Attribute-based Optimistic Fair Exchange: How to Restrict Brokers with Policies

Yang Wang\textsuperscript{a}, Man Ho Au\textsuperscript{a}, Willy Susilo\textsuperscript{a,1,*}

\textsuperscript{a}Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia

Abstract

Optimistic fair exchange (OFE) is a kind of protocols for solving the fair exchange problem between two participants with the help of an arbitrator that only needs to be involved when dispute occurs. As far as we are concerned, all previous work on OFE does not take into account user’s attributes such as nationality and age. We identify that in some applications, the attributes could play an important role in the exchange to take place, and OFE may not be suitable to these scenarios. We introduce a new notion named attribute-based optimistic fair exchange (ABOFE) to solve the fair exchange problem in the attribute-based setting. We formalize the notion of ABOFE and present a security model in the multi-user setting under the chosen-key attack. We also present a generic construction of ABOFE from existing cryptographic primitives and prove that our proposal is secure with respect to our definition in the standard model. An instantiation in the standard model is discussed.

Keywords: Optimistic fair exchange, Attribute-based encryption, Policy, Standard model

\textsuperscript{*}Corresponding author.

Email addresses: yw990@uowmail.uow.edu.au (Yang Wang), aau@uow.edu.au (Man Ho Au), wsusilo@uow.edu.au (Willy Susilo)

\textsuperscript{1}This work is supported by ARC Future Fellowship FT0991397.

Preprint submitted to Theoretical Computer Science November 27, 2013
1. Introduction

The brokerage business model has been used since the pre-Internet era, where the intermediary buys from the supplier (or producer) and owns the goods first, and then sells it. This model plays a very important role in the online business nowadays, as it enables fast and secure transactions without relying on a single merchant’s connection. Brokers have been known to be active in business-to-business (B2B), business-to-consumer (B2C) and consumer-to-consumer (C2C) or peer-to-peer (P2P) markets. Although this model is known to be very useful and practical, some issues have happened since the broker may incorporate some certain strategies to increase his/her sales, and it may damage even the supplier.

Consider the following real life case study. A broker $B$ buys games from the game developer $G$ and sells these games to its customer. Since $B$ is considered as a broker, the price that has been set to $B$ is certainly lower than the retail price of the game itself. In order to maximise its sales, $B$ is happy to make its margin to be very low, and this way $B$ will gain popularity among the customers and attract more buyers. In particular, this is certainly possible if $B$ purchases the games from different countries, since usually the price for each country will be different, due to the sales tax applied. This action is certainly damaging the market and $G$ will not be able to sell that particular game with the retail price to the customers as they would have preferred to acquire it from $B$ instead. This issue can be solved by placing required policies for the brokers. First of all, each broker must be licensed in order to sell the games. The license is limited to the country of residence. The game will be playable, if and only if, the required CD key sold by the broker matches with the country where the game will be played. To give an example, someone resides in the US will purchase a CD key from the broker. First of all, he/she needs to be sure that the broker is a licensed retailer for US. Once the CD key is issued by the broker, this CD key is only usable if it is used in the US and not in other countries. Another scenario involves limitation of the age restriction towards the game itself. For instance, consider the game “God of War” that is restricted to people over 18 years old. A US online reseller sells the activation key (or CD key) for this game. This price is lower than the price of the same game available in Canada. There are two issues need to be solved here. First, the buyer needs to be ensured that the reseller is a genuine reseller, and the reseller needs to be sure that the buyer is at least 18 years old. Subsequently, the activation key sold can only be
used in the US and not elsewhere. Hence, the buyer from Canada will not be able to make use of this particular activation key.

Our Approach. In this paper, we intend to present a cryptographic primitive that aims to solve the above scenario. We make use of the notion of optimistic fair exchange (OFE) and enhance this notion to enable policies for both parties, namely the seller and the buyer (or the signer and the verifier, resp.). A trivial solution for enabling the above scenario would be that prior to involving an OFE protocol, each participant will simply present an evidence, for instance a copy of the identity card or license, to the other participant, hence confirming that they satisfy the requirements. Nevertheless, this solution compromises the privacy of the users, and therefore it is not ideal. Furthermore, providing such kind of evidence over the network securely is rather challenging as well.

1.1. Optimistic Fair Exchange

Optimistic fair exchange (OFE), first introduced by Asokan, Schunter and Waidner [1], is a kind of protocols aiming to guarantee fairness for two parties Alice and Bob exchanging digital items. An trusted third party named “arbitrator” is needed in OFE, but involves only when there is a dispute between exchanging parties. Since a large number of digital items such as electronic checks and electronic airline tickets are implemented as digital signatures, the optimistic fair exchange of digital signatures constitutes an important part of any business transaction.

In a typical execution of optimistic fair exchange of digital signatures, Alice first sends a partial signature to Bob. Bob verifies the validity of Alice’s partial signature, and sends its full signature to Alice, after which Alice sends her full signature back to Bob and completes the exchange. In the case there is a network failure or Alice attempts to cheat by refusing to send her own full signature, the arbitrator will convert Alice’s partial signature into a full one and send it back to Bob. It is implicitly assumed that Bob should offer his own full signature when asking the arbitrator for help, and the arbitrator will send Bob’s full signature to Alice later. Thus at the end of this exchange, either both Alice and Bob gain the other’s full signature, or neither does. Thus the exchange is fair.

As a useful tool in applications such as contract signing and electronic commerce, OFE has been extensively researched [2, 3, 4, 5, 6, 7, 8] since its introduction. Several primitives are useful for the construction of OFE,
including verifiably encrypted signatures [9, 10, 11, 12, 13, 14, 15], and sequentially two-party multisignatures [16]. It was further showed that OFE can be constructed from OR signature [17], and from conventional signatures and ring signatures [18].

In an orthogonal dimension, formal OFE security models [17, 18] are proposed. While most optimistic fair exchange protocols are studied in the certified-key model (also known as the registered-key model [19]) in which the adversary is only allowed to make queries with respect to the registered public keys, the security of optimistic fair exchange in the multi-user setting and chosen-key model [18], where the adversary can choose its public key arbitrarily probably without knowing the corresponding private keys, is more desirable.

1.2. Our Contribution

In this paper, motivated by idea of attribute-based encryption (ABE) [20] and ciphertext policy attribute-based encryption (CPABE) [21], we introduce the notion of attribute-based optimistic fair exchange (ABOFE), which can be viewed as an extension of OFE, as a practical cryptographic solution to the aforementioned scenario.

In ABOFE, each user satisfying a set of attributes is assigned a credential by the credential center. This allows the signer Alice to generate a credential-protected package in such a way that only verifiers that possess appropriate credentials can convert it into a full signature, which naturally guarantees that the verifier should satisfy some particular set of attributes.

We propose the syntax and also define a security model for ABOFE in the multi-user setting under chosen-key attack. Our model captures the existing security requirements for OFE, namely, security against signers, security against verifiers and security against the arbitrator. As suggested by the respective names, they intend to cover the scenarios when the named party is dishonest.

Finally, we propose a generic construction of ABOFE from the two well established cryptographic primitives, OFE and CP-ABE, and provide the security proof of our proposal in the proposed model. Our generic construction works in the standard model and does not involve any extra assumptions. The efficiency of an instantiation about the generation construction is also discussed.
1.3. Paper Organization

In the next section, we review the notions and security models of OFE and CP-ABE, respectively. In Section 3, the syntax of ABOFE and its security definitions are presented. Then we present our construction in Section 4. An instantiation is discussed in Section 5. Finally, we conclude our paper in Section 6.

2. Preliminaries

Throughout the paper, the following notations are used. For a finite set $S$, $s \leftarrow S$ denotes that an element is randomly chosen from $S$. By $y \leftarrow A^O(x)$, we mean the algorithm $A$, on input $x$ and having access to oracle $O$, outputs $y$. By $x := y$, we mean variable $x$ is assigned with the value of $y$.

Definition 1 (Access structure [22]). Let $\{P_1, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, \cdots, P_n\}}$ is monotone if for any $B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \subseteq \mathbb{A}$. An access structure (respectively, monotonic access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, \cdots, P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.

In this paper, the attributes will take the role of the parties and by an access structure, we mean a monotone access structure. When we say a set $S$ satisfies an access structure $\mathbb{A}$, we mean $S \in \mathbb{A}$.

2.1. Ciphertext-Policy ABE

We review the notion and security model of ciphertext-policy attribute-based encryption introduced in [23]. A ciphertext-policy attribute-based encryption scheme $\mathcal{E}$ comprises four efficient algorithms: $\mathcal{E} := (\text{Setup}, \text{Encrypt}, \text{KeyGen}, \text{Decrypt})$.

Setup$(1^k, U)$. The setup algorithm takes as input security parameter $1^k$ and attribute universe description $U$, and outputs the public parameters PM and a master key MK.

Encrypt$(\text{PM}, M, \mathbb{A})$. The encryption algorithm takes as input the public parameters PM, a message $M$, and an access structure $\mathbb{A}$ over the universe of attributes, and outputs a ciphertext CT such that only a user that possesses
a set of attributes that satisfies the access structure $A$ will be able to decrypt the ciphertext. It is assumed that the ciphertext implicitly contains $A$.

