A publicly verifiable optimistic fair exchange protocol using decentralized CP-ABE

LIANG ZHANG[1], HAIBIN KAN[2], FEIYANG QIU[3] AND FENG HAO[4]

[1]School of Cyberspace Security(School of Cryptology), Hainan University, Haikou, China; Key Laboratory of Internet Information Retrieval of Hainan Province
[2]School of Computer Science, Fudan University, Shanghai, China; Shanghai Engineering Research Center of Blockchain, Shanghai 200433, China; Yiwu Research Institute of Fudan University Yiwu City 322000, China
[3]Department of Electrical Engineering, Katholieke Universiteit Leuven, 3001 Leuven, Belgium
[4]Department of Computer Science, University of Warwick, Coventry, CV4 7AL, U.K.
Email: zhangliang@hainanu.edu.cn; hbkan@fudan.edu.cn; fqiu@esat.kuleuven.be; feng.hao@warwick.ac.uk

Fair exchange (FE) is a challenging problem for two mutually distrusting players. It is widely known that fair exchange is impossible without a trusted third party (TTP). However, relying on a single TTP can cause a single-point failure. An intuitive idea is to adopt multiple TTPs to distribute trust. This paper constructs a two-party optimistic fair exchange (OFE) protocol using decentralized ciphertext-policy attribute-based encryption (CP-ABE), achieving decentralized TTPs. This is achievable because decentralized CP-ABE ciphertext supports a nested access control policy. A nested access control policy fits perfectly in a fair exchange protocol which contains multiple roles (i.e., players and TTPs). Further, we apply non-interactive zero knowledge (NIZK) proofs to prove the well-formedness of ciphertexts, so as to enforce players to follow the protocol specification honestly. Consequently, we construct an OFE protocol in which each player's operations are publicly verifiable without revealing secret information. Also, we obtain decentralized TTPs with optimism (i.e., the TTPs are involved only when arbitration is required), autonomy (i.e., the TTPs do not need to interact with each other), statelessness (i.e., the TTPs do not need to store data for the exchange protocol) and verifiability (i.e., the TTPs are publicly verifiable). Compared with previous work, our protocol assumes only a public communication channel and each party's operations are publicly verifiable. Besides, it achieves a favorable $O(n)$ verification complexity in the normal case, where n is the number of TTPs. Finally, we present a proof-of-concept implementation to demonstrate the feasibility.

Keywords: Optimistic fair exchange; Decentralized TTPs; Public verifiability; CP-ABE; NIZK

Received 18 July 2022; revised 21 February 2023

1. INTRODUCTION

Two-party fair exchange, where two players exchange their respective secret items, is one of the most fundamental protocols in the field of secure multi-party computation (MPC). The most critical property in fair exchange protocols is fairness, meaning that either both parties get the desired item, or neither does, even if some player maliciously deviates from the protocol. Unfortunately, no two-party protocol can guarantee fairness, so the reliance on a trusted third party (TTP) is shown to be unavoidable [1]. A TTP, also called an arbiter, is assumed to perform some specific operations fairly, securely, and efficiently. Decentralized TTPs are a group of parties replacing the single TTP, thus resolving the single-point failure problem. If TTP(s) are involved only in the case of a dispute, the protocol is called optimistic [2] and the TTP(s) are called passive [2] or invisible [3]. If all operations of the TTP(s) can be verified, they are considered verifiable [4] or accountable [2]. If no storage is required for an executing protocol, then the TTP(s) are stateless [5]. If the decentralized TTPs do not need to interact with each other, they are called autonomous. All in all, the less intervention of the TTP(s) and the weaker assumptions of the TTP(s) are, the more desirable the fair exchange protocol is.

Küpçü [6] summarized some known methods to construct decentralized TTPs. Protocols [6, 7, 8] use either threshold cryptography or split ciphertexts to eliminate the risk of a single-point failure. In 1997, Franklin [9] introduced a fair exchange protocol with a 2-out-of-2 verifiable secret sharing (VSS) scheme. It provided the rudiment of the idea to decentralize the trusted third party. In 2004, Avoine et al. [10] and Zhao et al. [11] proposed OFE protocols with fully decentralized trusted third parties, based on a publicly verifiable secret sharing (PVSS) scheme [12]. However, the two protocols have a few limitations. Firstly, they have a non-favorable verification cost of $O(nt)$ due to the underlying PVSS scheme, where n is the number of arbiters and t is the threshold value to recover the secret. Secondly, the protocols are not publicly verifiable. These issues will be addressed in our construction.

Public verifiability is a critical property of MPC applications [13], as it enables the parties to be accountable/auditable and enforces them to behave honestly. Blockchain is gathering massive momentum these years, mainly because it provides transparent computation and enhances credibility for distrusting distributed parties. Hence, non-interactive zero-knowledge (NIZK) proofs, which were research interests of specific mathematicians or computer scientists, become a topical issue to the public due to their ability to be verified publicly. Recently, zkSNARKs [14] and sigma protocol [15] (with Fiat-Shamir transformation [16]) attract increasing attention when building practical MPC applications.

This paper proposes an OFE protocol with publicly verifiable decentralized TTPs based on Rouselakis and Waters's multi-authorities ciphertext-policy attribute-based encryption (CP-ABE) scheme [17]. Briefly speaking, our protocol operates in two communication rounds. In Round-1, each player commits its item in a ciphertext using the CP-ABE *Encrypt* algorithm. In Round-2, each player reveals a decryption key (generated by CP-ABE *KeyGen* algorithm). In the end, one player obtains the other's item by invoking the CP-ABE *Decrypt* algorithm. If a player is malicious or corrupted, the other player complains to the TTPs in the recovery process ($\mathcal{RP}$). Then, each TTP invokes the *KeyGen* algorithm and sends a decryption key to the complainer. The motivation of our work is that a decentralized CP-ABE scheme brings several advantages to an OFE protocol. Firstly, since the key generation algorithm can be executed after the encryption algorithm, it allows distrusted users to construct commitment schemes. Secondly, it eliminates the single point of failure problem when multiple authorities are used as distributed arbiters. Moreover, the distributed arbiters are autonomous after the scheme setup. Thirdly, it provides a fine-grained access control policy when a user hides a secret in ciphertext. Hence, there is no need to communicate with arbiters if they are not involved. Fourthly, the unique attribute strings in an OFE protocol allow arbiters to make decisions without caring about protocol states.

We stress that all messages are transferred through a public channel, but the privacy of the exchanging items is guaranteed. In Round-1, CP-ABE ciphertexts protect privacy of the exchanging items. In Round-2, the ElGamal scheme [18] is utilized to protect privacy of the CP-ABE decryption keys. Further, NIZK proofs are attached in both rounds to acquire public verifiability; they will be explained in Section 4.1 and Section 4.2. However, the TTPs' messages (i.e., decryption keys) are delivered in plaintext in case they make a response in $\mathcal{RP}$.

The proposed OFE protocol is feasible because CP-ABE embeds an access control policy (ACP) in its ciphertext [19, 17]. By designing a nested threshold access control policy (i.e., Equation (7)), we obtain decentralized TTPs. Both PVSS [12, 20, 21] and CP-ABE schemes [19, 17] incorporate secret sharing technique, which is the key to realize decentralized TTPs. However, CP-ABE scheme has an advantage over PVSS schemes, i.e., CP-ABE enables a TTP to broadcast a decryption key, such that a ciphertext can be decrypted publicly. Consequently, compared with previous works [10, 11] that use a PVSS scheme, our CP-ABE-based protocol makes both the players and arbiters publicly verifiable. Moreover, our OFE protocol has a computation complexity of only $O(n)$.

The proof of concept implementation is published on Github[5]. In summary, our contributions are as below:

1. We are the first to build a two-party optimistic fair exchange protocol based on a multi-authorities CP-ABE scheme, achieving decentralized TTPs. We gain advantages of CP-ABE from two aspects: its ciphertexts support nested access control policies and its decryption keys can be sent to any related parties.

2. We achieve decentralized, passive (i.e., they are required only in abnormal cases), autonomous (i.e., they do not communicate with each other) and stateless (i.e., they do not store anything for the exchange protocol) TTPs leveraging the underlying CP-ABE scheme.

3. We propose a recovery process $\mathcal{RP}$ to tackle situations where one or both players behave maliciously. Besides, the protocol accommodates up to $\frac{n-1}{2}$ malicious TTPs when $\mathcal{RP}$ is invoked.

4. We enable both players and TTPs to be publicly verifiable or auditable. Particularly, CP-ABE

[5]Source code: `https://github.com/scottocs/OFE-ABE/`

Encrypt algorithm and ElGamal encryption are used to protect the transferred messages. Then, sigma protocol and Fiat-Shamir transformation are incorporated to obtain NIZK proofs for the ciphertexts.

5. The verification complexity for a player is $O(n)$, lower than $O(nt)$ in Avoine et al. [10] and Zhao et al. [11], where n is the number of TTPs and t is the minimum number of required honest TTPs.

2. PRELIMINARIES

2.1. Bilinear Map

$\mathbb{G}_1$ and $\mathbb{G}_T$ are two different groups of prime order p. Let g be a generator of $\mathbb{G}_1$ and e be a bilinear map, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The bilinear map e has the following properties:

- Bilinearity: for all $\mu \in \mathbb{G}_1$, $\upsilon \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, we have $e(\mu^a, \upsilon^b) = e(\mu, \upsilon)^{ab}$.
- Nondegeneracy: $e(g, g) \neq 1$.
- Computability: There is an efficient algorithm for computing $e(\mu, \upsilon)$, $\forall\ \mu \in \mathbb{G}_1, \upsilon \in \mathbb{G}_1$.

2.2. Threshold-based ACP and Decentralized CP-ABE

Access control is concerned with determining the allowed activities of legitimate users, mediating every attempt by a user to access a resource in the system. An access control policy (ACP) [19] is a proposition defined over attribute strings. A threshold-based ACP contains n propositions and only if at least t of them are True, the ACP is satisfied. A proposition can be either an attribute string or an ACP. When a proposition is an attribute string, it is True only when a given attribute set contains this attribute string. When a proposition is an ACP, its truth is defined recursively as the nested policy's truth. Thus, a threshold-based ACP can be expressed as "t/n *of* $\langle prop_1, prop_2, ...prop_n\rangle$", meaning the ACP is True if any t-out-of-n the propositions ($\{prop_i\}_{i\in\{1,2,...,n\}}$) are satisfied.

