# Optimistic Fair Exchange in a Multi-User Setting

Yevgeniy Dodis[1], Pil Joong Lee[2†], and Dae Hyun Yum[2†]

[1]Department of Computer Science, New York University, NY, USA
dodis@cs.nyu.edu
[2]Department of Electronic and Electrical Eng., POSTECH, Pohang, Korea
{pjl, dhyum}@postech.ac.kr

**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 [20, 21]; 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. The security in the multi-user setting was formally studied only recently [4, 18]. 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 [18]. Therefore, we only have to deal with the single-user security and

† The research of the second author and the third author was supported by the MIC of Korea, under the ITRC support program (IITA-2006-C1090-0603-0026) and BK21.

need not consider the multi-user security in the public key encryption and signature schemes. 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 (e.g. identity-based encryption schemes) where the single-user security is not enough.

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.

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" $\sigma'$ to Bob. Bob verifies the validity of the partial signature and fulfills his obligation, after which Alice sends her "full signature" $\sigma$ 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 $\sigma$ at the end, Bob will send $\sigma'$ (and proof of fulfilling his obligation) to the arbitrator who will convert $\sigma'$ into $\sigma$, sending $\sigma$ 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 [9], 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. [8] 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. [24], which was broken and repaired by Dodis and Reyzin [14]. While the schemes in [14] are very efficient, one important drawback of the approach based on the sequential two-party multisignature is that it is *setup-driven* [32]; 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 arbitrator. A more generalized and unified model for non-interactive optimistic fair exchange was suggested by Dodis and Reyzin [14]. 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 [14] is satisfactory. Otherwise, we should extend the 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 [14]. While the single-user model of [14] is setup-driven, our multi-user model is *setup-free* [32], 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 [32].

## 2 Preliminaries

### 2.1 NP-Relations and $\Sigma$-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 $\alpha$ for which there exists $\beta$ such that $(\alpha, \beta) \in R$, i.e., $\mathcal{L}_R = \{\alpha \mid \exists \beta \, [(\alpha, \beta) \in R]\}$.

A $\Sigma$-protocol [12] 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

$\alpha$, a valid **NP**-witness $\beta$ for $\alpha$, 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 $\alpha$ and the transcript $\pi = \{c, e, s\}$. The transcript $\pi$ is valid if the verifier outputs 1 (accept).

A $\Sigma$-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 $\beta$ for $\alpha$, 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 $\pi_1, \pi_2$ with the same commitment message $c$ outputs $\beta$ 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 $\beta$ for $\alpha$ and the verifier.

A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ 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 $\mathsf{td}$ such that the inverse of $f$ can efficiently be computed with $\mathsf{td}$. For simplicity, we let $f^{-1}$ be an inverse algorithm of $f$ with the trapdoor $\mathsf{td}$. It is known that any language in **NP** has a $\Sigma$-protocol if one-way functions exist.

**Theorem 1 ([15, 19]).** *A $\Sigma$-protocol for any* **NP***-relation can be constructed if one-way functions exist.*

While the $\Sigma$-protocol for any **NP**-relation can be constructed in generic ways [15, 19], there exist very efficient $\Sigma$-protocols for specific cases; for example, GQ protocol [22] and Schnorr protocol [31].

A $\Sigma$-protocol can be transformed into a signature scheme by using the Fiat-Shamir heuristic [17]. To sign a message $m$, the legal signer produces a valid transcript $\pi = \{c, e, s\}$ of the $\Sigma$-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 $\Sigma$-protocol is secure in the random oracle model [5, 26]. 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 $\Sigma$-protocols, i.e., $\Sigma_1$ for $R_1$ and $\Sigma_2$ for $R_2$, we can construct another $\Sigma$-protocol $\Sigma_{OR}$ (called OR-proof) [12] 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 indistin-

guishability property [16]). By applying the Fiat-Shamir heuristic to the OR-proof $\Sigma_{OR}$, 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 $\Sigma$-protocol.