**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 $SK$.

**Decrypt**($PM, CT, SK$). The decryption algorithm takes as input the public parameters $PM$, a ciphertext $CT$, which contains an access policy $A$, and a private key $SK$, which is a private key for a set of attributes $S$. If the set of attributes $S$ satisfies the access structure $A$ then the algorithm will decrypt the ciphertext and returns a message $M$.

Correctness for a CP-ABE scheme $\mathcal{E}$ states that $\text{Decrypt}(PM, \text{Encrypt}(PM, M, A), \text{KeyGen}(MK, S)) = M$, for any set of attributes $S$ that satisfies the access structure $A$.

### 2.1.1. Security Model.

The full security for CP-ABE [23] is described by a security game between a challenger and an adversary. The game proceeds as follows.

**Setup** The challenger runs **Setup** algorithm and sends the public parameters $PM$ to the adversary.

**Phase 1** The adversary adaptively queries the challenger for private keys corresponding to sets of attributes $S_1, \cdots, S_{q_1}$. For the query with respect to $S_k$, the challenger responds with a secret key outputted by **KeyGen**($MK, S_k$).

**Challenge** The adversary submits two equal length messages $M_0$ and $M_1$, and an access structure $A$ such that none of the previous sets $S_1, \cdots, S_{q_1}$ satisfies the access structure. The challenger flips a random coin $b \in \{0, 1\}$, and encrypts $M_b$ under $A$, producing $CT$. It sends $CT$ to the attacker.

**Phase 2** The adversary adaptively queries the challenger for private keys corresponding to sets of attributes $S_{q_1+1}, \cdots, S_q$, with the restriction that none of these satisfies $A$. For the query with respect to $S_k$, the challenger responds with a secret key outputted by **KeyGen**($MK, S_k$).

**Guess** The adversary outputs a guess $b' \in \{0, 1\}$.

The advantage of an adversary in this game is defined as $\Pr[b' = b] - \frac{1}{2}$. 
Definition 2. A ciphertext-policy attribute-based encryption scheme is fully
secure if all polynomial time adversaries have at most a negligible advantage
in the above security game.

2.2. Optimistic Fair Exchange

We review the notion and security model of the optimistic fair exchange
protocol introduced in [24].

Definition 3. An optimistic fair exchange scheme involves the users (sign-
ners and verifiers) and the arbitrator, and is formalized by the following (prob-
abilistic) polynomial-time algorithms:

- Setup\textsuperscript{TTP}: On input a security parameter \(1^k\), the algorithm generates a
  secret key ASK, and a public key APK of the arbitrator.

- Setup\textsuperscript{User}: On input \(1^k\) and (optionally) APK, it outputs a secret/public
  key pair (SK, PK). For a user \(U_i\), we use \((SK_i, PK_i)\) to denote the
  user’s key pair.

- Sig and Ver: \(\text{Sig}(M, SK_i, APK)\), outputs a (full) signature \(\sigma\) on mes-
  sage \(M\) of user \(U_i\), while \(\text{Ver}(M, \sigma, PK_i, APK)\) outputs \(\top\) or \(\bot\), in-
  dicating \(\sigma\) is \(U_i\)’s valid full signature on \(M\) or not, respectively.

- PSig and PVer: These are partial signing and partial verification al-
  gorithms respectively. \(\text{PSig}(M, SK_i, APK)\) outputs a partial signature
  \(\sigma_P\), while \(\text{PVer}(M, \sigma_P, PK_i, APK)\) outputs \(\top\) or \(\bot\).

- Res: This is the resolution algorithm. \(\text{Res}(M, \sigma_P, ASK, PK_i)\) outputs
  a full signature \(\sigma\), or \(\bot\) indicating the failure of resolving a partial
  signature.

Correctness property states that

- \(\text{Ver}(M, \text{Sig}(M, SK_i, APK), PK_i, APK) = \top\),
- \(\text{PVer}(M, \text{PSig}(M, SK_i, APK), PK_i, APK) = \top\),
- \(\text{Ver}(M, \text{Res}(M, \text{PSig}(M, SK_i, APK), ASK, PK_i, APK), PK_i, APK) = \top\).

Resolution ambiguity property states that
• any “resolved signature” \( \text{Res}(M, \text{PSig}(M, SK_i, APK), ASK, PK_i) \) is computationally indistinguishable from the “actual signature” \( \text{Sig}(M, SK_i, APK) \).

The security of an optimistic fair exchange scheme consists of three aspects: security against signers, security against verifiers, and security against the arbitrator. The security models of them in the multi-user setting and chosen-key model are reviewed below.

SECURITY AGAINST SIGNERS. We require that any PPT adversary \( \mathcal{A} \) succeeds with at most negligible probability in the following experiment.

\[
(ASK, APK) \leftarrow \text{Setup}^{\text{TTP}}(1^k) \\
(M, \sigma_P, PK_A) \leftarrow A^{O_{\text{Res}}}(APK) \\
\sigma \leftarrow \text{Res}(M, \sigma_P, ASK, PK_A) \\
\text{success of } \mathcal{A} := [\text{PVer}(M, \sigma_P, PK_A, APK) = \top] \\
\wedge [\text{Ver}(M, \sigma, PK_A, APK) = \bot]
\]

where the resolution oracle \( O_{\text{Res}} \) takes as input a valid partial signature \( \sigma_P \) of user \( U_i \) on message \( M \) (i.e. \( (M, \sigma_P, PK_i) \) such that \( \text{PVer}(M, \sigma_P, PK_i, APK) = \top \)) and outputs a full signature \( \sigma \) on \( M \) under \( PK_i \). In other words, no signer should be able to produce a partial signature that looks good to a verifier but cannot be resolved to a full signature by the honest arbitrator.

SECURITY AGAINST VERIFIERS. We require that any PPT adversary \( \mathcal{A} \) succeeds with at most negligible probability in the following experiment.

\[
(ASK, APK) \leftarrow \text{Setup}^{\text{TTP}}(1^k) \\
(SK_A, PK_A) \leftarrow \text{Setup}^{\text{User}}(APK) \\
(M, \sigma) \leftarrow A^{O_{\text{Sig}}, O_{\text{Res}}}(APK, PK_A) \\
\text{success of } \mathcal{A} := [\text{Ver}(M, \sigma, PK_A, APK) = \top] \\
\wedge (M, \cdot, PK_A) \notin \text{Query}(\mathcal{A}, O_{\text{Res}})]
\]

where oracle \( O_{\text{Res}} \) is described in the previous experiment, the partial signing oracle \( O_{\text{PSig}} \) takes a message \( M \) as input and outputs a partial signature \( \sigma_P \) on \( M \) under \( PK_A \), and \( \text{Query}(\mathcal{A}, O_{\text{Res}}) \) is the set of queries made by \( \mathcal{A} \) to oracle \( O_{\text{Res}} \). In other words, no verifier should be able to complete any partial signature \( \sigma_P \) into a full signature, without explicitly asking the arbitrator to do so. Note that there is no need to provide \( \mathcal{A} \) with the signing oracle \( O_{\text{Sig}} \).
as the resolution ambiguity property guarantees that it can be simulated by $O_{\text{PSig}}$ and $O_{\text{Res}}$.

SECURITY AGAINST THE ARBITRATOR. We require that any PPT two-stage adversary $A = (A_1, A_2)$ succeeds with at most negligible probability in the following experiment.

\[
\begin{align*}
(APK, ASK^*) &\leftarrow A_1(1^k) \\
(SK_A, PK_A) &\leftarrow \text{Setup}^{\text{User}}(APK) \\
(M, \sigma) &\leftarrow A_2^{\text{Opsig}}(ASK^*, APK, PK_A) \\
\text{success of } A &:= [\text{Ver}(M, \sigma, PK_A, APK) = \top \\
&\quad \land M \notin \text{Query}(A, O_{\text{PSig}})]
\end{align*}
\]

where $ASK^*$ is $A_1$’s state information, which might not be the corresponding private key of $APK$, oracle $O_{\text{PSig}}$ is described in the previous experiment, and $\text{Query}(A, O_{\text{PSig}})$ is the set of queries made by $A$ to oracle $O_{\text{PSig}}$. In other words, the arbitrator should not be able to produce a full signature without explicitly asking the signer to generate a partial one.

3. Attribute-based Optimistic Fair Exchange

We adapt the definitions and security models of OFE from various literatures for our attribute-based OFE. In ABOFE, besides the traditional secret/public key pair, each user also possesses a credential, generated by the credential center, corresponding to a set of attributes the user satisfies. We require that a user that does not possess a set of attributes should not be able to gain other users’ full signatures. Thus a full signature will not be sent by the signer in ABOFE, which is different from the case in OFE.

With this in mind, we introduce the the algorithms of which ABOFE scheme consists.

**Definition 4.** An attribute-based optimistic fair exchange scheme involves the users, the credential center and the arbitrator, and consists of the following (probabilistic) polynomial-time algorithms:

- **PMGen:** On input $1^k$ and attribute universe description $U$ where $k$ is a security parameter, this algorithm outputs a system parameter $PM$ and a credential secret key $CK$. 

---

9
• **Setup**<sup>TTP</sup>: On input $1^k$, the algorithm generates a secret key $ASK$, and a public key $APK$ of the arbitrator.

