Optimistic Fair Exchange in a Multi-user Setting

Yevgeniy Dodis
New York University
dodis@cs.nyu.edu

Pil Joong Lee
POSTECH
pjl@postech.ac.kr

Dae Hyun Yum
POSTECH
dhyum@postech.ac.kr

May 10, 2007

Abstract

This paper addresses the security of *optimistic fair exchange* in a *multi-user* setting. While the security of public key encryption and public key signature schemes in a single-user setting guarantees the security in a multi-user setting, we show that the situation is different in the optimistic fair exchange. First, we show how to break, in the multi-user setting, an optimistic fair exchange scheme provably secure in the single-user setting. This example separates the security of optimistic fair exchange between the single-user setting and the multi-user setting. We then define the formal security model of optimistic fair exchange in the multi-user setting, which is the first complete security model of optimistic fair exchange in the multi-user setting. We prove the existence of a generic construction meeting our multi-user security based on one-way functions in the random oracle model and trapdoor one-way permutations in the standard model. Finally, we revisit two well-known methodologies of optimistic fair exchange, which are based on the verifiably encrypted signature and the sequential two-party multisignature, respectively. Our result shows that these paradigms remain valid in the multi-user setting.

1 Introduction

Multi-user Security. In the early stage of modern cryptography, public key cryptography was usually studied in the single-user setting and the security model assumed only one public key [23, 24]; one receiver in the public key encryption and one signer in the public key signature. However, there are many users in the real world and the security in the single-user setting does not guard against the attacks by colluding dishonest users.

Even though the threats under multiple public keys were already pointed out in the 1980's (e.g., see [38, 26]), the security in the multi-user setting was formally studied only recently [4, 21]. Fortunately, these researches show that the security of encryption schemes in the single-user setting is preserved in the multi-user setting [4] and the same result holds good for signature schemes[1] [21]. Therefore, we only have to deal with the single-user security and need not consider the multi-user security in the public key encryption and signature schemes.[2]

[1]A slightly different result is presented in [27], where authors argue that the existential unforgeability against chosen message attacks in the single-user setting is not enough for the multi-user setting. However, their duplicate-signature key selection attack is not a flaw from the view of standard security notions and can be thwarted with ease.

[2]Sometimes, separate security analysis in the multi-user setting shows tighter security reduction [4], which is irrelevant to the subject of our paper.

While the security of public key encryption and public key signature schemes in the single-user setting guarantees the security in the multi-user setting, there are other cryptosystems where the single-user security is not enough. For example, identity-based encryption schemes [37, 10], by nature, must be analyzed in the multi-user setting and the security proof in the single-user setting is almost meaningless.

Optimistic Fair Exchange. A fair exchange scheme is a protocol by which two parties Alice and Bob swap items or services without allowing either party to gain an advantage by quitting prematurely or otherwise misbehaving. For instance, Alice signs some statement (e.g., e-cash) and Bob fulfills some obligation (e.g., delivery of goods). However, each party will play the role only if he (or she) is sure that the other party will keep the appointment. Of course, one could use an online trusted third party in every transaction to act as a mediator; each party sends the item to the trusted third party, who upon verifying the correctness of both items, forwards each item to the other party. A drawback of this approach is that the trusted third party is always involved in the exchange even if both parties are honest and no fault was occurred. In practice, sending messages via a trusted third party can lead to performance problems as it becomes a bottleneck.

A more desirable approach is that a semi-trusted arbitrator involves only in cases where one party attempts to cheat or simply crashes. We call such a fair exchange protocol *optimistic*. In this model, Alice first issues a verifiable "partial signature" σ' to Bob. Bob verifies the validity of the partial signature and fulfills his obligation, after which Alice sends her "full signature" σ to complete the transaction. Thus, if no problem occurs, the arbitrator does not participate in the protocol. However, if Alice refuses to send her full signature σ at the end, Bob will send σ' (and proof of fulfilling his obligation) to the arbitrator who will convert σ' into σ, sending σ to Bob.

Optimistic fair exchange was introduced by Asokan et al. [1] and formally studied in [2, 3] where several solutions were presented based on *verifiably encrypted signatures*. The approach of [2, 3] was later generalized by [12], but all these schemes involve expensive and highly interactive zero-knowledge proofs in the exchange phase. The first *non-interactive* verifiably encrypted signature was built by Boneh et al. [11] under a form of the computational Diffie-Hellman assumption over special elliptic curve groups.

A different approach for building non-interactive optimistic fair exchange based on *sequential two-party multisignatures* was proposed by Park et al. [29], which was broken and repaired by Dodis and Reyzin [17]. While the schemes in [17] are very efficient, one important drawback of the approach based on the sequential two-party multisignature is that it is *setup-driven* [39]; the registration is required between the user and the arbitrator.

Our Contribution. There have been attempts to formally define the security of optimistic fair exchange. The first formal security model was proposed by Asokan et al. [2, 3] but was not complete as their model did not consider a dishonest third party.[3] A more generalized and unified model for non-interactive optimistic fair exchange was suggested by Dodis and Reyzin [17]. Their model, called *verifiably committed signatures*, incorporates all aspects of non-interactive optimistic fair exchange but was defined in a single-user setting. If the security of optimistic fair exchange in the single-user setting guarantees the multi-user security, the model of [17] is satisfactory. Otherwise, we should extend the

[3]In the construction based on verifiably encrypted signatures, each user has a signing key and the third party has a decryption key [2, 3]. Therefore, the dishonest third party, who does not know the signing keys, cannot compromise the signature schemes of users. However, we can devise other constructions which are secure in the model of [2, 3] but they can be broken by a dishonest third party. For example, think of the optimistic fair exchange scheme where the third party simply holds the private keys of all users.

model to the multi-user setting.

In this paper, we show that the single-user security of optimistic fair exchange does not guarantee multi-user security. We present a simple counterexample based on a signature scheme and a trapdoor permutation. We then define the multi-user security model of optimistic fair exchange, extending the model of [17]. While the single-user model of [17] is setup-driven, our multi-user model is *setup-free* [39], which we feel is a more natural and advantageous realization of "optimistic" fair exchange in the multi-user setting; (1) If every fair exchange is performed normally (i.e., every user behaves honestly), it is desirable that users need not contact the arbitrator even for the registration purpose. (2) The arbitrator in setup-driven schemes should be semi-online to respond to registration requests, even when no dispute between users occurs. (3) If there are several arbitrators, the user in setup-free schemes can decide on a particular arbitrator in run-time.

After defining security notions, we address our attention to the basic theoretical question, namely whether or not a scheme satisfying the security notions exists, and, if so, what are the minimal computational complexity assumptions under which this existence can be proven. We answer this by providing a generic setup-free construction which relies on one-way functions in the random oracle model and trapdoor one-way permutations in the standard model. While the construction in the standard model is of theoretic interest, some specific instantiations in the random oracle model are efficient enough for practical use. Finally, we revisit two well-known techniques of optimistic fair exchange; the verifiably encrypted signature and the sequential two-party signature. Fortunately, our result shows that these paradigms remain valid in the multi-user setting if the underlying primitives satisfy some security properties. Furthermore, the construction based on the verifiably encrypted signature shows that trapdoor permutations imply optimistic fair exchange schemes that are *stand-alone* as well as setup-free; a fair exchange scheme is stand-alone if the full signature is the same as it were produced by an ordinary signature scheme only [39].

2 Preliminaries

2.1 Notation

If $k \in \mathbb{N}$, then 1^k denotes the string of k ones. If x is a string, then $|x|$ denotes its length, while if X is a finite set then $|X|$ denotes its size. If x and y are strings, then $x \| y$ denotes the concatenation of x and y. A function $f(n)$ is *negligible* if for all polynomials $p(n)$, $f(n) < 1/p(n)$ hold for all sufficiently large n. An *efficient algorithm* $A(\cdot)$ is a probabilistic polynomial-time (PPT) Turing machine. If $A(\cdot)$ is an efficient algorithm and x is an input for A, then $A(x)$ denotes the probability space that assigns to a string s the probability that A, on input x, outputs s. For a probability space P, $x \leftarrow P$ denotes the algorithm that samples a random element according to P. For a finite set X, $x \leftarrow X$ denotes the algorithm that samples an element uniformly at random from X. If $p(\cdot, \cdot, \cdots)$ is a boolean function, then $\Pr[p(x_1, x_2, \ldots) \mid x_1 \leftarrow P_1, x_2 \leftarrow P_2, \ldots]$ denotes the probability that $p(x_1, x_2, \ldots)$ is true after executing the algorithms $x_1 \leftarrow P_1$, $x_2 \leftarrow P_2, \ldots$.

2.2 NP-Relations and Σ-Protocols

An **NP**-relation R is a subset of $\{0,1\}^* \times \{0,1\}^*$ for which there is an efficient algorithm to decide whether $(\alpha, \beta) \in R$ or not in time polynomial in $|\alpha|$. The **NP**-language $\mathcal{L}_R$ associated with R is the set of α for which there exists β such that $(\alpha, \beta) \in R$, i.e., $\mathcal{L}_R = \{\alpha \mid \exists \beta \; [(\alpha, \beta) \in R]\}$.

A Σ-protocol [16] for an **NP**-relation R is an efficient 3-move two-party protocol between the prover and the verifier on a common input $\alpha \in \mathcal{L}_R$. Besides α, a valid **NP**-witness β for α, meaning $(\alpha, \beta) \in R$, is also given to the prover as a private input. The prover first sends a commitment message c to the receiver. After receiving the commitment message c, the verifier sends a challenge message e to the prover. Finally, the prover sends a response message s to the verifier who decides to output 1 (accept) or 0 (reject) based on the input α and the transcript $\pi = \{c, e, s\}$. The transcript π is valid if the verifier outputs 1 (accept).

A Σ-protocol should satisfy three properties: correctness, special soundness, and special (honest-verifier) zero-knowledge. Correctness property states that for all $\alpha \in \mathcal{L}_R$ and all valid witnesses β for α, if the prover and the verifier follow the protocol honestly, the verifier must output 1 (accept). Special soundness property states that there is an efficient extraction algorithm (called a knowledge extractor) that on input $\alpha \in \mathcal{L}_R$ and two valid transcripts π_1, π_2 with the same commitment message c outputs β such that $(\alpha, \beta) \in R$. Special zero-knowledge property states that there is an efficient simulation algorithm (called a simulator) that on input $\alpha \in \mathcal{L}_R$ and any challenge message e, outputs a valid transcript $\pi' = \{c', e, s'\}$. Moreover, the distribution of (c', s') is computationally indistinguishable from the corresponding distribution on (c, s) produced by the prover knowing a valid witness β for α and the verifier.

