International Journal of Security and Its Applications
Vol.9, No.1 (2015), pp.179-190
http://dx.doi.org/10.14257/ijsia.2015.9.1.19

# Universally Composable Attribute-based Group Key Exchange

Hui Xie, Yongjie Yan and Sihui Shu

*School of Mathematics & Computer Science, Jiangxi Science & Technology Normal University, Nanchang 330038, P. R. China*
*Xiehui822@gmail.com*

***Abstract***

*Several protocols implementing attribute-based group key exchange, which allows users with certain set of attributes to establish a session key, have been proposed in recent years. However, attacks on attribute-based group key exchange in current research have been considered only in stand-alone fashion. Thus these protocols may be vulnerable when run with other protocol sessions concurrently. We treat the security of attribute-based group key exchange in the universal composability framework to ensure that a protocol remains secure when run with arbitrary protocol sessions concurrently. More specifically, we define an ideal functionality for attribute-based group key exchange first, then propose a two-round protocol based on a primitive called encapsulation policy attribute-based key encapsulation mechanism. In addition, a complete security proof of our protocol in the universal composability framework under random oracle model is given.*

***Keywords:*** *attribute-based group key exchange; universal composability; random oracle; encapsulation policy attribute-based key encapsulation mechanism; ACK property*

## 1. Introduction

Recently, much attention has been attracted by attribute based cryptography (ABC). In ABC, an identity is a set of descriptive attributes instead of a single identity string. Attribute based encryption (ABE) has significant advantage over the traditional PKC primitives as it achieves flexible one to many encryption. Thus ABE [1-3] provides means for decentralized access control in large and dynamic networks and ubiquitous computing environments without the need of administrating large access control policies. Additionally, ABC achieves better identity privacy preserving since members do not necessarily have to know the identity of the other members with whom they communicate with.

In spite of the advantage of ABE, it is much more convenient and efficient using symmetric-key encryption with a key obtained in a group key exchange (GKE) protocol than using ABE to implement secure group communication in popular group-oriented applications and protocols. To efficiently secure group communication in ABC, we consider a scenario where participants in a GKE aim at obtaining a common session key with partners having certain attributes. In such a scenario, only members whose attributes satisfy the given policy can participate in the GKE without disclosing their identities. Such a GKE protocol is called an attribute-based group key exchange (AB-GKE) protocol.

Steinwandt and Corona proposed an attribute-based group key exchange protocol [5] which achieves forward secrecy recently. Their protocol uses an attribute-based signcryption scheme to authenticate the protocol messages. In independent work, Birkett and Stebila proposed a new concept of predicate-based key exchange [6] which encompasses attribute-based key exchange. In addition, they also proposed a predicate-based key exchange between two parties by combining a secure predicate-based signature scheme [6] with the Diffie-Hellman key exchange protocol [7]. Gorantla et al. proposed a

ISSN: 1738-9976 IJSIA

generic construction of one-round attribute-based group key exchange protocol [8] based on a new notion of encapsulation policy attribute-based key encapsulation mechanism (EP-AB-KEM).

All the protocols mentioned above achieve security only when the protocols are "stand-alone". Thus they may be totally insecure when they run with other protocols. In the framework of universal composability (UC) [9], the security of a protocol is defined by requiring the protocol mimics the behavior of an ideal functionality. The UC formulation allows cryptographic protocols to preserve security under arbitrary protocol composition as in the real world.

Katz and Shin propose a generic compiler [10] which transforms an authenticated secure GKE into a secure GKE in UC framework. An important component in Katz-Shin compiler is a signature scheme to ensure that the last round of messages is un-forgeable. However, deploying signature need to publish participants' identities with their public key. The exposure of participants' identities with their public key leads to privacy leakiness in ABC which is a security requirement in AB-GKE. In addition, attribute based signature (ABS) schemes [11] which are able to preserve privacy cannot ensure the un-forgeability of last round of messages in the Katz-Shin compiler, since every participants in the GKE can generate a valid signature for any other participants in the GKE when all of their attributes satisfy the given policy.

The motivation for the work presented in this paper is to develop a UC secure AB-GKE protocol. In doing so, we build our protocol on top of Gorantla *et al.'s* [8], but extend it to ensure the "ACK-property" [12] which is shown to be essential for proving UC security of key-exchange protocols. As Gorantla *et al.'s* protocol, our protocol is based on a primitive called encapsulation policy attribute-based key encapsulation mechanism (EP-AB-KEM) [8] and follows the framework of ciphertext policy attribute based encryption (CP-ABE) in which the public key is associated with a set of attributes and a policy defined over a set of attributes is attached to the ciphertext. Furthermore, we make use of an additional round of communication to exchange an extra message for each participant so that ACK-property is guaranteed.