For example, "2 *of* $\langle(1$ *of* $\langle attr_1, attr_2\rangle), attr_3, attr_4\rangle$" is a nested ACP, which can be satisfied by attribute set $\{attr_3, attr_4\}$ and attribute set $\{attr_2, attr_3\}$, respectively. The design of the attribute string in our protocol is detailed in Section 5.2. Decentralized CP-ABE [17] is such a primitive that it can generate ciphertexts containing threshold-based ACP, defined as below:

- $\mathsf{GP} \leftarrow GlobalSetup(\Lambda)$. It takes in the security parameter Λ and outputs global parameters GP.
- $\mathsf{sk}_\theta, \mathsf{pk}_\theta \leftarrow AuthSetup(\mathsf{GP}, \mathsf{ID}_\theta)$. Each authority P_θ takes GP and its identifier ID_θ as input to produce its secret key and public key pair (sk_θ, pk_θ).
- $K_{\mathsf{GID},u} \leftarrow KeyGen(\mathsf{GID}, \mathsf{GP}, u, \mathsf{sk}_\theta)$. The algorithm takes in an identity GID, the global parameters GP, an attribute u belonging to the authority P_θ, and the secret key sk_θ for this authority. It produces a key $K_{\mathsf{GID},u}$ for this attribute and identity pair (GID, u).
- $C_M \leftarrow Encrypt(M, acp, \mathsf{GP}, \{\mathsf{pk}_\theta\})$. The algorithm takes in a message M, an access control policy *acp*, a set of public keys, and the global parameters GP. It outputs a ciphertext C_M.
- $M \leftarrow Decrypt(C_M, \mathsf{GP}, \{K_{\mathsf{GID},u}\})$. The decryption algorithm takes in the global parameters GP, the ciphertext C_M, and a set of decryption keys

$\underline{GlobalSetup(\Lambda) \rightarrow \mathsf{GP} :}$

$\mathsf{GP} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_T)$

$\underline{AuthSetup(\mathsf{GP}, \mathsf{ID}_\theta) \rightarrow (\mathsf{sk}_\theta, \mathsf{pk}_\theta) :}$

$\mathsf{sk}_\theta \leftarrow (\alpha_\theta, \beta_\theta)$, where $\alpha_\theta \overset{R}{\leftarrow} \mathbb{Z}_p, \beta_\theta \overset{R}{\leftarrow} \mathbb{Z}_p$
$\mathsf{pk}_\theta \leftarrow (e(g, g)^{\alpha_\theta}, g^{\beta_\theta})$

(*AuthSetup* takes in an authority P_θ's unique identifier ID_θ, and outputs its long-term key pair $(\mathsf{sk}_\theta, \mathsf{pk}_\theta)$.)

$\underline{KeyGen(\mathsf{GID}, \mathsf{GP}, u, \mathsf{sk}_\theta) \rightarrow K_\theta :}$

$$K_\theta = \begin{cases} K_{\mathsf{GID},u} = g^{\alpha_\theta} H(\mathsf{GID})^{\beta_\theta} F(u)^{d_\theta} \\ K'_{\mathsf{GID},u} = g^{d_\theta} \end{cases}$$

(GID is an identifier, which can be arbitrary string. Only if GID is the same, can multiple decryption keys be integrated in *Decrypt* algorithm. u is an attribute string, in the form of *attribute*@ID_θ. d_θ is a random value. H and F map decryptors' identifiers and attribute strings to elements on $\mathbb{G}_1$, respectively. We use K_θ to denote $\{K_{\mathsf{GID},u}, K'_{\mathsf{GID},u}\}$ for authority θ.)

$\underline{Encrypt(M, acp, \mathsf{GP}, \{\mathsf{pk}_\theta\}) \rightarrow C_M :}$

$$C_M = \begin{cases} C_0 = Me(g, g)^s, \\ \{C_{1j} = e(g, g)^{\lambda_j} e(g, g)^{\alpha_{\rho(j)} r_j}, C_{2j} = g^{-r_j}, \\ C_{3j} = g^{\beta_{\rho(j)} r_j} g^{\omega_j}, C_{4j} = F(\sigma(j))^{r_j}\}_{\forall j \in [n]} \end{cases}$$

(s is a randomly chosen value by the encryptor to hide the plaintext M. *acp* is translated into linear secret sharing scheme (LSSS) matrix [22], denoted as $A_{n\times L}$. σ maps A's rows to attributes in *acp*. ρ maps A's rows to authorities. Suppose T maps each attribute to a unique authority, then $\rho(\cdot) \overset{\text{def}}{=} T(\sigma(\cdot))$. λ_j is $A_j \cdot v$, where A_j is the jth row of A and v is a random vector $v \in Z_p^L$ with s as its first entry. ω_j is $A_j \cdot \omega$, where ω is a random vector $\omega \in Z_p^L$ with 0 as its first entry. r_j is randomly chosen from Z_p. The size of the ciphertext C_M and the computation cost of the algorithm are linear with the number of attribute strings in *acp*.)

$\underline{Decrypt(C_M, \mathsf{GP}, \{K_{\mathsf{GID},u}, K'_{\mathsf{GID},u}\}) \rightarrow M :}$

$C_{1j} \cdot e(K_{\mathsf{GID},\sigma(j)}, C_{2j}) \cdot e(H(\mathsf{GID}), C_{3j}) \cdot e(K'_{\mathsf{GID},\sigma(j)}, C_{4j}) = e(g, g)^{\lambda_j} e(H(\mathsf{GID}), g)^{\omega_j}$

$\prod_j (e(g, g)^{\lambda_j} e(H(\mathsf{GID}), g)^{\omega_j})^{c_j} = e(g, g)^s$

$M = C_0 / e(g, g)^s$

($c_j \in Z_p$ are constants such that $\sum_j c_j A_j = (1, 0, ..., 0)$. The second step in the *Decrypt* algorithm is true because $\lambda_j = A_j \cdot v$ and $\omega_j = A_j \cdot w$. The computation cost of the algorithm is linear with the number of attribute strings of *acp* in C_M.)

FIGURE 1: Rouselakis and Waters's decentralized CP-ABE

$\{K_{\mathsf{GID},u}\}$. Suppose each of the decryption keys is generated with GID and an attribute string from an attribute set using $KeyGen$ algorithm. Only if the attribute set satisfies the ACP in C_M, it outputs the message M.

Decentralized CP-ABE scheme, particularly Rouselakis and Waters's scheme (summarized in Figure 1), is used as a building block in this paper. However, we extend the scheme with the $\mathsf{CheckKey}$ algorithm to verify whether a decryption key $K_{\mathsf{GID},u}$ is correctly generated by authority θ.

$\underline{\mathsf{CheckKey}(\mathsf{GP}, K_{\mathsf{GID},u}, \mathsf{pk}_\theta, attr)}$:

$$M_r \xleftarrow{R} \mathbb{G}_T$$
$$acp \leftarrow 1\ of\ \langle attr \rangle$$
$$C_{M_r} = Encrypt(M_r, acp, \mathsf{GP}, (\mathsf{pk}_\theta))$$
$$M_r \overset{?}{=} Decrypt(C_{M_r}, \mathsf{GP}, (K_{\mathsf{GID},u}))$$

COROLLARY 1. No PPT adversary $\mathcal{A}$ could generate a valid decryption key for an honest authority P_θ, meaning $\mathsf{CheckKey}$ $(\mathsf{GP}, K_{\mathcal{A}}, \mathsf{pk}_\theta, attr_\theta)$ always outputs False, where $K_{\mathcal{A}} = KeyGen$ $(\mathsf{GID}, \mathsf{GP}, attr_\theta, \mathsf{sk}_{\mathcal{A}})$.

Proof. This is easily deduced from the decentralized CP-ABE scheme. Since if $\mathcal{A}$ enables $\mathsf{CheckKey}$ algorithm to output True, it breaks the security defined in Definition 2 of the CP-ABE scheme [17]. Therefore, if the $\mathsf{CheckKey}$ algorithm outputs True, then K_θ must be generated by authority P_θ with $attr_\theta$ honestly. □

Rouselakis and Waters's CP-ABE is secure against a static attacker if a slightly modified q-Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption holds [17]. When Rouselakis and Waters's CP-ABE is utilized in our protocol, to distinguish with arbiters who run $AuthSetup$, players invoke $AuthSetupPlayer$ algorithm which extends $AuthSetup$, as below:

$\underline{AuthSetupPlayer(\mathsf{GP}, \mathsf{ID}_i) : \rightarrow (\mathsf{sk}_i, \mathsf{pk}_i)}$

$$\mathsf{sk}_i \leftarrow (\alpha_i, \beta_i, z_i), \text{ where } \alpha_i, \beta_i, z_i \xleftarrow{R} \mathbb{Z}_p$$
$$\mathsf{pk}_i \leftarrow (e(g,g)^{\alpha_i}, g^{\beta_i}, g^{z_i})$$

Remark: the additional key pair $(z_i,\ g^{z_i})$ of player i is used for ElGamal encryption, as required in Round-2 of the protocol, as introduced in Section 5.2.

2.3. ElGamal Scheme

$\mathbb{G}_1$ is a cyclic group of order p with generator g. ElGamal scheme [18] is an asymmetric encryption cryptographic primitive, defined over $\mathbb{G}_1$. It has three algorithms, namely $ElGamalKeyGen$, $ElGamalEnc$, and $ElGamalDec$:

ElGamal scheme

$\underline{ElGamalKeyGen:}$

$$(z \xleftarrow{R} \mathbb{Z}_p,\ g^z)$$

$\underline{ElGamalEnc(K, g^z):}$

$$EK = \begin{cases} EK_1 = K \cdot (g^z)^l \\ EK_2 = g^l \end{cases}$$, where K is a plaintext and $l \xleftarrow{R} \mathbb{Z}_p$.

$\underline{ElGamalDec(EK, z):}$

$$\frac{EK_1}{(EK_2)^z} = \frac{K \cdot (g^z)^l}{(g^l)^z} = K$$

2.4. Sigma Protocol and NIZK Proof

Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ be a relation based on hard problem(s). A sigma protocol [15] for $\mathcal{R}$ is a protocol between a prover P and a verifier V. P has a statement $(M, C_M) \in \mathcal{R}$. P and V work as Figure 2 shows.

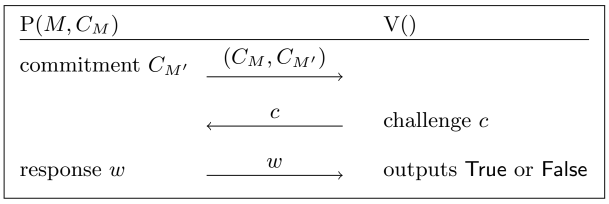


FIGURE 2: Sigma Protocol

$W = ((C_M, C_{M'}), c, w)$ constructs a conversation between P and V, where $C_{M'}$ is a commitment, c is a challenge value and w is a response. V outputs True only if the conversation is accepted. The proof $(C_{M'}, c, w)$ is denoted as proofs_M, thus the conversation $W = (C_M, \mathsf{proofs}_M)$. A sigma protocol has properties of *correctness*, *special knowledge soundness*, and *special honest verifier zero knowledge (HVZK)*, described as below:

- *correctness*: If P is honest, an honest V always accepts the conversation $((C_M, C_{M'}), c, w)$.
- *special Knowledge Soundness*: Given a statement $C_M \in \mathcal{Y}$, along with two accepting conversations $((C_M, C_{M'}),\ c, w)$ and $((C_M, C_{M'}), c', w')$, where c $\neq c'$, one can efficiently compute $M \in \mathcal{X}$ such that $(M, C_M) \in \mathcal{R}$.
- *special HVZK*: There exists a polynomial-time *simulator*, which inputs C_M and random value c, outputs an accepting conversation $((C_M, C_{M'}), c, w)$ with the same probability distribution as conversations between the honest P and V.