A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function, if there exists a polynomial time algorithm which computes $f(x)$ correctly for all x and the following probability is negligible for all PPT algorithm A: $\Pr[f(x') = y \mid x \leftarrow \{0,1\}^k; y = f(x); x' \leftarrow A(y, 1^k)]$. A one-way function f is called a trapdoor (one-way) permutation, if f is a permutation (that is, every $f(x)$ has a unique pre-image x) and there exists a polynomial-length trapdoor td such that the inverse of f can efficiently be computed with td. For simplicity, we let f^{-1} be an inverse algorithm of f with the trapdoor td. It is known that any language in **NP** has a Σ-protocol if one-way functions exist.

Theorem 1 ([18, 22]) *A Σ-protocol for any* **NP***-relation can be constructed if one-way functions exist.*

While the Σ-protocol for any **NP**-relation can be constructed in generic ways [18, 22], there exist very efficient Σ-protocols for specific cases; for example, GQ protocol [25] and Schnorr protocol [36].

A Σ-protocol can be transformed into a signature scheme by using the Fiat-Shamir heuristic [20]. To sign a message m, the legal signer produces a valid transcript $\pi = \{c, e, s\}$ of the Σ-protocol, where $e = H(c, m)$ and $H(\cdot)$ is a cryptographic hash function modeled as a random function. The signature scheme obtained by applying the Fiat-Shamir heuristic to the Σ-protocol is secure in the random oracle model [6, 31]. It is also known that the Fiat-Shamir heuristic provides a non-interactive proof of knowledge in the random oracle model (i.e., the witness can be extracted by rewinding the adversary).

If there are two Σ-protocols, i.e., Σ_1 for R_1 and Σ_2 for R_2, we can construct another Σ-protocol Σ_{OR} (called OR-proof) [16] which allows the prover to show that given two inputs x_1, x_2, he knows w such that either $(x_1, w) \in R_1$ or $(x_2, w) \in R_2$ without revealing which is the case (called the witness indistinguishability property [19]). By applying the Fiat-Shamir heuristic to the OR-proof Σ_{OR}, we obtain a signature scheme $\mathcal{S}_{OR}$ (called the OR-signature) secure in the random oracle model such that a valid signature can be generated by the signer who knows a valid witness w corresponding to either of the two inputs x_1, x_2. It is known that the Fiat-Shamir heuristic does not affect the witness indistinguishability property of the Σ-protocol.

2.3 Signatures

Syntax. A signature scheme $\mathcal{S}$ consists of three efficient algorithms: $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. The key generation algorithm Sig-Gen takes as input a security parameter 1^k and outputs a signing key sk and a verification key vk. The signing algorithm Sign takes as input a signing key sk and a message m from the associated message space $\mathcal{M}$, and outputs a signature σ. The verification algorithm Vrfy takes as input a verification key vk, a message m, and a signature σ; it outputs 1 if the signature is valid and 0 otherwise. We require that $\mathsf{Vrfy}_{vk}(m, \mathsf{Sign}_{sk}(m)) = 1$, for any $m \in \mathcal{M}$.

Security. We consider *existential unforgeability under adaptive chosen message attacks*, denoted by UF-CMA [24]. The adversary $\mathcal{A}$ is given oracle access to the signing oracle O_{Sign}, i.e., $\mathcal{A}$ is allowed to query the signing oracle O_{Sign} to obtain valid signatures $\sigma_1, \ldots, \sigma_n$ of arbitrary message $m_1, \ldots, m_n$ adaptively chosen by $\mathcal{A}$. Naturally, $\mathcal{A}$ is considered successful only if it forges a valid signature σ of a message m which has not been queried to O_{Sign}: $m \notin \{m_1, \ldots, m_n\}$. Quantitatively, we define

$$\mathsf{Adv}^{\mathcal{S}}_{\mathcal{A}}(k) = \Pr[\mathsf{Vrfy}_{vk}(m, \sigma) = 1 \mid (sk, vk) \leftarrow \mathsf{Sig\text{-}Gen}(1^k),\ (m, \sigma) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}}(vk)]$$

where m should not be queried to the signing oracle O_{Sign}.

Let $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme. An adversary $\mathcal{A}$ is said to (t, q_s, ε)-break $\mathcal{S}$, if $\mathcal{A}$ runs in time at most t, makes at most q_s signing queries to O_{Sign}, and succeeds in forgery with probability at least ε. $\mathcal{S}$ is said to be (t, q_s, ε)-secure, if no adversary can (t, q_s, ε)-break it. Asymptotically, $\mathcal{S}$ is UF-CMA-secure if $\mathsf{Adv}^{\mathcal{S}}_{\mathcal{A}}(k)$ is negligible for any PPT adversary $\mathcal{A}$.

2.4 Encryption

Syntax. An encryption scheme $\mathcal{E}$ consists of three algorithms: $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm Enc-Gen takes a security parameter 1^k as input and outputs an encryption key ek and a decryption key dk. The encryption algorithm Enc takes as input an encryption key ek and a message m from the associated message space $\mathcal{M}$, and outputs a ciphertext c. The decryption algorithm Dec takes a decryption key dk and a ciphertext c as input; it outputs some message $m \in \mathcal{M}$ if the ciphertext is valid and $\perp$ otherwise. We require that $\mathsf{Dec}_{dk}(\mathsf{Enc}_{ek}(m)) = m$, for any $m \in \mathcal{M}$.

Security. We consider *indistinguishability against adaptive chosen ciphertext attacks*, denoted by IND-CCA [32, 5]. Intuitively, no efficient adversary $\mathcal{A}$ can distinguish encryptions of any two equal-length messages m_0, m_1 for a randomly selected public key, even though $\mathcal{A}$ is given oracle access to the decryption oracle O_{Dec}. For an efficient algorithm $\mathcal{A}$, which runs in two stages of find and guess, we define the adversary's advantage as

$$\mathsf{CCA\text{-}Adv}^{\mathcal{E}}_{\mathcal{A}}(k) = \left| \Pr\left[b = \widetilde{b} \;\middle|\; \begin{array}{c} (ek, dk) \leftarrow \mathsf{Enc\text{-}Gen}(1^k),\ (m_0, m_1, \alpha) \leftarrow \mathcal{A}^{O_{\mathsf{Dec}}}(ek, \mathsf{find}), \\ b \leftarrow \{0,1\},\ c_b \leftarrow \mathsf{Enc}_{ek}(m_b),\ \widetilde{b} \leftarrow \mathcal{A}^{O_{\mathsf{Dec}}}(c_b, \alpha, \mathsf{guess}) \end{array} \right] - \frac{1}{2} \right|$$

where the challenge ciphertext c_b should not be queried to the decryption oracle in the guess stage and α is some internal state information that $\mathcal{A}$ saves and uses in the two stages.

Let $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. An adversary $\mathcal{A}$ is said to (t, q_d, ε)-break $\mathcal{E}$, if $\mathcal{A}$ runs in time at most t, makes at most q_d decryption queries to O_{Dec}, and succeeds in distinguishing the challenge ciphertext with advantage at least ε. The encryption scheme $\mathcal{E}$ is said to be (t, q_d, ε)-secure, if no adversary can (t, q_d, ε)-break it. Asymptotically, $\mathcal{E}$ is CCA-secure if $\mathsf{CCA\text{-}Adv}^{\mathcal{E}}_{\mathcal{A}}(k)$ is negligible for any efficient adversary $\mathcal{A}$.

3 Optimistic Fair Exchange in a Single-user Setting

3.1 Definition

We review the single-user model of optimistic fair exchange [17].

Definition 1 *A* non-interactive optimistic fair exchange *involves the signer Alice, the verifier Bob and the arbitrator Charlie, and is given by the following efficient algorithms:*

- Setup*. This is a registration protocol between Alice and Charlie, by the end of which Alice learns her secret signing key* SK*, Charlie learns his secret arbitration key* ASK*, and they publish Alice's public verification key* PK *and Charlie's partial verification key* APK*.*
- Sig *and* Ver*. These are similar to conventional signing and verification algorithms of an ordinary digital signature scheme.* $\mathsf{Sig}(m, \mathsf{SK}, \mathsf{APK})$ *— run by Alice — outputs a signature* σ *on* m*, while* $\mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK})$ *— run by Bob (or any verifier) — outputs* 1 *(accept) or* 0 *(reject).*
- PSig *and* PVer*. These are partial signing and verification algorithms.* PSig *together with* Res *is functionally equivalent to* Sig*.* $\mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK})$ *— run by Alice — outputs a partial signature* σ'*, while* $\mathsf{PVer}(m, \sigma', \mathsf{PK}, \mathsf{APK})$ *— run by Bob (or any verifier) — outputs* 1 *(accept) or* 0 *(reject).*
- Res*. This is a resolution algorithm run by Charlie in case Alice refuses to open her signature* σ *to Bob, who in turn possesses a valid partial signature* σ' *on* m *(and a proof that he fulfilled his obligation to Alice). In this case,* $\mathsf{Res}(m, \sigma', \mathsf{ASK}, \mathsf{PK})$ *should output a legal signature* σ *on* m*.*

Correctness property states that

- $\mathsf{Ver}(m, \mathsf{Sig}(m, \mathsf{SK}, \mathsf{APK}), \mathsf{PK}, \mathsf{APK}) = 1$, $\mathsf{PVer}(m, \mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK}), \mathsf{PK}, \mathsf{APK}) = 1$, *and* $\mathsf{Ver}(m, \mathsf{Res}(m, \mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK}), \mathsf{ASK}, \mathsf{PK}), \mathsf{PK}, \mathsf{APK}) = 1$.

Ambiguity property states that

- *Any "resolved signature"* $\mathsf{Res}(m, \mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK}), \mathsf{ASK}, \mathsf{PK})$ *is computationally indistinguishable from the "actual signature"* $\mathsf{Sig}(m, \mathsf{SK}, \mathsf{APK})$*.*

In a meaningful application, Charlie runs Res to produce a full signature σ from σ' only if Bob's obligation to Alice has been fulfilled. The definition does not deal with the application-specific question of how Bob proves to Charlie that he fulfilled his obligation to Alice. The definition assumes the authenticity of public keys.

The security of non-interactive optimistic fair exchange consists of ensuring three aspects: security against the signer, security against the verifier, and security against the arbitrator. In the following, we denote by O_{PSig} an oracle simulating the partial signing procedure PSig, and by O_{Res} an oracle simulating the resolution procedure Res.

Security against Alice. We require that any PPT adversary A succeeds with at most negligible probability in the following experiment.

$$\begin{array}{rcl} \mathsf{Setup}^*(1^k) & \to & (\mathsf{SK}^*, \mathsf{PK}, \mathsf{ASK}, \mathsf{APK}) \\ (m, \sigma') & \leftarrow & A^{O_{\mathsf{Res}}}(\mathsf{SK}^*, \mathsf{PK}, \mathsf{APK}) \\ \sigma & \leftarrow & \mathsf{Res}(m, \sigma', \mathsf{ASK}, \mathsf{PK}) \\ \text{success of } A & = & [\mathsf{PVer}(m, \sigma', \mathsf{PK}, \mathsf{APK}) \overset{?}{=} 1 \ \wedge \ \mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK}) \overset{?}{=} 0] \end{array}$$

where Setup^* denotes the run of Setup with dishonest Alice (run by A) and SK^* is A's state after this run. In other words, Alice should not be able to produce partial signature σ', which looks good to Bob but cannot be transformed into her full signature by honest Charlie.