• **Setup**<sup>User</sup>: On input $1^k$ and (optionally) $APK$, it outputs a secret/public key pair $(SK, PK)$. For a user $U_i$, we use $(SK_i, PK_i)$ to denote the user’s key pair.

• **Setup**<sup>Cred</sup>: On input the credential secret key $CK$ and a set of attributes $S$, the attribute key generation algorithm outputs a credential $CD_S$ corresponding to the set of attributes $S$.

• **Tran**<sub>P</sub>: The transaction promise generation algorithm $Tran_{P}(m_1, m_2, SK_i, APK, PM, A_1, A_2)$ takes as input two messages $m_1$ and $m_2$, respectively, user $U_i$’s secret key $SK_i$, the arbitrator’s public key $APK$, the system parameter $PM$ and two access structure $A_1$ and $A_2$ over the universe of attributes, respectively. This algorithm outputs a transaction promise $\omega$.

• **TPVer**: The transaction promise verification algorithm $TPVer(m_1, m_2, \omega, PK_i, APK, PM, A_1, A_2)$ takes as input two messages $m_1$ and $m_2$, respectively, a transaction promise $\omega$, user $U_i$’s public key $PK_i$, the arbitrator’s public key $APK$, the system parameter $PM$, and two access structure $A_1$ and $A_2$, respectively. The algorithm outputs $\top$ indicating valid or $\bot$ indicating invalid.

• **Tran**<sub>S</sub>: The transaction for signer algorithm $Tran_{S}(m, SK_i, APK, PM, A)$ takes as input a message $m$, user $U_i$’s secret key $SK_i$, the arbitrator’s public key $APK$, the system parameter $PM$, and an access structure $A$. The algorithm outputs a credential-protected package $\pi$.

• **Tran**<sub>V</sub>: The transaction for verifier algorithm $Tran_{V}(m, \pi, PK_i, APK, PM, CD_S)$ takes a message $m$, a credential-protected package $\pi$, user $U_i$’s public key $PK_i$, the arbitrator’s public key $APK$, the system parameter $PM$, and a credential $CD_S$. This algorithm outputs a full signature $\sigma$ or $\bot$ indicating failure.

• **Ver**: The full signature verification algorithm $Ver(m, \sigma, PK_i, APK)$ outputs $\top$ indicating valid or $\bot$ indicating invalid.

• **Res**<sup>User</sup>: The resolution for user algorithm $Res_{User}(m_1, m_2, \omega, SK_j, PK_i, APK, PM, A_1, A_2)$ takes as input two messages $m_1$ and $m_2$, respectively,
Correctness states that, for \( PMGen(1^k) \rightarrow (PM, CK) \), \( Setup^{TTP}(1^k) \rightarrow (ASK, APK) \), \( Setup^{User}(1^k) \rightarrow (SK_i, PK_i) \), \( Setup^{User}(1^k) \rightarrow (SK_j, PK_j) \), \( Tran_p(m_1, m_2, SK_i, APK, PM, A_1, A_2) \rightarrow \omega \), \( Tran_s(m_1, SK_i, APK, PM, A_2) \rightarrow \pi_1 \), \( Tran_s(m_2, SK_j, APK, PM, A_1, A_2) \rightarrow \omega \), \( Res_{TTP}(m_1, PK_i, A_2, request, ASK) \rightarrow (\tilde{\pi}_1, \tilde{\pi}_2) \), \( Setup^{Cred}(CK, S) \rightarrow CD_S \) where the sets of attributes \( S \) satisfies the access structures \( A_1 \), and \( Setup^{Cred}(CK, S') \rightarrow CD_{S'} \) where the sets of attributes \( S' \) satisfies the access structures \( A_2 \), the following conditions hold:

- TPVer\((m_1, m_2, \omega, PK_i, APK, PM, A_1, A_2) = \top \).
- \( Ver(m_1, Tran_v(m_1, \pi_1, PK_i, APK, PM, CD_{S'}), PK_i, APK) = \top \).
- \( Ver(m_2, Tran_v(m_2, \pi_2, PK_j, APK, PM, CD_S), PK_j, APK) = \top \).
- \( Ver(m_1, Tran_v(m_1, \tilde{\pi}_1, PK_i, APK, PM, CD_{S'}), PK_i, APK) = \top \).
- \( Ver(m_2, Tran_v(m_2, \tilde{\pi}_2, PK_j, APK, PM, CD_S), PK_j, APK) = \top \).

Resolution ambiguity states that, any “resolved credential-protected packages” \( \tilde{\pi}_1 \) and \( \tilde{\pi}_2 \) outputted by \( Res_{\text{Abr}} \) are computationally indistinguishable from the “actual credential-protected packages” \( \pi_1 \) and \( \pi_2 \) generated by \( Tran_s \), respectively.

3.1. A Typical Usage of ABOFE

For simplicity, we will describe a typical usage of ABOFE between two users Alice and Bob. Before the exchange phase, the credential center runs \( PMGen \) to set up the public parameters and keep the credential secret key as private. Alice and Bob acquire their respective credentials generated by \( Setup^{Cred} \) from the credential center. Besides, the arbitrator sets its key.
pair by invoking $\text{Setup}^{\text{TP}}$. Alice and Bob generate their own key pairs by invoking $\text{Setup}^{\text{User}}$, respectively.

Suppose Alice intends to exchange her own full signature $\sigma_1$ on message $m_1$ for Bob’s full signature $\sigma_2$ on message $m_2$. Besides, Alice should satisfy an access structure $A_1$, and Bob should satisfy an access structure $A_2$. The exchange phase consists of three steps.

1. Alice invokes $\text{Tran}_P$ to generate a transaction promise $\omega$ on message $m_1$ and $m_2$ with respect to access structures $A_1$ and $A_2$. Alice sends $\omega$ to Bob.
2. Bob invokes $\text{TPVer}$ to ensure the validity of $\omega$. Bob then invokes $\text{Tran}_S$ to generate a credential-protected package $\pi_B$ on message $m_2$ with respect to access structure $A_1$. Bob sends $\pi_B$ to Alice.
3. If satisfying the access structure $A_1$, Alice can invoke $\text{Tran}_V$ to gain a full signature $\sigma_B$ of Bob’s. Then Alice invokes $\text{Tran}_S$ to generate a credential-protected package $\pi_A$ on message $m_1$ with respect to access structures $A_2$. Alice sends $\pi_A$ to Bob, who can invoke $\text{Tran}_V$ to gain Alice’s full signature $\sigma_A$ if satisfying the access structure $A_2$.

Note that in the normal cases, the exchange will finish and both Alice and Bob can gain the full signature of the other’s. In the case Alice refuses to send her credential-protected package $\pi_A$ in the third step, or that credential-protected package is created with respect to improper access structures, Bob can approach the arbitrator for assistance. Specifically, he approaches the arbitrator and sends a resolution request generated by invoking $\text{Res}^{\text{User}}$. To make a resolution, the arbitrator invokes $\text{Res}^{\text{TP}}$ to generate two credential-protected packages $\tilde{\pi}_1$ and $\tilde{\pi}_2$. $\tilde{\pi}_1$ is sent to Bob, and $\tilde{\pi}_2$ is sent to Alice. Thus both Alice and Bob can invoke $\text{Tran}_V$ to gain the other’s full signature if satisfying the expectant attributes.

### 3.2. Security model

We modify the traditional OFE model to make it suitable in the attribute-based setting.

- **Security Against Signers**: This property guarantees that a transaction promise generated by a signer can always be resolved to a full signature of the signer’s if the arbitrator honestly makes a resolution and the verifier satisfies the expectant attributes. Formally, we consider
the following experiment, in which the adversary $A$ models a dishonest signer that can even have access to the credential secret key and the verifier’s secret key.

Experiment $SAS$:

\[
\begin{align*}
(PM, CK) & \leftarrow \text{PMGen}(1^k, U) \\
(ASK, APK) & \leftarrow \text{Setup}^{\text{TTP}}(1^k) \\
(SK_B, PK_B) & \leftarrow \text{Setup}^{\text{User}}(1^k) \\
(m_1^*, m_2^*, \omega^*, PK_A, A_1, A_2, S) & \leftarrow A^{\text{ORes}}(PM, CK, APK, SK_B, PK_B) \\
CD_S & \leftarrow \text{Setup}^{\text{Cred}}(CK, S) \\
\text{request}^* & \leftarrow \text{Res}_{\text{User}}(m_1^*, m_2^*, \omega^*, SK_B, APK, PM, A_1, A_2) \\
(\pi_1^*, \pi_2^*) & \leftarrow \text{Res}^{\text{TTP}}(m_1^*, PK_A, A_2, \text{request}^*, ASK) \\
\sigma^* & \leftarrow \text{Tran}_V(m_1^*, \pi_1^*, PK_A, APK, PM, CD_S) \\
\text{success of } A & := [\text{TPVer}(m_1^*, m_2^*, \omega^*, PK_A, APK, PM, A_1, A_2) = \top \\
& \quad \land \text{Ver}(m_1^*, \sigma^*, PK_A, APK) = \bot]
\end{align*}
\]

where $S$ is a set of attributes that satisfies the access structure $A_2$, the resolution for arbitrator oracle $O_{\text{Res}}$ takes as input $(m, PK_i, A, \text{request})$, and outputs $\text{Res}_{\text{TTP}}(m, PK_i, A, \text{request}, ASK)$. In this experiment, the adversary can arbitrarily choose a public key $PK_i$, and it may not know the corresponding private key of $PK_i$.