## 2. Background

We give a brief overview of EP-AB-KEM introduced in [8], the UC framework, and ACK property in this section.

### 2.1. Encapsulation Policy Attribute-based Key Encapsulation Mechanism (EP-AB-KEM)

The EP-AB-KEM introduced by Gorantla *et al.* and used by our protocol follows the framework of ciphertext policy attribute based encryption (CP-ABE).

An EP-AB-KEM consists of five probabilistic polynomial time algorithm listed below.

(1)Setup: inputs system security parameter and attribute universe description, then outputs system public parameters PK and system master key MK.

(2)Encapsulation: inputs the system master key and a policy $A$ over a set of attributes with the attribute universe $U$, outputs encapsulation $C$ and a symmetric key $SK$. Only the user with attributes satisfying $A$ can recover $SK$ from $C$.

(3)KeyGen: inputs the system master key MK, system public parameters and a set of attributes S, outputs a private key pk for S.

(4)Decapsulation: inputs system public parameters, an encapsulation C, and a private key pk, if pk is the private key of a user with attributes satisfying a policy specified in the computation of encapsulation C outputs a symmetric key SK else outputs an error symbol $\perp$.

 

(5)Delegate: inputs system public parameters, a private key pk corresponding to a set of attributes S, and a set $S^* \subset S$, outputs another private key pk* for S*. Delegate is an optional algorithm.

The security of EP-AB-KEM is a semantic security which ensures that any probabilistic polynomial time adversary cannot distinguish the symmetric key SK generated in encapsulation algorithm and a random value in the probability distribution of the symmetric key. We refer to [8] for more detail.

### 2.2. UC Framework

Canetti introduced the UC framework [9] to design and analyze the security of cryptographic primitives and protocols. In the UC framework, entities involved in the real execution of a protocol are a number of players, an adversary, and in addition an entity called the environment which represents everything outside of the protocol. In the ideal process of UC framework an ideal functionality of the protocol is defined and an adversary in the ideal process is introduced. Essentially, the ideal functionality is a trusted party that produces the desired functionality of the given protocol. Furthermore, the players in ideal process are replaced by dummy players, who do not communicate with each other but transfer messages between the ideal functionality and the environment.

Security is defined from the environment's point of view such that no environment can distinguish the execution of the real protocol and a simulation in the ideal process. More precisely, a protocol securely realizes an ideal functionality if for any real execution adversary, there exists an ideal process adversary such that no environment, on any input, can distinguish whether it is interacting with the real execution adversary and players running the protocol, or with the ideal process adversary and the ideal functionality. We refer to [9] for more detail.

### 2.3. ACK Property

The ACK property is first introduced by Canetti and Krawczyk [12] to implement UC security of two-party key exchange protocols, and is generalized to group key exchange in [10]. Roughly speaking, a GKE protocol has the ACK property if by the time the first player outputs a session key and no player in the session is corrupted, the internal states of all players in the session can be simulated given the session key and messages exchanged among these players.

Consider an interaction among an environment machine $\Omega$, an adversary A, a GKE protocol $\pi$, and an algorithm $I$. $\pi$ runs with $\Omega$ and A; furthermore, A has not corrupted any player when the first player output a session key. Then $I$ is given the session key and messages exchanged among these players, and generates "simulated" internal state for each player in the protocol. Next, the internal state of each player is replaced with the "simulated" internal state for this player, and the interaction of $\Omega$ and A with $\pi$ continues. $I$ is called a good internal state simulator if no probabilistic polynomial time $\Omega$ can distinguish between the above interaction with $\pi$ and $I$ and an interaction with $\pi$ in the real execution.

**Definition 1**. (ACK property) Let $\pi$ be a GKE protocol. $\pi$ is said to have ACK property if there exists a good internal state simulator for $\pi$.

### 2.4. Divisible Computational Diffie-Hellman Problem and Divisible Computational Diffie-Hellman Assumption

Divisible computational Diffie-Hellman (DCDH) problem is a variation of CDH problem. Let $q$ be a large prime number, $G$ be a cyclic additive group of order $q$, and $P$ be a generator of $G$.

DCDH problem: On input $P$, $aP$, and $bP$, where $a,b \in Z_q^*$, outputs $(a/b)P$.

DCDH assumption: There is no probabilistic polynomial time Turing machine solves the DCDH problem with non- negligible probability.

## 3. Universally Composable Attribute-based GKE

To formally define the notion of UC security for attribute-based group key exchange, we present an ideal functionality $\Phi_{\text{AB-ГКЕ}}$ as follows.

