{DecryptNode}$ algorithm on the node R, which is the root of the subtree $\mathcal{T}$. If $\mathcal{T}$ is satisfied by the attribute set S, then we have $F_R = \mathsf{DecryptNode}(\mathcal{C}, SK, R) = e(g,g)^{r \cdot q_R(0)}$. We now compute $F_{R'}$ from F_{vk} and F_R using polynomial interpolation as follows:

$$\begin{aligned} F_{R'} &= \prod_{x \in \{R, vk\}} F_x^{\Delta_{\mathsf{index}(x), \{R, vk\}}} \\ &= e(g,g)^{r \cdot q_{R'}(0)} \\ &= e(g,g)^{rs} \end{aligned}$$

Let $A = e(g,g)^{rs}$. The symmetric key is recovered as

$$e(C_1, D)/A = e(h_1^s, g^{(\alpha+r)/\beta_1})/e(g,g)^{rs} = e(g,g)^{s(\alpha+r)}/e(g,g)^{rs} = e(g,g)^{\alpha s} = K \tag{3}$$

Note that in Equation 1 we implicitly verify that the one-time verification key has not been replaced. If the vk was replaced the symmetric key computed in Equation 3 would be $\perp$. Alternatively, the verification check can be done explicitly at the cost of an additional pairing operation. In Appendix C, we show that the proposed EP-AB-KEM is IND-CCA secure in the generic group and random oracle models.

3 Attribute-based Authenticated Key Exchange

An AB-AKE protocol consists of three polynomial-time algorithms: Setup, KeyGen and $\mathsf{KeyExchange}$. The Setup and KeyGen algorithms are identical to those defined for EP-AB-KEM in Section 2. Each party in the AB-AKE protocol executes the $\mathsf{KeyExchange}$ algorithm which initially takes as input the master public key PK, an access structure $\mathbb{A}$ and a private key for a set of attributes S. If S satisfies $\mathbb{A}$, $\mathsf{KeyExchange}$ proceeds as per specification and may generate outgoing messages and also accept incoming messages from other parties as inputs. The output of $\mathsf{KeyExchange}$ is either a session key κ or the $\perp$ symbol.

Communication Model Let $\mathcal{U}= \{U_1, \cdots, U_n\}$ be a set of n users. The protocol may be executed among any subset $\tilde{U} \subseteq \mathcal{U}$ of size $\tilde{n} \geq 2$. We assume that each user has a set of descriptive attributes. Let SK_i be the private key corresponding to an attribute set S_i of user U_i. We assume that an access structure $\mathbb{A}$ is given as input to all the users. Note that this $\mathbb{A}$ may be specified by a higher level protocol. Alternatively, the users can run an interactive protocol to negotiate a common access structure $\mathbb{A}$. We also assume that all the users execute the protocol honestly. If a user U_i wants to establish a key with other users in $\mathcal{U}$, it first checks whether its attribute set S_i satisfy $\mathbb{A}$ or not. U_i proceeds with the protocol execution only if S_i satisfy $\mathbb{A}$. Thus, any user U_j with attribute set S_j that satisfy $\mathbb{A}$ is a potential participant in the key exchange protocol. The set of parties whose individual attributes satisfy $\mathbb{A}$ can compute a common session key.

An AB-AKE protocol π executed among $\tilde{n} \leq n$ users is modelled as a collection of $\tilde{n}$ programs running at the $\tilde{n}$ parties. Each instance of π within a party is defined as a session and each party may have multiple such sessions running concurrently. Let π_i^j be the j-th run of the protocol π at party $U_i \in \tilde{\mathcal{U}}$. Each protocol instance at a party is identified by a unique session ID. We assume that the session ID is derived during the run of the protocol. The session ID of an instance π_i^j is denoted by sid_i^j. An instance π_i^j enters an *accepted* state when it computes a session key sk_i^j. Note that an instance may terminate without ever entering into an accepted state. The information of whether an instance has terminated with acceptance or without acceptance is assumed to be public.

Note that there may be more than one party whose attributes can satisfy $\mathbb{A}$, hence we consider group setting for AB-AKE. We define partnership in AB-AKE protocol as follows: A set of $\tilde{n}$ instances at $\tilde{n}$ different parties $\tilde{\mathcal{U}} \subseteq \mathcal{U}$ are called partners if

1. they all have the same session ID **and**
2. the attributes of each $U_i \in \tilde{\mathcal{U}}$ satisfy $\mathbb{A}$

Adversarial Model The communication network is assumed to be fully controlled by the adversary, which schedules and mediates the sessions among all the parties. The adversary is allowed to insert, delete or modify the protocol messages. If it honestly forwards the protocol messages among all the participants, then all the instances are partnered and output identical session keys. We also assume that it is the adversary that may select the protocol participants from the set $\mathcal{U}$. While the adversary may not know the attribute set that a user possesses, it can initiate an instance of the AB-AKE protocol with an access structure of its choice. In addition to controlling the message transmission, the adversary is allowed to ask the following queries.

– $\mathsf{Send}(\pi_i^j, m)$ sends a message m to the instance π_i^j. If the message is $\mathbb{A}$, the instance π_i^j is initiated the access structure $\mathbb{A}$. Otherwise, the message is processed as per the protocol specification. The response of π_i^j to any Send query is returned to the adversary.
– $\mathsf{RevealKey}(\pi_i^j)$ If π_i^j has accepted, the adversary is given the session key sk_i^j established at π_i^j.
– $\mathsf{Corrupt}(S_i)$ This query returns the private key SK_i corresponding to the attribute set S_i.
– $\mathsf{Test}(\pi_i^j)$ A random bit b is secretly chosen. If $b = 1$,the adversary is given sk_i^j established at π_i^j. Otherwise, a random value chosen from the session key probability distribution is given. Note that a Test query is allowed only on an accepted instance.