- **Security Against Verifiers**: This property captures two cases: the first case is that no verifier, even given the credential secret key, should be able to generate a full signature of the signer’s while not possessing a credential-protected package. The second case is that the verifier that does not possess the expectant attributes should not be able to generate a full signature even if given a corresponding credential-protected package. Formally, we require that no PPT adversary $A$, who models a dishonest verifier, succeeds with non-negligible probability in either of the following two experiments:
Experiment SAV1:

\[(\text{PM, CK}) \leftarrow \text{PMGen}(1^k, U)\]
\[(\text{ASK, APK}) \leftarrow \text{Setup}^{\text{TTP}}(1^k)\]
\[(\text{SK}_A, \text{PK}_A) \leftarrow \text{Setup}^{\text{User}}(1^k, \text{APK})\]
\[(m^*, \sigma^*) \leftarrow \mathcal{A}^{O_{\text{Trans},O_{\text{Res}}}(\text{PM, CK, APK, PK}_A)}\]

success of \(\mathcal{A}\) := \[\text{Ver}(m^*, \sigma^*, \text{PK}_A, \text{APK}) = \top \]
\[\land (m^*, \text{PK}_A, \cdot, \cdot) \notin \text{Query}(\mathcal{A}, O_{\text{Res}})\]

where oracle \(O_{\text{Res}}\) is described in the previous experiment, oracle \(O_{\text{Trans}}\) takes as input \((m_1, m_2, \mathcal{A}, \mathcal{A}')\) and outputs \(\text{Tran}_P(m_1, m_2, \text{SK}_A, \text{APK}, \text{PM}, \mathcal{A}, \mathcal{A}')\), and \(\text{Query}(\mathcal{A}, O_{\text{Res}})\) is the set of queries \(\mathcal{A}\) issued to the resolution oracle. Note that there is no need to provide \(\mathcal{A}\) with access to the transaction for signer oracle \(O_{\text{Trans}}\), as resolution ambiguity property guarantees that its functionality could be achieved by executing \(O_{\text{Trans}}\) and \(O_{\text{Res}}\).

Experiment SAV2:

\[(\text{PM, CK}) \leftarrow \text{PMGen}(1^k, U)\]
\[(\text{ASK, APK}) \leftarrow \text{Setup}^{\text{TTP}}(1^k)\]
\[(\text{SK}_A, \text{PK}_A) \leftarrow \text{Setup}^{\text{User}}(1^k, \text{APK})\]
\[(m^*, \sigma^*) \leftarrow \mathcal{A}^{O_{\text{SetupCred},O_{\text{Trans},O_{\text{Res}}}}(\text{PM, APK, PK}_A)}\]

success of \(\mathcal{A}\) := \[\text{Ver}(m^*, \sigma^*, \text{PK}_A, \text{APK}) = \top \]
\[\land (m^*, \text{PK}_A, \cdot, \cdot) \notin \text{Query}(\mathcal{A}, O_{\text{Res}})\]
\[\land \text{for any } S \in \text{Query}(\mathcal{A}, O_{\text{SetupCred}}),\]
\[\text{for any } (m^*, \text{PK}_A, \mathcal{A}, \cdot) \in \text{Query}(\mathcal{A}, O_{\text{Res}}),\]
\[S \text{ does not satisfy } \mathcal{A}\]

where oracles \(O_{\text{Trans}}, O_{\text{Res}}\) and \(\text{Query}(\mathcal{A}, O_{\text{Res}})\) are described in the previous experiment, oracle \(O_{\text{SetupCred}}\) takes as input a set of attributes \(S\), and outputs a credential \(\text{CD}_S\), and \(\text{Query}(\mathcal{A}, O_{\text{SetupCred}})\) is the set of queries \(\mathcal{A}\) issued to the oracle \(O_{\text{SetupCred}}\).

- **Security Against the Arbitrator:** This property guarantees that even when given the credential secret key, the arbitrator should not be able to create a valid full signature unless it has seen a corresponding
transaction promise. Formally, we consider the following experiment, in which the two-stage adversary \( A = (A_1, A_2) \) models the dishonest arbitrator that may even have access the master key.

Experiment SAA:

\[
(PM, CK) \leftarrow PMGen(1^k, U)
\]

\[
(APK, ASK^*) \leftarrow A_1(PM, MK)
\]

\[
(SK_A, PK_A) \leftarrow Setup^{User}(1^k, APK)
\]

\[
(m^*, \sigma^*) \leftarrow A_2^{O_{Tran}}(PM, CK, ASK^*, APK, PK_A)
\]

success of \( A \) := \([\text{Ver}(m^*, \sigma^*, PK_A, APK) = \top \wedge (m^*, \cdot, \cdot, \cdot) \notin \text{Query}(A, O_{Tran})]\)]

where the partial signing oracle \( O_{Tran} \) is described in the previous experiment, \( ASK^* \) is \( A \)'s state information, which might not be the corresponding secret key of \( APK \), and \( \text{Query}(A, O_{Tran}) \) is the set of queries \( A \) issued to oracle \( O_{Tran} \).

Definition 5. An attribute-based optimistic fair exchange scheme is said to be secure in the multi-user setting and chosen-key model if there is no PPT adversary that wins any of the experiments above with non-negligible probability.

4. Construction

In this section, we give a generic construction of ABOFE that is secure in our defined model.

Let \( \mathcal{E} := (\text{Setup}, \text{Encrypt}, \text{KeyGen}, \text{Decrypt}) \) be a ciphertext-policy attribute-based encryption scheme and \( \text{OFE} = (\text{Setup}^{TTP}, \text{Setup}^{User}, \text{PSig}, \text{PVer}, \text{Sig}, \text{Ver}, \text{Res}) \) be a conventional optimistic fair exchange scheme. Below are the details of our generic construction of an attribute-based optimistic fair exchange scheme.

- **PMGen**: On input \( 1^k \) and attribute universe description \( U \), this algorithm runs \( \mathcal{E}.\text{Setup}(1^k, U) \rightarrow (PM, MK) \). The public parameter and credential secret key are set as \( PM := PM \) and \( CK := MK \), respectively.
• **Setup**\textsuperscript{TTP}: On input $1^k$, this algorithm runs \textsc{OFE.Setup}\textsuperscript{TTP}($1^k$) → ($\text{ASK}, \text{APK}$). The arbitrator’s secret and public key pair is set as $\text{ASK} := \text{ASK}$ and $\text{APK} := \text{APK}$.

• **Setup**\textsuperscript{User}: This algorithm runs \textsc{OFE.Setup}\textsuperscript{User}($1^k, \text{APK}$) → ($\text{SK}_i, \text{PK}_i$). The user’s secret and public key pair is set as $\text{SK}_i := \text{SK}_i$, $\text{PK}_i := \text{PK}_i$.

• **Setup**\textsuperscript{Cred}: On input the credential secret key $\text{CK}$ and a set of attributes $S$, this algorithm runs \textsc{E.KeyGen} (MK, $S$) → $\text{SK}$. The credential is set as $\text{CD}_S := \text{SK}$.

• **Tran\textsuperscript{P}**: Taking as input ($m_1, m_2, \text{SK}_i, \text{APK}, \text{PM}, A_1, A_2$), user $U_i$ runs \textsc{OFE.PSig}($m_1, \text{SK}_i, \text{APK}$) → $\sigma_P$, and generates a non-repudiation information $\theta$ about the transaction\footnote{This is used to identify the transaction that is going on and will be necessary when the arbitrator makes a resolution. Typically this can be achieved by signing $\sigma_P || m_1 || m_2 || A_1 || A_2$ using an independent signature key pair.} with respect to the tuple ($\sigma_P, m_1, m_2, A_1, A_2$). The transaction promise $\omega$ is set as ($\sigma_P, \theta$)\footnote{In most research of optimistic fair exchange, it is normally assumed that only the full signatures are useful to the exchanging parties. Even if Bob holds a partial signature from Alice, Bob can not acquire Alice’s service, since only the full signature represents the commitment of a service. Thus, for simplicity and clarity, we send the partial signature $\sigma_P$ in clear. In practice, if we would like to achieve the property that Bob cannot gain Alice’s partial signature unless he satisfies some attributes, then Alice can encrypt $\sigma_P$ under a proper access structure.}.

• **TPVer**: Taking as input ($m_1, m_2, \omega, \text{PK}_i, \text{APK}, \text{PM}, A_1, A_2$), it outputs \textsc{OFE.PVer}($m_1, \sigma_P, \text{PK}_i, \text{APK}$).

• **Tran\textsuperscript{S}**: Taking as input ($m, \text{SK}_i, \text{APK}, \text{PM}, A$), user $U_i$ runs \textsc{OFE.Sig}($m, \text{SK}_i, \text{APK}$) → $s$, and then encrypts $s$ under the access structure $A$, i.e., \textsc{E.Encrypt}($\text{PM}, s, A$) → $\text{CT}$. The credential-protected package is set as $\pi := \text{CT}$.