Functionality $\Phi_{\text{AB-ГКЕ}}$

$\Phi_{\text{AB-ГКЕ}}$ proceeds as follows, running on security parameter $k$, with players $U_1$, $U_2$,…, $U_n$, and an ideal adversary $\Sigma$.

**Initialization**: Upon receiving (*sid*, *number*, *policy*, $U_i$, *attributes*, *new-session*) from some player $U_i$ for the first time where *number*$\geq$2 represents the number of players involved in this session, *policy* represents a policy that attributes of each player involved should satisfy, and *attributes* represents attributes of this player, checks that whether *attributes* satisfies *policy* or not. If true, records (*sid*, *number*, $U_i$, *policy*) and sends it to $\Sigma$, else aborts.

**Key Generation**: if there are already *number* recorded tuples (*sid*, *number*, $U_i$, *policy*) then stores (*sid*, *policy*, *U*, *ready*) and sends it to $\Sigma$, where *U* is the set of all players involved in this session. Upon receiving a message (*sid*, *policy*, ok) from $\Sigma$ and there is a recorded tuple (*sid*, *policy*, *U*, *ready*), checks if any players in this session is corrupt. If all players in this session are uncorrupted, chooses $\kappa \leftarrow \{0, 1\}^k$ randomly, and stores (*sid*, *policy*, *U*, $\kappa$); if any player in this session is corrupted, waits for $\Sigma$ to send $\kappa$ and then stores (*sid*, *policy*, *U*, $\kappa$).

**Key Delivery**: Upon receiving a message (deliver, $U_i$) from $\Sigma$ where there is a recorded tuple (*sid*, *policy*, *U*, $\kappa$) and $U_i \in U$, sends (*sid*, *policy*, $\kappa$) to player $U_i$.

**Player Corruption**: when $\Sigma$ corrupts $U_i \in U$, (*sid*, $U_i$) is marked as corrupted from this point on. In addition, if there is a recorded tuple (*sid*, *policy*, *U*, $\kappa$) and (*sid*, *policy*, $\kappa$) has not been sent to $U_i$ yet, then $\Sigma$ is given $\kappa$, else $\Sigma$ is given nothing.

**Figure 1. Attribute-based Group Key Exchange Functionality $\Phi_{\text{AB-ГКЕ}}$**

Unlike traditional group key exchange functionality, $\Phi_{\text{AB-ГКЕ}}$ checks whether *attributes* satisfies *policy* or not for each tuple (*sid*, *number*, *policy*, *attributes*, *new-session*). This ensures that only players whose attributes satisfy a given policy can participate in the protocol session. Furthermore, each player only knows the number of players in the session other than the identities of the players in initialization stage, so that the identity privacy is achieved.

Though our ideal functionality only handles a single protocol session, it is easy to use universal composition with joint state [13] to obtain the multi-session extension which handles multiple sessions of the protocol. In addition, focusing on single-session protocols simplifies the definitions and the analysis.

 

Forward secrecy is the notion that corruption of a player should not reveal previous session keys generated by the player. To model forward secrecy, the adversary is not given the session key on corruption of a party if the key has already been delivered to the player in $\Phi_{AB\text{-}\Gamma KE}$.

## 4. UC Secure AB-GKE Protocol

This section shows our attribute-based group key exchange protocol followed with the security proof of the protocol. Our protocol is obtained by enhancing Gorantla *et al.'s* protocol [8] and is composed of two rounds. Each player sends a message to all other players in each round. Our basic strategy is ensuring ACK property by adding a new round of message sending. The new message of each player contains an ephemeral key which is used to ensure the ACK property. Additionally, in order to prevent impersonation attack in the new round of message sending, each player add an "blind" ephemeral value in first message and prove himself a valid player in the protocol by combining this value in the second message.

Since identities of players keep undiscovered in AB-GKE, impersonation-resilient in AB-GKE only guarantees each player participates in the second round of message sending is indeed a player participates in the first round of message sending and each participant in the first round indeed participates in the second round of message sending.

### 4.1. Notations

The notations in the protocol are listed below.

k: system security parameter.

q: a large prime of k bits.

G1,G2: two groups of prime order q. G1 is an additive group.

e:G1×G1→G2: a bilinear map.

P: a generator of the group G1.

f1: {0, 1}*→G1 is a secure hash function that models a random oracle.

f2: {0, 1}*→Zq* is a collision-resistant hash function.

PK : the system public parameters of the EP-AB-KEM in the protocol.

A : the description of a policy.

pki : the private key of a player Ui in the EP-AB-KEM.