Using Fiat-Shamir transformation [16], P can generate c as $H_0(C_M, C_{M'})$, where H_0 is a hash function. Thus, we get a non-interactive or communication round-efficient sigma protocol, which is depicted in Figure 3. We denote the non-interactive process of obtaining proofs_M as $GenProofs(M)$, i.e., $\mathsf{proofs}_M \leftarrow GenProofs(M)$. A sigma protocol with the Fiat-Shamir transformation constructs a non-interactive zero knowledge (NIZK) proof system, proving $(M, C_M) \in \mathcal{R}$. For simplicity of presentation, we denote the "non-interactive" conversation $W = (C_M, \mathsf{proofs}_M)$ as a transcript.

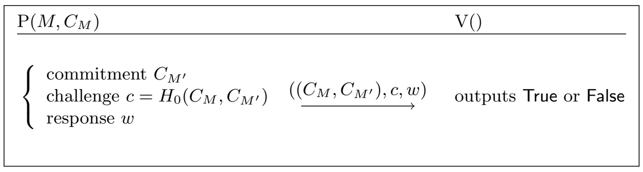


FIGURE 3: Sigma Protocol with Fiat-Shamir transformation

Theorem 1 (Generic Sigma Protocol For Arbitrary Linear Relations [23]). A prover can prove knowledge of $a_1, a_2, ..., a_n \in \mathbb{Z}_p$ in $u = g_1^{a_1} g_2^{a_2} ... g_n^{a_n}$ using the sigma protocol with Fiat-Shamir transformation (as described in Figure 3), where $g_1, g_2, ..., g_n$ are generators of $\mathbb{G}_1$. Schnorr's sigma protocol [24] and Okamoto's sigma protocol [25] are special cases of the generic sigma protocol, where $n = 1$ and $n = 2$, respectively.

3. SYSTEM AND SECURITY MODEL

3.1. System Assumption

We aim to implement an optimistic fair exchange protocol based on a decentralized CP-ABE scheme. At the end of the protocol, either both parties obtain each other's secret (Alice obtains Bob's secret $y \in \mathbb{G}_T$ and Bob obtains Alice's secret $x \in \mathbb{G}_T$), or no one obtains anything useful. If the exchange is accomplished, we say it has a success termination. Otherwise, we say the protocol has a failure termination.

As usual, secret descriptions or commitments, i.e., D_x and D_y, are assumed to be correctly shared beforehand [10]. In our situation, secret M is an element on $\mathbb{G}_T$, which can be represented using $M = e(g, g)^m$ and m is only known by secret holder. Hence, we set $D_M = h^m$, where g and h are randomly chosen generators on $\mathbb{G}_1$.

We demonstrate an OFE protocol with $n + 2$ key roles: player Alice, player Bob and n arbiters (TTPs). If a player or an arbiter obeys the protocol, it is honest; otherwise, it is dishonest or malicious. We assume that Rouselakis and Waters's decentralized CP-ABE [17] *GlobalSetup* algorithm is correctly initialized. Each arbiter P_θ has a unique identifier ID_θ. We assume the n arbiters are authenticated, meaning they invoke *AuthSetup* honestly. We use the item description as the identifier for both players, i.e., $\mathsf{ID}_A = D_x$ and $\mathsf{ID}_B = D_y$. During the protocol execution, a majority of the arbiters are assumed to be honest and we set $t = n/2 + 1$.

Optimism (or passive arbiters) indicates that no arbiters are involved in the normal case, where both players are honest. In the normal case, the protocol is described with two communication rounds. In Round-1, each player commits to its secret value using Rouselakis and Waters's CP-ABE encryption algorithm. In Round-2, each player transfers its CP-ABE decryption key in an encrypted mode using a ElGamal encryption. In the abnormal case, where at least a player is malicious, the arbiters are involved in a recovery process.

We assume synchronized clocks for the communication channel such that a message is promised to arrive in Δt, similar to related works [10, 6, 26]. We also assume an expiry τ for the protocol.[6] Only before τ, the arbiters respond to the players in the recovery process. An honest player waits for the protocol to terminate no earlier than τ. Our protocol also aims to achieve privacy against arbiters and external users.

3.2. Security Model

The protocol involves two players and n arbiters. Since the protocol assumes a public channel, external users should also be considered. If a role is malicious in the protocol, we mean a player or an arbiter deviates the protocol by sending invalid messages or sending nothing in each communication round. To protect fairness of the protocol against a PPT adversary $\mathcal{A}$, we consider the following game.

Phase 1. Before Round-1, the adversary $\mathcal{A}$ controls a player and chooses f arbiters to corrupt. The corrupted player and the f arbiters follow $\mathcal{A}$'s choice in the proposed protocol. The other player obeys the rules throughout the protocol.
Phase 2. Waiting for the other player's message in Round-1, the corrupted player responds to the protocol, following $\mathcal{A}$'s command.
Phase 3. $\mathcal{A}$ tries to break the fairness after Round-1.
Phase 4. If timeouts, the protocol aborts. Otherwise, the protocol enters Round-2, and the corrupted player responds to the protocol, following $\mathcal{A}$'s command.
Phase 5. $\mathcal{A}$ tries to break the fairness in Round-2.

The adversary $\mathcal{A}$ wins the game if fairness is broken or honest player's secret privacy is leaked.

3.3. Properties

We inherit the properties of **Completeness**, **Timeliness**, **Fairness**, **Privacy**, **Optimism** from Avoine et al. [10]:

- **Completeness:** when both players are honest, then player Alice gets y and player Bob gets x at the end of the protocol (i.e., success termination).
- **Timeliness:** the exchange protocol eventually terminates.
- **Fairness:** the exchange protocol ends with either success termination or failure termination.
- **Privacy:** Except the two players, no one can acquire x and y.

[6]Küpçü et al. [27] have concluded that it is impossible to implement an optimistic fair exchange protocol without timeouts.

- **Optimism:** the arbiters are needed only when one player crashes or attempts to cheat.

Further, the TTPs are stateless, passive, autonomous and verifiable throughout the protocol.

4. NIZK PROOFS FOR CP-ABE

4.1. Proving Plaintext Knowledge in CP-ABE Ciphertext

In this section, we describe how to add NIZK proofs to Rouselakis and Waters's decentralized CP-ABE encryption algorithm using sigma protocol and Fiat-Shamir transformation. Thus, the NIZK proofs enable an encryptor to prove knowledge of $M = e(g,g)^m, s, r, \{r_j\}$ in the ciphertext C_M. Without affecting correctness of the CP-ABE scheme, we represent C_0 a tuple as $[g, g^{m+s}]$, because the ciphertext can be caluculated as $e(C_0[0], C_0[1])$. Also, we add $C_5 = h^s$, where s is the random value used in C_0. C_5 reveals no information of s, g^s or $e(g,g)^s$, which is essential to obtain the plaintext M. Then, the ciphertext C_M and the description D_M are as below:

$$\begin{cases} C_M = (C_0, C_1, C_2, C_3, C_4), \\ \quad = \begin{cases} C_0 = [g, g^{s+m}], \\ C_1 = \{C_{1j} = e(g,g)^{\lambda_j} e(g,g)^{\alpha_{\rho(j)} r_j}\}_{\forall j}, \\ C_2 = \{C_{2j} = g^{-r_j}\}_{\forall j}, \\ C_3 = \{C_{3j} = g^{\beta_{\rho(j)} r_j} g^{\omega_j}, \}_{\forall j}, \\ C_4 = \{C_{4j} = F(\sigma(j))^{r_j}\}_{\forall j}, \\ C_5 = h^s \end{cases} \\ D_M \, (= h^m) \end{cases} \tag{1}$$

Suppose P is the prover, who outputs C_M given by Equations (1). Then, P encrypts $M' (\xleftarrow{R} \mathbb{G}_T)$ to $C_{M'}$. Both C_M and $C_{M'}$ are encrypted with the same threshold-based access control policy.

$$\begin{cases} C_{M'} = (C'_0, C'_1, C'_2, C'_3, C'_4) \\ \quad = \begin{cases} C'_0 = [g, g^{s'+m'}], \\ C'_1 = \{C'_{1j} = e(g,g)^{\lambda'_j} e(g,g)^{\alpha_{\rho(j)} r'_j}\}_{\forall j}, \\ C'_2 = \{C'_{2j} = g^{-r'_j}\}_{\forall j}, \\ C'_3 = \{C'_{3j} = g^{\beta_{\rho(j)} r'_j} g^{\omega'_j}\}_{\forall j}, \\ C'_4 = \{C'_{4j} = F(\sigma(j))^{r'_j}\}_{\forall j}, \\ C'_5 = h^{s'} \end{cases} \\ D_{M'} = h^{m'} \end{cases} \tag{2}$$

Thus, the commitment of the sigma protocol is $(C_{M'}, D_{M'})$. Then, P calculates a challenge value $c_1 = H_1(C_M, C_{M'}, D_{M'})$, where H_1 is a random oracle that maps arbitrary data to an element in $\mathbb{Z}_p$. Therefore, the responses $(\tilde{m}, , \tilde{s}, \{\tilde{r_j}, \tilde{\lambda_j}, \tilde{\omega_j}\}_{\forall j})$ are calculated as below:

$$\begin{cases} \tilde{m} = m' - c_1 m, \\ \tilde{s} = s' - c_1 s, \\ \{\tilde{r_j} = r'_j - c_1 r_j, \tilde{\lambda_j} = \lambda'_j - c_1 \lambda_j, \tilde{\omega_j} = \omega'_j - c_1 \omega_j\}_{\forall j} \end{cases}$$

Hence, the NIZK proof $\mathsf{proofs}_M \leftarrow GenProofs(M)$ is $((C_{M'}, D_{M'}), (c_1), (\tilde{m}, \tilde{s}, \{\tilde{r_j}, \tilde{\lambda_j}, \tilde{\omega_j}\}_{\forall j}))$.

Then, transcript of the non-interactive sigma protocol is as below:

$$\begin{aligned} W &= ((C_M), \mathsf{proofs}_M) \\ &= ((C_M, C_{M'}), (c_1), (\tilde{m}, \tilde{s}, \{\tilde{r_j}, \tilde{\lambda_j}, \tilde{\omega_j}\}_{\forall j})) \end{aligned}$$

Upon receiving the transcript W, any verifier V can check the correctness of c_1 and verify the transcript W via Equations (3). We denote the process of verifying Equations (3) as $\mathsf{CheckCiphertext}(W)$. If all the equations hold, $\mathsf{CheckCiphertext}(W)$ outputs True. Otherwise, $\mathsf{CheckCiphertext}(W)$ outputs False.