### 2.2 Signatures

A signature scheme $\mathcal{S}$ consists of three efficient algorithms: $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. We consider *existential unforgeability under adaptive chosen message attacks*, denoted by $\mathsf{UF\text{-}CMA}$ [21]. The adversary $\mathcal{A}$ is given oracle access to the signing oracle $O_{\mathsf{Sign}}$. Naturally, $\mathcal{A}$ is considered successful only if it forges a valid signature $\sigma$ of a message $m$ which has not been queried to $O_{\mathsf{Sign}}$. 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 $O_{\mathsf{Sign}}$. An adversary $\mathcal{A}$ is said to $(t, q_s, \varepsilon)$-break $\mathcal{S}$, if $\mathcal{A}$ runs in time at most $t$, makes at most $q_s$ signing queries to $O_{\mathsf{Sign}}$, and succeeds in forgery with probability at least $\varepsilon$. $\mathcal{S}$ is said to be $(t, q_s, \varepsilon)$-secure, if no adversary can $(t, q_s, \varepsilon)$-break it. Asymptotically, $\mathcal{S}$ is $\mathsf{UF\text{-}CMA}$-secure if $\mathsf{Adv}^{\mathcal{S}}_{\mathcal{A}}(k)$ is negligible for any PPT adversary $\mathcal{A}$.

### 2.3 Encryption

An encryption scheme $\mathcal{E}$ consists of three algorithms: $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. We consider *indistinguishability against adaptive chosen ciphertext attacks*, denoted by $\mathsf{IND\text{-}CCA}$ [27]. For an efficient algorithm $\mathcal{A}$, which runs in two stages of $\mathsf{find}$ and $\mathsf{guess}$, we define the adversary's advantage $\mathsf{CCA\text{-}Adv}^{\mathcal{E}}_{\mathcal{A}}(k)$ as

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

where the challenge ciphertext $c_b$ should not be queried to the decryption oracle in the $\mathsf{guess}$ stage. An adversary $\mathcal{A}$ is said to $(t, q_d, \varepsilon)$-break $\mathcal{E}$, if $\mathcal{A}$ runs in time at most $t$, makes at most $q_d$ decryption queries to $O_{\mathsf{Dec}}$, and succeeds in distinguishing the challenge ciphertext with advantage at least $\varepsilon$. The encryption scheme $\mathcal{E}$ is said to be $(t, q_d, \varepsilon)$-secure, if no adversary can $(t, q_d, \varepsilon)$-break it. Asymptotically, $\mathcal{E}$ is $\mathsf{CCA}$-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 [14].

**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:*

- $\mathsf{Setup}$. *This is a registration protocol between Alice and Charlie, by the end of which Alice learns her secret signing key* $\mathsf{SK}$*, Charlie learns his secret arbitration key* $\mathsf{ASK}$*, and they publish Alice's public verification key* $\mathsf{PK}$ *and Charlie's partial verification key* $\mathsf{APK}$.
- $\mathsf{Sig}$ *and* $\mathsf{Ver}$. *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* $\sigma$ *on* $m$*, while* $\mathsf{Ver}(m, \sigma, \mathsf{PK}, \mathsf{APK})$ *— run by Bob (or any verifier) — outputs* $1$ $(accept)$ *or* $0$ $(reject)$.
- $\mathsf{PSig}$ *and* $\mathsf{PVer}$. *These are partial signing and verification algorithms.* $\mathsf{PSig}$ *together with* $\mathsf{Res}$ *is functionally equivalent to* $\mathsf{Sig}$. $\mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK})$ *— run by Alice — outputs a partial signature* $\sigma'$*, while* $\mathsf{PVer}(m, \sigma', \mathsf{PK}, \mathsf{APK})$ *— run by Bob (or any verifier) — outputs* $1$ $(accept)$ *or* $0$ $(reject)$.
- $\mathsf{Res}$. *This is a resolution algorithm run by Charlie in case Alice refuses to open her signature* $\sigma$ *to Bob, who in turn possesses a valid partial signature* $\sigma'$ *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* $\sigma$ *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 $\mathsf{Res}$ to produce a full signature $\sigma$ from $\sigma'$ only if Bob's obligation to Alice has been fulfilled. 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 $O_{\mathsf{PSig}}$ an oracle simulating the partial signing procedure $\mathsf{PSig}$, and by $O_{\mathsf{Res}}$ an oracle simulating the resolution procedure $\mathsf{Res}$.