(Ci, SKi): the encapsulation pair obtained in encapsulation algorithm run by a player Ui.

### 4.2. Protocol Description

Figure 2 shows our protocol. Let U={U1, U2,…, Un} be the set of players who wish to establish a group key. Before the protocol, the Setup algorithm of an EKP-AB-KEM is executed, and the KeyGen algorithm is executed for each player in the protocol, so that each Ui∈U obtains a private key pki for the EKP-AB-KEM. Unlike the protocol of Gorantla et al. [8], the protocol in Figure 2 has ACK property by using of eki and αP and by erasing internal state before finishing round 2.

Before accepting the group key, each player Ui checks that whether the players who send messages in round 2 are indeed the players who send messages in round 1 and whether the ephemeral key held by other players are valid, so that any uncorrupted

 

players in the same protocol session output the same group session key. Note that we do not take into account the case that the adversary who is a valid player in the protocol impersonates another valid player from the beginning, because identity privacy in ABC makes it impossible for a player to verify the identity of another player in a protocol session. All we can do is to guarantee the players who send messages in round 2 are indeed the players who send messages in round 1. Therefore we also assume broadcast messages sent by all players in round 1 are received by all players without being modified.

Our AB-GKE protocol

**Round 1**.

Each $U_i \in U$ executes encapsulation algorithm of the EP-AB-KEM on the input ($PK$, $\mathcal{A}$) where $PK$ is the system public parameters and $\mathcal{A}$ is the description of a policy that attributes of each player in the protocol should satisfy, and then obtains an encapsulation pair ($C_i$, $SK_i$). In addition, $U_i$ chooses $\alpha_i \in Z_q^*$ randomly and broadcasts ($C_i$, $\alpha_i P$).

**Round 2**.

Let $\alpha$ be a list of $\alpha_i P$ for each $i$, i.e., $\alpha=(\alpha_1 P, \alpha_2 P, \ldots, \alpha_n P)$. Upon receipt of all encapsulations $C_j$ for $i \neq j$, each $U_i$ executes the decapsulation using private key $pk_i$, on each $C_j$ and obtains $SK_j$. $U_i$ computes $K=f_1(SK_1||SK_2||\ldots||SK_n)$, $sid=f_2(C_1||C_2||\ldots||C_n||SK_1||SK_2||\ldots||SK_n)$, $\kappa= f_2(\mathcal{A})K$, and $ek_i=\alpha_i^{-1} sid \cdot K$, erases its internal state other than $\alpha$, $\kappa$, $sid \cdot K$, and $ek_i$, then broadcasts $ek_i$.

**Key computation**.

Upon receipt of all $ek_j$ for $i \neq j$, each $U_i$ verifies that whether $e(\alpha_j P, ek_j)=e(sid \cdot K, P)$. If true for each $j \neq i$, $U_i$ erases its internal state and outputs $\kappa$ as the session key; else $U_i$ terminates without accepting any session key.

**Figure 2. Our AB-GKE Protocol**

### 4.3. Proof of Security

The security proof of our protocol is given in theorem 1. In order to prove the theorem, we introduce four lemmas first. Then the theorem is given based on these lemmas.

**Lemma 1**. Except with negligible probability, if an uncorrupted player $U_i$ outputs a session key $\kappa$, then every uncorrupted player $U_j \in U$ either holds internal state ($\alpha P$, $\kappa$, $sid \cdot K$, $ek_j$) which is held by $U_i$ before it outputs the session key or holds no state and outputs $\kappa$ as session key.

**Proof**. Assume that an uncorrupted player $U_i$ outputs a session key $\kappa$, when some uncorrupted player $U_j \in U$ does not hold internal state ($\alpha P$, $\kappa$, $sid \cdot K$, $ek_j$) and does not output $\kappa$ as its session key.

(1) We prove that $U_j$ holds $ek_j=\alpha_j^{-1} sid \cdot K$ first. According to the description of protocol, $U_i$ outputs a session key after checking that whether $e(\alpha_j P, ek_j)=e(sid \cdot K, P)$. Moreover, $U_j$ broadcasts $ek_j$ when it has already held the internal state ($\alpha P$, $\kappa$, $sid \cdot K$, $ek_j$). Thus there exists an adversary computes the $ek_j$ and sends it to $U_i$ without knowing $\alpha_j$ (if the adversary computes $\alpha_j$, then it solves discrete logarithm) with non-negligible probability $\varepsilon$. If such an adversary A exists, we construct an algorithm B that solves DCDH problem [4]

 

with a non-negligible probability. The algorithm B proceeds as follows when being input $(P_1, P_2)$, where $P_1, P_2 \in G_1$:

B takes A as its subroutine. Moreover, B plays the role of random oracle $f_1$. B simulates an execution of the protocol among $n$ players $U_1, U_2, \ldots U_n$ and chooses all parameters for $U_1, U_2, \ldots U_n$. B chooses one player $U_j$, and sets $\alpha_j P = P_1$. Moreover, when queried $SK_1 \| SK_2 \| \ldots \| SK_n$ to $f_1$, B replies $f_1(SK_1 \| SK_2 \| \ldots \| SK_n) = sid^{-1} \cdot P_2$. If $U_j$ is corrupted by A in the simulation, B aborts, else B takes $ek_j$ computed by A as its answer to the divisible computation Diffie-Hellman problem. Because $\alpha_j$ is chosen randomly in a real execution of the protocol and $f_1$ is a random oracle, B simulates an execution of the protocol and the random oracle $f_1$ perfectly from A's view.

In a non-trivial execution of the protocol, at least one player keeps uncorrupted. Thus the probability that $U_j$ keeps uncorrupted in the above simulation is no less than $1/n$. The probability that B solves DCDH problem on $G_1$ is at least $\varepsilon/n$. This contradicts the DCDH assumption.

Note that it also can be concluded that $sid \cdot K$ held by $U_j$ is equal to the $sid \cdot K$ held by $U_i$ before it outputs the session key, since $ek_j = \alpha_j^{-1} sid \cdot K$.

(2) We then prove $\kappa$ held by $U_j$ is equal to the session key output by $U_i$. Assume $U_j$ holds $\kappa' \neq \kappa$. Since session key $\kappa = f_2(\mathcal{A})K$ and $\mathcal{A}$ is public known before the session execution, $U_i$ and $U_j$ hold different $K$. Assume the $K$ and $sid$ held by $U_j$ be $K^*$ and $sid^*$. Because $sid^* \cdot K^* = sid \cdot K$, $K / K^* = sid^* / sid$. Note that $sid$ and $sid^*$ can be computed before $K$ and $K^*$. Therefore after computing $sid^*$ and $sid$, $K / K^*$ is already fixed. This contradicts that $K$ and $K^*$ are computed from a random oracle.

Furthermore, $\alpha P$ held by $U_j$ is equal to the $\alpha P$ held by $U_i$ before it outputs the session key according to the assumption that broadcast messages sent by all players in round 1 are received by all players without being modified.

In conclusion, $U_j$ either holds internal state $(\alpha P, \kappa, sid \cdot K, ek_j)$ which is held by $U_i$ before it outputs the session key or holds no state and outputs $\kappa$ as session key.

**Lemma 2**. If no player is corrupted in the protocol, any outsiders can not distinguish the session key and a random value in $G_1$ with non- negligible probability.

**Proof**. If no player is corrupted, the session key computed in the protocol is $\kappa = f_2(\mathcal{A})K = f_2(\mathcal{A}) f_1(SK_1 \| SK_2 \| \ldots \| SK_n)$, where $f_1$ is a random oracle. Due to the nature of a random oracle, an outsider who can distinguish $\kappa$ and a random value in $G_1$ must obtain $SK_1$, $SK_2, \ldots$, and $SK_n$. This contradicts the security of EP-AB-KEM.

**Lemma 3**. Except with negligible probability, any uncorrupted players in the same session who output a session key output the same session key.

**Proof**. From lemma 1, whenever an uncorrupted player in a protocol session outputs a session key $\kappa$, every uncorrupted player $U_j \in U$ either holds internal state $(\alpha P, \kappa, sid \cdot K, ek_j)$ or holds no state and outputs $\kappa$ as session key. Thus each uncorrupted player output $\kappa$ as the session key if it outputs the session key.

**Lemma 4**. Our protocol has the ACK property.

**Proof**. When the first player in a protocol session outputs its session key, the internal state of another player in the session, say $U_j$, consists of $(\alpha, \kappa, sid \cdot K, ek_j)$. To prove the ACK property of our protocol, we construct an internal state simulator $I$ as follows. Assume the first player $U_i$ output $\kappa$ as its session key. Given $\kappa$, $I$ chooses $\alpha_1', \alpha_2', \ldots, \alpha_n' \in Z_q^*$ and $\beta \in G_1$ at random, then simulates the internal state for any $U_j$ $(1 \leqslant j \leqslant n, i \neq j)$ as $((\alpha_1' P, \alpha_2' P, \ldots, \alpha_n' P), \kappa, \beta, (\alpha_j')^{-1}\beta)$. Consequently, the verification of $U_j$'s second message is $e((\alpha_j')^{-1}\beta, \alpha_j' P) = e(\beta, P)$ rather than $e(\alpha_j P, ek_j) = e(sid \cdot K, P)$.