Definition 3 (Freshness)**.** Let $\mathbb{A}$ be the access structure for an instance π_i^j. π_i^j is called fresh if the following the conditions hold: (1) the instance π_i^j or its partners have not been asked a $\mathsf{RevealKey}$ query **and** (2) there has not been a $\mathsf{Corrupt}$ query on an input S_i such that S_i satisfies $\mathbb{A}$.

Definition 4 (AKE-Security)**.** An adversary $\mathcal{A}_{ake}$ against the AKE-security notion is allowed to make Send, RevealKey and Corrupt queries in Stage 1. $\mathcal{A}_{ake}$ makes a Test query to an instance π_i^j at the end of Stage 1 and is given a challenge key K_b as described above. It can continue asking queries in Stage 2. Finally, $\mathcal{A}_{ake}$ outputs a bit b' and wins the AKE security game if (1) $b' = b$ **and** (2) the Test instance π_i^j remains fresh till the end of $\mathcal{A}_{ake}$'s execution. Let $\mathsf{Succ}_{\mathcal{A}_{ake}}$ be the success probability of $\mathcal{A}_{ake}$ in winning the AKE-security game. The advantage of $\mathcal{A}_{ake}$ in winning this game is $\mathsf{Adv}_{\mathcal{A}_{ake}} = |2 \cdot \Pr[\mathsf{Succ}_{\mathcal{A}_{ake}}] - 1|$. A protocol is called AKE-secure if $\mathsf{Adv}_{\mathcal{A}_{ake}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}_{ake}$.

Remark 2. By allowing the adversary to reveal the private keys corresponding to attribute sets which individually do not satisfy the given access structure $\mathbb{A}^*$ in the test session, our definition naturally considers collusion resistance. In other words, any number of parties whose individual attribute sets do not satisfy $\mathbb{A}^*$ may collude among themselves and try to violate the AKE-security of the protocol. An AB-AKE protocol satisfying our AKE-security notion will still remain secure against such collusion attacks.

4 A Generic One Round AB-AKE Protocol

We now present a simple generic AB-AKE protocol based on IND-CCA secure EP-AB-KEM. Informally, each party executes an EP-AB-KEM in parallel and combines the symmetric key it has generated with the symmetric keys extracted from the incoming messages to establish a common session key. Our construction is an extension of Boyd et al.'s [7] one round generic protocol for two-party key exchange to the attribute-based setting. Figure 1 presents our generic one round AB-AKE protocol.

At the beginning of the protocol each party is given an access structure $\mathbb{A}$ represented via an access tree $\mathcal{T}$. The protocol uses an EP-AB-KEM scheme (Setup, Encapsulation, KeyGen, Decapsulation). Each U_i is issued a private key SK_i corresponding to the attributes set S_i that it possesses. Each party U_i who has attribute set S_i satisfying the access structure $\mathbb{A}$ runs the Encapsulation algorithm and obtains a symmetric key-encapsulation pair (K_i, C_i). The parties broadcast the encapsulations to other parties. Upon receiving the encapsulations, each party runs the Decapsulation algorithm using the its private key on each of the incoming encapsulations and extracts the symmetric keys. The session key is finally computed by each party from the symmetric key that it has generated and all the symmetric keys decapsulated from the incoming encapsulations.

A pseudo-random function f is applied to derive the session key. We assume that the symmetric key output by the Decapsulation algorithm can be directly used as a seed for f. Otherwise, we have to extract and then expand the randomness from the output of the Decapsulation algorithm as done by Boyd et al. [7].

Theorem 1. *The AB-AKE protocol in Fig. 1 is AKE-secure as per Definition 4 assuming that the underlying EP-AB-KEM is IND-CCA secure. The advantage of $\mathcal{A}_{ake}$ is*

$$Adv\mathcal{A}_{ake} \le \tilde{n} \cdot \frac{q_s^2}{|C|} + q_s \cdot (\tilde{n} \cdot Adv_{\mathcal{A}_{PRF}} + Adv_{\mathcal{A}^{CCA}})$$

where n is the number of parties in the protocol, q_s is the number of sessions $\mathcal{A}_{ake}$ is allowed to activate, $|C|$ is the size of the ciphertext space, $\mathcal{A}^{CCA}$ is a polynomial adversary against the IND-CCA security of the underlying EP-AB-KEM and $\mathcal{A}_{PRF}$ is a polynomial adversary adversary against the pseudo random function f.

Computation

Each U_i executes an EP-AB-KEM on the input $(PK, \mathcal{T})$ where PK is the master public key and $\mathcal{T}$ is the access tree that represents an access structure $\mathbb{A}$. As a result, a symmetric key and encapsulation pair (K_i, C_i) is obtained.

$$(K_i, C_i) \leftarrow \mathsf{Encapsulation}(PK, \mathcal{T})$$

Broadcast

Each U_i broadcasts the generated encapsulation C_i.

$$U_i \rightarrow \tilde{\mathcal{U}} \setminus \{U_i\}: \ \ C_i$$

Key Computation

1. Each U_i executes the decapsulation algorithm using its private key SK_i on each of the incoming encapsulations C_j and obtains the symmetric keys K_j, for $j \neq i$.

$$K_j \leftarrow \mathsf{Decapsulation}(sk_i, C_j) \text{ for each } j \neq i$$

2. Each U_i then computes the session ID as the concatenation of all the outgoing and incoming messages exchanged i.e. $\mathsf{sid} = (C_1 \| \cdots \| C_{\tilde{n}})$.
3. The session key κ is then computed as

$$\kappa \ = f_{K_1}(\mathsf{sid}) \oplus f_{K_2}(\mathsf{sid}) \oplus \cdots \oplus f_{K_n}(\mathsf{sid})$$

where f is a pseudo-random function.

Fig. 1. A Generic One round AB-AKE Protocol

The proof of the above theorem is given in Appendix D.

Concrete Instantiation. From the EP-AB-KEM proposed in Section 2.2, a concrete AB-AKE protocol can be directly realized. It follows from the security of the EP-AB-KEM and the generic AB-AKE protocol that the instantiated protocol is AKE-secure in the generic group model and the random oracle model. We leave it an open problem to construct an adaptive and IND-CCA secure EP-AB-KEM in the standard model. By our generic protocol, constructing such an EP-AB-KEM will immediately result in a one-round AB-AKE protocol secure in the standard model.