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

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

where $\mathsf{Setup}^*$ denotes the run of $\mathsf{Setup}$ with dishonest Alice (run by $A$) and $\mathsf{SK}^*$ is $A$'s state after this run. In other words, Alice should not be able to produce

partial signature $\sigma'$, 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.

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

where $\mathit{Query}(B, O_{\mathsf{Res}})$ is the set of valid queries of $B$ has asked to the resolution oracle $O_{\mathsf{Res}}$ (i.e., $(m, \sigma')$ such that $\mathsf{PVer}(m, \sigma', \mathsf{PK}, \mathsf{APK}) = 1$). In other words, Bob should not be able to complete any partial signature $\sigma'$ that he received from Alice into a complete signature $\sigma$, without explicitly asking Charlie to do so. Note that there is no need to provide $B$ with access to the signing oracle $O_{\mathsf{Sig}}$, since it could be simulated by $O_{\mathsf{PSig}}$ and $O_{\mathsf{Res}}$. Finally, we remark that we also want Bob to be unable to generate a valid partial signature $\sigma'$ which was not produced by Alice (via a query to $O_{\mathsf{PSig}}$). However, this guarantee will follow from a stronger security against Charlie, which is defined below.

SECURITY AGAINST CHARLIE. We require that any PPT adversary $C$ succeeds with at most negligible probability in the following experiment.

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

where $\mathsf{Setup}^*$ denotes the run of $\mathsf{Setup}$ with dishonest Charlie (run by $C$), $\mathsf{ASK}^*$ 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 $O_{\mathsf{PSig}}$. 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 $\mathsf{ASK}$).

### 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.

- $\mathsf{Setup}$. 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.
- $\mathsf{Sig}$ and $\mathsf{Ver}$. 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 $\mathsf{Vrfy}_{vk}(m \| y_A, \delta_A) \stackrel{?}{=} 1$.

– $\mathsf{PSig}$ and $\mathsf{PVer}$. 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 $\mathsf{Vrfy}_{vk}(m\|y_A, \delta_A) \stackrel{?}{=} 1$.
– $\mathsf{Res}$. Given a partial signature $(m, y_A, \delta_A)$, the arbitrator Charlie first verifies its validity by checking $\mathsf{Vrfy}_{vk}(m\|y_A, \delta_A) \stackrel{?}{=} 1$. 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 [14].

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 $(m_A, y_A, \delta_A)$ to Bob, where $y_A = f(r_A)$ and $\delta_A = \mathsf{Sign}_{\mathsf{SK}_A}(m_A\|y_A)$.
2. Bob gives $(m_B, y_B, \delta_B)$ 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 $(m_B, y_B, \delta_B)$ 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, $(m_A, r_A, \delta_A)$, 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. This counterexample entails the following theorem.

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

## 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 [14] to the multi-user setting. Firstly, we separate the $\mathsf{Setup}$ 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 $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i})$.

**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* $\mathsf{ASK}$ *and a public partial verification key* $\mathsf{APK}$.
– $\mathsf{Setup}^{\mathsf{User}}$. *The user setup algorithm takes as input a security parameter and (optionally)* $\mathsf{APK}$. *It returns a private signing key* $\mathsf{SK}$ *and a public verification key* $\mathsf{PK}$.
– $\mathsf{Sig}$ *and* $\mathsf{Ver}$. *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* $\sigma_{U_i}$ *on* $m$, *while* $\mathsf{Ver}(m, \sigma_{U_i}, \mathsf{PK}_{U_i}, \mathsf{APK})$ *— run by a verifier — outputs* $1$ $(accept)$ *or* $0$ $(reject)$.
– $\mathsf{PSig}$ *and* $\mathsf{PVer}$. *These are partial signing and verification algorithms.* $\mathsf{PSig}$ *together with* $\mathsf{Res}$ *is functionally equivalent to* $\mathsf{Sig}$. $\mathsf{PSig}(m, \mathsf{SK}_{U_i}, \mathsf{APK})$ *— run by a signer* $U_i$ *— outputs a partial signature* $\sigma'_{U_i}$, *while* $\mathsf{PVer}(m, \sigma'_{U_i}, \mathsf{PK}_{U_i}, \mathsf{APK})$ *— run by a verifier — outputs* $1$ $(accept)$ *or* $0$ $(reject)$.
– $\mathsf{Res}$. *This is a resolution algorithm run by the arbitrator in case a signer* $U_i$ *refuses to open his signature* $\sigma_{U_i}$ *to a user* $U_j$, *who possesses a valid partial signature* $\sigma'_{U_i}$ *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* $\sigma_{U_i}$ *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 $\kappa$ inside the message, as suggested in [3].