We now prove that $I$ is a good internal state simulator. Assume there is a probabilistic polynomial time environment Ω can distinguish the simulated internal states of all players and the internal states of all players in a real execution of the protocol with non-negligible probability. Because $\alpha_1$', $\alpha_2$', …, $\alpha_n$' and $\alpha_1$, $\alpha_2$, …, $\alpha_n$ are all chosen at random in $Z_q^*$ as in a real execution, Ω only distinguishes $\beta$ and $sid \cdot K$. According to the security of EP-AB-KEM, Ω cannot distinguish $SK_i$ and a random value in $Z_q^*$ for $1 \leq i \leq n$. Thus Ω cannot distinguish $sid$ and a random value in $Z_q^*$ because $sid=f_2(C_1||C_2||\ldots||C_n||SK_1||SK_2||\ldots||SK_n)$. This contracts that Ω distinguishes $\beta$ and $sid \cdot K$ with non-negligible probability.

**Theorem 1**. Our AB-GKE protocol securely realizes the ideal functionality $\Phi_{\text{AB-ГКЕ}}$.

**Proof(Sketch)**. Let A be an arbitrary real-life adversary against our protocol. We show how to construct an ideal-process adversary Σ such that no probabilistic polynomial-time environment Z can tell whether it interacts with A and players running our protocol in the real world, or with Σ and players communicating with $\Phi_{\text{AB-ГКЕ}}$ in the ideal process. Σ proceeds as follows:

(1) Σ internally executes a copy of A. Messages from Z to Σ are forwarded to A, and messages from A to environment are forwarded to Z.

(2) Σ generates public and private parameters for system and players, including all public parameters, system master key, and private keys for all players. All public parameters are given to A.

(3) Upon receiving a message (*sid*, *number*, $U_i$, *policy*) from $\Phi_{\text{AB-ГКЕ}}$ for an uncorrupted player $U_i$, Σ invokes internally a simulated copy of the protocol being run by $U_i$ with session ID *sid* and gives the private key for $U_i$ generated in step(2) to this simulated copy. Any messages sent by A to $U_i$ are processed by this simulated copy, and any messages output by the simulated copy are forwarded to A.

(4) When A corrupts a player, say $U_i$, Σ corrupts the corresponding player in the ideal process. Σ gives A private key of $U_i$. In addition, Σ gives A the current internal state of $U_i$ in following way:

ⅰ. If Σ has not sent (*sid*, *policy*, ok) to $\Phi_{\text{AB-ГКЕ}}$ yet, then Σ gives A the internal state of the simulated copy of the protocol being run on behalf of $U_i$.

ⅱ. If Σ has already sent (*sid*, *policy*, ok) to $\Phi_{\text{AB-ГКЕ}}$, but has not sent (deliver, $U_i$) to $\Phi_{\text{AB-ГКЕ}}$ yet, then Σ checks that whether the internal state of the simulated copy of the protocol being run on behalf of $U_i$ includes $\alpha P$, $sid \cdot K$, and $ek_i$. If not Σ aborts; otherwise Σ gives A the internal state of $U_i$ as ( $\kappa$, $\alpha P$, $sid \cdot K$, $ek_i$) where $\kappa$ is the key obtained from $\Phi_{\text{AB-ГКЕ}}$ when Σ corrupts $U_i$ in ideal process.

ⅲ. If Σ has already sent (*sid*, *policy*, ok) and (deliver, $U_i$) to $\Phi_{\text{AB-ГКЕ}}$, an empty internal state is given to A.

(5) When a simulated copy of the protocol being run on behalf of an uncorrupted player, say $U_i$, outputs a session key $\kappa$' , S checks whether any of players have been corrupted and whether it has received a message (*sid*, *policy*, *U*, *ready*) from $\Phi_{\text{AB-ГКЕ}}$, then proceeds as follows:

ⅰ. If no player in the session are corrupted and Σ has not received (*sid*, *policy*, *U*, *ready*) from $\Phi_{\text{AB-ГКЕ}}$, Σ aborts.

ⅱ. If no player in the session are corrupted and Σ has already received (*sid*, *policy*, *U*, *ready*) from $\Phi_{\text{AB-ГКЕ}}$, Σ sends (*sid*, *policy*, ok) and (deliver, $U_i$) to $\Phi_{\text{AB-ГКЕ}}$.