5 Conclusion

We have initiated the concept of attribute based authenticated key exchange (AB-AKE) in the ciphertext-policy attribute-based system. Our modelling of AB-AKE assumes that each party has a set of attributes and a corresponding private key. A policy is defined (or negotiated) for each execution of the protocol and the parties satisfying the policy can establish a common shared key by executing the protocol.

In the security model for AB-AKE, we have considered only outsider adversaries. Our security model can be extended by considering insider attackers who try to impersonate other protocol participants [19]. However, constructing AB-AKE protocols secure against insider attackers may require additional setup assumptions since we assume that a given policy can be satisfied by multiple parties at the same time. One possible solution to avoid impersonation in such case is to employ some authentication mechanism like signatures (not attribute-based).

We have also introduced the concept of EP-AB-KEM. We then proposed a one-round generic AB-AKE protocol based on IND-CCA secure EP-AB-KEMs. For concrete instantiation of this protocol, we have presented an EP-AB-KEM and shown it secure under the IND-CCA notion in the generic group and random oracle models. As a consequence, a concrete AB-AKE protocol based on this EP-AB-KEM would also be secure in the group group and random oracle models.

References

1. Ateniese, G., Kirsch, J., Blanton, M.: Secret Handshakes with Dynamic and Fuzzy Matching. In: Proceedings of the Network and Distributed System Security Symposium–NDSS'07, The Internet Society (2007)
2. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-Policy Attribute-Based Encryption. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (2007) 321–334
4. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Advances in Cryptology–CRYPTO'01. Volume 2139 of LNCS., Springer (2001) 213–229
5. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Advances in Cryptology–EUROCRYPT 2005. Volume 3494 of LNCS., Springer (2005) 440–456
6. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. SIAM J. Comput. **36**(5) (2007) 1301–1328
7. Boyd, C., Cliff, Y., González Nieto, J.M., Paterson, K.G.: One-Round Key Exchange in the Standard Model. International Journal of Applied Cryptography **1**(3) (2009) 181–199
8. Burmester, M., Desmedt, Y.: A Secure and Efficient Conference Key Distribution System (Extended Abstract). In: Advances in Cryptology–EUROCRYPT'94. (1994) 275–286
9. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Advances in Cryptology–EUROCRYPT 2004. Volume 3027 of LNCS., Springer (2004) 207–222
10. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, New York, NY, USA, ACM (2007) 456–465
11. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM J. Comput. **33**(1) (2004) 167–226
12. Dent, A.W.: A Designer's Guide to KEMs. In: Cryptography and Coding, 9th IMA International Conference, Cirencester. Volume 2898 of LNCS., Springer (2003) 133–151
13. Fujisaki, E., Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost. In Imai, H., Zheng, Y., eds.: Public Key Cryptography–PKC '99. Volume 1560 of LNCS., Springer (1999) 53–68
14. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In Wiener, M.J., ed.: Advances in Cryptology–CRYPTO '99. Volume 1666 of LNCS., Springer (1999) 537–554
15. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded Ciphertext Policy Attribute Based Encryption. In: Automata, Languages and Programming, 35th International Colloquium–ICALP'08. Volume 5126 of LNCS., Springer (2008) 579–591
16. Goyal, V., Pandey, O., Sahai, A., 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–CCS'06, ACM (2006) 89–98
17. Jarecki, S., Liu, X.: Private Mutual Authentication and Conditional Oblivious Transfer. In Halevi, S., ed.: Advances in Cryptology–CRYPTO'09. Volume 5677 of LNCS., Springer (2009) 90–107
18. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: Algorithmic Number Theory, 4th International Symposium. Volume 1838 of LNCS., Springer (2000) 385–394
19. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security–CCS'05, ACM (2005) 180–189
20. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Advances in Cryptology–CRYPTO'03. Volume 2729 of LNCS., Springer (2003) 110–125
21. Maji, H., Prabhakaran, M., Rosulek, M.: Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. Cryptology ePrint Archive, Report 2008/328 (2008) `http://eprint.iacr.org/2008/328`.
22. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In Cramer, R., ed.: Advances in Cryptology–EUROCRYPT'05. Volume 3494 of LNCS., Springer (2005) 457–473
23. Schwartz, J.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM **27**(4) (1980) 701–717
24. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. Cryptology ePrint Archive, Report 2008/290 (2008) `http://eprint.iacr.org/`.
25. Zippel, R.: Probabilistic algorithms for sparse polynomials. In Ng, E., ed.: EUROSAM. Volume 72 of LNCS., Springers (1979) 216–226

A Extensions

The security model in Section 3 is concerned only about the basic notion of AKE-security without forward secrecy. Forward secrecy is one of the most important security attributes for key exchange protocols since it limits the damage of long-term key exposure. A key exchange protocol with forward secrecy ensures that even if the long-term key of a party is exposed, all the past session keys established using that long-term key will remain uncompromised.

Forward secrecy seems to be more important in the case of AB-AKE protocols than it is for normal key exchange protocols. To see why, let us assume that the adversary obtains the private key of a user U_i who possesses a set of attributes S_i. If an AB-AKE protocol does not achieve forward secrecy, then the adversary can compromise all the protocol sessions which have been established with access structures that can be satisfied by S_i. Note that the party U_i does not even have to participate in any of these sessions. We now define a notion of freshness that takes forward secrecy into account.

A.1 AKE-security with Forward Secrecy

Definition 5 (FS-Freshness)**.** Let $\mathbb{A}$ be the access structure for an instance π_i^j. π_i^j is called fs-fresh if the following the conditions hold: (1) the instance π_i^j or its partners have not been asked a $\mathsf{RevealKey}$ query **and** (2) there has not been a $\mathsf{Corrupt}$ query on an input S_i before π_i^j or its partner instances have terminated, such that S_i satisfies $\mathbb{A}$.