Security against Bob. We require that any PPT adversary B succeeds with at most negligible probability in the following experiment.

$$\begin{aligned} \mathsf{Setup}(1^k) &\rightarrow (\mathsf{SK}, \mathsf{PK}, \mathsf{ASK}, \mathsf{APK}) \\ (m, \sigma) &\leftarrow B^{O_{\mathsf{PSig}}, O_{\mathsf{Res}}}(\mathsf{PK}, \mathsf{APK}) \\ \text{success of } B &= [\mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK}) \overset{?}{=} 1 \ \wedge \ (m, \cdot) \notin \mathit{Query}(B, O_{\mathsf{Res}})] \end{aligned}$$

where $\mathit{Query}(B, O_{\mathsf{Res}})$ is the set of valid queries of B has asked to the resolution oracle O_{Res} (i.e., (m, σ') such that $\mathsf{PVer}(m, \sigma', \mathsf{PK}, \mathsf{APK}) = 1$). In other words, Bob should not be able to complete any partial signature σ' that he received from Alice into a complete signature σ, without explicitly asking Charlie to do so.

Note that there is no need to provide B with access to the signing oracle O_{Sig}, since it could be simulated by O_{PSig} and O_{Res}. Finally, we remark that we also want Bob to be unable to generate a valid partial signature σ' which was not produced by Alice (via a query to O_{PSig}). However, this guarantee will follow from a stronger security against Charlie, which is defined below. Indeed, we will ensure that even Charlie, who knows more than Bob (i.e., ASK), cannot succeed in this attack.

Security against Charlie. We require that any PPT adversary C succeeds with at most negligible probability in the following experiment.

$$\begin{aligned} \mathsf{Setup}^*(1^k) &\rightarrow (\mathsf{SK}, \mathsf{PK}, \mathsf{ASK}^*, \mathsf{APK}) \\ (m, \sigma) &\leftarrow C^{O_{\mathsf{PSig}}}(\mathsf{ASK}^*, \mathsf{PK}, \mathsf{APK}) \\ \text{success of } C &= [\mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK}) \overset{?}{=} 1 \ \wedge \ m \notin \mathit{Query}(C, O_{\mathsf{PSig}})] \end{aligned}$$

where Setup^* denotes the run of Setup with dishonest Charlie (run by C), ASK^* is C's state after this run, and $\mathit{Query}(C, O_{\mathsf{PSig}})$ is the set of queries of C asked to the partial signing oracle O_{PSig}. In other words, Charlie should not be able to produce a valid signature on m without explicitly asking Alice to produce a partial signature on m (which Charlie can complete into a full signature by himself using ASK).

3.2 Single-user Security $\not\Rightarrow$ Multi-user Security

We show that the single-user security of optimistic fair exchange does not imply the multi-user security by presenting a counter-example.

Scheme. Let $f(\cdot)$ be a trapdoor permutation and $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme.

- Setup. Charlie generates a trapdoor permutation (f, f^{-1}) and publishes $\mathsf{APK} = f$, while he keeps $\mathsf{ASK} = f^{-1}$ secret. Alice generates $(sk, vk) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$ and publishes $\mathsf{PK}_A = vk$ and keeps $\mathsf{SK}_A = sk$ secret.
- Sig and Ver. To sign a message m, Alice chooses a random number r_A, and computes $y_A = f(r_A)$ and $\delta_A = \mathsf{Sign}_{sk}(m \| y_A)$. The signature of m is $\sigma_A = (r_A, \delta_A)$. To verify Alice's signature $\sigma_A = (r_A, \delta_A)$ of m, Bob computes $y_A = f(r_A)$ and checks $\mathsf{Vrfy}_{vk}(m \| y_A, \delta_A) \overset{?}{=} 1$.

- PSig and PVer. To generate a partial signature, Alice chooses a random number r_A and computes $y_A = f(r_A)$ and $\delta_A = \mathsf{Sign}_{sk}(m\|y_A)$. The partial signature of m is $\sigma'_A = (y_A, \delta_A)$. Bob verifies $\sigma'_A = (y_A, \delta_A)$ by checking $\mathsf{Vrfy}_{vk}(m\|y_A, \delta_A) \overset{?}{=} 1$.
- Res. Given a partial signature (m, y_A, δ_A), the arbitrator Charlie first verifies its validity by checking $\mathsf{Vrfy}_{vk}(m\|y_A, \delta_A) \overset{?}{=} 1$. If valid, he computes $r_A = f^{-1}(y_A)$ and returns $\sigma_A = (r_A, \delta_A)$.

The Single-user Security. The above scheme is secure in the single-user setting, which can easily be shown following the proofs in [17]. For details, see appendix A.

Attack Scenario. We observe that y_A can be re-used by a dishonest user without knowing the corresponding r_A, which causes the scheme to be insecure in the multi-user setting. Dishonest users Bob and Eve attack Alice as follows:

1. Alice gives a partial signature (m_A, y_A, δ_A) to Bob, where $y_A = f(r_A)$ and $\delta_A = \mathsf{Sign}_{\mathsf{SK}_A}(m_A\|y_A)$.
2. Bob gives (m_B, y_B, δ_B) to his dishonest friend Eve, where $m_B \neq m_A$, $y_B = y_A$ and $\delta_B = \mathsf{Sign}_{\mathsf{SK}_B}(m_B\|y_B)$.
3. Eve comes to the arbitrator with (m_B, y_B, δ_B) and claims that Bob refuses to open his signature (and maybe gives a proof to the arbitrator that Eve fulfilled her obligation to Bob).
4. The arbitrator does not suspect anything and completes this signature by giving $r_A = f^{-1}(y_B)$ to Eve.
5. Eve gives r_A to Bob, who now has completed the signature of Alice, (m_A, r_A, δ_A), although Alice never intended to open this and Bob did not fulfill his duty to Alice.

Therefore, the above optimistic fair exchange scheme is secure in the single-user setting but insecure in the multi-user setting.[4] This counterexample entails the following theorem.

Theorem 2 *The single-use security of optimistic fair exchange does not imply the multi-user security.*

Remark 1 (Disclaimer) *Theorem 2 does not claim that all previous schemes (e.g., [2, 3, 11, 17]) are insecure in the multi-user setting. Even though the previous schemes were proved secure in the single-user models (or in the incomplete models), we believe that they are also secure in the multi-user model of Section 4.*

4 Optimistic Fair Exchange in a Multi-user Setting

4.1 Definition

Instead of defining the syntax and security from scratch, we extend the model of [17] to the multi-user setting. Firstly, we separate the Setup algorithm of the single-user setting into two algorithms $\mathsf{Setup}^{\mathsf{TTP}}$ and $\mathsf{Setup}^{\mathsf{User}}$ to model the setup-free optimistic fair exchange. By running $\mathsf{Setup}^{\mathsf{User}}$, each user U_i generates his own key pair $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i})$.

[4] A naive countermeasure such as including the signer's public key in the message m does not defeat the collusion attacks.

Definition 2 *A* non-interactive optimistic fair exchange *involves the users (signers and verifiers) and the arbitrator, and is given by the following efficient algorithms:*

- $\mathsf{Setup}^{\mathsf{TTP}}$*. The arbitrator setup algorithm takes as input a security parameter and returns a secret arbitration key* ASK *and a public partial verification key* APK*.*
- $\mathsf{Setup}^{\mathsf{User}}$*. The user setup algorithm takes as input a security parameter and (optionally)* APK*. It returns a private signing key* SK *and a public verification key* PK*.*
- Sig *and* Ver*. These are similar to conventional signing and verification algorithms of an ordinary digital signature scheme.* $\mathsf{Sig}(m, \mathsf{SK}_{U_i}, \mathsf{APK})$ *— run by a signer* U_i *— outputs a signature* σ_{U_i} *on* m*, while* $\mathsf{Ver}(m, \sigma_{U_i}, \mathsf{PK}_{U_i}, \mathsf{APK})$ *— run by a verifier — outputs* 1 *(accept) or* 0 *(reject).*
- PSig *and* PVer*. These are partial signing and verification algorithms.* PSig *together with* Res *is functionally equivalent to* Sig*.* $\mathsf{PSig}(m, \mathsf{SK}_{U_i}, \mathsf{APK})$ *— run by a signer* U_i *— outputs a partial signature* σ'_{U_i}*, while* $\mathsf{PVer}(m, \sigma'_{U_i}, \mathsf{PK}_{U_i}, \mathsf{APK})$ *— run by a verifier — outputs* 1 *(accept) or* 0 *(reject).*
- Res*. This is a resolution algorithm run by the arbitrator in case a signer* U_i *refuses to open his signature* σ_{U_i} *to a user* U_j*, who in turn possesses a valid partial signature* σ'_{U_i} *on* m *(and a proof that* U_j *fulfilled his obligation to* U_i*). In this case,* $\mathsf{Res}(m, \sigma'_{U_i}, \mathsf{ASK}, \mathsf{PK}_{U_i})$ *should output a legal signature* σ_{U_i} *on* m*.*

Correctness property states that

- $\mathsf{Ver}(m, \mathsf{Sig}(m, \mathsf{SK}_{U_i}, \mathsf{APK}), \mathsf{PK}_{U_i}, \mathsf{APK}) = 1$*,* $\mathsf{PVer}(m, \mathsf{PSig}(m, \mathsf{SK}_{U_i}, \mathsf{APK}), \mathsf{PK}_{U_i}, \mathsf{APK}) = 1$*, and* $\mathsf{Ver}(m, \mathsf{Res}(m, \mathsf{PSig}(m, \mathsf{SK}_{U_i}, \mathsf{APK}), \mathsf{ASK}, \mathsf{PK}_{U_i}), \mathsf{PK}_{U_i}, \mathsf{APK}) = 1$*.*

Ambiguity property states that

- *Any "resolved signature"* $\mathsf{Res}(m, \mathsf{PSig}(m, \mathsf{SK}_{U_i}, \mathsf{APK}), \mathsf{ASK}, \mathsf{PK}_{U_i})$ *is computationally indistinguishable from the "actual signature"* $\mathsf{Sig}(m, \mathsf{SK}_{U_i}, \mathsf{APK})$*.*

We do not deal with the subtle issue of timely termination addressed by [2, 3]. We remark, however, that the technique of [2, 3] can easily be added to our solutions to resolve this problem. The security of non-interactive optimistic fair exchange is composed of ensuring three aspects: security against signers, security against verifiers, and security against the arbitrator. To clarify the identity of the signer, we hereinafter assume that the message m (implicitly) includes the identity of the signer. One simple and trivial solution is to include the signer's identity inside the message. If the included signer's identity does not correspond to the subject of the alleged signer's public key, we consider the signature (or the partial signature) is invalid. We also remark that it is a good practice to include an enforcing resolution policy κ inside the message, as suggested in [3].