$\underline{\mathsf{CheckCiphertext}(W):}$

$$\begin{cases} C'_0[1] \overset{?}{=} g^{\tilde{s}} g^{\tilde{m}} C_0[1]^{c_1} \\ \{C'_{1j} \overset{?}{=} e(g,g)^{\tilde{\lambda_j}} e(g,g)^{\alpha_{\rho(j)} \tilde{r_j}} C_{1j}^{c_1}, C'_{2j} \overset{?}{=} g^{-\tilde{r_j}} C_{2j}^{c_1} \\ C'_{3j} \overset{?}{=} g^{\beta_{\rho(j)} \tilde{r_j}} g^{\tilde{\omega_j}} C_{3j}^{c_1}, C'_{4j} \overset{?}{=} F(\sigma(j))^{\tilde{r_j}} C_{4j}^{c_1}\}_{\forall j} \\ C'_5 \overset{?}{=} h^{\tilde{s} C_5^{c_1}} \\ D_{M'} \overset{?}{=} h^{\tilde{m} D_M^{c_1}} \\ e(C_0[1], h) \overset{?}{=} e(C_5 D_M, g) \\ \tilde{s} \overset{?}{=} interpolate(\{\tilde{\lambda_j}\}_{\forall j}) \\ 0 \overset{?}{=} interpolate(\{\tilde{\omega_j}\}_{\forall j}) \end{cases} \tag{3}$$

In Equations (3), *interpolate* is the Lagrange interpolation [28] algorithm, where authorized shares could recover the secret value defined in *acp*. On one hand, *interpolate* is used to prove binding between the secret s (or 0) and the shares $\{\lambda_j\}$ (or $\{\omega_j\}$). On the other hand, it proves that threshold values in *acp* are correct. Besides, $e(C_0[1], h) \overset{?}{=} e(C_5 D_M, g)$ is used to prove binding relationship of m and s, which are used in C_0, C_5 and D_M.

We prove the properties of the sigma protocol constructed in Section 4.1 when adding NIZK proofs to Rouselakis and Waters's CP-ABE ciphertexts.

Proof of correctness The verification in Equations (3) related *interpolate* is straightforward due to the linearity of polynomial functions. By checking all rest the verification equations (3) from right to left, we prove *Correctness* as follows:

$$g^{\tilde{s}} g^{\tilde{m}} C_0[1]^{c_1} = g^{s\prime - c_1 s + m' - c_1 m} \cdot (g^{s+m})^{c_1} = C'_0[1] \tag{4}$$

$$\begin{aligned} & e(g,g)^{\tilde{\lambda_j}} e(g,g)^{\alpha_{\rho(j)} \tilde{r_j}} C_{1j}^{c_1} \\ & \quad = e(g,g)^{\lambda'_j - c_1 \lambda_j} e(g,g)^{\alpha_{\rho(j)} (r'_j - c_1 r_j)} (e(g,g)^{\lambda_j} e(g,g)^{\alpha_{\rho(j)} r_j})^{c_1} \\ & \quad = e(g,g)^{\lambda'_j} e(g,g)^{\alpha_{\rho(j)} r'_j} = C'_{1j} \\ & g^{-\tilde{r_j}} C_{2j}^{c_1} = g^{-(r'_j - c_1 r_j)} g^{-r_j c_1} = g^{-r'_j} = C'_{2j} \\ & g^{\beta_{\rho(j)} \tilde{r_j}} g^{\tilde{\omega_j}} C_{3j}^{c_1} = g^{\beta_{\rho(j)} (r'_j - c_1 r_j)} g^{\omega'_j - c_1 \omega_j} (g^{\beta_{\rho(j)} r_j} g^{\omega_j})^{c_1} \\ & \quad = g^{\beta_{\rho(j)} r'_j} g^{\omega'_j} = C'_{3j} \end{aligned}$$

$$F(\sigma(j))^{\tilde{r_j}}C_{4j}^{c_1} = F(\sigma(j))^{r'_j-c_1r_j}F(\sigma(j))^{r_jc_1} = F(\sigma(j))^{r'_j} = C'_{4j}$$

$$h^{\tilde{s}}C_5^{c_1} = h^{s'-c_1s}(h^s)^{c_1} = C'_5 \tag{5}$$

$$h^{\tilde{m}}D_M^{c_1} = h^{m'-c_1m}(h^m)^{c_1} = h^{m'} = D_{M'}$$

$$e(C_5D_M, g) = e(h^sh^m, g) = e(g^{s+m}, h) = e(C_0[1], h)$$

Proof of special knowledge soundness Given two transcripts (C_M, proofs_M) and $(C_M, \mathsf{proofs}'_M)$, where $\mathsf{proofs}_M = (C_{M'}, c_1, (\tilde{m}, \tilde{s}, \{\tilde{r_j}, \tilde{\lambda}_j, \tilde{\omega_j}\}_{\forall j})$, $\mathsf{proofs}'_M = (C_{M'}, c'_1, (\tilde{m}', \tilde{s}', \{\tilde{r_j}', \tilde{\lambda}_j', \tilde{\omega_j}'\}_{\forall j}))$. The two transcripts share the same sigma protocol commitment $C_{M'}$. with $\begin{cases} \tilde{s} = s' - c_1s, \\ \tilde{s}' = s' - c'_1s \end{cases}$, anyone can calculate $s = \frac{\tilde{s}'-\tilde{s}}{c_1-c'_1}$. Hence, M can be obtained as $M = \frac{C}{e(g,g)^s}$.

Proof of special HVZK In sigma protocol, a *simulator* could generate a conversation in arbitrary order between the prover P and verifier V. Denote $q = e(g, g)$, thus, C_0 is in the form of Mq^s. The simulator randomly chooses $M \xleftarrow{R} \mathbb{G}_T$, $s \xleftarrow{R} \mathbb{Z}_p$, and on input $C_0 \in \mathbb{G}_T$ and c $\in \mathcal{C}$ ($\mathcal{C}$ is the challenge space), then computes $C_{0t} \leftarrow Mq^s/C_0^c$. It is evident that $(C_{0t}, c, (M, s))$ is always an accepting conversation.

In a real conversation, c, M, s are mutually independent where c is uniformly distributed over $\mathbb{Z}_p$, M is uniformly distributed over $\mathbb{G}_T$, s is uniformly distributed over $\mathbb{Z}_p$. With the equation $Mq^s = C_{0t}C_0^c$, C_{0t} is uniquely determined then. Thus, the simulator has the same output distribution with the conversation distribution between P and V.

Similar spcial HVZK proofs can also be applied to other parts of ciphertext C, i.e., C_1, C_2, C_3, C_4.

4.2. Proving Plaintext Knowledge in ElGamal-encrypted Keys

We have proved that the messages (CP-ABE ciphertexts) in Round-1 are publicly verifiable in Section 4.1. Similarly, we add NIZK proofs to the messages in Round-2 (i.e., EK_A and EK_B) in this section, such that EK_A and EK_B are publicly verifiable. Refer to Section 5.2 for more details about the protocol. Without loss of generality, we only add NIZK proofs for Alice in this section. Recall $K_A = \begin{cases} g^{\alpha_A}H(\mathsf{GID})^{\beta_A}F(u)^{d_A} \xrightarrow{\text{as}} K_{A_1} \\ g^{d_A} \xrightarrow{\text{as}} K_{A_2} \end{cases}$. Hence, the encrypted key $EK_A = ElGamalEnc(K_A, g^{z_B}) = ElGamalEnc(K_{A_1}, g^{z_B}) = \begin{cases} g^{\alpha_A}H(\mathsf{GID})^{\beta_A}F(u)^{d_A}(g^{z_B})^{l_A} \xrightarrow{\text{as}} EK_{A_1} \\ g^{d_A} \xrightarrow{\text{as}} EK_{A_2} \\ g^{l_A} \xrightarrow{\text{as}} EK_{A_3} \end{cases}$. Thus, Alice should prove knowledge of α, β, d_A, l_A in EK_A.

Since $\mathsf{pk}_A = \begin{cases} e(g,g)^{\alpha_A} \\ g^{\beta_A} \\ g^{z_A} \end{cases}$ is also publicly known, Alice generates a sigma protocol commitment as $\left(EK'_A = \begin{cases} g^{\alpha'_A}H(\mathsf{GID})^{\beta'_A}F(u)^{d'_A}(g^{z_B})^{l'_A} \xrightarrow{\text{as}} EK'_{A_1} \\ g^{d'_A} \xrightarrow{\text{as}} EK'_{A_2} \\ g^{l'_A} \xrightarrow{\text{as}} EK'_{A_3} \end{cases}, \mathsf{pk}'_A = \begin{cases} e(g,g)^{\alpha'_A} \\ g^{\beta'_A} \\ g^{z'_A} \end{cases} \right)$, where $\alpha'_A, \beta'_A, d'_A, l'_A, z'_A$ are randomly chosen from $\mathbb{Z}_p$. Next, Alice calculates the challenge $c_3 = H_3(EK_A,\ EK_{A'}, \mathsf{pk}_A, \mathsf{pk}'_A)$, where H_3 is a random oracle that maps arbitrary data to an element in $\mathbb{Z}_p$. Then, the sigma protocol responses $(\tilde{d_A}, \tilde{\alpha_A}, \tilde{\beta_A}, \tilde{l_A}, \tilde{z_A})$ are calculated as below:

$$\tilde{d_A} = d'_A - c_3d_A, \tilde{\alpha_A} = \alpha'_A - c_3\alpha_A, \tilde{\beta_A} = \beta'_A - c_3\beta_A,$$
$$\tilde{l_A} = l'_A - c_3l_A, \tilde{z_A} = z'_A - c_3z_A$$

Thus, the $\mathsf{proofs}_{K_A} \leftarrow GenProofs(\alpha_A, \beta_A, d_A, l_A)$ is $((EK'_A, g^{\alpha'_A},\ g^{\beta'_A}), c_3,\ (\tilde{d_A}, \tilde{\alpha_A}, \tilde{\beta_A}, \tilde{l_A}))$ and the corresponding sigma protocol transcript $U = (EK_A$, $\mathsf{proofs}_{K_A})$. The verification algorithm $\mathsf{CheckEncKey}$ is defined as below:

$\underline{\mathsf{CheckEncKey}(U):}$

$$\begin{cases} EK'_{A_1} \stackrel{?}{=} g^{\tilde{\alpha_A}}H(\mathsf{GID})^{\tilde{\beta_A}}F(u)^{\tilde{d_A}}(g^{z_B})^{\tilde{l_A}}(EK_{A_1})^{c_3} \\ EK'_{A_2} \stackrel{?}{=} g^{\tilde{d_A}}(EK_{A_2})^{c_3} \\ EK'_{A_3} \stackrel{?}{=} g^{\tilde{l_A}}(EK_{A_3})^{c_3} \\ e(g,g)^{\alpha'_A} \stackrel{?}{=} e(g,g)^{\tilde{\alpha_A}}(e(g,g)^{\alpha_A})^{c_3} \\ g^{\beta'_A} \stackrel{?}{=} g^{\tilde{\beta_A}}(g^{\beta_A})^{c_3} \\ g^{z'_A} \stackrel{?}{=} g^{\tilde{z_A}}(g^{z_A})^{c_3} \end{cases} \tag{6}$$

Hence, any external user can invoke $\mathsf{CheckEncKey}(U)$, described by Equations (6), to judge whether Alice is honest or not in Round-2. By Thereom 1, we can conclude that our solution is correct when making EK_A publicly verifiable. Due to the similarity with adding NIZK proofs to CP-ABE ciphertext in Section 4.1, we omit the details in proving properties of the sigma protocol (i.e., *Correctness*, *special knowledge soundness* and *special HVZK*).