Definition 5 can be coupled with the AKE-security notion in Definition 4 to arrive at AKE-security notion with forward secrecy for AB-AKE protocols.

A.2 Constructing AB-AKE Protocols with Forward Secrecy

Our one-round AB-AKE protocol can be modified to achieve AKE-security with forward secrecy for two-party and three-party settings using known techniques. For a two-party AB-AKE protocol with forward secrecy, one can use the technique of Boyd et al. [7] where ephemeral Diffie-Hellman public keys are appended with the encapsulations. Similarly, for a three-party AB-AKE protocol with forward secrecy, the protocol of Joux [18] can be executed in the same round with our EP-AB-KEM based protocol. The session keys in both the protocols will include the ephemeral Diffie-Hellman and ephemeral bilinear Diffie-Hellman keys, which ensure forward secrecy. However, the protocols will achieve *weak forward secrecy*, wherein the adversary has to remain passive during protocol execution. The security of the resulting two-party and three-party AB-AKE protocols will depend on the hardness of the computational Diffie-Hellman and bilinear Diffie-Hellman problems respectively along with the security of the underlying AB-AKE protocol (the security of the latter has been proven already).

Constructing AB-AKE protocols in the more general group setting needs more than one round. The compiler of Katz and Yung (KY) [20] turns an unauthenticated group key exchange protocol into an authenticated one. The compiler uses a public key based signature as an "authenticator" for this purpose. One may adapt the KY compiler to the attribute-based setting by replacing the normal public key based signature with an attribute-based signature [21]. The resulting compiler can then be applied to the two-round unauthenticated Burmester and Desmedt (BD) protocol [8] to achieve a three-round AB-AKE protocol with forward secrecy. Since the session key established by the BD protocol is ephemeral it achieves forward secrecy, where as the attribute-based KY compiler provides authentication. Although the attribute-based version of the KY compiler can be constructed with necessary changes to the KY compiler, it may not be straightforward.

B Preliminaries

B.1 Bilinear Pairing

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative groups of prime order p. Let g be an arbitrary of $\mathbb{G}_0$. The pairing $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ is called an admissible bilinear map if it has the following properties:

Bilinearity: $\forall u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$
Non-degeneracy: $e(g, g) \neq 1$
Computable: There exists an efficient algorithm to compute $e(g, g)$

B.2 Strong Existential Unforgeability

A signature scheme Σ consists of three polynomial time algorithms: SigKeyGen, Sign and Verify. The probabilistic algorithm SigKeyGen generates a signing-verification key pair (sk, vk). Sign is also a probabilistic algorithm that produces a signature σ on an input message m using the signing key sk. Verify is a deterministic algorithm that takes a tuple (m, σ, vk) as input and outputs a boolean value. If σ is a valid signature on m under vk, Verify returns 1. Otherwise 0 is returned.

A signature is said to be strongly existentially unforgeable against chosen message attacks (sUF-CMA) if there exists no probabilistic polynomial time adversary $\mathcal{A}^{CMA}$ that has non-negligible success probability in the security game below:

Setup: The challenger runs the SigKeyGen algorithm to generate a key pair (sk, vk) and passes the verification key vk on to $\mathcal{A}^{CMA}$.

Sign Queries: This query is asked by $\mathcal{A}^{CMA}$ with a message m as input. The challenger runs the Sign algorithm with signing key sk and returns the signature σ to $\mathcal{A}^{CMA}$. $\mathcal{A}^{CMA}$ is allowed to issue multiple **Sign** queries in an adaptive manner.

Forgery: The adversary outputs a tuple (m^*, σ^*). It wins the sUF-CMA security game if (1) σ^* is a valid signature on the message m^* under vk and (2) (m^*, σ^*) has not been an output of any of the Sign queries issued earlier.

C Security Proof of EP-AB-KEM

We prove the security of our EP-AB-KEM in the generic group and random oracle models. Intuitively, our security proof implies that if there are any weaknesses in the our EP-AB-KEM, they will only have come from exploiting specific mathematical structures of the underlying groups or the cryptographic hash functions used in the instantiation. Our proof closely follows the proof of the CP-ABE scheme of Bethencourt et al. [3].

The Generic Group Model [5] . We consider two random encodings ψ_0, ψ_1 of the additive group $\mathbb{F}_p$ i.e., injective maps $\psi_0, \psi_1 : \mathbb{F}_p \to \{0,1\}^m$, where $m > 3\log(p)$. We write $\mathbb{G}_0 = \{\psi_0(x) | x \in \mathbb{F}_p\}$ and $\mathbb{G}_1 = \{\psi_1(x) | x \in \mathbb{F}_p\}$. We are given oracles to compute the group operations in both the groups and also a non-degenerate bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. The identity elements in the groups can be accessed by the queries $\psi_0(0)$ and $\psi_1(0)$, while the generators by $\psi_0(1)$ and $\psi_1(1)$. We denote $\psi_0(1)$, $\psi_0(x)$ and $\psi_1(y)$ by g, g^x and $e(g,g)^y$ respectively.

We are also given access to a random oracle to represent the hash function $H : \{0,1\}^* \to \mathbb{G}_0$.

Theorem 2. *Let ψ_0, ψ_1, $\mathbb{G}_0$ and $\mathbb{G}_1$ be defined as above. For any $\mathcal{A}^{CCA}$, let q be the total number of group elements it receives from the oracles and during the interaction with EP-AB-KEM security game. Let $Adv_{\mathcal{A}^{CMA}}$ be the advantage of a polynomial adversary $\mathcal{A}^{CMA}$ against the signature scheme Σ. We have the advantage of $\mathcal{A}^{CCA}$ as $\max\{Adv_{\mathcal{A}^{CMA}}, O(q^2/p)\}$.*

Proof. Note that in the Challenge phase of the EP-AB-KEM security game, the adversary has to distinguish between real symmetric key and a value randomly chosen from symmetric key probability distribution i.e., with respect to our scheme the adversary has to distinguish between $e(g,g)^{\alpha s}$ and $e(g,g)^{\theta}$ for a randomly chosen $\theta \in \mathbb{F}_p$.