• **Tran\textsuperscript{V}**: Taking as input ($m, \pi, \text{PK}_i, \text{APK}, \text{PM}, \text{CD}_S$), the verifier that possesses a set of attributes $S$ that satisfies the access structure $A$ will be able to convert the credential-protected package to a full signature. The verifier outputs $\sigma \leftarrow \textsc{E.Decrypt}($PM, $\pi, \text{CD}_S$).
Ver: Taking as input \((m, \sigma, \text{PK}_i, \text{APK})\), it outputs \(\text{OFE.}\text{Ver}(m, \sigma, \text{PK}_i, \text{APK})\).

\textbf{Res}_{\text{User}}: Taking as input \((m_1, m_2, \omega, \text{SK}_j, \text{PK}_i, \text{APK}, \text{PM}, A_1, A_2)\), it runs \(\text{OFE.}\text{Sig}(m_2, \text{SK}_j, \text{APK}) \rightarrow \sigma_2\). The resolution request is set as \(\text{request} := (m_2, \omega, \text{PK}_j, \sigma_2, A_1)\).

\textbf{Res}_{\text{TTP}}: Taking as input \((m, \text{PK}_i, A, \text{request}, \text{ASK})\), the arbitrator firstly parses \(\text{request}\) as \((m_2, \omega, \text{PK}_j, \sigma_2, \hat{A}_1)\), and checks whether \(\text{TPVer}(m_1, m_2, \omega, \text{PK}_i, \text{APK}, \text{PM}, A_1, \hat{A}_2) = \top\) and whether \(\text{Ver}(m_2, \sigma_2, \text{PK}_j, \text{APK}) = \top\). If either does not hold, it returns \(\bot\). Otherwise, the arbitrator parses \(\omega\) as \((\sigma_P, \theta)\), computes \(\sigma_1 \leftarrow \text{OFE.}\text{Res}(m, \sigma_P, \text{ASK}, \text{PK}_i)\) and then encrypts \(\sigma_1\) under the access structure \(A_2\), i.e., \(E.\text{Encrypt}(\text{PM, } \sigma_1, \hat{A}_2) \rightarrow \text{CT}_1\). The arbitrator also encrypts \(\sigma_2\) under the access structure \(A_1\), i.e., \(E.\text{Encrypt}(\text{PM, } \sigma_2, A_1) \rightarrow \text{CT}_2\). The credential-protected packages are set as \(\pi_1 := \text{CT}_1\) and \(\pi_2 := \text{CT}_2\).

4.1. Security Analysis

Regarding the security of our generic construction of ABOFE, we have the following theorem.

**Theorem 1.** Our generic construction of ABOFE is secure in the multi-user setting and chosen-key model if the optimistic fair exchange scheme OFE is secure in the multi-user setting and chosen-key model and the ciphertext-policy attribute-based encryption scheme \(E\) is fully secure.

We prove Theorem 1 by the following three lemmas.

**Lemma 1 (Security against Signers).** The generic construction is secure against signers if \(\text{OFE} = (\text{Setup}^{\text{TP}}\text{, Setup}^{\text{User}}, \text{PSig, PVer, Sig, Ver, Res})\) is secure against signers.

**Proof.** To show security against signers, we convert any adversary \(A\) that wins the experiment \(\text{SAS}\) into an adversary \(A'\) that breaks the security against signers of \(\text{OFE}\). Recall that \(A'\) gets \(\text{APK}\) as input and has access to oracle \(O'_{\text{Res}}\). \(A'\) runs \(\text{PMGen}(1^k, U) \rightarrow (\text{PM, CK})\) and invokes \(A\) on input \(\text{PM, CK}\) and \(\text{APK} := \text{APK}\).

Given a resolution for arbitrator query \((m, \text{PK}_i, A, \text{request})\) to \(O_{\text{Res}}\), \(A'\) firstly parses \(\text{request}\) as \((m_2, \omega, \text{PK}_j, \sigma_2, \hat{A}_2)\) and checks whether \(\text{TPVer}...\)
\[(m_1, m_2, \omega, \text{PK}_i, \text{APK}, \text{PM}, A', A) = \top\] and whether \(\text{Ver}(m_2, \sigma_2, \text{PK}_j, \text{APK}) = \top\). If does not hold, it returns \(\bot\). Otherwise, \(A'\) parses \(\omega\) as \((\sigma_p, \theta)\), and makes a query \((m, \sigma_p, \text{PK}_i)\) to its own oracle \(O'_{\text{Res}}\). Denote the answer from \(O'_{\text{Res}}\) is \(\sigma_1\). \(A'\) encrypts \(\sigma_1\) under the access structure \(A\), i.e., \(\mathcal{E}.\text{Encrypt}(\text{PM}, \sigma_1, A) \rightarrow \text{CT}_1\). \(A'\) also encrypts \(\sigma_2\) under the access structure \(A'\), i.e., \(\mathcal{E}.\text{Encrypt}(\text{PM}, \sigma_2, A') \rightarrow \text{CT}_2\). The credential-protected packages are set as \(\pi_1 := \text{CT}_1\) and \(\pi_2 := \text{CT}_2\). \(A'\) returns \((\pi_1, \pi_2)\) to \(A\).

It can be seen that the resolution for arbitrator oracle \(O_{\text{Res}}\) is perfectly simulated by \(A'\). Finally, \(A\) outputs a tuple \((m_1^*, m_2^*, (\sigma_p^*, \theta^*), \text{PK}_A, A_1, A_2, S)\) where the set of attributes \(S\) satisfies the access structure \(A_2\). \(A'\) outputs \((m_1^*, \sigma_p^*, \text{PK}_A)\).

Denote \((\pi_1^*, \pi_2^*)\) as the output of \(\text{Res}_{\text{TTP}}(m_1^*, \text{PK}_A, A_2, \text{Request}^*, \text{ASK})\), and \(\sigma^*\) as the output of \(\text{Tran}_\mathcal{V}(m_1^*, \pi_1^*, \text{PK}_A, \text{APK}, \text{PM}, \text{CD}_S)\) where \(\text{CD}_S = \text{Setup}^\text{Cred}(\text{CK}, S)\). Due to the fact that the set of attributes \(S\) satisfies the access structure \(A_2\), \(\sigma^*\) is in fact also the output of \(\text{OFE}.\text{Res}(m_1^*, \sigma_p^*, \text{ASK}, \text{PK}_A)\). Since \(\text{TPVer}(m_1^*, m_2^*, (\sigma_p^*, \theta^*), \text{PK}_A, \text{APK}, \text{PM}, A_1, A_2) = \top\) but \(\text{Ver}(m_1^*, \sigma_p^*, \text{PK}_A, \text{APK}) = \bot\), it means \(\text{OFE}.\text{PVer}(m_1^*, \sigma^*, \text{PK}_A, \text{APK}) = \top\) but \(\text{OFE}.\text{Ver}(m_1^*, \sigma^*, \text{PK}_A, \text{APK}) = \bot\). Thus if \(A\) succeeds in the experiment, \(A'\) also succeeds with the same probability in breaking the security against signers of \(\text{OFE}\).

**Lemma 2 (Security against Verifiers).** The generic construction is secure against verifiers if \(\text{OFE} = (\text{Setup}^\text{TTP}, \text{Setup}^\text{User}, \text{PSig}, \text{PVer}, \text{Sig}, \text{Ver}, \text{Res})\) is secure against verifiers and the ciphertext-policy attribute-based encryption \(\mathcal{E} := (\text{Setup}, \text{Encrypt}, \text{KeyGen}, \text{Decrypt})\) is fully secure.

**Proof.** To show security against verifiers, we consider the experiments \(\text{SAV}_1\) and \(\text{SAV}_2\), respectively. We firstly convert any adversary \(A\) that wins the experiment \(\text{SAV}_1\) into an adversary \(A'\) that breaks the security against verifiers of \(\text{OFE}\).

Recall that \(A'\) gets \((\text{APK}, \text{PK}_A)\) as input and has access to oracles \(O'_{\text{PSig}}\) and \(O'_{\text{Res}}\). \(A'\) runs \(\text{PMGen}(1^k, U) \rightarrow (\text{PM}, \text{MK})\) and invokes \(A\) on input \(\text{PM}, \text{CK}, \text{APK} := \text{APK}\) and \(\text{PK}_A := \text{PK}_A\).

Given a transaction promise query \((m_1, m_2, A, A')\) to oracle \(O'_{\text{TTP}}\), \(A'\) makes a query \(m\) to its own oracle \(O'_{\text{PSig}}\). Denote the answer from \(O'_{\text{PSig}}\) is \(\sigma_p\). \(A'\) returns \((\sigma_p, \theta)\) to \(A\) where \(\theta\) is a non-repudiation information about the transaction with respect to the tuple \((\sigma_p, m_1, m_2, A_1, A_2)\).