In order to consider the collusion attack of dishonest users, we modify the resolution oracle O_{Res}. In the single-user setting, the input to O_{Res} is (m, σ'), assuming that σ' is the partial signature value of the single signer Alice and the oracle checks the validity of σ' by using Alice's public key. In the multi-user setting, we define the input to O_{Res} as $(m, \sigma', \mathsf{PK}_{U_i})$ where PK_{U_i} is the public key of the alleged signer U_i. As usual, we assume that the authenticity of public keys can be verified and each user should show his knowledge of the legitimate private key in the public key registration stage to defend against key substitution attacks.

For simplicity but without loss of generality, when we model either the dishonest verifier or the dishonest arbitrator, we suppose that the adversary attacks an honest user Alice and the adversary can collude with all other (dishonest) users.[5] Therefore, the dishonest verifier or the dishonest arbitrator has access to private keys of all users except Alice, and the partial signing oracle O_{PSig}, taking as input a message m, always returns Alice's partial signature σ'_A on m.

Security against Signers. We require that any PPT adversary A, who models the dishonest signer Alice, succeeds with at most negligible probability in the following experiment.

$$\begin{array}{rcl}\mathsf{Setup}^{\mathsf{TTP}}(1^k) & \to & (\mathsf{ASK}, \mathsf{APK}) \\ (m, \sigma', \mathsf{PK}_A) & \leftarrow & A^{O_{\mathsf{Res}}}(\mathsf{APK}) \\ \sigma & \leftarrow & \mathsf{Res}(m, \sigma', \mathsf{ASK}, \mathsf{PK}_A) \\ \text{success of } A & = & [\mathsf{PVer}(m, \sigma', \mathsf{PK}_A, \mathsf{APK}) \stackrel{?}{=} 1 \ \wedge\ \mathsf{Ver}(m, \sigma, \mathsf{PK}_A, \mathsf{APK}) \stackrel{?}{=} 0]\end{array}$$

In the single-user setting, the signer Alice wins if she comes up with a partial signature (m, σ') which is valid with respect to her public key but cannot be transformed into her full signature by the honest arbitrator. In the multi-user setting, Alice wins if she comes up with $(m, \sigma', \mathsf{PK}_A)$ where σ' is a valid partial signature with respect to PK_A but cannot be completed to the full signature (w.r.t. PK_A) by the honest arbitrator.

Note that there is no need to provide A with access to any kind of the partial signing oracle, since she has access to private keys of all users and can simulate all partial signing oracles by herself.

Security against Verifiers. We require that any PPT adversary B succeeds with at most negligible probability in the following experiment.

$$\begin{array}{rcl}\mathsf{Setup}^{\mathsf{TTP}}(1^k) & \to & (\mathsf{ASK}, \mathsf{APK}) \\ \mathsf{Setup}^{\mathsf{User}}(1^k) & \to & (\mathsf{SK}_A, \mathsf{PK}_A) \\ (m, \sigma) & \leftarrow & B^{O_{\mathsf{PSig}}, O_{\mathsf{Res}}}(\mathsf{PK}_A, \mathsf{APK}) \\ \text{success of } B & = & [\mathsf{Ver}(m, \sigma, \mathsf{PK}_A, \mathsf{APK}) \stackrel{?}{=} 1 \ \wedge\ (m, \cdot, \mathsf{PK}_A) \notin \mathit{Query}(B, O_{\mathsf{Res}})]\end{array}$$

where $\mathit{Query}(B, O_{\mathsf{Res}})$ is the set of valid queries of B has asked to the resolution oracle O_{Res} (i.e., $(m, \sigma', \mathsf{PK}_{U_i})$ such that $\mathsf{PVer}(m, \sigma', \mathsf{PK}_{U_i}, \mathsf{APK}) = 1$). Even though the adversary B is not allowed to ask a valid query $(m, \cdot, \mathsf{PK}_A)$ with the target message m, it can freely ask $(\cdot, \cdot, \mathsf{PK}_{U_i})$ to the resolution oracle O_{Res} as long as PK_{U_i} is not Alice's public key. This very property was used to attack the scheme of Section 3.2. Note that there is no need to provide B with access to the signing oracle O_{Sig}, since it can be simulated by O_{PSig} and O_{Res}.

Security against the Arbitrator. We require that any PPT adversary C succeeds with at most negligible probability in the following experiment.

$$\begin{array}{rcl}\mathsf{Setup}^{\mathsf{TTP}^*}(1^k) & \to & (\mathsf{ASK}^*, \mathsf{APK}) \\ \mathsf{Setup}^{\mathsf{User}}(1^k) & \to & (\mathsf{SK}_A, \mathsf{PK}_A) \\ (m, \sigma) & \leftarrow & C^{O_{\mathsf{PSig}}}(\mathsf{ASK}^*, \mathsf{PK}_A, \mathsf{APK}) \\ \text{success of } C & = & [\mathsf{Ver}(m, \sigma, \mathsf{PK}_A, \mathsf{APK}) \stackrel{?}{=} 1 \ \wedge\ m \notin \mathit{Query}(C, O_{\mathsf{PSig}})]\end{array}$$

[5]In identity-based cryptosystems, fixing the identity of the target user also fixes the corresponding public key and consequently weakens the security level (so-called "selective-ID security" [15, 8]). However, fixing the identity of the target user in our context does not impose any constraint on the corresponding public key.

where $\mathsf{Setup}^{\mathsf{TTP}^*}$ denotes the run of $\mathsf{Setup}^{\mathsf{TTP}}$ with the dishonest arbitrator (run by C), ASK^* is C's state after this run, and $Query(C, O_{\mathsf{PSig}})$ is the set of queries of C asked to the partial signing oracle O_{PSig}.

4.2 Generic Construction

We present a generic construction of non-interactive setup-free[6] optimistic fair exchange based on the OR-proof where the signer has one witness and the arbitrator has the other witness. We use the Fiat-Shamir heuristic in the random oracle model and the non-interactive witness indistinguishable proof of knowledge in the standard model.

Scheme. Let $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be an ordinary signature scheme.

- $\mathsf{Setup}^{\mathsf{TTP}}$. The arbitrator chooses (sk, vk) by running $\mathsf{Sig\text{-}Gen}(1^k)$ and sets $(\mathsf{ASK}, \mathsf{APK}) = (sk, vk)$.
- $\mathsf{Setup}^{\mathsf{User}}$. Each user U_i chooses (sk_i, vk_i) by running $\mathsf{Sig\text{-}Gen}(1^k)$ and sets $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}) = (sk_i, vk_i)$.
- Sig. When a user U_i wants to sign a message m, the signer generates an ordinary signature s_1 on "$0||m$" (i.e., $s_1 = \mathsf{Sign}_{sk_i}(0||m)$) and then generates an OR-signature s_2 on "$1||m$" for the knowledge of sk_i or $\mathsf{Sign}_{sk}(1||m)$. Since the signer U_i knows sk_i, he can generate the valid OR-signature s_2. The signature value on m is $\sigma_{U_i} = (s_1, s_2)$.
- Ver. To verify the signature $\sigma_{U_i} = (s_1, s_2)$ on m, a verifier checks that (1) $\mathsf{Vrfy}_{vk_i}(0||m, s_1) \stackrel{?}{=} 1$ and (2) s_2 is a valid OR-signature on "$1||m$" for the knowledge of sk_i or $\mathsf{Sign}_{sk}(1||m)$.
- PSig and PVer. The same as Sig and Ver except that the partial signature σ'_{U_i} on m is s_1.
- Res. For the user U_i's partial signature $\sigma'_{U_i} = s_1$ on m, the arbitrator first checks that $\mathsf{Vrfy}_{vk_i}(0||m, s_1) \stackrel{?}{=} 1$ and then computes an OR-signature s_2 on "$1||m$" for the knowledge of sk_i or $\mathsf{Sign}_{sk}(1||m)$. Since the arbitrator knows sk, he can compute an ordinary signature $\mathsf{Sign}_{sk}(1||m)$ and then the valid OR-signature s_2. The arbitrator outputs $\sigma_{U_i} = (s_1, s_2)$.

The correctness property of the scheme is obvious and the ambiguity property follows from the witness indistinguishability of the OR-signature s_2. We now analyze the security.

Theorem 3 *The generic construction of the optimistic fair exchange scheme is multi-user secure in the random oracle model if the underlying signature scheme is secure.*

Proof: See Appendix B. $\square$

Theorem 4 *If there are one-way functions, we can build the setup-free optimistic fair exchange schemes that are multi-user secure in the random oracle model.*

[6]If we allow the registration between the signer and the arbitrator, there are trivial setup-driven solutions. For example, the signer chooses two signature key pairs $(sk_1, pk_1), (sk_2, pk_2)$ and gives only sk_2 to the arbitrator. The user's full signature is $(\mathsf{Sign}_{sk_1}(m), \mathsf{Sign}_{sk_2}(m))$, while the partial signature is $\mathsf{Sign}_{sk_1}(m)$.

Proof: Secure signatures exist if and only if one-way functions exist [28, 33]. Together with Theorem 3, we obtain Theorem 4 . □

The proof of Theorem 3 only requires two properties from the Fiat-Shamir proofs: (1) witness indistinguishability and (2) proof of knowledge. Hence, we can use the straight-line extractable witness indistinguishable proof [30] instead of the Fiat-Shamir proof. Like the Fiat-Shamir heuristic, the construction of the straight-line extractable witness indistinguishable proof starts with the Σ-protocol but the length of the resulting proof is much longer. However, non-programmable random oracle is used and better exact security can be obtained.

Instead of the Fiat-Shamir proof, we can also use the non-interactive witness indistinguishable proofs of knowledge for sk_i or $\mathsf{Sign}_{sk}(m)$. In this case, we do not need the random oracle and can instead use a common reference string (which could be generated by the arbitrator).[7] The construction of non-interactive witness indistinguishable proofs of knowledge requires the existence of trapdoor permutations [35] and this observation leads to the following theorem.