At the setup time, the simulation chooses α, β_1, β_2 at random from $\mathbb{F}_p$. If $\beta_1 = \beta_2$, $\beta_1 = 0$ or $\beta_2 = 0$ the setup is aborted just as it would be in the actual construction. The public parameters $h_1 = g^{\beta_1}$, $h_2 = g^{\beta_2}$, $f_1 = g^{\frac{1}{\beta_1}}$, $f_2 = g^{\frac{1}{\beta_2}}$ and $e(g,g)^{\alpha}$ are sent to the adversary. The answers to the queries asked by $\mathcal{A}^{CCA}$ as part of the EP-AB-KEM security game are simulated as below:

H-queries: The simulation maintains a list for the random oracle H with the input and response as entries. When a query is issued to the random oracle with input i, the simulation first checks if there is an entry for i in the list. If there exists an entry, it returns the previously returned response. Otherwise a new random value t_i is chosen from $\mathbb{F}_p$ and the value g^{t_i} is returned. The queries with input vk are answered in the same way.

Extract queries: When the $\mathcal{A}^{CCA}$ makes j-th key generation query on a set of attributes S_j, a new random value $r^{(j)} \in \mathbb{F}_p$ and for each $i \in S_j$ new random value $r_i^{(j)} \in \mathbb{F}_p$ are chosen. The simulator then generates the private key corresponding to S_j as in the scheme. It computes $D = g^{(\alpha + r^{(j)})/\beta_1}$, $E = g^{r^{(j)}/\beta_2}$ and for each $i \in S_j$, $D_i = g^{r^{(j)}} \cdot H(i)^{r_i^{(j)}}$ and $D'_i = g^{r_i^{(j)}}$. The private key is passed onto $\mathcal{A}^{CCA}$.

Decap queries: When $\mathcal{A}^{CCA}$ asks for a decapsulation query on an input encapsulation C, the simulation first parses the access tree $\mathcal{T}'$ from C. It then extracts the verification key vk and the subtree $\mathcal{T}$ from $\mathcal{T}'$. The simulation first verifies the signature on the encapsulation using vk and if it is valid proceeds with decapsulation as follows: It computes F_{vk} and F_x for each leaf node and interior node in $\mathcal{T}$ as specified in the decapsulation algorithm. Note that this can be performed using appropriate queries to ψ_0, ψ_1 and the random oracle H. Finally, $F_{R'}$ is computed and the symmetric key K recovered. Note that as in the decapsulation algorithm if vk was replaced, the simulation would set K to $\perp$. Finally, K is returned.

In the **Challenge** phase, $\mathcal{A}^{CCA}$ outputs a challenge access structure $\mathcal{T}^*$. The simulation does the following: It generates a one-time key pair (sk^*, vk^*) and constructs an access tree $\mathcal{T}^{*'}$ from $\mathcal{T}^*$ and vk^*. It then chooses $s \in \mathbb{F}_p$. It uses the linear secret sharing scheme associated with the access tree $\mathcal{T}^{*'}$ (as described in the scheme) to construct shares λ_i and λ_{vk^*} of s for all relevant attributes i and the verification key vk^*. Note that all the λ_i are chosen uniformly and independently at random from $\mathbb{F}_p$ subject to the condition imposed on them by the linear secret sharing scheme. The choice of λ_i's can be perfectly simulated by choosing l random values $\mu_1, \cdots, \mu_l$ uniformly at random from $\mathbb{F}_p$ for some value l and then letting λ_i be fixed as public linear combination of $\mu_1, \cdots, \mu_l$ and s. Later in proof, we will think of λ_i as such linear combination of these independent random variables.

Finally, the simulation chooses a random $\theta \in \mathbb{F}_p$ and constructs the challenge symmetric key and encapsulation as follows: $K^* = e(g,g)^\theta$ and $C_1^* = h_1^s$. For each relevant attribute i, $C_i^* = g^{\lambda_i}, C_i'^* = g^{t_i \lambda_i}$. For the verification key vk^*, $C_{vk^*} = h_2^{\lambda_{vk^*}}$, $C'_{vk^*} = g^{t_{vk^*} \lambda_{vk^*}}$. Let $\mathcal{C}^* = (\mathcal{T}^{*\prime}, C^*, C_i^*, C_i'^*, C_{vk^*}, C'_{vk^*})$. It then computes a signature σ^* on $\mathcal{C}^*$ using the one-time secret key sk^*. The encapsulation values $(\mathcal{C}^*, vk^*, \sigma^*)$ are sent to $\mathcal{A}^{CCA}$.

Following the generic proof of Boneh et al. [6], we divide the proof into the following two cases:

Case 1: Let Forge be the event that $\mathcal{A}^{CCA}$ submits a decapsulation query with input $(\mathcal{C}, vk, \sigma)$ that is different from the challenge encapsulation given to it but with $vk = vk^*$. We now show that Pr[Forge] is negligible.
With the simulation of $\mathcal{A}^{CCA}$'s queries as described above we now construct a forger $\mathcal{A}^{CMA}$ against the signature scheme. We assume that $\mathcal{A}^{CMA}$ is given the challenge verification key vk^* at the beginning the experiment. As described above, the public parameters are generated and answers to $\mathcal{A}^{CCA}$'s queries are simulated. If $\mathcal{A}^{CCA}$ outputs a query $(\mathcal{C}, vk^*, \sigma)$ even before the **Challenge** phase, then $\mathcal{F}$ outputs $(\mathcal{C}, \sigma)$ as its forgery and stops. Let $(\mathcal{C}^*, vk^*, \sigma^*)$ be the challenge encapsulation given to $\mathcal{A}^{CCA}$. If $\mathcal{A}^{CCA}$ submits a valid encapsulation $(\mathcal{C}, vk^*, \sigma)$ in a decapsulation query, as per the EP-AB-KEM security game we must have $(C, \sigma) \neq (C^*, \sigma^*)$. In this case $\mathcal{A}^{CMA}$ submits $(\mathcal{C}, \sigma)$ as its forgery. Hence, the success probability of $\mathcal{A}^{CMA}$ is at least Pr[Forge]. Since, the one-time signature scheme is assumed to be strongly unforgeable, Pr[Forge] $\leq Adv_{\mathcal{A}^{CMA}}$ must be negligible. Note that in this case (i.e., when Forge occurs), $\mathcal{A}^{CCA}$'s view would have been identical even if we had set $\theta = \alpha s$.