In order to consider the collusion attack of dishonest users, we modify the resolution oracle $O_{\mathsf{Res}}$. In the single-user setting, the input to $O_{\mathsf{Res}}$ is $(m, \sigma')$, assuming that $\sigma'$ is the partial signature value of the single signer Alice and the oracle checks the validity of $\sigma'$ by using Alice's public key. In the multi-user setting, we define the input to $O_{\mathsf{Res}}$ as $(m, \sigma', \mathsf{PK}_{U_i})$ where $\mathsf{PK}_{U_i}$ 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. Therefore, the dishonest verifier or the dishonest arbitrator has access to private keys of all users except Alice, and the partial signing oracle $O_{\mathsf{PSig}}$, taking as input a message $m$, always returns Alice's partial signature $\sigma'_A$ 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.

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

In the single-user setting, the signer Alice wins if she comes up with a partial signature $(m, \sigma')$ 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 $\sigma'$ is a valid partial signature with respect to $\mathsf{PK}_A$ but cannot be completed to the full signature (w.r.t. $\mathsf{PK}_A$) 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.

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

where $\mathit{Query}(B, O_{\mathsf{Res}})$ is the set of valid queries of $B$ has asked to the resolution oracle $O_{\mathsf{Res}}$ (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 $O_{\mathsf{Res}}$ as long as $\mathsf{PK}_{U_i}$ 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 $O_{\mathsf{Sig}}$, since it can be simulated by $O_{\mathsf{PSig}}$ and $O_{\mathsf{Res}}$.

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

$$\mathsf{Setup}^{\mathsf{TTP}^*}(1^k) \rightarrow (\mathsf{ASK}^*, \mathsf{APK})$$

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

where $\mathsf{Setup}^{\mathsf{TTP}*}$ denotes the run of $\mathsf{Setup}^{\mathsf{TTP}}$ with the dishonest arbitrator (run by $C$), $\mathsf{ASK}^*$ 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 $O_{\mathsf{PSig}}$.

### 4.2 Generic Construction

We present a generic construction of non-interactive setup-free[i] 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 $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}) = (sk_i, vk_i)$.
- $\mathsf{Sig}$. 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 $\sigma_{U_i} = (s_1, s_2)$.
- $\mathsf{Ver}$. 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 $\mathsf{Sign}_{sk}(1||m)$.
- $\mathsf{PSig}$ and $\mathsf{PVer}$. The same as $\mathsf{Sig}$ and $\mathsf{Ver}$ except that the partial signature $\sigma'_{U_i}$ on $m$ is $s_1$.
- $\mathsf{Res}$. 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 $s_2$.

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

[i] If we allow the registration between the signer and the arbitrator, there are trivial setup-driven solutions.

*Proof.* See [13].

**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.*

*Proof.* Secure signatures exist if and only if one-way functions exist [23, 28]. 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 [25] instead of the Fiat-Shamir proof. Like the Fiat-Shamir heuristic, the construction of the straight-line extractable witness indistinguishable proof starts with the $\Sigma$-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).[ii] The construction of non-interactive witness indistinguishable proofs of knowledge requires the existence of trapdoor permutations [30] and this observation leads to the following theorem.

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

*Remark 1.* 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 $\Sigma$-protocols for the knowledge of a signature value and for the knowledge of a secret key corresponding to a given public key (e.g., [22, 31, 10, 11, 7]).

## 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. 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.

[ii] 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.