Theorem 5 *If there are trapdoor one-way permutations, we can build the setup-free optimistic fair exchange schemes that are multi-user secure in the standard model.*

Remark 2 *While the construction using non-interactive witness indistinguishable proofs of knowledge in the standard model is mainly of theoretic interest, the construction using the Fiat-Shamir heuristic in the random oracle is efficient for specific cases, as there are efficient Σ-protocols for the knowledge of a signature value and for the knowledge of a secret key corresponding to a given public key (e.g., [25, 36, 13, 14, 9]). For readers who are not familiar with all the theory, Appendix C provides a concrete (efficient) example of the generic construction in the random oracle model.*

5 Previous Paradigms Revisited

5.1 Optimistic Fair Exchange from Verifiably Encrypted Signature

Suppose Alice wants to show Bob that she has signed a message, but does not want Bob to possess her signature. Alice first encrypts her signature using the public encryption key of the arbitrator, and sends the ciphertext to Bob with proof that she has given him a valid encryption of her signature. Bob can verify that Alice has signed the message, but cannot deduce any information on her signature. Later in the protocol, if Alice is unable or unwilling to reveal her signature, Bob can ask the arbitrator to decrypt the ciphertext of Alice's signature.

Scheme. Let (P, V) be a non-interactive zero-knowledge (NIZK) proof system for the NP-language $L = \{(c, m, ek, vk) \mid \exists s\ [c = \mathsf{Enc}_{ek}(s) \wedge \mathsf{Vrfy}_{vk}(m, s) = 1]\}$, where $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an encryption scheme and $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is a signature scheme.[8]

- $\mathsf{Setup}^{\mathsf{TTP}}$. The arbitrator chooses (dk, ek) by running $\mathsf{Enc\text{-}Gen}(1^k)$ and sets $(\mathsf{ASK}, \mathsf{APK}) = (dk, ek)$.

[7]We use a common "reference" string rather than a common "random" string. The arbitrator can indeed publish the common reference string because in our particular scheme cheating in OR-signature or NIZK does not help the arbitrator.

[8]For brevity's sake, we omit the description of a common reference string, which could be generated by the arbitrator.

- $\mathsf{Setup}^{\mathsf{User}}$. Each user U_i chooses (sk_i, vk_i) by running $\mathsf{Sig\text{-}Gen}(1^k)$ and sets $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}) = (sk_i, vk_i)$.
- Sig. When a user U_i wants to sign a message m, the signer generates a signature $s = \mathsf{Sign}_{sk_i}(m)$. The signature value of m is $\sigma_{U_i} = s$.
- Ver. To verify the signature $\sigma_{U_i} = s$ of m, a verifier checks $\mathsf{Vrfy}_{vk_i}(m, s) \overset{?}{=} 1$.
- PSig. When a user U_i wants to generate a partial signature of m, the signer first computes a signature $s = \mathsf{Sign}_{sk_i}(m)$ and then encrypts s with APK, i.e., $c = \mathsf{Enc}_{ek}(s)$. The partial signature of m is $\sigma'_{U_i} = (c, \pi)$, where π is a proof showing $(c, m, ek, vk_i) \in L$.
- PVer. To verify the partial signature $\sigma'_{U_i} = (c, \pi)$ of m, a verifier checks that π is an accepting proof for the statement $(c, m, ek, vk_i) \in L$. If so, 1 is returned and otherwise, 0 is returned.
- Res. For the user U_i's partial signature $\sigma'_{U_i} = (c, \pi)$ of m, the arbitrator first checks that π is an accepting proof for the statement $(c, m, ek, vk_i) \in L$ and then decrypts $s = \mathsf{Dec}_{dk}(c)$. The arbitrator outputs $\sigma_{U_i} = s$.

For security analysis, we need the concept of simulation-sound NIZK proofs [34]. The soundness property of ordinary proof systems states that with overwhelming probability, the prover should be incapable of convincing the verifier of a false statement. Intuitively, the simulation-sound property requires that this remains the case even after a polynomially bounded party has seen a simulated proof of its choosing. The formal definition of simulation soundness can be found in [34].

Theorem 6 *The optimistic fair exchange scheme based on a verifiably encrypted signature is secure if the underlying $\mathcal{E}$ is* CCA*-secure, $\mathcal{S}$ is* UF-CMA*-secure, and (P, V) is a simulation-sound NIZK proof system.*

Proof: See Appendix D. □

We observe that the full signature $\sigma_{U_i} = s$ is a signature value of the underlying ordinary signature scheme $\mathcal{S}$, which means that the fair exchange scheme is stand-alone.[9] In addition, CCA-secure encryption $\mathcal{E}$, UF-CMA-secure signature $\mathcal{S}$, and simulation-sound NIZK proof system (P, V) can be built from trapdoor permutations [34, 28, 33]. Hence, we obtain the following existence theorem of setup-free and stand-alone fair exchange schemes.

Theorem 7 *If there are trapdoor one-way permutations, we can build the optimistic fair exchange schemes that are setup-free and stand-alone.*

5.2 Optimistic Fair Exchange from Sequential Two-Party Multisignature

A multisignature scheme allows any subgroup of users to jointly sign a document such that a verifier is convinced that each user of the subgroup participated in signing. To construct an optimistic fair exchange, we can use a simple type of multisignature, which is called a sequential two-party multisignature.

[9]Recall that a fair exchange scheme is *stand-alone* if the full signature is the same as it were produced by an ordinary signature scheme only [39].

A sequential two-party multisignature $\mathcal{MS}$ consists of five efficient algorithms: $\mathcal{MS} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{MSign}, \mathsf{MVrfy})$. Key generation algorithm $\mathsf{Sig\text{-}Gen}$, signing algorithm Sign, and verification algorithm Vrfy are similar to the conventional algorithms of an ordinary signature scheme. MSign takes as input (m, s_i, vk_i, sk_j) and returns a multisignature s_{ij}, where $m \in \mathcal{M}$ is a message, sk_j is a signing key, s_i is a valid signature w.r.t. a verification key vk_i, and s_{ij} is a multisignature w.r.t. verification keys vk_i and vk_j. MVrfy takes (m, s_{ij}, vk_i, vk_j) as input and returns 1 (accept) or 0 (reject). Correctness property requires that $\mathsf{Vrfy}_{vk_i}(m, \mathsf{Sign}_{sk_i}(m)) = 1$ and $\mathsf{MVrfy}(m, \mathsf{MSign}(m, s_i, vk_i, sk_j), vk_i, vk_j) = 1$, for any $m \in \mathcal{M}$. A multisignature scheme is *symmetric* if s_{ij} and s_{ji} are computationally indistinguishable.[10]

For security consideration, we allow the adversary $\mathcal{A}$, who tries to forge a multisignature w.r.t. a given verification key, to have access to the signing oracle O_{Sign} and the multi-signing oracle O_{MSign}. $\mathcal{A}$'s query to O_{Sign} is (m, vk_i) and O_{Sign} returns $\mathsf{Sign}_{sk_i}(m)$. $\mathcal{A}$'s query to O_{MSign} is (m, s_i, vk_i, vk_j) and O_{MSign} returns s_{ij} if $\mathsf{Vrfy}_{vk_i}(m, s_i) = 1$. While the adversary $\mathcal{A}$ is allowed to create arbitrary keys for corrupted users, we require $\mathcal{A}$ to prove knowledge of secret keys during the public key registration. For simplicity, we follow the model of [7] which asks $\mathcal{A}$ to output the public key and secret key of a corrupted user in the key registration stage. Let $Query(\mathcal{A}, O_{\mathsf{Sign}})$ and $Query(\mathcal{A}, O_{\mathsf{MSign}})$ be the set of valid queries of $\mathcal{A}$ to O_{Sign} and O_{MSign}, respectively. We define $\mathcal{A}$'s advantage of attacking $\mathcal{MS}$ as follows.

$$\mathsf{Adv}^{\mathcal{MS}}_{\mathcal{A}}(k) = \Pr\left[\mathsf{MVrfy}(m, s, vk_i, vk_j) = 1 \vee \mathsf{MVrfy}(m, s, vk_j, vk_i) = 1 \;\middle|\; \begin{array}{l} (sk_i, vk_i) \leftarrow \mathsf{Sig\text{-}Gen}(1^k), \\ (m, s, vk_j) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}, O_{\mathsf{MSign}}}(vk_i) \end{array}\right]$$

where $(m, vk_i) \notin Query(\mathcal{A}, O_{\mathsf{Sign}})$ and $(m, \cdot, vk_j, vk_i) \notin Query(\mathcal{A}, O_{\mathsf{MSign}})$.

Definition 3 *Let $\mathcal{MS} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{MSign}, \mathsf{MVrfy})$ be a sequential two-party signature scheme. An adversary $\mathcal{A}$ is said to $(t, q_s, q_{ms}, \varepsilon)$-break $\mathcal{MS}$, if $\mathcal{A}$ runs in time at most t, makes at most q_s signing queries to O_{Sign} and q_{ms} multi-signing queries to O_{MSign}, and succeeds in forgery with probability at least ε. $\mathcal{MS}$ is said to be $(t, q_s, q_{ms}, \varepsilon)$-secure, if no adversary can $(t, q_s, q_{ms}, \varepsilon)$-break it. Asymptotically, $\mathcal{MS}$ is* UF-CMA-*secure if $\mathsf{Adv}^{\mathcal{MS}}_{\mathcal{A}}(k)$ is negligible for any PPT adversary $\mathcal{A}$.*

Remark 3 *If a sequential two-party signature scheme $\mathcal{MS} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{MSign}, \mathsf{MVrfy})$ is* UF-CMA-*secure, the induced signature scheme $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is also* UF-CMA-*secure.*

By relaxing the definition of optimistic fair exchange to allow interactive registration during setup (i.e., setup-driven), we can have much simpler (almost trivial) schemes based on the sequential two-party multisignature. Each user U_i generates four keys SK_{U_i}, PK_{U_i}, ASK_{U_i}, APK_{U_i} and sends PK_{U_i}, ASK_{U_i}, APK_{U_i} to the arbitrator, who checks if the keys were properly generated. The arbitrator will then store ASK_{U_i} and certify APK_{U_i}. A verifier will accept partial signatures from U_i only if they are valid w.r.t. APK_{U_i}.

SCHEME. Let $\mathcal{MS} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{MSign}, \mathsf{MVrfy})$ be a sequential two-party multisignature scheme.

- $\mathsf{Setup}^{\mathsf{TTP}}$ and $\mathsf{Setup}^{\mathsf{User}}$. Each user U_i chooses $(sk^0_{U_i}, vk^0_{U_i})$ and $(sk^1_{U_i}, vk^1_{U_i})$ by running $\mathsf{Sig\text{-}Gen}(1^k)$ twice, and sends $(vk^0_{U_i}, sk^1_{U_i}, vk^1_{U_i})$ to the arbitrator. After checking validity of the keys, the arbitrator stores $sk^1_{U_i}$ and certifies $vk^1_{U_i}$. If we use a simplified notation such as $sk_{i_0} = sk^0_{U_i}$, $vk_{i_1} = vk^1_{U_i}$, the output is $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}, \mathsf{ASK}_{U_i}, \mathsf{APK}_{U_i}) = ((sk_{i_0}, sk_{i_1}), (vk_{i_0}, vk_{i_1}), sk_{i_1}, vk_{i_0})$.