Case 2: In this case, we assume that the event Forge does not occur. We now show that decapsulation queries with an input verification key $vk \neq vk^*$ does not give $\mathcal{A}^{CCA}$ any advantage. Note that since we have assumed that Forge does not occur, a decapsulation query with input $vk = vk^*$ must contain an invalid signature. For such a query $\mathcal{A}^{CCA}$ is returned $\perp$. The rest of the proof below deals with **Case 2**.

When $\mathcal{A}^{CCA}$ makes a query to the group oracles, we may condition on the event that (1) $\mathcal{A}^{CCA}$ provides as input only the values it received from the simulation or intermediate values it obtained as response from the oracles and (2) there are p distinct values in the ranges of both ψ_0 and ψ_1. This event happen with the overwhelming probability of $1 - O(q/p^2)$, where q is the upper bound on the number of queries that can be made during simulation. We may even keep track of the algebraic expressions being called for from the oracles as long as "accidental collisions" do not occur. Specifically, we can think of an oracle query as being a rational function $\nu = \eta/\xi$ in the variables θ, α, β_1, β_2, s, t_i's, $r^{(j)}$'s, $r_i^{(j)}$'s and μ_k's. An accidental collision would be when for queries corresponding to any two distinct formal rational functions $\eta/\xi \neq \eta'/\xi'$, we have that the values of η/ξ and η'/ξ' coincide due to random choices of these independent variables' values.

We now condition that no such accidental collisions occur in either $\mathbb{G}_0$ or $\mathbb{G}_1$. For any pair of distinct queries η/ξ and η'/ξ' within a group, a collision occurs only if the non-zero polynomial $\eta/\xi - \eta'/\xi'$ evaluates to zero. The total degree of this polynomial in our case is at most 5 (a constant). By Schwart-Zippel lemma [23, 25], the probability of this event is $O(1/p)$. By a union bound, the probability that any such collision happens in our simulation is at most $O(q^2/p)$. Hence, we can condition on no such collision happening and still maintain $1 - O(q^2/p)$ of the probability mass.

We now consider what the adversary's view would have been, if we had set $\theta = \alpha s$. In this part of **Case 2** of the proof, subject to the above conditioning, we show that the adversary's view would have been identically distributed. Since we are in the generic group model, where each group element's representation is uniformly and independently chosen, the only way that adversary's view can differ in the case $\theta = \alpha s$ is if there are two queries ν and ν' into $\mathbb{G}_1$ such that $\nu \neq \nu'$ but $\nu|_{\theta=\alpha s} = \nu'|_{\theta=\alpha s}$. Since θ only occurs as $e(g,g)^\theta$ in $\mathbb{G}_1$, the only dependence ν or ν' can have on θ is by having some additive terms of the form $\gamma\theta$ for some constant γ. Therefore we must have $\nu - \nu' = \gamma\alpha s - \gamma\theta$ for some constant $\gamma \neq 0$. We can then artificially add the query $\nu - \nu' + \gamma\theta = \gamma\alpha s$ to the adversary's queries. We will now show that based on the information given to the adversary it can never construct a query for $e(g,g)^{\gamma\alpha s}$.

t_i	λ_i	$\lambda_i t_i$	$r^{(j)} + t_i r_i^{(j)}$
$r_i^{(j)}$	$t_i t_{i'}$	$\lambda_i t_{i'}$	$t_i t_{i'} \lambda_{i'}$
$t_i(r^{(j)} + t_{i'} r_{i'}^{(j)})$	$t_i r_{i'}^{(j)}$	$\alpha + r^{(j)}$	s
$\alpha s + r^{(j)} s$	$r^{(j)}$	$\lambda_i \lambda_{i'}$	$t_i \lambda_i \lambda_{i'}$
$\lambda_{i'}(r^{(j)} + t_i r_i^{(j)})$	$\lambda_{i'} r_i^{(j)}$	$t_i t_{i'} \lambda_i \lambda_{i'}$	$t_i \lambda_i (r^{(j)} + t_{i'} r_{i'}^{(j)})$
$t_i \lambda_i r_{i'}^{(j)}$	$(r^{(j)} + t_i r_i^{(j)})(r^{(j')} + t_{i'} r_{i'}^{(j')})$	$(r^{(j)} + t_i r_i^{(j)}) r_{i'}^{(j')}$	$r_i^{(j)} r_{i'}^{(j')}$
$\mathrm{s}t_{vk}$; λ_{vk}	$t_{vk} t_i$	$t_{vk} t_{vk'}$	$t_{vk} \lambda_i$
$t_{vk} t_i \lambda_i$	$t_{vk} t_{vk'} \lambda_{vk'}$	$t_{vk}(r^{(j)} + t_i r_i^{(j)})$	$t_{vk} r_i^{(j)}$
$t_{vk} \lambda_{vk}$	$t_{vk} t_{vk'} \lambda_{vk} \lambda_{vk'}$	$t_{vk} \lambda_{vk} t_i$	$t_{vk} \lambda_{vk} \lambda_i$
$t_{vk} t_i \lambda_{vk} \lambda_i$	$t_{vk} \lambda_{vk} (r^{(j)} + t_i r_i^{(j)})$	$t_{vk} \lambda_{vk} r_i^{(j)}$	$r^{(j)} \lambda_{vk}$

Table 1. Possible query types into $\mathbb{G}_{01}$ from the adversary

Table 1 enumerates all the possible query types into $\mathbb{G}_1$ by means of the bilinear map and the group elements given to the adversary except for those that contain β_1 or β_2 in every monomial as they will not be relevant for constructing a query involving the term αs. In the table, the variables i and i' are possible attribute strings, j and j' are indices of secret key queries made by the adversary and vk and vk' are the verification keys generated by KeyGen algorithm of the signature scheme. Note that all the possible queries are given in terms of λ_i's, not μ_k's. It can be checked that the query terms in the table can be formed by the adversary from the information available to it. In addition to the polynomials in the table, the adversary also has access to 1 and α. The adversary can query for arbitrary linear combination of these terms. We will now show that no such combination can produce a polynomial of the form $\gamma\alpha s$ for some constant $\gamma \neq 0$.