 

iii. If a subset of players, say $\overline{U} \subseteq U/U_i$, are corrupted and Σ has not received (*sid*, *policy*, *U*, *ready*) from $\Phi_{\text{AB-ГКЕ}}$, then Σ sends (*sid*, *number*, *policy*, $U_j$, *attributes*, *new-session*) to $\Phi_{\text{AB-ГКЕ}}$ on behalf of corrupted players who have not done so, where $U_j \in \overline{U}$. If Σ does not receive (*sid*, *policy*, *U*, *ready*) after doing above, Σ aborts. Otherwise Σ sends (*sid*, *policy*, ok), $\kappa$', and (deliver, $U_i$) in sequence to $\Phi_{\text{AB-ГКЕ}}$ after receiving (*sid*, *policy*, *U*, *ready*).

iv. If a subset of players, say $\overline{U} \subseteq U/U_i$, are corrupted, in addition Σ has sent (*sid*, *policy*, ok) to $\Phi_{\text{AB-ГКЕ}}$ already and no player in this session was corrupted at that point in time, then Σ sends (deliver, $U_i$) to $\Phi_{\text{AB-ГКЕ}}$.

v. If a subset of players, say $\overline{U} \subseteq U/U_i$, are corrupted, in addition Σ has sent a session key $\kappa$'' to $\Phi_{\text{AB-ГКЕ}}$ already, then Σ aborts if $\kappa' \neq \kappa''$, otherwise Σ sends (deliver, $U_i$) to $\Phi_{\text{AB-ГКЕ}}$.

**Analysis of the Simulation**. From an environment Z's view, the differences between an interaction with A and with Σ are concluded as follows:

① Step (1), (2), (3), (4) i , and (4) iii introduce no differences from the view of Z.

② In step (4) ii, Σ may abort. In this situation, Σ has already sent (*sid*, *policy*, ok) to $\Phi_{\text{AB-ГКЕ}}$ and the internal state of the simulated copy of the protocol being run on behalf of $U_i$ does not include $\alpha P$, *sid*·*K*, or $ek_i$ . The description of step(5) shows that Σ only sends (*sid*, *policy*, ok) to $\Phi_{\text{AB-ГКЕ}}$ when some uncorrupted player in simulation outputs the session key. It can be obtained from lemma 1 that when some uncorrupted player outputs a session key and uncorrupted $U_i$($U_i \in U$) does not output the session key, $U_i$ does not hold the internal state ($\alpha P$, $\kappa$, *sid*·*K*, $ek_j$) with negligible probability. Thus the probability for Σ to abort in this step is negligible.

③ Difference occurs when Σ aborts in step (5) i . In this situation, Σ has not received (*sid*, *policy*, *U*, *ready*) from $\Phi_{\text{AB-ГКЕ}}$ and all players in the session is uncorrupted. From the description of Σ and $\Phi_{\text{AB-ГКЕ}}$, $\Phi_{\text{AB-ГКЕ}}$ sends (*sid*, *policy*, *U*, *ready*) to Σ once all uncorrupted players in ideal process sends (*sid*, *number*, *policy*, $U_j$, *attributes*, *new-session*) to $\Phi_{\text{AB-ГКЕ}}$; while Σ invokes the simulated copy of the protocol for some uncorrupted player in ideal process, say $U_j$, whenever Σ receives (*sid*, *number*, $U_j$, *policy*) form $\Phi_{\text{AB-ГКЕ}}$.

This means the simulated copy for $U_j$ has not been invoked by Σ when a simulated copy of the protocol for some uncorrupted player outputs a session key. According to the description of $\Phi_{\text{AB-ГКЕ}}$, this occurs only with negligible probability. Thus Σ aborts in this step only with negligible probability.

④ In step (5) ii, the key generated by $\Phi_{\text{AB-ГКЕ}}$ and output by players in ideal process is chosen randomly, not $\kappa$'. If A never corrupts any player in this session, it is computationally indistinguishable whether a player in the ideal process outputs a random session key or a player in the real execution of the protocol outputs the session key $\kappa$' from Z's view according to lemma 2. If A corrupts some players later in this session before these players output the session key, then the situation is identical to step (5) iv.

⑤ Step (5) iii introduces difference when Σ aborts. This only occurs if there is some uncorrupted player in ideal process, say $U_j$, has not sent (*sid*, *number*, *policy*, $U_j$, *attributes*, *new-session*) to $\Phi_{\text{AB-ГКЕ}}$, while a simulated copy of the protocol being run on behalf of an uncorrupted player has outputs a session key. In the light of analysis in ③, this occurs only with negligible probability.