It is interesting to compare the algorithm $\mathsf{CheckKey}$, which verifies an original CP-ABE decryption key K_A, and the algorithm $\mathsf{CheckEncKey}$, which verifies an ElGamal-encrypted CP-ABE decryption key EK_A. $\mathsf{CheckKey}$ works, which can be deduced from CP-ABE algorithm, as Corollary 1 shows. Obviously, the $\mathsf{CheckKey}$ algorithm does not provide zero knowledge proof for K_θ. However, $\mathsf{CheckEncKey}$ constructs a NIZK proof system. That is because it is based on sigma protocol with Fiat-Shamir transformation.

5. OUR PROTOCOL IN DETAILS

5.1. Setup

We describe the OFE protocol between Alice and Bob, who hold secret values $x(\in \mathbb{G}_T)$ and $y(\in \mathbb{G}_T)$, respectively. Figure 4 demonstrates how to setup for the two players. After setup, the public parameters are $(\mathsf{GP}, D_x, D_y, \{\mathsf{ID}_\theta\}, \mathsf{pk}_A,\ \mathsf{pk}_B,\ \{\mathsf{pk}_\theta\}, \tau)$ for both players, where g^{z_A}, g^{z_B} are included in pk_A and pk_B, respectively. Similarly, z_A,z_B are included in sk_A and pk_B.

```
Setup$(\mathsf{ID}_A = D_x, \mathsf{ID}_B = D_y)$ :
    $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow AuthSetupPlayer(\mathsf{GP}, \mathsf{ID}_A)$
    $(\mathsf{sk}_B, \mathsf{pk}_B) \leftarrow AuthSetupPlayer(\mathsf{GP}, \mathsf{ID}_B)$
```

FIGURE 4: Setup

5.2. Exchanging Values/Elements on $\mathbb{G}_T$

Since attribute strings are with the form of $attribute@\mathsf{ID}_\theta$, we define attribute string $attr_i$ as "$\tau||D_x||D_y||\mathsf{pk}_i@\mathsf{ID}_i$". Thus, the attribute string can be calculated by anyone, but can only be used by arbiter P_i in $KeyGen$ algorithm. This can be deduced by Corollary 1.

Figure 5 gives the details of the protocol. Without loss of generality, we depict the operations of Alice in the following of this section. In abnormal cases, a recovery process ($\mathcal{RP}_{\theta\leftarrow Alice}$) for each arbiter P_θ is required, which is defined in Section 5.3.

1. Input $(x, \mathsf{sk}_A, D_x, D_y, \{\mathsf{pk}_\theta\}, \mathsf{pk}_A, \mathsf{pk}_B, \mathsf{GP}, \tau, \{\mathsf{ID}_\theta\})$, Alice calculates $attr_i = \tau||D_x||D_y||\mathsf{pk}_i@\mathsf{ID}_i$, $\mathsf{GID} = D_x||D_y$ for each role P_i and the threshold-based access control policy acp:

$$acp = 2\ of\ \langle (t\ of\ \langle attr_1, attr_2, ..., attr_n\rangle), \\ attr_A, attr_B\rangle \quad (7)$$

   The acp construction means if a message is encrypted with the access control policy, at least 2 out of 3, i.e., Alice, Bob and at least $n/2+1$ arbiters (the arbiters are as a whole), can successfully decrypt the corresponding ciphertext. (Note that only at least $n/2 + 1$ of the arbiters are enough when the arbiters are required.)

2. Alice obtains a sigma protocol transcript X, which is comprised of $D_x, C_x = Encrypt(x, acp, \mathsf{GP}, (\mathsf{pk}_A, \mathsf{pk}_B, \{\mathsf{pk}_\theta\}))$ and $\mathsf{proofs}_x \leftarrow GenProofs(x)$, where $GenProofs(x)$ is defined in Section 2.4 and implemented in Section 4.1. Alice initiates two local variables $FlagR1$ = False and $FlagR2$ = True, indicating whether messages of Round-1 and Round-2 are valid, respectively. Then, she sends X to Bob in the Round-1.

3. In the normal case (before τ), upon receiving valid Y (i.e., $\mathsf{CheckCiphertext}(Y)$ == True) in Round-1, Alice first generates a CP-ABE decryption key $K_A = KeyGen(\mathsf{GID}, \mathsf{GP}, attr_A, \mathsf{sk}_A)$. Next, Alice sets $FlagR1$ = True. Then, Alice encrypts K_A to EK_A using $ElGamalEnc$ algorithm and generates the corresponding NIZK proofs proofs_{K_A}. proofs_{K_A} is used for public verifiability when audit is required by external users and it is introduced in Section 4.2. Lastly, Alice sends transcript $U = (EK_A, \mathsf{proofs}_{K_A})$ to Bob.

4. Upon receiving transcript $V = (EK_B, \mathsf{proofs}_{K_B})$ from Bob in Round-2, Alice decrypts EK_B to get K_B using $ElGamalDec$ algorithm. Then, Alice directly uses CheckKey to verify whether K_B is valid. Finally, Alice decrypts C_y to get y with decryption keys K_A and K_B, i.e., $Decrypt(C_y, \mathsf{GP}, \{K_A, K_B\}) \rightarrow y$.

5. In case Bob sends invalid EK_B ($\mathsf{CheckKey}(\mathsf{GP}, K_B, \mathsf{pk}_B,\ attr_B)$ == False) or does not send anything to Alice in Round-2 before τ, Alice invokes the recovery process through $\mathcal{RP}_{\theta\leftarrow Alice}(X, Y, \tau)$ for each arbiter P_θ; In case Bob calls $\mathcal{RP}$ after receiving Alice's X in a rush, then Alice will get Y and K_θ from the arbiters before $\tau + \Delta t$ (where Δt is a short time in which honest messages should be arrived). In both cases, Alice decrypts C_y to get y with decryption keys K_A and $\{K_\theta\}$ which contains at least $n/2+1$ valid keys, i.e., $Decrypt(C_y, \mathsf{GP}, (K_A, \{K_\theta\})) \rightarrow y$.

6. If time exceeds $\tau + \Delta t$ (this is the case Bob has behaved maliciously in Round-1), the protocol ends with failure termination.

5.3. Recovery Process $\mathcal{RP}$

In this section, we present a recovery process $\mathcal{RP}$, where the arbiters are involved in helping a player to recover the other player's value. Without loss of generality, we demonstrate Alice as the invoker of the recovery process, which is depicted in Figure 6. The input of $\mathcal{RP}_{\theta\leftarrow Alice}$ for each arbiter P_θ is (X, Y, τ) and the output is X and K_θ. Thus, once the $\mathcal{RP}$ is successfully executed, both players obtain $X, Y, \{K_\theta\}$ and at least $n/2 + 1$ of the keys $\{K_\theta\}$ are valid due to honest majority of the arbiters.

1. If it is before τ when receiving (X, Y, τ), each arbiter P_θ checks the validity of X using $\mathsf{CheckCiphertext}(X)$. If X is valid, P_θ calculates decryption key $K_\theta = KeyGen(\mathsf{GID}, \mathsf{GP}, attr_\theta, \mathsf{sk}_\theta)$, where $\mathsf{GID} = D_x||D_y$ and $attr_\theta = \tau||D_x||D_y||\mathsf{pk}_\theta@\mathsf{ID}_\theta$. ($D_x$ is contained in X, and D_y is contained in Y.)

2. P_θ sends K_θ to Alice and sends (X, K_θ) to Bob at the same time. (Bob can attach his HTTP address together with Y in Round-1. When Alice invokes $\mathcal{RP}$, she also sends Bob's address to arbiters. Thus, arbiters can contact Bob with his address. The address can be proved to be authenticated with

```
Alice $(x, z_A, \mathsf{sk}_A, D_x, D_y, \{\mathsf{pk}_\theta\}, \mathsf{pk}_A, \mathsf{pk}_B, \mathsf{GP}, \tau, \{\mathsf{ID}_\theta\})$    External user    Bob $(y, z_B, \mathsf{sk}_B, D_x, D_y, \{\mathsf{pk}_\theta\}, \mathsf{pk}_A, \mathsf{pk}_B, \mathsf{GP}, \tau, \{\mathsf{ID}_\theta\})$

Alice:
$\mathsf{GID} = D_x||D_y$
$attr_\theta = \tau||D_x||D_y||\mathsf{pk}_\theta@\mathsf{ID}_i (i = \theta, A, B)$
$acp = 2\ of\ \langle (t\ of\ \langle attr_1, ..., attr_n \rangle), attr_A, attr_B \rangle$
$X = \begin{cases} D_x \\ C_x = Encrypt(x, acp, \mathsf{GP}, (\mathsf{pk}_A, \mathsf{pk}_B, \{\mathsf{pk}_\theta\})) \\ \mathsf{proofs}_x \leftarrow GenProofs(x) \end{cases}$
$FlagR1 = \mathsf{False}$
$FlagR2 = \mathsf{True}$

Bob:
$\mathsf{GID} = D_x||D_y$
$attr_i = \tau||D_x||D_y||\mathsf{pk}_i@\mathsf{ID}_i (i = \theta, A, B)$
$acp = 2\ of\ \langle (t\ of\ \langle attr_1, ..., attr_n \rangle), attr_A, attr_B \rangle$
$Y = \begin{cases} D_y \\ C_y = Encrypt(y, acp, \mathsf{GP}, (\mathsf{pk}_A, \mathsf{pk}_B, \{\mathsf{pk}_\theta\})) \\ \mathsf{proofs}_y \leftarrow GenProofs(y) \end{cases}$
$FlagR1 = \mathsf{False}$
$FlagR2 = \mathsf{True}$

                    Round-1 : $(X, Y)$  $\longleftrightarrow$

Alice:
while $ts$=current time $\leq \tau + \Delta t$ :
  if CheckCiphertext($Y$) and $ts<\tau$ and !$FlagR1$
    $K_A = KeyGen(\mathsf{GID}, \mathsf{GP}, attr_A, \mathsf{sk}_A)$
    $FlagR1 = \mathsf{True}$
    $U = \begin{cases} EK_A = ElGamalEnc(K_A, g^{z_B}) \\ \mathsf{proofs}_{K_A} = GenProofs(K_A) \end{cases}$

Bob:
while $ts$=current time $\leq \tau + \Delta t$ :
  if CheckCiphertext($X$) and $ts<\tau$ and !$FlagR1$
    $K_B = KeyGen(\mathsf{GID}, \mathsf{GP}, attr_B, \mathsf{sk}_B)$
    $FlagR1 = \mathsf{True}$
    $V = \begin{cases} EK_B = ElGamalEnc(K_B, g^{z_A}) \\ \mathsf{proofs}_{K_B} = GenProofs(K_B) \end{cases}$

                    Round-2 : $(U, V)$  $\longleftrightarrow$

Alice:
  else if $V$ from Bob and $ts<\tau$
    $K_B \leftarrow ElGamalDec(EK_B, z_A)$
    if CheckKey($\mathsf{GP}, K_B, \mathsf{pk}_B, attr_B$) == True
      $Decrypt(C_y, \mathsf{GP}, \{K_A, K_B\}) \rightarrow y$
      success termination
    else
      $FlagR2 = \mathsf{False}$
  else if $FlagR1$ and (!$FlagR2$ or $ts \geq \tau$)
    $(X, K_\theta) \leftarrow \mathcal{RP}_{\theta \leftarrow Alice}(X, Y, \tau)$
    break
  else if message from $\mathcal{RP}_{\theta \leftarrow Bob}$
    $(Y, K_\theta) \leftarrow \mathcal{RP}_{\theta \leftarrow Bob}$
    break
if current time $>\tau + \Delta t$
  failure termination
$Decrypt(C_y, \mathsf{GP}, (K_A, \{K_\theta\})) \rightarrow y$
success termination

Bob:
  if $U$ from Alice and $ts<\tau$
    $K_A \leftarrow ElGamalDec(EK_A, z_B)$
    if CheckKey($\mathsf{GP}, K_A, \mathsf{pk}_A, attr_A$) == True
      $Decrypt(C_x, \mathsf{GP}, \{K_A, K_B\}) \rightarrow x$
      success termination
    else
      $FlagR2 = \mathsf{False}$
  else if $FlagR1$ and (!$FlagR2$ or $ts \geq \tau$)
    $(Y, K_\theta) \leftarrow \mathcal{RP}_{\theta \leftarrow Bob}(X, Y, \tau)$
    break
  else if message from $\mathcal{RP}_{\theta \leftarrow Alice}$
    $(X, K_\theta) \leftarrow \mathcal{RP}_{\theta \leftarrow Alice}$
    break
if current time $>\tau + \Delta t$
  failure termination
$Decrypt(C_x, \mathsf{GP}, (K_B, \{K_\theta\})) \rightarrow x$
success termination
```