In Table 1 the only term that contains αs is $\alpha s + r^{(j)} s$, which can be formed by pairing $s\beta_1$ with $\alpha + r^{(j)}/\beta_1$. By such queries, the adversary could create a polynomial of the form $\gamma\alpha s + \sum_{j\in T} \gamma_j s r^{(j)}$ for some set T and constants $\gamma, \gamma_j \neq 0$. To obtain a query polynomial of the form $\gamma\alpha s$ the adversary must add other linear combinations in order to cancels the terms of the form $\sum_{j\in T} \gamma_j s r^{(j)}$. From the table, the only other terms that the adversary has access to that could involve terms of the form $sr^{(j)}$ are obtained by pairing $r^{(j)} + t_i r_i^{(j)}$ with some $\lambda_{i'}$ and also by pairing $\beta_2 \lambda_{vk}$ with $r^{(j)}/\beta_2$. This is so since, $\lambda_{i'}$ and λ_{vk} terms are public linear combinations of $\mu_1, \cdots, \mu_l$ and s. The adversary can create a query polynomial of the form:

$$\gamma\alpha s + \sum_{j\in T}\left(\gamma_j s r^{(j)} + \sum_{(i,i',vk)\in T_{j'}} \lambda_{i'} r^{(j)} + \lambda_{i'} t_i r_i^{(j)} + \lambda_{vk} r^{(j)}\right) + \text{other terms}$$

We now complete the proof with the following case analysis that shows that any of the adversary's query polynomials cannot be of the form $\gamma\alpha s$.

Case 2a: In this case, let us assume that there exists some $j \in T$ such that the set of secret shares $L_j = \{\lambda_{i'}, \lambda_v k : \exists i : (i, i', vk) \in T'_j\}$ do not allow for reconstruction of s. If this is the case, then the term $sr^{(j)}$ will not be cancelled and hence the adversary's query cannot be of the form $\gamma\alpha s$.

Case 2b: Now we assume that for all $j \in T$, the set of secret shares $L_j = \{\lambda_{i'}, \lambda_v k : \exists i : (i, i', vk) \in T'_j\}$ do allow for the reconstruction of the secret s. Fix any $j \in T$. Consider the set of attributes S_j that belongs to the j-th Extract query from the adversary. By the restriction that no requested key should pass the challenge access structure and by the properties of the secret sharing scheme, the set of shares $L'_j = \{\lambda_i : i \in S_j\}$ cannot reconstruct s. Thus, there must exist at least one share $\lambda_{i'}$ in L_j such that $\lambda_{i'}$ is linearly dependent of L'_j when written in terms of s and $\mu_1, \cdots, \mu_l$. Thus for some $i \in S_j$, there must be a term of the form $\lambda_{i'} t_i r_i^{(j)}$ in the adversary's queries. However, it is evident from Table 1 that the adversary has no access to a term of this form. Hence, none of the queries can be of the form $\gamma\alpha s$.

□

D Security Proof of the Generic AB-AKE Protocol

Proof. We prove the theorem in a sequence of games. Let S_i be the event that $\mathcal{A}_{ake}$ wins the AKE-security game in Game i.

Game 0. This is the original AKE-security game as per Definition 4. We have

$$Adv_{\mathcal{A}_{ake}} = |2 \cdot \Pr[S_0] - 1| \tag{4}$$

Game 1. This game is the same as the previous one except that if two different sessions at user U_i output identical message C_i, then the game aborts. Let Repeat be such an event. As there are $\tilde{n}$ users in the protocol, we have

$$|\Pr[S_1] - \Pr[S_0]| \le \tilde{n} \cdot \Pr[\mathsf{Repeat}] \tag{5}$$

As the adversary is allowed to activate at most q_s number of sessions, we have

$$\Pr[\mathsf{Repeat}] \le \frac{q_s^2}{|C|} \tag{6}$$

Game 2. This is the same as the previous game except that a value $t \xleftarrow{R} [1, q_s]$ is chosen. If the Test query does not occur in the t-th session the game aborts and outputs a random value. Let E_2 be the event that the guess is correct.

$$\Pr[S_2] = \Pr[S_2|E_2]\Pr[E_2] + \Pr[S_2|\neg E_2]\Pr[\neg E_2] = \Pr[S_1]\frac{1}{q_s} + \frac{1}{2}\left(1 - \frac{1}{q_s}\right) \tag{7}$$