[10]Symmetric multisignature schemes have natural symmetric properties such as $\mathsf{MVrfy}(m, s_{ij}, vk_i, vk_j) = \mathsf{MVrfy}(m, s_{ij}, vk_j, vk_i)$.

- Sig. When a user U_i wants to sign a message m, the signer computes $s_{i_0} = \mathsf{Sign}_{sk_{i_0}}(m)$ and a multisignature $s_{i_0 i_1} = \mathsf{MSign}(m, s_{i_0}, vk_{i_0}, sk_{i_1})$. The signature value of m is $\sigma_{U_i} = s_{i_0 i_1}$.
- Ver. To verify the signature $\sigma_{U_i} = s_{i_0 i_1}$ of m, a verifier checks $\mathsf{MVrfy}(m, s_{i_0 i_1}, vk_{i_0}, vk_{i_1}) \overset{?}{=} 1$.
- PSig. When a user U_i wants to generate a partial signature of a message m, the signer computes a signature $s_{i_0} = \mathsf{Sign}_{sk_{i_0}}(m)$. The partial signature of m is $\sigma'_{U_i} = s_{i_0}$.
- PVer. To verify the partial signature $\sigma'_{U_i} = s_{i_0}$ of m w.r.t. PK_{U_i}, a verifier checks $\mathsf{Vrfy}_{vk_{i_0}}(m, s_{i_0}) \overset{?}{=} 1$. If so, 1 is returned and otherwise, 0 is returned.
- Res. For the user U_i's partial signature $\sigma'_{U_i} = s_{i_0}$ of m, the arbitrator first checks $\mathsf{Vrfy}_{vk_{i_0}}(m, s_{i_0}) \overset{?}{=} 1$ and then generates a multisignature $s_{i_0 i_1} = \mathsf{MSign}(m, s_{i_0}, vk_{i_0}, sk_{i_1})$. The arbitrator outputs $\sigma_{U_i} = s$.

Remark 4 *While the generic construction based on the sequential two-party multisignature is almost trivial, specific instantiations could be very efficient by directly using the combined signing key* $sk_{U_i} = sk^0_{U_i} \diamond sk^1_{U_i}$ *to generate multisignatures and the combined verification key* $pk_{U_i} = pk^0_{U_i} \circ pk^1_{U_i}$ *to verify multisignatures.*

Theorem 8 *The setup-driven optimistic fair exchange scheme based on a sequential two-party multisignature is secure if the underlying multisignature is* $\mathsf{UF\text{-}CMA}$*-secure.*

Proof: See Appendix E. □

References

[1] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *ACM Conference on Computer and Communications Security*, pages 7–17. ACM, 1997.

[2] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer, 1998.

[3] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communication*, 18(4):593–610, 2000.

[4] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.

[5] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[7] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *PKC 2003*, Lecture Notes in Computer Science, pages 31–46. Springer, 2003.

[8] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[9] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[10] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[11] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[12] J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2000.

[13] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002.

[14] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

[15] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.

[16] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[17] Y. Dodis and L. Reyzin. Breaking and repairing optimistic fair exchange from PODC 2003. In *the 2003 ACM Workshop on Digital Rights Management*, pages 47–54. ACM, 2003.

[18] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544. Springer, 1989.

[19] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *the 22nd Annual ACM Symposium on Theory of Computing*, pages 416–426. ACM, 1990.

[20] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[21] S. D. Galbraith, J. Malone-Lee, and N. P. Smart. Public key signatures in the multi-user setting. *Inf. Process. Lett.*, 83(5):263–266, 2002.

[22] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[23] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[24] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[25] L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *CRYPTO 1988*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, 1988.

[26] J. Håstad. Solving simultaneous modular equations of low degree. *SIAM J. Comput.*, 17(2):336–341, 1988.

[27] A. J. Menezes and N. P. Smart. Security of signature schemes in a multi-user setting. *Designs, Codes and Cryptography*, 33(3):261–274, November 2004.

[28] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43. ACM, 1989.

[29] J. M. Park, E. K. P. Chong, and H. J. Siegel. Constructing fair-exchange protocols for e-commerce via distributed computation of RSA signatures. In *PODC 2003*, pages 172–181. ACM, 2003.

[30] R. Pass. On deniability in the common reference string and random oracle model. In *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337. Springer, 2003.

[31] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.

[32] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO 1991*, Lecture Notes in Computer Science, pages 433–444. Springer, 1991.

[33] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM, 1990.

[34] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *the 40th Annual Symposium on Foundations of Computer Science*, pages 543–553. IEEE, 1999.

[35] A. D. Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *the 33rd Annual Symposium on Foundations of Computer Science*, pages 427–436. IEEE, 1992.

[36] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[37] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[38] G. J. Simmons. A "weak" privacy protocol using the RSA crypto algorithm. *Cryptologia*, 7(2):180–182, Apr. 1983.

[39] H. Zhu and F. Bao. Stand-alone and setup-free verifiably committed signatures. In *CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 2006.

Appendix A. Security Proof for the Scheme of Section 3.2

Theorem 9 *The optimistic fair exchange scheme of Section 3.2 is single-user secure if the underlying trapdoor permutation and signature scheme are secure.*[11]

Proof: SECURITY AGAINST ALICE. Security against Alice follows unconditionally. If a partial signature $\sigma'_A = (y_A, \delta_A)$ passes PVer, we have $\mathsf{Vrfy}_{vk}(m\|y_A, \delta_A) = 1$. Now, the honest arbitrator Charlie can compute $r_A = f^{-1}(y_A)$ and the resolved signature $\sigma_A = (r_A, \delta_A)$ passes Ver.

SECURITY AGAINST BOB. To show security against Bob, we convert any attacker B that attacks the fair exchange scheme into an inverter Inv for the trapdoor one-way permutation f. Recall that Inv gets as input a trapdoor one-way permutation $g : D \to D$ and $b \in D$, and wins if Inv outputs $a \in D$ satisfying $b = g(a)$. On the other hand, B expects $(\mathsf{PK}, \mathsf{APK})$ and oracle access to both O_{PSig} and O_{Res}, and wins if B forges a signature σ of some message m without asking a valid query (m, σ') to O_{Res}. Let (m_B, σ_B) be the successful forgery of the attacker B. We can assume that B obtained the corresponding partial signature σ'_B on m_B from O_{PSig}, since the underlying signature scheme $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is existentially unforgeable.

On input of g and b, Inv begins simulating the attack environment of B. It picks a random signing/verification key pair (sk, vk) by running $\mathsf{Sig\text{-}Gen}(1^k)$, sets $\mathsf{PK} = vk$, $\mathsf{SK} = sk$, $\mathsf{APK} = g$, and gives $(\mathsf{PK}, \mathsf{APK})$ to B. Let q_{PSig} be the total number of O_{PSig} queries made by B and j be a random number chosen by Inv in the interval of $\{1, 2, \cdots, q_{\mathsf{PSig}}\}$. Now, Inv knowing $\mathsf{SK} = sk$ responds to the i-th O_{PSig} query m_i of B as follows.

- If $i = j$, Inv sets $y_i = b$ and computes $\delta_i = \mathsf{Sign}_{sk}(m_i\|y_i)$. Inv returns $\sigma'_i = (y_i, \delta_i)$ to B.
- If $i \neq j$, Inv picks $r_i \in D$ randomly and computes $y_i = g(r_i)$, $\delta_i = \mathsf{Sign}_{sk}(m_i\|y_i)$. Inv returns $\sigma'_i = (y_i, \delta_i)$ to B.

Inv maintains a list $H = \{(m_i, r_i, \sigma'_i) \mid 1 \leq i \leq q_{\mathsf{PSig}}\}$, where $r_j = \perp$. To simulate O_{Res}'s response to a resolution query (m_i, σ'_i), Inv checks the validity of the partial signature σ'_i and retrieves the corresponding r_i from the list H. If $r_i = \perp$ (meaning $i = j$), Inv aborts. Note that if the query is valid but m_i is not in the list, it is an existential forgery of the signature scheme $\mathcal{S}$.

When B outputs the forgery (m_B, σ_B) where $\sigma_B = (r, \delta)$, Inv verifies whether $b = g(r)$ or not. If $b \neq g(r)$, Inv outputs a random number. Otherwise, Inv outputs r. Let ϵ be the success probability of B's forgery in the real attack environment. Since $b \in D$ and $r \in \{1, \cdots, q_{\mathsf{PSig}}\}$ are randomly chosen and g is a permutation, Inv succeeds in inverting g with a probability $\epsilon' \geq \epsilon / q_{\mathsf{PSig}}$.

SECURITY AGAINST CHARLIE. To show security against Charlie, we convert any arbitrator C that attacks the optimistic fair exchange scheme into a forger F for the underlying signature $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. The forger F, on input vk, generates a trapdoor permutation (f, f^{-1}) and gives $(\mathsf{ASK}, \mathsf{PK}, \mathsf{APK}) = (f^{-1}, vk, f)$ to C. Now, F responds to the i-th O_{PSig} query m_i of C with $\sigma' = (y, \delta_A)$, where y is chosen randomly and $\delta = \mathsf{Sign}(m_i\|y)$ is obtained from its own signing oracle O_{Sign}. When C outputs the forgery (m, σ_A) where $\sigma_A = (r_A, \delta_A)$, F computes $y_A = f(r_A)$ and outputs $(m\|y_A,\ \delta_A)$. We see that the simulation is perfect and F succeeds in producing a new forgery if and only if C succeeds. □

[11]Actually, we need not assume secure signature, as signature schemes can be built from trapdoor permutations.

Appendix B. Security Proof for Theorem 3

Theorem 3 *The optimistic fair exchange scheme of Section 4.2 is multi-user secure in the random oracle model if the underlying signature scheme is secure.*

Proof: Security against Signers. Security against the signer follows unconditionally. Since the OR-signatures are non-interactive proofs of knowledge of sk_i or $\mathsf{Sign}_{sk}(1||m)$, the arbitrator who knows sk can always generate valid OR-signatures. Therefore, if the partial signature $\sigma' = s_1$ passes PVer, the honest arbitrator can transform the partial signature σ' into the valid full signature σ by generating an OR-signature s_2.

Security against Verifiers. To show security against the verifier, we convert any verifier B that attacks the optimistic fair exchange scheme into a forger F for the underlying signature $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. Recall that F gets vk as input and has access to the signing oracle O_{Sign}. The forger F wins if it forges a signature which has not been queried to O_{Sign}. On the other hand, B expects $(\mathsf{PK}_A, \mathsf{APK})$ as input and has access to both O_{PSig} and O_{Res} oracles. B wins if it forges a signature σ of a message m without asking a valid query $(m, \sigma', \mathsf{PK}_A)$ to O_{Res}. Let (m, σ) be a successful forgery of B, where s_1 is a signature on "$0||m$" (w.r.t. PK_A) and $\sigma = (s_1, s_2)$. If B did not obtain the corresponding partial signature $\sigma' = s_1$ from O_{PSig}, s_1 becomes an existential forgery of $\mathcal{S}$ and the analysis of this type of attack is covered in the security against the arbitrator discussed later. Hence, we assume that B has obtained the corresponding partial signature $\sigma' = s_1$ from O_{PSig}.