FIGURE 5: Optimistic fair exchange protocol between Alice and Bob

```
Bob          Arbiter $\theta$ $(\mathsf{GP}, \mathsf{sk}_\theta, \mathsf{pk}_\theta, \mathsf{ID}_\theta)$          Alice
                                                  $\xleftarrow{X, Y, \tau}$
             assert current time $\leq \tau$
             assert CheckCiphertext($X$) = True
             $\mathsf{GID} = D_x||D_y$
             $attr_\theta = \tau||D_x||D_y||\mathsf{pk}_\theta@\mathsf{ID}_\theta$
$\xleftarrow{X, K_\theta}$  $K_\theta \leftarrow KeyGen(\mathsf{GID}, \mathsf{GP}, attr_\theta, \mathsf{sk}_\theta)$  $\xrightarrow{K_\theta}$
```

FIGURE 6: Recovery process $\mathcal{RP}_{\theta \leftarrow Alice}$

Bob's verification key. Due to limited space, we do not go further about the details in this paper.)

5.4. Complexity in the Normal Case

The normal case is the scenario where both players are honest and the recovery process is not required. Since X, Y are of size $O(n)$ and EK_A, EK_B are of size $O(1)$, there are 4 messages transferred in total with communication complexity of $O(n)$. To reduce the communication messages, Alice can send the data of Round-1 and Round-2 (i.e., X, EK_A) simultaneously to Bob if she receives a valid Y.

The computation complexity of generating the transcript in Round-1 for Alice is $O(n)$, which can be deduced from the decentralized CP-ABE *Encrypt* algorithm. Hence, the verification complexity (i.e., CheckCiphertext) in Round-1 for Alice is $O(n)$. The verification complexity (i.e., CheckKey) in Round-2 for Alice is $O(1)$, as defined in Section 2.2. The computation complexity of decrypting the message in Round-2 for Alice is $O(1)$, where only two decryption keys are involved. Thus, the total computation complexity for a player is $O(n)$.

We enable any external user to verify whether a role has behaved dishonestly. For the external user, checking the messages in Round-1 costs $O(n)$ using CheckCiphertext algorithm; checking the messages in Round-2 costs $O(1)$ using CheckEncKey algorithm; checking the message of an arbiter, in case the recovery process is invoked, is $O(1)$ using CheckKey algorithm.

6. EVALUATION

6.1. Experimentation

We implement the proposed fair exchange protocol with Charm-Crypto library[7], which is a framework for constructing cryptographic schemes. Charm-Crypto also provides classic cryptographic primitives as its built-in examples. Based on this library, we make it compatible with the threshold-based access control policy. We utilize curve "SS512" throughout the experiment. The experiments are conducted on MacBook Air macOS 11.3.1, 1.6 GHz Dual-Core Intel Core i5 CPU, 16 GB RAM, with Python 3.7.7.

Figure 7 shows the performance of Rouselakis and Waters's decentralized CP-ABE algorithms. The encryption and decryption cost scales linearly with the number of attributes (equal to the number of all roles). Since an arbiter only uses one attribute, the key generation algorithm costs constant time. The figure depicts the time cost when the number of arbiters ranges from 10 to 100. When there are 100 arbiters, the encryption cost is about 1.33s and the decryption cost is about 0.12s.

Figure 8 shows the ciphertext size and the message size, which is ciphertext size plus the proofs size. We can conclude that the decentralized CP-ABE ciphertext also increases linearly with the number of attributes. The results show that the ciphertext (C_M) size is of 57KB and a player sends only 145KB data (the transcript X) when there are 100 arbiters in the protocol. The decryption key is of constant size, which is negligible to the message size.

Figure 9 depicts the time cost when a player verifies different parts of the other player's CP-ABE ciphertext of Equations (3), namely $C_0, C_1, C_2, C_3, C_4,\ C_5\ , D_M$. From Figure 9, we can see that C_0 and C_5 are verified in constant time, while $C_1, C_2,\ C_3$ and C_4 cost linearly with the number of attributes in the ciphertext. Figure 10 gives the total verification cost of Equations (3). The figure indicates that total verification complexity is $O(n)$, since C_1, C_2, C_3 and C_4 all have $O(n)$ equations to verify, where n is the number of arbiters.

[7]https://github.com/JHUISI/charm, Accessed: 2022-03-21

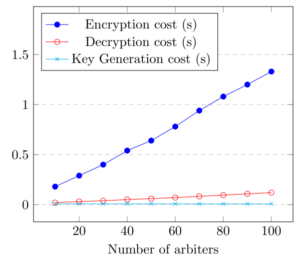


FIGURE 7: Time cost (s) of the CP-ABE algorithms

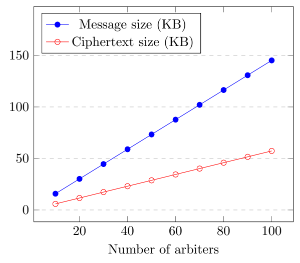


FIGURE 8: Message size and Ciphertext Size

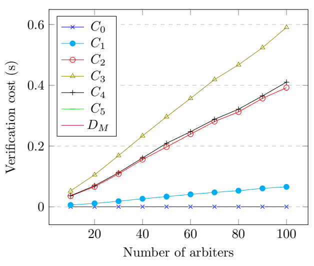


FIGURE 9: Verification cost (s) of different parts

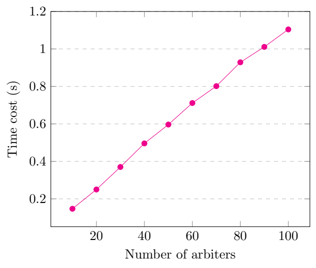


FIGURE 10: Total cost (s) of verifying the ciphertext

The verification of the messages (EK_A and EK_B) in Round-2 costs 13 exponentiations and 8 multiplications on $\mathbb{G}_1$ or $\mathbb{G}_T$, as Equations (6) show. Thus, the verification time cost is constant and it is 0.025s in our experiment.

6.2. Security Analysis

LEMMA 1. If both players are honest in Round-1, the protocol has fairness with success termination.

Proof. If both players continue to behave honestly in Round-2, then the two players exchange their values successfully, as the normal case demonstrates. If a player, say Alice, does not receive a valid EK_B in Round-2, she invokes the recovery process to obtain decryption keys of the arbiters, enabling her to get y as well. Therefore, the protocol ends with success termination if both players are honest in Round-1. $\square$

Without loss of generality, we consider Bob as the malicious player in this section. We prove security against a PPT adversary $\mathcal{A}$ in the game defined in Section 3.2, as follows.

Phase 1. Before Round-1, the adversary $\mathcal{A}$ controls a player (Say Bob) and chooses f arbiters to corrupt. The player and the f arbiters follow $\mathcal{A}$'s choice in the proposed protocol. The other player (Alice) obeys the rules throughout the protocol.

Phase 2. Waiting for Alice's message X in Round-1, if Bob behaves honestly, the protocol would ends with success termination as Lemma 1 demonstrates.

Otherwise, $\mathcal{A}$ holds Bob's message Y or lets Bob send an invalid Y' to the Alice.

Phase 3. $\mathcal{A}$ tries to gain advantage from Round-1. We prove that $\mathcal{A}$ does not violate the protocol fairness as follows. The f malicious arbiters are insufficient in recovering x from X, since *acp* in X cannot be satisfied with only $f(< n/2 < t)$ keys. Hence, we consider the scenario that Bob starts rushing attack by invoking the recovery process $\mathcal{RP}$. Firstly, if Bob correctly invokes $\mathcal{RP}_{\theta \leftarrow Bob}(X, Y, \tau)$, the arbiters in $\mathcal{RP}$ check the transcript Y, so that Alice can obtain Y and the arbiters' $\{K_\theta\}$. As a consequence, the protocol ends with $\mathsf{success}$ termination; Secondly, if Bob forges an invalid Y' (i.e., $\mathsf{CheckCiphertext}(Y') == \mathsf{False}$), the honest arbiters will abort the recovery process, leading to $\mathsf{failure}$ termination; Thirdly, since the arbiters are stateless, Bob can forge the expiry τ', a valid $X' = (D'_x, C'_x, \mathsf{proofs}'_x)$ or a valid $Y' = (D'_y, C'_y, \mathsf{proofs}'_y)$. In this case, the corresponding decryption key K'_θ, generated from attribute string $\tau' || D'_x || D'_y || \mathsf{pk}_\theta @ \mathsf{ID}_\theta$, will be useless. Thus, the protocol ends with $\mathsf{failure}$ termination. Therefore, Lemma 2 can be deduced.

Phase 4. If Bob does not send the valid message Y and timeout occurs $(\tau + \Delta t)$, the protocol ends with $\mathsf{failure}$ termination, where no player learns the other player's secret. Otherwise, the protocol enters Round-2.

Phase 5. By Lemma 1, $\mathsf{success}$ termination is achieved when both players behave honestly in Round-1 and the protocol enters Round-2. In Round-2, Alice can at least obtain Bob's secret value from the honest arbiters. Hence, the adversary $\mathcal{A}$ cannot break fairness of the protocol.

LEMMA 2. A PPT adversary $\mathcal{A}$ is unable to break fairness through cheating the arbiters (or by rushing attack) through invoking $\mathcal{RP}$.

6.3. Protocol Properties

We demonstrate that all required properties, defined in Section 3.3, of the proposed optimistic fair exchange protocol hold in this section.

THEOREM 2 (Completeness). When both players are honest, then player Alice gets y and player Bob gets x at the end of the protocol.

Proof. In section 5.2, we depict how the protocol is designed in detail. When both players are honest, they commit to their corresponding secret values in Round-1. In Round-2, they exchange their respective decryption keys. With the decryption keys, Alice decrypts to get y and Bob decrypts to get x. Therefore, the protocol is accomplished with completeness. □

THEOREM 3 (Timeliness). The exchange protocol eventually terminates.