Given a resolution for arbitrator query \((m, \text{PK}_i, A, \text{request})\) to \(O_{\text{Res}}\), \(A'\) firstly parses \(\text{request}\) as \((m_2, \omega, \text{PK}_j, \sigma_2, A')\) and checks whether \(\text{TPVer}\)....
\[(m_1, m_2, \omega, PK_i, APK, PM, A', A) = \top\) and whether Ver\((m_2, \sigma_2, PK_j, APK) = \top\). If either does not hold, it returns \(\bot\). Otherwise, \(A'\) parses \(\sigma_1\) as \((\sigma_P, \theta)\), and makes a query \((m, \sigma_P, PK_i)\) to its own oracle \(O'_{\text{Res}}\). Denote the answer from \(O'_{\text{Res}}\) is \(\sigma_1\). \(A'\) encrypts \(\sigma_1\) under the access structure \(A\), i.e., \(E.\text{Encrypt}(PM, \sigma_1, A) \rightarrow CT_1\). \(A'\) also encrypts \(\sigma_2\) under the access structure \(A'\), i.e., \(E.\text{Encrypt}(PM, \sigma_2, A') \rightarrow CT_2\). The credential-protected packages are set as \(\pi_1 := CT_1\) and \(\pi_2 := CT_2\). \(A'\) returns \((\pi_1, \pi_2)\) to \(A\).

It can be seen that the oracles \(O_{\text{PSig}}\) and \(O_{\text{Res}}\) are perfectly simulated by \(A'\). Finally, \(A\) outputs a tuple \((m^*, \sigma^*)\) such that Ver\((m^*, \sigma^*, PK_A, APK) = \top\). This means OFE Ver\((m^*, \sigma^*, PK_A, APK) = \top\). \(A'\) outputs \((m^*, \sigma^*)\). Since \(A\) is prohibited from making a query \((m^*, PK_A, \cdot, \cdot)\) to oracle \(O_{\text{Res}}\), \(A'\) has never made a query with respect to a tuple \((m^*, \cdot, PK_A)\) to its own oracle \(O'_{\text{Res}}\). If \(A\) succeeds in the experiment \(SAV1\), \(A'\) also succeeds in breaking the security against verifiers of OFE. Thus \(A'\)'s advantage is also non-negligible.

Next, we consider the experiment \(SAV2\). We convert any adversary \(A\) that wins the experiment \(SAV2\) into an algorithm \(B\) that breaks the full security of the encryption scheme \(E\).

Recall that \(B\) gets \(PM\) as input and can query its own challenger for private keys corresponding to a sequence of sets of attributes. \(B\) runs \(\text{Setup}^{\text{TTP}}(1^k) \rightarrow (ASK, APK)\) and \(\text{Setup}^{\text{User}}(1^k, APK) \rightarrow (SK_A, PK_A)\), and invokes \(A\) as a subroutine by forwarding \(PM := PM, APK\) and \(PK_A\).

When \(A\) makes a query \(S\) to oracle \(O_{\text{Setup}^{\text{card}}}\), \(B\) supplies \(S\) to its own challenger and forwards the reply to \(A\). When \(A\) makes a transaction promise query \((m_1, m_2, A, A')\) to oracle \(O_{\text{TranP}}\), \(B\) runs \(\text{TranP}(m_1, m_2, SK_A, APK, PM, A_1, A_2)\) and sends \(\omega\) to \(A\) as the reply.

Suppose \(A\) makes \(q\) valid resolution for arbitrator queries to \(O_{\text{Res}}\). \(B\) chooses a random value \(z \in \{1, \cdots, q\}\). When \(A\) makes a \(j\)-th resolution query \((m_j, PK_j, A_j, \text{request}_j)\) to \(O_{\text{Res}}\), if \(j \neq z\), \(B\) runs \(\text{Res}^{\text{TTP}}(m_z, PK_j, A_j, \text{request}_j, ASK)\) and sends the outputs to \(A\) as the reply. When \(j = z\) and \(PK_z \neq PK_A\), \(B\) aborts and returns failure. When \(j = z\) and \(PK_z = PK_A\), \(B\) parses \(\text{request}_z\) as \((m_2, \omega_j, PK_i, \sigma_2, A'_z)\) and parses \(\omega_z\) as \((\sigma_{P_z}, \theta_z)\). \(B\) runs \(\text{OFE.\text{Res}}(m_z, \sigma_{P_z}, ASK, PK_A) \rightarrow \sigma_z\), randomly chooses \(M_1\) that is of the same bit length with \(\sigma_z\), and sends \(M_0 := \sigma_z, M_1\) and the access structure \(A'_z\) to its own challenger. Denote the challenge ciphertext is \(CT\). \(B\) also encrypts \(\sigma_2\) under the access structure \(A'_z\), i.e., \(E.\text{Encrypt}(PM, \sigma_2, A') \rightarrow CT'\). The credential-protected packages are set as \(\pi_1 := CT\) and \(\pi_2 := CT'\). \(A'\) returns \((\pi_1, \pi_2)\) to \(A\) as the reply to the \(z\)-th resolution for arbitrator query.
Finally, $\mathcal{A}$ either outputs failure or wins by outputting a tuple $(m^*, \sigma^*)$ such that $\text{Ver}(m^*, \sigma^*, \text{PK}_A, \text{APK}) = \top$. Note that when $\mathcal{A}$ wins, $\mathcal{A}$ must have made a resolution query $(m^*, \text{PK}_A, \cdot, \cdot, \cdot)$ to $O_{\text{Res}}$. Otherwise the analysis of this type of attack is covered in the experiment $\text{SAV1}$ discussed above. If $m^* = m_z$, $\text{PK}_A = \text{PK}_z$, and $\mathcal{A}_2 = \mathcal{A}'_z$, then $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs a random bit.

If the challenge ciphertext CT is the encryption of $\sigma_z$ (i.e., $b = 0$), CT is a valid attribute-related signature and the distribution of $B$’s view in the simulated environment is identical with that in the real attack environment. If the challenge ciphertext CT is the encryption of $M_1$ (i.e., $b = 1$), CT has no information on the full signature of a message $m_z$ and $B$’s chance of forging a valid full signature on message $m_z$ (with respect to $\text{PK}_A$) is negligible. Thus if $\mathcal{A}$ in the real experiment $\text{SAV2}$ wins with non-negligible probability, then $\mathcal{B}$ wins with non-negligible probability in breaking the full security of the encryption scheme $\mathcal{E}$.

\begin{lemma}[Security against the Arbitrator] The generic construction is secure against the arbitrator if $\text{OFE} = (\text{Setup}^{\text{TTP}}, \text{Setup}^{\text{User}}, \text{PSig}, \text{PVer}, \text{Sig}, \text{Ver}, \text{Res})$ is secure against the arbitrator.
\end{lemma}

\begin{proof}
To show security against the arbitrator, we convert any adversary $\mathcal{A}$ that wins the experiment $\text{SAA}$ into an adversary $\mathcal{A}'$ that breaks the security against the arbitrator of $\text{OFE}$. $\mathcal{A}$ firstly chooses a public adjudication key $\text{APK}$ and outputs it, keeps a corresponding secret state information $\text{ASK}^*$ private. $\mathcal{A}'$ sets $\text{APK} := \text{APK}$, gets $\text{PK}_A$ as input, and has access to oracles $O'_{\text{PSig}}$. $\mathcal{A}'$ runs $\text{PMGen}(1^k, U) \to (\text{PM}, \text{CK})$ and forwards $\text{PM}, \text{CK}$ and $\text{PK}_A := \text{PK}_A$ to $\mathcal{A}$.

Given a transaction promise query $(m_1, m_2, A, A')$ to oracle $O_{\text{Tran}}$, $\mathcal{A}'$ makes a query $m$ to its own oracle $O'_{\text{PSig}}$. Denote the answer from $O'_{\text{PSig}}$ is $\sigma_p$. $\mathcal{A}'$ returns $(\sigma_p, \theta)$ to $\mathcal{A}$ where $\theta$ is a non-repudiation information about the transaction with respect to the tuple $(\sigma_p, m_1, m_2, A_1, A_2)$.

It can be seen that the oracle $O_{\text{Tran}}$ is perfectly simulated by $\mathcal{A}'$. Finally, $\mathcal{A}$ outputs a tuple $(m^*, \sigma^*)$ such that $\text{Ver}(m^*, \sigma^*, \text{PK}_A, \text{APK}) = \top$. This means $\text{OFE.Ver}(m^*, \sigma^*, \text{PK}_A, \text{APK}) = \top$. $\mathcal{A}'$ outputs $(m^*, \sigma^*)$. Since $\mathcal{A}$ is prohibited from making a query $(m^*, \cdot, \cdot, \cdot)$ to oracle $O_{\text{Tran}}$, $\mathcal{A}'$ has never made a query message $m^*$ to its own oracle $O'_{\text{PSig}}$. If $\mathcal{A}$ succeeds in the experiment $\text{SAV2}$, $\mathcal{A}'$ also succeeds with the same probability in breaking the security against the arbitrator of $\text{OFE}$.
\end{proof}
5. Instantiation

In the following, we provide an instantiation to demonstrate the flexibility of our generic construction. It was shown in [18] that OFE secure in the multi-user setting and chosen key model can be constructed from conventional signatures and ring signatures. More specifically, in this paradigm, the partial signature in OFE is a conventional signature, and the full signature in OFE is the partial signature, together with a two party ring signature generated between the signer and the arbitrator. The authors [18] suggest a concrete OFE scheme can be built on waters signature scheme [25] in group of composite order and Shacham-Waters’ ring signature scheme [26] so that they share the same set of system parameters. For the CP-ABE scheme, we employ the one proposed by Lewko and Waters in [23], which is known to be fully secure. Note that there is a global setup process before execution of the scheme due to the requirement of having such a setup process of Shacham-Waters’ ring signature.