On input vk, the forger F begins simulating the attack environment of B. It sets $vk_1 = vk$ and picks a random key pair (sk_2, vk_2) by running $\mathsf{Sig\text{-}Gen}(1^k)$. F flips a coin and gets a random bit b. According to the random bit b, F performs one of the following two games.

- Game 0 ($b = 0$): The forger F gives $(\mathsf{PK}_A, \mathsf{APK}) = (vk_1, vk_2)$ to B and answers the i-th O_{PSig} query m_i of B by getting an ordinary signature $\mathsf{Sign}_{sk_1}(0||m_i)$ from its own signing oracle O_{Sign}. To simulate O_{Res} to a resolution query $(m_i, \sigma'_i, \mathsf{PK}_i) = (m_i, s_1^i, \mathsf{PK}_i)$ of B, the forger F checks the validity of (m_i, s_1^i) w.r.t. PK_i and then computes $\mathsf{Sign}_{sk_2}(1||m_i)$ with sk_2. From the knowledge of the signature $\mathsf{Sign}_{sk_2}(1||m_i)$, F can generate a valid OR-signature s_2^i on the message $1||m_i$. F returns $\sigma_i = (s_1^i, s_2^i)$ to B.

- Game 1 ($b = 1$): The forger F gives $(\mathsf{PK}_A, \mathsf{APK}) = (vk_2, vk_1)$ to B and answers the i-th O_{PSig} query m_i of B by computing a signature $\mathsf{Sign}_{sk_2}(0||m_i)$ with sk_2. To simulate O_{Res} to a resolution query $(m_i, \sigma'_i, \mathsf{PK}_i) = (m_i, s_1^i, \mathsf{PK}_i)$ of B, the forger F checks the validity of (m_i, s_1^i) w.r.t. PK_i and then obtains a signature $\mathsf{Sign}_{sk_1}(0||m_i)$ from its own signing oracle O_{Sign}. Now, F generates a valid OR-signature s_2^i on the message $1||m_i$ from the knowledge of $\mathsf{Sign}_{sk_1}(0||m_i)$, and returns $\sigma_i = (s_1^i, s_2^i)$ to B.

If the attacker B succeeds in attacking, i.e., forges a valid OR-signature s_2 on $1||m$ without asking the arbitrator, the forger F rewinds B and obtains one of the two witnesses SK_A and $\mathsf{Sign}_{\mathsf{ASK}}(1||m)$. The forger wins if (1) he performs Game 0 and extracts the witness SK_A (total break recovering the signing key), or (2) he performs Game 1 and extracts the witness $\mathsf{Sign}_{\mathsf{ASK}}(1||m)$ (existential forgery of a new message "$1||m$"). Note that B cannot distinguish between Game 0 and Game 1 because the simulation is perfect and OR-signatures are witness indistinguishable. Therefore, if B succeeds with a probability ϵ, the forger F succeeds with a probability $\epsilon/2$.

Security against the Arbitrator. To show security against the arbitrator, we convert any arbitrator C that attacks the optimistic fair exchange scheme into a forger F for the underlying signature $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. Recall that F gets vk as an input and has access to the signing oracle O_{Sign}. On the other hand, C expects $(\mathsf{ASK}, \mathsf{PK}_A, \mathsf{APK})$ as input and has access to O_{PSig}. C wins if it generates a valid signature σ_A of some message m without asking m to O_{PSig}.

Thus, here is how F, on input vk, simulates the run of C. To choose $(\mathsf{ASK}, \mathsf{APK})$, F runs $\mathsf{Sig\text{-}Gen}(1^k)$ and obtains (sk_C, vk_C). Then, F gives $(\mathsf{ASK}, \mathsf{PK}_A, \mathsf{APK}) = (sk_C, vk, vk_C)$ to C.[12] Now, F responds to the i-th O_{PSig} query m_i of C by getting a signature on "$0||m_i$" from its own signing oracle O_{Sign}. When C outputs the forgery (m, σ_A) where $\sigma_A = (s_1, s_2)$ and s_1 is an ordinary signature on "$0||m$" (w.r.t. vk), F outputs $(0||m,\ s_1)$. We see that the simulation is perfect and F succeeds in producing a new forgery if and only if C succeeds.

Random Oracles. Careful readers could find that we omitted to simulate random oracles. The random oracles are (1) implicitly used in the OR-signatures that are based on the Fiat-Shamir heuristic and (2) explicitly called if needed. The simulation of the random oracles can be done simply by answering randomly but consistently since we do not need to manipulate the answers of the random oracles. The only exception is the knowledge extraction by rewinding, which can be treated easily. □

Appendix C. Concrete Example of Generic Construction

The main purpose of generic construction is to find out minimal computational complexity assumptions under which setup-free optimistic fair exchange exists in the multi-user setting. However, many efficient Σ-protocols are already known for specific cases (e.g., [25, 36, 13, 14, 9]) and, by using them, the generic construction gives efficient fair exchange schemes in the random oracle model. Here, we present an example based on the Schnorr protocol [36] and the GQ protocol [25]. The output length is $|\sigma'_{U_i}| \simeq 320$ and $|\sigma_{U_i}| \simeq 1824$ (for typical parameters of $|p| = |n| = 1024, |q| = 160$) and each procedure requires only a few exponentiations, which is a comparable performance to the state-of-the-art setup-free schemes. The security is based on the standard RSA and discrete logarithm problems. If we use other efficient Σ-protocols, we can obtain optimistic fair exchange schemes of different performance characteristics.

Scheme. Let (p, q, g, t, H_1, H_2) be a domain parameter, where (1) p and q are primes such that $q \mid p-1$, (2) g is a generator for the subgroup of $\mathbb{Z}_p^*$ of order q, (3) t is an integer such that $2^t < q$, and (4) $H_1 : \{0,1\}^* \to \mathbb{Z}_n^*$ and $H_2 : \{0,1\}^* \to \{0,1\}^t$ are cryptographic hash functions.

- $\mathsf{Setup}^{\mathsf{TTP}}$. The arbitrator chooses an RSA modulus $n = p'q'$ and exponents e, d, where p' and q' are safe primes, e is a small prime, and d satisfies $ed \equiv 1 \bmod \varphi(n)$. The arbitrator sets $\mathsf{ASK} = d$ and $\mathsf{APK} = (n, e)$.
- $\mathsf{Setup}^{\mathsf{User}}$. Each user U_i chooses $x_i \in_R \mathbb{Z}_q^*$, computes $y_i = g^{x_i} \bmod p$, and sets $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}) = (x_i, y_i)$.
- Sig. When a user U_i wants to sign a message m, the signer first generates a Schnorr signature

[12] Precisely, (sk_C, vk_C) should be generated by the adversary C, who keeps sk_C secret and vk_C public. However, in that case, it is a common practice that one requests C to perform zero-knowledge proof of knowledge of sk_C, which, in turn, allows the forge F to obtain sk_C by rewinding C. Therefore, the effect of C's generation of (sk_C, vk_C) makes no practical difference in our simulation.

$s_1 = (c_0, z_0)$ on "$0\|m$" and an OR-signature $s_2 = (c_1, z_1, c_2, z_2)$ on "$1\|m$" by using x_i. The signature is $\sigma_{U_i} = (s_1, s_2)$.

s_1	s_2	
(c_0, z_0)	(c_1, z_1)	(c_2, z_2)
$r_0 \leftarrow \mathbb{Z}_q^*$	$r_1 \leftarrow \mathbb{Z}_q^*$	$c_2 \leftarrow \mathbb{Z}_{2^t}, z_2 \leftarrow \mathbb{Z}_n^*$
$a_0 = g^{r_0} \bmod p$	$a_1 = g^{r_1} \bmod p$	$a_2 = z_2^e \cdot H_1(1\|m)^{-c_2} \bmod n$
$c_0 = H_2(0\|m, a_0)$	$c = H_2(1\|m, a_1, a_2)$	
$z_0 = r_0 + c_0 x_i \bmod q$	$c_1 = c \oplus c_2$	
	$z_1 = r_1 + c_1 x_i \bmod q$	

- Ver. To verify the signature $\sigma_{U_i} = (s_1, s_2)$ on m, a verifier checks the following conditions.

s_1	s_2
(c_0, z_0)	(c_1, z_1, c_2, z_2)
$a_0 = g^{z_0} y_i^{-c_0} \bmod p$	$a_1 = g^{z_1} y_i^{-c_1} \bmod p$
$c_0 \stackrel{?}{=} H_2(0\|m, a_0)$	$a_2 = z_2^e \cdot H_1(1\|m)^{-c_2} \bmod n$
	$c_1 \oplus c_2 \stackrel{?}{=} H_2(1\|m, a_1, a_2)$

- PSig and PVer. The same as Sig and Ver except that the partial signature σ'_{U_i} on m is s_1.
- Res. For the user U_i's partial signature $\sigma'_{U_i}(= s_1 = (c_0, z_0))$ on m, the arbitrator first checks $c_0 \stackrel{?}{=} H_2(0\|m, g^{z_0} y_i^{-c_0} \bmod p)$ and then computes an OR-signature $s_2 = (c_1, z_1, c_2, z_2)$ on "$1\|m$" by using an RSA signature value $\omega = H_1(1\|m)^d \bmod n$. The arbitrator outputs $\sigma_{U_i} = (s_1, s_2)$.

s_2	
(c_1, z_1)	(c_2, z_2)
$c_1 \leftarrow \mathbb{Z}_{2^t}, z_1 \leftarrow \mathbb{Z}_q^*$	$r_2 \leftarrow \mathbb{Z}_n^*$
$a_1 = g^{z_1} y_i^{-c_1} \bmod p$	$a_2 = r_2^e \bmod n$
$c = H_2(1\|m, a_1, a_2)$	
	$c_2 = c \oplus c_1$
	$z_2 = r_2 \omega^{c_2} \bmod n$

Appendix D. Security Proof for Theorem 6

Theorem 6 *The optimistic fair exchange scheme based on a verifiably encrypted signature is secure if the underlying $\mathcal{E}$ is* CCA*-secure, $\mathcal{S}$ is* UF-CMA*-secure, and (P, V) is a simulation-sound NIZK proof system.*

Proof: Security against Signers. To break the security against signers, a dishonest signer has to generate a partial signature $\sigma'_{U_i} = (c, \pi)$ of m, where π is an accepting proof but $(c, m, ek, vk_i) \notin L$. However, this is infeasible by the soundness of the NIZK proof system (P, V).