Proof. If both players are honest, the protocol ends with $\mathsf{success}$ termination in 2 rounds before the expiry τ. If arbitration is required (i.e., $\mathcal{RP}$ is invoked), the arbiters respond to a player only if the request time is earlier than the expiry τ. Thus, players wait for the response from the recovery process only until $\tau + \Delta t$, where Δt is a short time in which honest messages should arrive. Therefore, the exchange protocol eventually terminates no later than $\tau + \Delta t$. In particular, if the protocol ends before $\tau + \Delta t$, the result is $\mathsf{success}$ termination; if the protocol ends at $\tau + \Delta t$, the result is $\mathsf{failure}$ termination. □

THEOREM 4 (Fairness). The protocol ends with either $\mathsf{success}$ or $\mathsf{failure}$ termination.

Proof. Section 6.2 proves that the protocol keeps fairness against a PPT adversary. Hence, the protocol ends with either $\mathsf{success}$ or $\mathsf{failure}$ termination. □

THEOREM 5 (Privacy). No one, except the two players, can get x and y.

Proof. All transferred messages are sent publicly, and any external user (including dishonest arbiters) can access them. The exchanged values x and y are committed in CP-ABE ciphertexts C_x and C_y, whose ACP is as Equation (7) shows. Only with decryption keys of at least 2 out of Alice, Bob and the arbiters (as a whole), can Alice (or Bob) decrypt C_y (or C_x) successfully. In addition, the CP-ABE decryption keys of the two players are encrypted using the ElGamal scheme in Round-2. Thus, with only the arbiters' CP-ABE decryption keys, no one could decrypt C_x or C_y. Moreover, the NIZK proofs obtained from sigma protocol and Fiat-Shamir transformation reveal no useful information about the secret x and y in both rounds. Therefore, no matter the protocol ends with $\mathsf{success}$ or $\mathsf{failure}$ termination, no arbiter or external user acquires x and y. □

THEOREM 6 (Optimism). The arbiters are needed only when one player crashes or attempts to cheat.

Proof. As described in Figure 5 of Section 5.2, Alice and Bob accomplish the exchange in two rounds without any help from arbiters if they both follow the protocol. In Round-1, they exchange committed values, which are publicly verifiable. In Round-2, they exchange the decryption keys for decrypting the committed ciphertext. Therefore, no other party is needed if both players are honest. □

THEOREM 7. The TTPs are stateless, passive, autonomous and verifiable throughout the protocol.

Proof. As described in proving Lemma 2, players cannot cheat the arbiters with forged input of $\mathcal{RP}$. Thus, the arbiters do not need to store any information, indicating they are stateless. The TTPs are passive, which is implied by the property of optimism by Theorem 6. In $\mathcal{RP}$, the arbiters respond to the players with decryption keys, generated only relying on the players' input, and they do not interact with each other.

Therefore, the TTPs are autonomous. In addition, the only message of an arbiter P_θ is the decryption key K_θ, which can be verified via $\mathsf{CheckKey}(\mathsf{GP}, K_\theta, \mathsf{pk}_\theta, attr_\theta)$ by any external user. Thus, we get verifiable TTPs. □

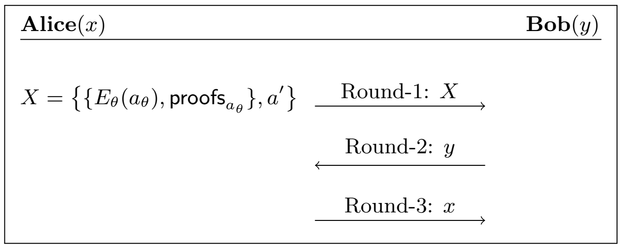
THEOREM 8. The protocol is publicly verifiable.

Proof. The players' messages in Round-1 (i.e., (X, Y)) and Round-2 (i.e., (U, V)) are publicly verifiable with respective NIZK proofs. Also, the arbiters' messages can be verified by Theorem 7. Hence, the protocol is publicly verifiable. □

7. RELATED WORKS

7.1. Comparison with OFE Protocols using Decentralized Arbiters

We give a comparison with related works [10, 11], which all incorporate decentralized arbiters, in the normal case. Avoine et al. [10] and Zhao et al. [11] propose OFE protocols based on Stadler's [12] PVSS scheme. Figure 11 depicts Avoine et al. [10] protocol in the normal case. (PVSS is similarly used in Zhao et al. [11], which is not presented due to space limitation.)



Alice(x)
Bob(y)
$X = \{\{E_\theta(a_\theta), \mathsf{proofs}_{a_\theta}\}, a'\}$
Round-1: X
Round-2: y
Round-3: x

FIGURE 11: Avoine et al. fair exchange protocol

1. Alice selects a random value $a \in \mathbb{Z}_p$, and calculates $a' = x - a$, where $x \in \mathbb{Z}_p$ is Alice's secret item. Alice runs the Stadler's PVSS protocol to share the secret a and gets the encrypted shares $\{E_\theta(a_\theta)\}$ along with $\{\mathsf{proofs}_{a_\theta}\}$, where $\mathsf{proofs}_{a_\theta}$ proves plaintext of knowledge a_θ in ciphertext $E_\theta(a_\theta)$ of Elgamal cryptosystem. Then, Alice sends $\{E_\theta(a_\theta), \mathsf{proofs}_{a_\theta}\}$ and a' to Bob in Round-1.

2. Bob checks the validity of each encrypted share $E_\theta(a_\theta)$ using $\mathsf{proofs}_{a_\theta}$ and checks whether $g^a = g^x \cdot g^{-a'}$ holds. If the verification passes, Bob sends his secret item $y \in \mathbb{Z}_p$ to Alice securely in Round-2.

3. If y is correct, that is the discrete logarithm of g^y with respec to the base g, Alice sends x to Bob securely in Round-3.

4. If Bob does not receive valid x in Round-3, he asks the arbiters to help him recover a and x.

However, Stadler's PVSS has a verification complexity of $O(nt)$ [12, 21], where n is the number of arbiters and t is the threshold value. Usually, t is assumed to be greater than $n/2$ to get rid of collusion attack. Avoine et al. [10] used private/secure channel to transfer secret data, indicating both arbiters and players are not publicly verifiable (**P.V.**). Compared to Avoine et al. [10], Zhao et al. [11] eliminate private communication channel and make the arbiters' behaviors publicly verifiable.

In our protocol, $n + 2$ attribute strings are included in *acp*, leading to $O(n)$ pairings and exponentiations in *Encrypt* and *Decrypt* algorithms. Thus, each player has only $O(n)$ computation (**Comp.**) complexity. Moreover, arbiters and players are publicly verifiable due to the NIZK proofs acquired using sigma protocol and Fiat-Shamir transformation. CP-ABE has size of $O(n)$ ciphertexts and PVSS has n size of $O(1)$ encrypted shares. Hence, all above protocols have a communication (**Comm.**) complexity of $O(n)$.

Table 1 demonstrates the comparisons.

Protocol	Item	Channel	**Comp.**	**Comm.**	**P.V.** players	**P.V.** TTPs
Avoine et al. [10]	$\in \mathbb{Z}_p$	private	$O(nt)$	$O(n)$	✗	✗
Zhao et al. [11]	$\in \mathbb{Z}_p$	public	$O(nt)$	$O(n)$	✓	✗
Ours	$\in \mathbb{G}_T$	public	$O(n)$	$O(n)$	✓	✓

TABLE 1: Protocols with decentralized TTPs

When arbiters are required in the abnormal case, our protocol has a communication complexity of $O(n^2)$ for a player and each arbiter costs $O(n)$ to verify the player's recovery request. Avoine et al. [10] and Zhao et al. [11] cost $O(n)$ communication complexity and each arbiter costs $O(1)$ to verify the player's request. However, since our protocol is publicly verifiable, we can fill the gap by incorporating Ethereum blockchain [29], which provides public available storage and verifiable computation. That is our future work.

7.2. Fair Exchange of Digital Signatures

Asokan et al. [2] propose the first optimistic fair exchange (OFE) of digital signatures in 1998. They assume asynchronous communication model such that no synchronized clock is assumed. The protocol can be easily adapted to exchange encrypted data. However, Pfitzmann et al. [30] prove that it is impossible to obtain stateless TTP(s) in an asynchronous setting. In traditional OFE of digital signature protocols, a player's partial signature sometimes leaks information. Due to this problem, Garay et al. [31] present the abuse-free OFE protocol. Later, OFE protocols of threshold signatures [33] and ring signatures [32] are considered. Huang et al. [34] give an OFE protocol with properties of accountability and transparency, where accountability means some misbehaving parties should be detected and transparency means the arbiter's involvement is transparent. Huang et al. [35] introduce the privacy-preserving property of digital signatures fair exchange protocol against completely malicious arbitrator. Recently, Zhang et al. [36]

propose a modified verifiable encrypted signature scheme leveraging the Ethereum smart contract, which assumed to decentralized, publicly verifiable and tamper-resistant, to guarantee fairness.

7.3. Blockchain-based Fair Exchange

A significant amount of secure multi-party computation protocols, including fair exchange protocols, are implemented over blockchain, due to its decentralization. The monetary system of a blockchain is also a critical property for fair exchange protocols. Blockchain can easily provide incentivization and penalty for parties in the protocol. Since physical goods exchange still requires a trusted third party in case of disputes, we only consider digital goods. We distinguish two protocols based on blockchain. 1). Exchanging items: two digital items (i.e., secrets) are exchanged; 2). Selling item: trading an item with digital currency. One of the two protocols does not imply the other. Most known blockchain-based fair exchange protocols without any TTP are "Selling items" protocols.

A significant job on blockchain we can do is to utilize the "claim-or-refund" functionality [37]. Claim-or-refund is used in fair exchange protocols as follows. In the claim phase, as receiver, a party claims to reward m coins if $\Phi(x) = 1$, where x is the desired digital item and Φ is a transparent function in a smart contract. In the refund phase, the receiver can refund its m coins if nobody has claimed the reward yet. It is easy to see that the protocol belongs to "Selling items". Followed by FairSwap [38], OptiSwap [39] and FileBounty [40], all of which focus on "Selling files". As a file, x is super large, leading to the impossibility to store it and take computation over it on blockchain. Delgado-Segura et al. [41] take advantage of private key locked transactions [42] to implement "Selling item" in which the transaction output can be redeemed by anyone who provides a private key corresponding to a predefined public key. Wagner et al. [43] construct a SmartJudge, which moderates two-party interactions in case of a dispute. SmartJudge, built on Ethereum, provides the functionality of cross-blockchain trades and exchanging digital goods with high efficiency.

7.4. Multi-party Fair Exchange

In multi-party fair exchange (MFE) protocols, each of the multiple players has an item, and at the end of the protocol, either every player receives every other party's item, or no player receives anything.