Game 3. This is identical to the previous game except that the output of each f_{K_i} for $1 \leq i \leq \tilde{n}$ is replaced by a random value chosen uniformly from $\{0,1\}^k$. We have,

$$|\Pr[S_3] - \Pr[S_2]| \leq \tilde{n} \cdot Adv_{\mathcal{A}_{PRF}} \tag{8}$$

Game 4. This game is identical to the previous game except that the queries asked of $\mathcal{A}_{ake}$ are now answered by $\mathcal{A}^{CCA}$, an adversary against the IND-CCA security of the underlying EP-AB-KEM as follows: $\mathcal{A}^{CCA}$ forwards the public parameters that it received from its challenger to $\mathcal{A}_{ake}$. Note that if we allow $\mathcal{A}_{ake}$ to choose the access structure in the Test session, $\mathcal{A}_{ake}$ chooses $\mathbb{A}^*$ and sends it to $\mathcal{A}^{CCA}$ at the beginning of the Test session. Otherwise, $\mathcal{A}^{CCA}$ itself may choose $\mathbb{A}^*$. Once $\mathcal{A}_{ake}$ chooses the Test session, $\mathcal{A}^{CCA}$ gives the challenge access structure $\mathbb{A}^*$ to its challenger. The EP-AB-KEM challenger returns (K_b, C^*) to $\mathcal{A}^{CCA}$ as described in Definition 2. The goal of $\mathcal{A}^{CCA}$ is output whether K_b is encapsulated within C^* or not. $\mathcal{A}^{CCA}$ finally chooses a user U_i^* whose attributes S_i^* satisfy the challenge access structure $\mathbb{A}^*$. With these choices, $\mathcal{A}^{CCA}$ now starts simulating answers to the queries of $\mathcal{A}_{ake}$ as below. Note that we explain only the simulation done in the test session. The queries issued in all the other sessions can be trivially answered by $\mathcal{A}^{CCA}$, since it is allowed to extract private keys corresponding to attributes that satisfy all the access structures except $\mathbb{A}^*$.

$\mathsf{Send}(\pi_i^t, m)$: If m contains only $\mathbb{A}^*$ as per the protocol it has to initiate the test session at U_i. If $U_i = U_i^*$, $\mathcal{A}^{CCA}$ returns the challenge encapsulation C^* as the outgoing message from the instance π_i^t. otherwise, $\mathcal{A}^{CCA}$ runs the $\mathsf{Encapsulation}$ algorithm on behalf of the U_i and obtains the pair (K_i, C_i). It keeps K_i with itself and returns C_i as the outgoing message.
On the other hand, if the message contains an encapsulation C_i, $\mathcal{A}^{CCA}$ proceeds as follows:

1. If $U_i = U_i^*$, it issues a Decap query to its challenger with C_i and the attributes of U_i^* as input. If the challenger returns a key K_i, $\mathcal{A}^{CCA}$ stores K_i and accepts the session. Otherwise, the session is rejected. Note that if $U_i = U_i^*$, then C_i cannot be equal to C_i^* conditioning on the event Repeat in Game 1.
2. If $U_i \neq U_i^*$, $\mathcal{A}^{CCA}$ first checks if $C_i = C_i^*$. If it matches $\mathcal{A}^{CCA}$ accepts the session. Otherwise, as described above it issues Decap query to its challenger with C_i and the attributes of U_i^* as input. Note that the attributes of U_i^* satisfy the access structure $\mathbb{A}^*$ embedded in C_i. If the challenger returns a key K_i, $\mathcal{A}^{CCA}$ stores K_i and accepts the session. Otherwise, $\mathcal{A}^{CCA}$ rejects the session.

$\mathsf{RevealKey}(\pi_i^j)$: Note that a $\mathsf{RevealKey}$ query on the test session is not allowed. In all other sessions $\mathcal{A}^{CCA}$ can answer this query by simply asking Decap query on all the encapsulations exchanged in that session. Since $\mathcal{A}^{CCA}$ is also allowed to extract private keys corresponding to attributes that do not satisfy $\mathbb{A}^*$, it can trivially answer the $\mathsf{RevealKey}$ queries of all the sessions other than the test session.

$\mathsf{Corrupt}(S_i)$: If S_i do not satisfy $\mathbb{A}^*$, then $\mathcal{A}^{CCA}$ can trivially answer this query using the $\mathsf{Extract}$ query available to it as part of EP-AB-KEM game.

$\mathsf{Test}(\pi_i^t)$: $\mathcal{A}^{CCA}$ now embeds the challenge key K_b into the response to $\mathcal{A}_{ake}$. It computes the challenge key $\kappa^* = f_{K_1}(\mathsf{sid}) \oplus \cdots \oplus f_{K_b}(\mathsf{sid}) \oplus \cdots \oplus f_{K_n}(\mathsf{sid})$. Note that, as described in the simulation of Send queries above, all the symmetric other than K_b are either generated by $\mathcal{A}^{CCA}$ or obtained from its challenger via Decap queries. The key κ^* is returned to $\mathcal{A}_{ake}$.

Since the simulation by $\mathcal{A}^{CCA}$ for $\mathcal{A}_{ake}$ is perfect without any aborts, Game 3 and Game 4 are indistinguishable. We have $\Pr[S_4] = \Pr[S_3]$.

Let b' be the output of $\mathcal{A}_{ake}$. $\mathcal{A}^{CCA}$ simply passes this bit onto its challenger. This game is essentially $\mathcal{A}_{ake}$ playing IND-CCA security game against the EP-AB-KEM's challenger. $\mathcal{A}^{CCA}$ succeeds whenever $\mathcal{A}_{ake}$ does so. Hence, the advantage of $\mathcal{A}^{CCA}$ is at least the same as that of $\mathcal{A}_{ake}$. We have

$$|2 \cdot \Pr[S_4] - 1| \leq Adv_{\mathcal{A}^{CCA}} \tag{9}$$

From Equations 4 to 9,we have the claimed advantage for $\mathcal{A}_{ake}$.

□

Remark 3. From Game 4 of the above proof, it is evident that $\mathcal{A}^{CCA}$ obtains the challenge access structure $\mathbb{A}^*$ only at initiation of the Test session. However, $\mathcal{A}^{CCA}$ has to answer the queries asked by $\mathcal{A}_{ake}$ on sessions established prior to the Test session for which $\mathcal{A}^{CCA}$ has to interact with its challenger. As in the selective security model for EP-AB-KEM, if $\mathcal{A}^{CCA}$ commits to an access structure at the start of its game, it cannot simulate answers to the queries asked by $\mathcal{A}_{ake}$. Hence, we need an IND-CCA secure EP-AB-KEM secure in the adaptive model.