Security against Verifiers. To show security against the verifier, we convert any verifier B that attacks the optimistic fair exchange scheme into a distinguisher $\mathcal{D}$ for the underlying encryption scheme $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$ which is CCA-secure. Recall that $\mathcal{D}$ gets ek as an input and has access to the decryption oracle O_{Dec}. The distinguisher $\mathcal{D}$ wins if it distinguishes encryptions of two equal-length messages of its own choosing. On the other hand, B expects $(\mathsf{PK}_A, \mathsf{APK})$ as input and has access to both O_{PSig} and O_{Res} oracles and wins if B forges a signature σ of some message m without asking a valid query $(m, \sigma', \mathsf{PK}_A)$ to O_{Res}. Let (m, σ) be a successful forgery of the adversary B. If B did not obtain the corresponding partial signature σ' from O_{PSig}, σ becomes an existential forgery of $\mathcal{S}$ and the analysis of this type of attack is covered in the security against the arbitrator discussed later. Hence, we assume that B has obtained the corresponding partial signature σ' from O_{PSig}.

Let q be the total number of O_{PSig} queries made by B. On input ek, the distinguisher $\mathcal{D}$ begins simulating the attack environment of B by generating a random key pair $(sk, vk) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$ and setting $(\mathsf{PK}_A, \mathsf{APK}) = (vk, ek)$. After choosing j randomly in the interval of $\{1, 2, \cdots, q\}$, the distinguisher simulates O_{PSig}'s response to the i-th query m_i of B as follows.

- If $i = j$, $\mathcal{D}$ chooses a random message $\dot{m}_0$, sets $\dot{m}_1 = m_i$, and computes $(\dot{s}_0, \dot{s}_1)$=$(\mathsf{Sign}_{sk}(\dot{m}_0)$, $\mathsf{Sign}_{sk}(\dot{m}_1))$. $\mathcal{D}$ sends the two "messages" $(\dot{s}_0, \dot{s}_1)$ to the CCA challenger. Let $\dot{c}_b$ be the challenge ciphertext returned by the CCA challenger, which equals to either $\mathsf{Enc}_{ek}(\dot{s}_0)$ or $\mathsf{Enc}_{ek}(\dot{s}_1)$. Finally, $\mathcal{D}$ returns $\sigma'_i = (c_i, \pi_i)$, where $c_i = \dot{c}_b$ and π_i is a *simulated* proof showing $(c_i, m_i, ek, vk) \in L$, and stores (m_i, σ'_i) for a later use.
- If $i \neq j$, $\mathcal{D}$ returns $\sigma'_i = (c_i, \pi_i)$, where $s_i = \mathsf{Sign}_{sk}(m_i)$, $c_i = \mathsf{Enc}_{ek}(s_i)$, and π_i is a proof showing $(c_i, m_i, ek, vk) \in L$.

To simulate O_{Res}'s response to B's resolution query $(m_i, \sigma'_i, \mathsf{PK}_i)$ where $\sigma'_i = (c_i, \pi_i)$, the distinguisher $\mathcal{D}$ checks the validity of π_i.

- If π_i is valid and $\mathsf{PK}_i \neq vk$, $\mathcal{D}$ obtains the plaintext s_i of c_i from its own decryption oracle O_{Dec}. The distinguisher $\mathcal{D}$ returns $\sigma_i = s_i$, which is a signature of m_i w.r.t. PK_i by the soundness of NIZK.
- If π_i is valid and $\mathsf{PK}_i = vk$, $\mathcal{D}$ checks whether $m_i = m_j$ or not. If $m_i = m_j$, $\mathcal{D}$ aborts the simulation and outputs a random bit to the CCA challenger. Otherwise, $\mathcal{D}$ obtains the plaintext s_i of c_i from its own decryption oracle O_{Dec} and returns $\sigma_i = s_i$ to B.
- If π_i is invalid, $\mathcal{D}$ returns a random value to B.

If the challenge ciphertext $\dot{c}_b$ is the encryption of $\dot{s}_0$ (i.e., $b = 0$), c_j has no information on the signature of m_j and B's chance of forging a valid signature of m_j (w.r.t. vk) is negligible. If the challenge ciphertext $\dot{c}_b$ is the encryption of $\dot{s}_1$ (i.e., $b = 1$), c_j is an encryption of a valid signature of m_j and the distribution of B's view in the simulated environment is identical with that in the real attack environment. Even after seeing a simulated proof π_j in case of $b = 0$, B cannot generate O_{Res} queries containing an accepting proof of a false statement by the simulation-sound property of the proof system (P, V). Let (m, σ) be the final output of B. If $m \neq m_j$, $\mathcal{D}$ outputs a random bit to the CCA challenger. If $m = m_j$ and σ is a valid signature[13] of m with respect to vk, $\mathcal{D}$ outputs 1 and otherwise, $\mathcal{D}$ outputs a random bit to the CCA challenger. If B in the real attack environment

[13]As the signature scheme $\mathcal{S}$ dose not need to be secure in the sense of "strong unforgeability," which means that an adversary should be unable to forge a new signature even on a previously-signed message, σ can be different from $\dot{s}_1$.

succeeds with a non-negligible probability ϵ, the distinguisher $\mathcal{D}$'s advantage is also non-negligible and given as follows.

$$\mathsf{CCA\text{-}Adv}_{\mathcal{D}} = \left| \left\{ \frac{q-1}{q} \times \frac{1}{2} + \frac{1}{q} \left(\frac{\epsilon}{2} \times 1 + \left(1 - \frac{\epsilon}{2}\right) \times \frac{1}{2} \right) \right\} - \frac{1}{2} \right| \approx \frac{2+\epsilon}{4q}$$

Security against the Arbitrator. We convert an arbitrator C that attacks the optimistic fair exchange scheme into a forger F for the underlying signature $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. The forger F, on input vk, runs $\mathsf{Enc\text{-}Gen}(1^k)$ and obtains (dk_C, ek_C). Then, F gives $(\mathsf{ASK}, \mathsf{PK}_A, \mathsf{APK}) = (dk_C, vk, ek_C)$ to C. Now, F responds to the i-th O_{PSig} query m_i of C by returning $(c_i = \mathsf{Enc}_{ek_C}(s_i), \pi_i)$ where s_i is a signature of m_i from its own signing oracle O_{Sign} and π_i is a proof showing $(c_i, m_i, ek_C, vk) \in L$. When C outputs the forgery (m, σ) where $\sigma = s$ is a signature of m (w.r.t. vk), F outputs (m, s). The simulation is perfect and F succeeds in producing a new forgery if and only if C succeeds. $\square$

Appendix E. Security Proof for Theorem 8

Theorem 8 *The optimistic fair exchange scheme based on a sequential two-party multisignature is secure if the underlying multisignature is* $\mathsf{UF\text{-}CMA}$*-secure.*

Proof: Security against Signers. If a partial signature $\sigma'_{U_i} = s_{i_0}$ passes PVer, we have $\mathsf{Vrfy}_{vk_{i_0}}(m, s_{i_0}) = 1$ and s_{i_0} is a valid signature. By the correctness property of multisignature, the arbitrator who knows sk_{i_1} can computer a multisignature $s_{i_0 i_1} = \mathsf{MSign}(m, s_{i_0}, vk_{i_0}, sk_{i_1})$ which satisfies $\mathsf{MVrfy}(m, s_{i_0 i_1}, vk_{i_0}, vk_{i_1}) = 1$.

Security against Verifiers. We convert a verifier B attacking the optimistic fair exchange scheme into a forger F against $\mathcal{MS}$. The forger F, given vk_i, generates $(sk_j, vk_j) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$ and gives $(\mathsf{PK}_A, \mathsf{APK}_A) = (vk_{A_0}, vk_{A_1}) = (vk_j, vk_i)$ to B. We know that the induced signature scheme $\mathcal{S}$ is $\mathsf{UF\text{-}CMA}$-secure and hence assume that B always makes a partial signature query m to forge a full signature of m.

When B makes a partial signature query m to O_{PSig}, F returns $\sigma'_A = \mathsf{Sign}_{sk_j}(m)$. When B makes a resolution query $(m, \sigma'_l, \mathsf{PK}_l)$ to O_{Res} where $\sigma'_l = s_{l_0}$, F first checks $\mathsf{Vrfy}_{vk_{l_0}}(m, s_{l_0}) \overset{?}{=} 1$ and then $\mathsf{PK}_i \overset{?}{=} \mathsf{PK}_A$. (1) If $\mathsf{PK}_l = \mathsf{PK}_A$ (i.e., $\mathsf{PK}_l{=}vk_{A_0}{=}vk_j$ and $s_{l_0}{=}s_{A_0}$), F makes a multisignature query (m, s_{l_0}, vk_j, vk_i) to its own oracle O_{MSign}. The answer from O_{MSign} is s_{ji} w.r.t. vk_j and vk_i. F returns $\sigma_l = s_{l_0 l_1}{=}s_{A_0 A_1}{=}s_{ji}$ to B. (2) If $\mathsf{PK}_l \neq \mathsf{PK}_A$, F knows the corresponding secret keys (sk_{l_0}, sk_{l_1}) during the public key registration. F generates the multisignature $s_{l_0 l_1} = \mathsf{MSign}(m, s_{l_0}, vk_{l_0}, sk_{l_1})$ and returns $\sigma_l = s_{l_0 l_1}$ to B.

The simulation is perfect and the constraint $(m, \cdot, \mathsf{PK}_A) \notin \mathit{Query}(B, O_{\mathsf{Res}})$ in the fair exchange implies $(m, \cdot, vk_j, vk_i) \notin \mathit{Query}(F, O_{\mathsf{MSign}})$ in the multisignature. Therefore, B's successful forgery (m, σ_A) where $\sigma_A{=}\ s_{A_0 A_1}{=}\ s_{ji}$ implies a forged multisignature (m, s_{ji}, vk_j, vk_i) for $\mathcal{MS}$.

Security against the Arbitrator. We convert an arbitrator C attacking the optimistic fair exchange scheme into a forger F against $\mathcal{MS}$. The forger F, given vk_i, generates $(sk_j, vk_j) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$ and gives $(vk_{i_0}, sk_{i_1}, vk_{i_1}) = (vk_i, sk_j, vk_j)$ to C. Now, F simulates O_{PSig} simply by relaying queries and answers between C and its own signing oracle O_{Sign}. The simulation is perfect and C's successful forgery (m, σ_i), where $\sigma_i{=}\ s_{i_0 i_1}{=}\ s_{ij}$, implies a forged multisignature (m, s_{ij}, vk_j) for $\mathcal{MS}$, which satisfies $\mathsf{MVrfy}(m, s_{ij}, vk_i, vk_j) = 1$. $\square$