Dodis et al. [44] point out the security of public key encryption and public key signature schemes in a single-user setting do not guarantee security multi-user setting. They define the multi-user security model of optimistic fair exchange and make a generic construction. Kılınç et al. [45] construct an efficient multi-party fair exchange protocol that requires only $O(1)$ rounds and $O(n^2)$ transferred messages, where n is the number of players. Payeras-Capellà et al. [5] present a blockchain-based fair certified notification system on Bitcoin, which allows a confidential exchange of notifications among a sender and multiple recipients. Choudhuri et al. [46] formalize a fairness model with dishonest majority using public bulletin board. Their construction can be implemented either with extractable witness encryption or trusted hardware. More recently, Kılınç et al. [26] extend their previous work [45] to implement a MEF protocol tolerating $n - 1$ malicious parties. Further, they adapt their MFE protocol to a fair secure multi-party computation protocol. Meanwhile, Kılınç et al. [47] enable multiple parties to exchange large files and employ electronic payments to penalize dishonest behaviors, compared with previous work [45].

8. CONCLUSION

We adopt decentralized CP-ABE as a cryptographic primitive to build a publicly verifiable optimistic fair exchange protocol. A secure optimistic fair exchange protocol involves two players, who want to exchange items, and TTP(s), who are required in case of dispute (i.e., $\mathcal{RP}$ is invoked). Decentralized TTPs, facilitated by decentralized CP-ABE, solve the single-point failure problem caused by a single TTP. In addition, the TTPs are autonomous, passive and verifiable and their long-term key pairs can be used for different exchanging couples. The public verifiable OFE protocol has two communication rounds in a normal case. Decentralized CP-ABE ciphertext is leveraged to guarantee privacy of exchanging items in Round-1. NIZK proofs, obtained using sigma protocol and Fiat-Shamir transformation, are attached to prove plaintext knowledge of the CP-ABE ciphertext. ElGamal-encrypted CP-ABE decryption keys, which can be used to decrypt ciphertext of Round-1, are exchanged in Round-2. Similarly, NIZK proofs are also used to prove the correctness of the encrypted CP-ABE decryption keys. Lastly, we achieve verification and computation complexities of only $O(n)$ for both players.

ACKNOWLEDGEMENTS

This work was supported by National Key R & D Program of China (No. 2019YFB2101703), the Foundation of Hainan University (No. KYQD22168), the Key Laboratory of Internet Information Retrieval of Hainan Province Research Found (No. 2022KY01), National Natural Science Foundation of China (Grant No. 62272107 and U19A2066), the Innovation Action Plan of Shanghai Science and Technology (Nos. 20511102200 and 21511102200), and the Key R & D Program of Guangdong Province (No. 2020B0101090001). The last author is supported by Royal Society (ICA\R1\180226) and EPSRC (EP/T014784/1).

DATA AVAILABILITY

The data and source codes of this paper are available at Harvard Dataverse: `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/MRA14V`.

REFERENCES

[1] Even, S. and Yacobi, Y. (1980) Relations among public key signature systems (No. CS Technion report CS0175). Computer Science Department, Technion.

[2] Asokan, N., Shoup V. and Waidner, M. (2000) Optimistic fair exchange of digital signatures. *IEEE J. Sel. Areas Commun.*, 18, 593-610.

[3] Micali, S. (2003) Simple and fast optimistic protocols for fair electronic exchange. *In ACM PODC*, New York, USA, July 2003, pp. 12-19.

[4] Ferrer-Gomila, J.L., Hinarejos, M.F., Draper-Gil, G., and Rotger, L.H. (2018) Optimistic protocol for certified electronic mail with verifiable TTP. *Comput. Stand. Interfaces*, 57, 20-30.

[5] Payeras-Capellà, M., Mut-Puigserver, M. and Cabot-Nadal, M. À. (2018) Smart Contract for Multiparty Fair Certified Notifications. *In CANDARW*, Takayama, Japan, 27-30 November, pp. 459-465.

[6] Küpçü, A. (2013) Distributing trusted third parties.*SIGACT News*, 44, 92-112.

[7] Ateniese, G., Medeiros, B.D., and Goodrich, M.T. (2001) TRICERT: A Distributed Certified E-Mail Scheme. *In NDSS*, San Diego, California, USA.

[8] Paulin, A. and Welzer, T. (2013) A universal system for fair non-repudiable certified e-mail without a trusted third party. *Comput. Secur.*, 32, 207-218.

[9] Franklin, M.K. and Reiter, M.K. (1997) Fair exchange with a semi-trusted third party. *In ACM CCS*, Zurich, Switzerland, 1997, pp. 1-5.

[10] Avoine, G. and Vaudenay, S. (2004) Optimistic Fair Exchange Based on Publicly Verifiable Secret Sharing. *In ACISP*, Sydney, Australia, 13-15 July, pp. 3108.

[11] Zhao, Y. and Qin, Z.G. (2012) An optimistic protocol for distributed fair exchange. *In IMIS*, Palermo, Italy, 04-06 July, pp. 395-399

[12] Stadler, M. (1996) Publicly Verifiable Secret Sharing. *In EUROCRYPT*, 12–16 May, pp. 190-199

[13] Baum, C., Damgård, I. and Orlandi, C. (2014) Publicly auditable secure multi-party computation. *In SCN*, Amalfi, Italy, 3-5 September, pp. 175-196.

[14] Reitwiessner, C. (2016) zkSNARKs in a nutshell. Ethereum blog.

[15] Damgård, I. (2010) On Σ-Protocols, [Online], Available: `https://www.cs.au.dk/~ivan/Sigma.pdf`

[16] Fiat, A. and Shamir, A. (1986) How to prove yourself: Practical solutions to identification and signature problems. *In CRYPTO*, pp. 186–194.

[17] Rouselakis, Y. and Waters, B. (2015) Effcient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption. *In FC*, San Juan, Puerto Rico, 26-30 January, pp. 315–332.

[18] Gamal, T.E. (1984). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31, 469-472.

[19] Bethencourt, J., Sahai, A. and Waters, B. (2007) Ciphertext-Policy Attribute-Based Encryption, *In IEEE S&P*, Berkeley, CA, USA, 20-23 May, pp. 321-334.

[20] Schoenmakers, B. (1999) A simple publicly verifiable secret sharing scheme and its application to electronic voting. *In Crypto*, Santa Barbara, California, USA, 15-19 August, pp. 148-164.

[21] Cascudo, I. and David, B. (2017) SCRAPE: Scalable Randomness Attested by Public Entities. *In ACNS*, Kanazawa, Japan, 10-12 July, pp. 537–556

[22] Beimel, A. (1996) Secure Schemes for Secret Sharing and Key Distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel.

[23] Boneh, D. and Shoup, V. (2020) A graduate course in applied cryptography, [Online]. Available: `http://toc.cryptobook.us/book.pdf`, version 0.5.

[24] Schnorr, C.P. (1991) Efficient signature generation by smart cards. *J. Cryptology*, 4, 161–174.

[25] Okamoto, T. (1992) Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. *In Crypto*, Santa Barbara, California, USA, 16–20 August, pp. 31–53.

[26] Alper, H. K. and Küpçü, A. (2021) Optimally efficient multi-party fair exchange and fair secure multi-party computation. *ACM TOPS*, 25(1), 1-34.

[27] Küpçü, A. and Lysyanskaya, A. (2010) Optimistic Fair Exchange with Multiple Arbiters. *In ESORICS*, Athens, Greece, 20-22 September, pp. 488–507.

[28] Berrut, J. and Trefethen, L.N. (2004). Barycentric Lagrange Interpolation. *SIAM Rev.*, 46, 501-517.

[29] Wood, G. (2014) Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper.

[30] Pfitzmann, B., Schunter, M. and Waidner, M. (1998) Optimal Efficiency of Optimistic Contract Signing. *In PODC*, Puerto Vallarta, Mexico, 28 June, pp. 113-122.

[31] Garay, J. A., Jakobsson, M. and MacKenzie, P. (1999) Abuse-Free Optimistic Contract Signing. *In CRYPTO*. Santa Barbara, California, USA, 15-19 August, pp. 449–466.

[32] Lie, Q., Wang, G. and Mu, Y. (2011) Optimistic fair exchange of ring signatures. *In SecureComm*, London, 7-9 September, pp. 227-242.

[33] Wang, Y., Au, M. H., Liu, J. K., Yuen, T. H. and Susilo, W. (2013) Threshold-Oriented Optimistic Fair Exchange. *In NSS*, Madrid, Spain, 3-4 June, pp. 424-438.

[34] Huang, X., Mu, Y., Susilo, W., Wu, W., Zhou, J., & Deng, R.H. (2011) Preserving Transparency and Accountability in Optimistic Fair Exchange of Digital Signatures. *IEEE TIFS*, 6, 498-512.

[35] Huang, Q., Wong, D.S. and Susilo, W. (2015) How to protect privacy in Optimistic Fair Exchange of digital signatures. *Inf. Sci.*, 325, 300-315.

[36] Zhang, L., Zhang, H., Yu, J. and Xian, H. (2020). Blockchain-based two-party fair contract signing scheme. *Inf. Sci.*, 535, 142-155.

[37] Bentov, I. and Kumaresan, R. (2014) How to Use Bitcoin to Design Fair Protocols. *In CRYPTO*, Santa Barbara, CA, USA, 17-21 August, pp. 421–439.

[38] Dziembowski, S., Eckey, L. and Faust, S. (2018) Fairswap: How to fairly exchange digital goods. *In ACM CCS*, Toronto, Canada, 15-19 October, pp. 967–984.

[39] Eckey, L., Faust, S. and Schlosser B. (2020) Optiswap: Fast optimistic fair exchange. *In ACM CCS*, Taipei, Taiwan, 5-9 October, pp. 543–557.

[40] S. Janin, K. Qin, A. Mamageishvili and A. Gervais. (2020) FileBounty: Fair Data Exchange. *In EuroS&PW*, Genoa, Italy, 7-11 September, pp. 357-366.

[41] Delgado-Segura, S., Pérez-Solà, C., Navarro-Arribas, G. and Herrera-Joancomartí, J. (2020) A fair protocol for data trading based on Bitcoin transactions.*FGCS*, 107, 832-840.

[42] Delgado-Segura, S., Pérez-Solà, C., Herrera-Joancomartí, J. and Navarro-Arribas, G. (2018) Bitcoin private key locked transactions. *Information Processing Letters*, 140, 37-41.

[43] Wagner, E., Völker, A., Fuhrmann, F., Matzutt, R. and Wehrle, K. (2019) Dispute Resolution for Smart Contract-based Two-Party Protocols. *In IEEE ICBC*, Seoul, Korea, 14-17 May, pp. 422-430.

[44] Dodis, Y., Lee, P. J. and Yum, D. H. (2007) Optimistic Fair Exchange in a Multi-user Setting. *In PKC*, Beijing, China, 16-20 April, pp. 118–133.

[45] Kilinç, H. and Küpçü, A. (2015) Optimally Efficient Multi-Party Fair Exchange and Fair Secure Multi-Party Computation. *ACM Trans. Priv. Secur.*, 25, 1-34.

[46] Choudhuri, A. R., Green, M., Jain, A., Kaptchuk, G. and Miers, I. (2017) Fairness in an unfair world: Fair multiparty computation from public bulletin boards. *In ACM CCS*, Dallas Texas USA, 30 October, pp. 719–728.

[47] H. Kılınç Alper and A. Küpçü. (2021) Coin-based multi-party fair exchange. *In ACNS*, Kamakura, Japan, 21–24 June, pp. 130-160.