⑥ Step (5)iv may introduce difference when the key generated by $\Phi_{AB\text{-}\Gamma KE}$, $\kappa$ does not equal $\kappa'$. In this situation, Σ obtains the key $\kappa$ from $\Phi_{AB\text{-}\Gamma KE}$ and replaces $\kappa'$ with $\kappa$ as the output session key. Since our protocol has ACK property according to lemma 4, Σ simulates the internal states for all uncorrupted players and replaces the internal states of simulated copies of the protocol being run on behalf of all uncorrupted players with the simulated internal states. According to lemma 4, Z can not tell the difference between the simulated internal states and real internal states of all players with non- negligible probability.

⑦ In step (5)v , Σ may abort. Σ sends a session key $\kappa''$ to $\Phi_{AB\text{-}\Gamma KE}$ only when some simulated copy of the protocol being simulated by Σ on behalf of an uncorrupted player $U_i$ outputs $\kappa''$ previously according to the description of Σ. In addition, lemma 3 indicates that all uncorrupted players who output a session key will output the same session key. Thus $\kappa''=\kappa'$, and Σ aborts in this step with negligible probability.

From above analysis, we can conclude that no probabilistic polynomial-time environment Z can tell whether it interacts with A and players running our protocol in the real world, or with Σ and players communicating with $\Phi_{AB\text{-}\Gamma KE}$ in the ideal process. This completes our sketch of the proof.

## 5. Conclusions

We focus on secure AB-GKE protocol in UC framework in this paper. After presenting an ideal functionality of AB-GKE in UC framework, we propose a two-round AB-GKE which securely realizes the functionality. Our protocol is constructed from any secure EP-AB-KEM scheme follows the framework of CP-ABE. According to the security analysis, our protocol is universally composable.

## Acknowledgments

These authors would like to thank the anonymous reviewers for their valuable and constructive comments. The work was sponsored by the National Natural Science Foundation of China (Grant No. U1304606), the Fund for Humanities and Social Sciences in Jiangxi Province, China (No. 14TQ07), and the Research Foundation for Humanities and Social Sciences at Universities in Jiangxi Province, China (No. JC1428).

## References

[1] A. Sahai and B. Waters, “Fuzzy identity-based encryption”, Proceedings of the EUROCRYPT 2005, LNCS, vol. 3494, pp. 457–473.
[2] H. Lin, Z. Cao, X. Liang and J. Shao, “Secure threshold multi authority attribute based encryption without a central authority”, Proceedings of INDOCRYPT 2008, Kharagpur, India, (**2008**) December.
[3] M. Chase, “Multi-authority attribute based encryption”, Proceedings of TCC’07, Amsterdam, Netherlands, (**2007**) February, pp. 515–534.
[4] F. Bao, R. Deng and H. Zhu, “Variations of Diffie–Hellman problem”, in: Proceedings of ICICS’03, LNCS 2836, Springer-Verlag, (**2003**), pp. 301-312.
[5] R. Steinwandt and A. S. Corona, “Attribute-based group key establishment”, Advances in Mathematics of Communications, vol. 4, no. 3, (**2010**), pp.381–398.
[6] J. Birkett and D. Stebila, “Predicate-Based Key Exchange”, Proceedings of ACISP 2010, Sydney, Australia, (**2010**) July.
[7] W. Diffie and M. E. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, vol. IT-22, (**1976**), November, pp. 644–654.
[8] M. C. Gorantla, C. Boyd, and J. M. G. Nieto, “Attribute-based Authenticated Key Exchange”, Proceedings of ACISP 2010, Sydney, Australia, (**2010**) July.
[9] R. Canetti, “Universally Composable Security”, A New Paradigm for Cryptographic Protocols’,
[10] Manuscript dated Jan. 28, 2005, available at http://eprint.iacr.org/2000/067.

 

[11] J. Katz and J. Shin, "Modeling insider attacks on group key-exchange protocols", in: Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05), Alexandria, USA, (**2005**), pp. 180–189.
[12] S. F. Shahandashti and R. Safavi-Naini, "Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems", In: Proceedings of AFRICACRYPT 2009, Gammarth, Tunisia, (**2009**) June, pp. 198-216.
[13] R. Canetti and H. Krawczyk, "Universally Composable Notions of Key Exchange and Secure Channels", Proceedings of Eurocrypt 2002, Amsterdam, Holland, vol. 2332, (**2002**) April, pp. 337–351.
[14] R. Canetti and T. Rabin, "Universal Composition with Joint State", Proceedings of Crypto 2003, Santa Barbara, USA, Lecture Notes in Computer Science, Springer, Berlin, vol. 2729, (**2003**), pp. 265–281.