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.[iii]

- $\mathsf{Setup}^{\mathsf{TTP}}$. The arbitrator chooses $(dk, ek)$ by running $\mathsf{Enc\text{-}Gen}(1^k)$ and sets $(\mathsf{ASK}, \mathsf{APK}) = (dk, ek)$.
- $\mathsf{Setup}^{\mathsf{User}}$. Each user $U_i$ chooses $(sk_i, vk_i)$ by running $\mathsf{Sig\text{-}Gen}(1^k)$ and sets $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}) = (sk_i, vk_i)$.
- $\mathsf{Sig}$. 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 $\sigma_{U_i} = s$.
- $\mathsf{Ver}$. To verify the signature $\sigma_{U_i} = s$ of $m$, a verifier checks $\mathsf{Vrfy}_{vk_i}(m, s) \stackrel{?}{=} 1$.
- $\mathsf{PSig}$. 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 $\mathsf{APK}$, i.e., $c = \mathsf{Enc}_{ek}(s)$. The partial signature of $m$ is $\sigma'_{U_i} = (c, \pi)$, where $\pi$ is a proof showing $(c, m, ek, vk_i) \in L$.
- $\mathsf{PVer}$. To verify the partial signature $\sigma'_{U_i} = (c, \pi)$ of $m$, a verifier checks that $\pi$ is an accepting proof for the statement $(c, m, ek, vk_i) \in L$. If so, 1 is returned and otherwise, 0 is returned.
- $\mathsf{Res}$. For the user $U_i$'s partial signature $\sigma'_{U_i} = (c, \pi)$ of $m$, the arbitrator first checks that $\pi$ 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 $\sigma_{U_i} = s$.

**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 [13].

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. 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 [29, 23, 28]. Hence, we obtain the following existence theorem of setup-free and stand-alone fair exchange schemes.

**Theorem 7.** *If there are trapdoor 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

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

signing. To construct an optimistic fair exchange, we can use a simple type of multisignature, which is called a sequential two-party multisignature.

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 $\mathsf{Sign}$, and verification algorithm $\mathsf{Vrfy}$ are similar to the conventional algorithms of an ordinary signature scheme. $\mathsf{MSign}$ 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 $vk_j$. $\mathsf{MVrfy}$ 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.

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 $O_{\mathsf{Sign}}$ and the multi-signing oracle $O_{\mathsf{MSign}}$. $\mathcal{A}$'s query to $O_{\mathsf{Sign}}$ is $(m, vk_i)$ and $O_{\mathsf{Sign}}$ returns $\mathsf{Sign}_{sk_i}(m)$. $\mathcal{A}$'s query to $O_{\mathsf{MSign}}$ is $(m, s_i, vk_i, vk_j)$ and $O_{\mathsf{MSign}}$ 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 [6] 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 $O_{\mathsf{Sign}}$ and $O_{\mathsf{MSign}}$, respectively. We define $\mathcal{A}$'s advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathcal{MS}}(k)$ of attacking $\mathcal{MS}$ as follows.

$$\Pr[\mathsf{MVrfy}(m, s, vk_i, vk_j) = 1 \vee \mathsf{MVrfy}(m, s, vk_j, vk_i) = 1 \mid (sk_i, vk_i) \leftarrow \mathsf{Sig\text{-}Gen}(1^k), (m, s, vk_j) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}, O_{\mathsf{MSign}}}(vk_i)]$$

**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 $O_{\mathsf{Sign}}$ and $q_{ms}$ multi-signing queries to $O_{\mathsf{MSign}}$, and succeeds in forgery with probability at least $\varepsilon$. $\mathcal{MS}$ 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 $\mathsf{UF\text{-}CMA}$-secure if $\mathsf{Adv}_{\mathcal{A}}^{\mathcal{MS}}(k)$ is negligible for any PPT adversary $\mathcal{A}$.*

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 $\mathsf{SK}_{U_i}$, $\mathsf{PK}_{U_i}$, $\mathsf{ASK}_{U_i}$, $\mathsf{APK}_{U_i}$ and sends $\mathsf{PK}_{U_i}$, $\mathsf{ASK}_{U_i}$, $\mathsf{APK}_{U_i}$ to the arbitrator, who checks if the keys were properly generated. The arbitrator will then store $\mathsf{ASK}_{U_i}$ and certify $\mathsf{APK}_{U_i}$. A verifier will accept partial signatures from $U_i$ only if they are valid w.r.t. $\mathsf{APK}_{U_i}$.