**Global Setup**: On input $1^k$ where $k$ is a security parameter, the setup algorithm generates a multiplicative cyclic group $G$ of composite order $n_1$ where $n_1 = pq$ and a bilinear pairing $e : G \times G \to G_T$ where $G_T$ is a multiplicative group of order $n_1$. Let $G_p$ and $G_q$ be the cyclic order-$p$ and order-$q$ subgroups of $G$, respectively. Let $g$ be a generator of $G$ and $h$ be a generator of $G_q$. It then chooses random exponents $a, b \in \mathbb{Z}_{n_1}$ and sets

$$A := g^a, B := g^b, \hat{A} := h^a.$$ 

Let $H_0 : \{0,1\}^* \to \{0,1\}^k$ be a collision-resistant hash function. The setup algorithm picks Waters hash generators

$$u', u_1, \cdots, u_k \leftarrow G.$$ 

The common reference string is set as $(n_1, A, B, \hat{A}, u', u_1, \cdots, u_k, H_0)$.

After the global setup is finished, the algorithms in ABOFE can be executed as follows.

- **PMGen**: On input $1^k$ where $k$ is a security parameter, the setup algorithm outputs a multiplicative cyclic group $\hat{G}$ of order $n_2$ where $n_2 = p_1p_2p_3$ (3 distinct primes) and a bilinear pairing $\hat{e} : \hat{G} \times \hat{G} \to \hat{G}_T$ where $\hat{G}_T$ is a multiplicative group of order $n_2$. Let $\hat{G}_{p_i}$ denote the
subgroup of order \( p_i \) in \( \hat{G} \). Let \( g_i \) be a generator of \( \hat{G}_{p_i} \) and \( H_1 : \hat{G}_T \to \{0, 1\}^l \) be a collision-resistant hash function. It then chooses random exponents \( \alpha, c, \gamma \in \mathbb{Z}_n \), and for each attribute \( i \in U \), it chooses a random group element \( h_i \in \hat{G}_{p_i} \). Let \( SE \) be a symmetric encryption scheme whose symmetric key space is \( \{0, 1\}^l \). The public parameters \( PM \) are set as

\[
n_2, g_1, A, g_1^c, g_1^\gamma, e(g_1, g_1)^\alpha, H_1, h_1, \ldots, h_{|U|}, SE.
\]

The master key are set as \( MK := (g_1^\alpha, g_3) \).

- **Setup\(^\text{TTP}\)**: The arbitrator chooses a random exponent \( y \in \mathbb{Z}_{n_1} \), and sets \( APK := g^y \in G \) and \( sk := A^y \in G \).

- **Setup\(^\text{User}\)**: Each user \( U_i \) randomly chooses exponents \( x_1, x_3 \in \mathbb{Z}_{n_1} \) and computes \( \bar{g}_1 = g^{x_1} \) and \( \bar{g}_3 = g^{x_3} \). Besides, user \( U_i \) randomly chooses \( \bar{g}_2, \bar{u}'_1, \bar{u}_1, \ldots, \bar{u}_k \leftarrow G \) and sets \( PK_i := (\bar{g}_1, \bar{g}_2, \bar{g}_3, \bar{u}', \bar{u}_1, \ldots, \bar{u}_k) \) and \( SK := (\bar{g}_2^{x_1}, A^{x_3}) \).

- **Setup\(^\text{Cred}\)**: On input the credential secret key \( CK \) and a set of attributes \( S \), the algorithm chooses random exponents \( t, u \in \mathbb{Z}_{n_2} \), and random elements \( R, R', R'', \{R_i\}_{i \in S} \in G_{p_3} \) (this can be done by raising a generator of \( G_{p_3} \) to random exponents modulo \( n_2 \)). The credential is:

\[
S, K = g_1^o g_1^c g_1^u R, \quad K' = g_1^t R', \quad K'' = g^t R'', \quad K_i = h_i^{t} R_i \text{ for } i \in S.
\]

- **Tran\(^\text{P}\)**: \( \text{Tran}\(^\text{P}\)(\( m_A, m_B, SK_i, APK, PM, (M, \rho), (M', \rho') \)) \), where \( M \) is an \( L \times N \) matrix and \( \rho \) a map from each row \( M_j \) of \( M \) to an attribute \( \rho(j) \), and \( M' \) is an \( L' \times N' \) matrix and \( \rho' \) a map from each row \( M'_j \) of \( M' \) to an attribute \( \rho'(j) \), does as follows:

1. Compute \( (m_1, \ldots, m_k) \leftarrow H_0(m_A) \),
2. Choose a random exponent \( r \in \mathbb{Z}_{n_1} \) and computes

\[
S_1 = \bar{g}^{x_1} \cdot \left( \bar{u}' \prod_{i=1}^{k} \bar{u}_i^{m_i} \right)^r, \quad \text{and} \quad S_2 = g^r.
\]
3. \( \sigma_P \) is set as \( (S_1, S_2) \).
4. Generate a non-repudiation information $\theta$ about the transaction with respect to the tuple $((\sigma_p, m_A, m_B, (M, \rho), (M', \rho'))).$ The transaction promise $\omega$ is set as $((\sigma_p, \theta)).$

- **TPVer**: $\text{TPVer}(m_A, m_B, \omega, PK_i, APK_i, PM_i, (M, \rho), (M', \rho'))$ does as follows.

  1. Parse $\omega$ as $((\sigma_p, \theta)).$
  2. Verify whether $\theta$ is a non-repudiation information about the transaction with respect to the tuple $((\sigma_p, m_A, m_B, (M, \rho), (M', \rho'))).$ If so, it continues; otherwise, it aborts and outputs $\bot.$
  3. Compute $(m_1, \ldots, m_k) \leftarrow H_0(m_A),$ and verify whether

$$e(S_1, g) \cdot e(S_2^{-1}, \bar{u}^m \prod_{j=1}^k \bar{u}^m_j) = e(\bar{g}_1, \bar{g}_2)$$

holds. If so, it outputs $\top;$ if not, it outputs $\bot.$

- **Trans**: $\text{Trans}(m, SK_i, APK_i, PM_i, (M, \rho)),$ where $M$ is an $L \times N$ matrix and $\rho$ a map from each row $M_j$ of $M$ to an attribute $\rho(j),$ does as follows:

  1. Compute $(m_1, \ldots, m_k) \leftarrow H_0(m),$
  2. Choose a random exponent $r \in \mathbb{Z}_{n_1}$ and computes

$$S_1 = g_2^{r_1} \cdot (\bar{u}' \prod_{i=1}^k \bar{u}^{m_i}_j)^r, \quad \text{and} \quad S_2 = g^r.$$

  3. Compute $(m'_1, \ldots, m'_k) \leftarrow H_0(m || S_1 || S_2 || PK_i).$
  4. Choose two random exponents $t_0, t_1 \in \mathbb{Z}_{n_1}$ and sets

$$C_0' = (\bar{g}_3/B)h^{t_0}, \quad \pi_0 = ((\bar{g}_3/B)h^{t_0})^{t_0}, \quad C_1' = h^{t_1}, \quad \pi_1 = ((APK_i/B)^{-1}h^{t_1})^{t_1}.$$

  5. Choose $r' \leftarrow \mathbb{Z}_{n_1},$ and compute first $t = t_1 + t_2$ and then

$$S_1' = A^{r'} \cdot (u' \prod_{j=1}^k u'^{m'_j}_j)^{r'} \cdot \tilde{A}^t, \quad \text{and} \quad S_2' = g^{r'}.$$
6. Choose a random vector \( v \in \mathbb{Z}_{n_2}^N \), denoted \( v = (s, v_2, \ldots, v_N) \). For each row \( M_j \) of \( M \), it chooses a random \( r_j \in \mathbb{Z}_{n_2} \).

7. Choose a random element \( D \in G_T \), computes \( sk = H_1(D) \), and uses \( sk \) as a symmetric key of SE to encrypt the bit strings \( S_1 || S_2 || S_1' || S_2' || C_0' || C_1' || \pi_0 || \pi_1 \) and gains a ciphertext \( \tilde{c} \).

8. The credential-protected package \( \pi \) is set as

\[
\tilde{c}, \ C_0 = De(g_1, g_1)^{\alpha s}, \ C = g_1^s, \ C' = (g_1^\gamma)^s, \\
C_j = (g_1^c)^{M_j \cdot v_h^{-r_j}}, \ D_j = g_1^{r_j}, \ \text{for } j = 1, \ldots, L.
\]

- **Tran**: \( \text{Tran}_V(m, \pi, PK_i, APK, PM, CD_S) \) does as follows.

  1. Computes constants \( \omega_j \in \mathbb{Z}_{n_2} \) such that \( \sum_{\rho(j) \in S} \omega_j M_j = (1, 0, \ldots, 0) \). It then computes

\[
\hat{e}(C, K) \hat{e}(C', K')^{-1} / \prod_{\rho(j) \in S} (\hat{e}(C_j, K'') \hat{e}(D_j, K_{\rho(j)}))^{\omega_j} = \hat{e}(g_1, g_1)^{\alpha s}.
\]

  2. Recover \( D \) as \( C_0 / \hat{e}(g_1, g_2)^{\alpha s} \), compute \( sk = H_1(D) \) and uses \( sk \) as a symmetric key of SE to decrypt the ciphertext \( c \) to gain a full signature \( \sigma \).