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 $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}, \mathsf{ASK}_{U_i}, \mathsf{APK}_{U_i}) = ((sk_{i_0}, sk_{i_1}), (vk_{i_0}, vk_{i_1}), sk_{i_1}, vk_{i_0})$.
- $\mathsf{Sig}$. 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 $\sigma_{U_i} = s_{i_0 i_1}$.
- $\mathsf{Ver}$. A verifier checks $\mathsf{MVrfy}(m, s_{i_0 i_1}, vk_{i_0}, vk_{i_1}) \stackrel{?}{=} 1$.
- $\mathsf{PSig}$. When a user $U_i$ wants to generate a partial signature of a message $m$, the signer computes $s_{i_0} = \mathsf{Sign}_{sk_{i_0}}(m)$. The partial signature is $\sigma'_{U_i} = s_{i_0}$.
- $\mathsf{PVer}$. To verify the partial signature $\sigma'_{U_i} = s_{i_0}$ of $m$ w.r.t. $\mathsf{PK}_{U_i}$, a verifier checks $\mathsf{Vrfy}_{vk_{i_0}}(m, s_{i_0}) \stackrel{?}{=} 1$. If so, 1 is returned and otherwise, 0 is returned.
- $\mathsf{Res}$. For the user $U_i$'s partial signature $\sigma'_{U_i} = s_{i_0}$ of $m$, the arbitrator first checks $\mathsf{Vrfy}_{vk_{i_0}}(m, s_{i_0}) \stackrel{?}{=} 1$ 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 2.* 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 [13].

## References


1. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. *ACM CCS*, pages 7–17. ACM, 1997.
2. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). *EUROCRYPT 1998*, pages 591–606, 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. *EUROCRYPT 2000*, pages 259–274.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *ACM CCS*, pages 62–73, 1993.
6. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. *PKC 2003*, pages 31–46, 2003.
7. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. *CRYPTO 2004*, pages 41–55, 2004.
8. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *EUROCRYPT 2003*, pages 416–432, 2003.

9. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. *ASIACRYPT 2000*, pages 331–345, 2000.
10. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, pages 268–289, 2002.
11. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. *CRYPTO 2004*, pages 56–72, 2004.
12. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *CRYPTO 1994*, pages 174–187, 1994.
13. Y. Dodis, P.J. Lee, and D.H. Yum. Optimistic fair exchagne in a multi-user setting. *IACR ePrint Archive*, http://eprint.iacr.org/, 2007.
14. Y. Dodis and L. Reyzin. Breaking and repairing optimistic fair exchange from PODC 2003. *2003 ACM Workshop on Digital Rights Management*, pages 47–54.
15. U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. *CRYPTO 1989*, pages 526–544, 1989.
16. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. *the 22nd STOC*, pages 416–426. ACM, 1990.
17. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *CRYPTO 1986*, pages 186–194, 1986.
18. 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.
19. 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.
20. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
21. 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.
22. L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. *CRYPTO 1988*, pages 216–231, 1988.
23. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. *the 21st STOC*, pages 33–43. ACM, 1989.
24. J. M. Park, E. K. P. Chong, and H. J. Siegel. Constructing fair-exchange protocols for e-commerce via distributed computation of RSA signatures. *PODC 2003*, pages 172–181. ACM, 2003.
25. R. Pass. On deniability in the common reference string and random oracle model. *CRYPTO 2003*, pages 316–337, 2003.
26. D. Pointcheval and J. Stern. Security proofs for signature schemes. *EUROCRYPT 1996*, pages 387–398, 1996.
27. C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *CRYPTO 1991*, pages 433–444, 1991.
28. J. Rompel. One-way functions are necessary and sufficient for secure signatures. *the 22nd STOC*, pages 387–394. ACM, 1990.
29. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. *the 40th FOCS*, pages 543–553. IEEE, 1999.
30. A. D. Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. *the 33rd FOCS*, pages 427–436. IEEE, 1992.
31. C.-P. Schnorr. Efficient identification and signatures for smart cards. *CRYPTO 1989*, pages 239–252, 1989.
32. H. Zhu and F. Bao. Stand-alone and setup-free verifiably committed signatures. *CT-RSA 2006*, pages 159–173, 2006.