- **Ver**: \( \text{Ver}(m, \sigma, PK_i, APK) \) does as follows.

  1. Parse \( \sigma \) as \( (S_1, S_2, S_1', S_2', C_0, C_1, \pi_0, \pi_1) \), compute \( (m_1, \ldots, m_k) \leftarrow H_0(m) \), and verify whether

\[
e(S_1, g) \cdot e(S_2^{-1}, \tilde{u} \prod_{j=1}^k \tilde{u}_j^{m_j}) = e(\tilde{g}_1, \tilde{g}_2)
\]

holds. If so, output \( \top \); if not, output \( \bot \).

  2. Check whether

\[
e(C_0', C_0'/(\tilde{g}_3/B)) = e(h, \pi_0) \quad \text{and} \quad e(C_1', C_1'/(APK/B)) = e(h, \pi_1)
\]

hold. If either does not hold, output \( \bot \).
3. Compute \((m_1', \ldots, m_k') \leftarrow H_0(m||S_1||S_2||PK_i)\) and \(C'_2 = C'_0 \cdot C'_1\), and verify whether

\[ e(A, BC'_2) = e(S'_1, g) \cdot e((S'_2)^{-1}, u \prod_{j=1}^{k} u_j^{m'_j}) \]

holds. If so, output \(\top\); if not, output \(\bot\).

- **ResUser**: 
  \(\text{ResUser}(m_A, m_B, \omega, SK_j, PK_i, APK, PM, (M, \rho), (M', \rho'))\) does as follows.

  1. Parses \(PK_j\) as \((\tilde{g}_1, \tilde{g}_2, \tilde{g}_3, \tilde{u}', \tilde{u}_1, \ldots, \tilde{u}_k)\) and \(SK_j := (\tilde{g}_2^{x'1}, A^{x'_3})\).
  2. Compute \((m_1, \ldots, m_k) \leftarrow H_0(m_B),\)
  3. Choose a random exponent \(r \in \mathbb{Z}_{n_1}\) and computes

  \[ S_1 = \tilde{g}_2^{x'1} \cdot (\tilde{u}' \prod_{i=1}^{k} \tilde{u}_i^{m_i})^r, \quad \text{and} \quad S_2 = g^r. \]

  4. Compute \((m_1', \ldots, m_k') \leftarrow H_0(m||S_1||S_2||PK_j).\)
  5. Choose two random exponents \(t_0, t_1 \in \mathbb{Z}_{n_1}\) and sets

  \[ C'_0 = (\tilde{g}_3/B)h^{t_0}, \pi_0 = ((\tilde{g}_3/B)h^{t_0})^{t_0}, C'_1 = h^{t_1}, \pi_1 = ((APK/B)^{-1}h^{t_1})^{t_1}. \]

  6. Choose \(r' \leftarrow \mathbb{Z}_{n_1}\), and compute first \(t = t_1 + t_2\) and then

  \[ S'_1 = A^{x_3} \cdot (u' \prod_{j=1}^{k} u_j^{m'_j})^{r'} \cdot A^t, \quad \text{and} \quad S'_2 = g^{r'}. \]

  7. Set \(\sigma_B := (S_1, S_2, S'_1, S'_2, C_0, C_1, \pi_0, \pi_1).\)
  8. Set \(\text{request} := (\omega, m_B, \sigma_B, PK_j, (M, \rho)).\)

- **ResTTP**: 
  \(\text{ResTTP}(m_A, PK_i, (M', \rho'), \text{request}, ASK)\) does as follows.

  1. Parse \(\text{request}\) as \((\omega, m_B, \sigma_B, PK_j, (M, \rho))\) and \(\omega\) as \(\sigma_P\) and \(\theta\). If the non-repudiation information \(\theta\) is not with respect to the tuple \((\sigma_P, m_A, m_B, (M, \rho), (M', \rho'))\), outputs \(\bot\).
2. Parse $\sigma_P$ as $(S_1, S_2)$, compute $(m_1, \cdots, m_k) \leftarrow H_0(m_A)$, and verify whether
\[
e(S_1, g) \cdot e(S_2^{-1}, \bar{u}' \prod_{j=1}^{k} a_j^{m_j}) = e(\bar{g}_1, \bar{g}_2)
\]
holds. If not, output $\bot$.
3. Check whether $\mathbf{Ver}(m_B, \sigma_B, \mathbf{PK}_j, \mathbf{APK}) = \top$. If not, output $\bot$.
4. Compute $(m'_1, \cdots, m'_k) \leftarrow H_0(m_A||S_1||S_2||\mathbf{PK}_i)$.
5. Choose two random exponents $t_0, t_1 \in \mathbb{Z}_{n_1}$ and sets
\[
C'_0 = h^{t_0}, \pi_0 = ((\bar{g}_3/B)^{-1}h^{t_0})^{t_0}, C'_1 = (\mathbf{APK}/B)h^{t_1}, \pi_1 = ((\mathbf{APK}/B)h^{t_1})^{t_1}.
\]
6. Choose $r' \leftarrow \mathbb{Z}_{n_1}$, and compute first $t = t_1 + t_2$ and then
\[
S'_1 = A^{\rho} \cdot (u' \prod_{j=1}^{k} u_j^{m'_j})^{r'} \cdot \hat{A}^t, \text{ and } S'_2 = g^{r'}.
\]
7. Choose a random vector $v \in \mathbb{Z}_{n_2}^{N'}$, denoted $v = (s, v_2, \cdots, v_N)$.
   For each row $M_j$ of $M$, it chooses a random $r_j \in \mathbb{Z}_{n_2}$.
8. Choose a random element $D \in G_T$, computes $sk = H_1(D)$, and uses $sk$ as a symmetric key of $\mathbf{SE}$ to encrypt the bit strings $S_1||S_2||S'_1||S'_2||C'_0||C'_1||\pi_0||\pi_1$ and gains a ciphertext $\tilde{c}_A$.
9. The credential-protected package $\pi_A$ is set as
\[
\tilde{c}_A, \quad C_0 = D e(g_1, g_1)^{\alpha s}, \quad C = g_1^s, \quad C' = (g_1)^s,
\]
\[
C_j = (g_1^{\alpha s})^{M_j \cdot h^{-r_j}} = g_1^{\rho h_j}, \text{ for } j = 1, \cdots, L'.
\]
10. Choose a random vector $v' \in \mathbb{Z}_{n_2}^N$, denoted $v' = (s', v'_2, \cdots, v'_N)$.
    For each row $M'_j$ of $M'$, it chooses a random $r'_j \in \mathbb{Z}_{n_2}$.
11. Choose a random element $D' \in G_T$, computes $sk' = H_1(D')$, and uses $sk'$ as a symmetric key of $\mathbf{SE}$ to encrypt the $\sigma_B$ and gains a ciphertext $\tilde{c}_B$.
12. The credential-protected package $\pi_B$ is set as
\[
\tilde{c}_B, \quad C_0 = D e(g_1, g_1)^{\alpha s'}, \quad C = g_1^{s'}, \quad C' = (g_1)^{s'},
\]
\[
C_j = (g_1^{\alpha s'})^{M_j \cdot h^{-r'_j}} = g_1^{\rho h_j}, \text{ for } j = 1, \cdots, L.
\]
13. The output is $(\pi_A, \pi_B)$. 26
<table>
<thead>
<tr>
<th>Algorithms:</th>
<th>Tran _p</th>
<th>TPVer</th>
<th>Tran _s</th>
<th>Tran _v</th>
<th>Ver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost:</td>
<td>2E</td>
<td>3P</td>
<td>1SE, (12 + 3L)E</td>
<td>1SD, (2s + 2)P, (s)E</td>
<td>10P</td>
</tr>
</tbody>
</table>

Table 1: Costs of algorithms in our instantiation of ABOFE.

5.1. Efficiency Analysis

Since pairing and exponentiation operations take more time than multiplication operations do, we will simply ignore the costs of multiplication computations here. Let SE, SD, P and E denote a symmetric encryption operation, a symmetric decryption operation and a pairing operation, and an exponentiation operation, respectively. Let $M$ be an $L \times N$ matrix, $\rho$ a map from each row $M_j$ of $M$ to an attribute $\rho(j)$, and $S$ a set of attributes that satisfies the access structure $(M, \rho)$. Denote $s$ as the number of elements of the set $\{1 \leq j \leq L | \rho(j) \in S\}$.

The costs of our instantiation of ABOFE is presented in Table 1, in which $\text{Tran}_s$ is evaluated with respect to the access structure $(M, \rho)$, and $\text{Tran}_v$ is evaluated with respect to the set of attributes $S$.

6. Conclusion

We proposed the notion of attribute-based optimistic fair exchange, and gave a formal security model. We then proposed a generic construction of PAOFE, and proved its security under the proposed model in the standard model. Since ABOFE can be used to solve the fair exchange problem when the attributes of the participants need to be taken into account, ABOFE can be viewed as an extension of OFE in the attribute-based setting.

Acknowledgement

We would like to thank the anonymous reviewers who have helped us to improve the clarity and quality of this paper.

References


[18] Q. Huang, G. Yang, D. S. Wong, W. Susilo, Efficient optimistic fair exchange secure in the multi-user setting and chosen-key model


