Generic Construction of Fair Exchange Scheme with Semi-Trusted Adjudicator

Yang Wang, Willy Susilo, Joonsang Baek, Jongkil Kim, and Intae Kim*
University of Wollongong, Wollongong, New South Wales 2522, Australia
{ywang, wsusilo, baek, jongkil, intaekim}@uow.edu.au

Received: October 14, 2019; Accepted: December 4, 2019; Published: December 31, 2019

Abstract

A fair exchange of digital signatures has been considered as a fundamental problem in cryptography. The most widely adopted approach to ensure the fairness of exchanging signatures is to resort to a trusted third party (TTP) whenever required. The TTP is assumed to be entirely honest and neutral, hence never colludes with the other parties in the system. In practice, such a TTP may not be available. To overcome this difficulty, Shao introduced a new idea of building fair exchange schemes with a semi-trusted adjudicator that only needs to be trusted by one party (the signer). These new fair exchange schemes are more practical than previous ones since there is no necessity for two mutually distrusted parties to commonly trust the same adjudicator. In the new approach, an adjudicator is also trusted unilaterally. Nevertheless, the two schemes that Shao proposed in 2008 and 2010, respectively, are both only provably secure in the random oracle model. In this paper, we revisited the schemes proposed by Shao and revealed that some subtle issues might have been overlooked. In particular, the number of interactions between the signer and the adjudicator is more involved than it is anticipated. Our investigation leads to a refined definition of this kind of fair exchange scheme with a semi-trusted adjudicator (FESTA). We proposed a generic construction in our model based on pseudo-random functions, collision-resistant hash functions and any digital signature schemes that are existentially unforgeable against adaptive chosen message attacks. Based on our generic construction, FESTA in the standard model can be realized. We provide such instantiations to demonstrate the practicability of our generic construction.

Keywords: fair exchange, digital signature, semi-trusted adjudicator

1 Introduction

With the widely use of open networks such as the Internet, online businesses such as electronic commerce are growing rapidly. Under normal circumstances, the networks where the exchanges of items take place are insecure and the participants may not trust each other. Furthermore, there is no assurance that a digital item will eventually be delivered to the intended recipient. Even if it has been delivered properly, the recipient could claim otherwise. There may be a dispute during the exchange even if both participants act honestly. Thus fair exchanging, *i.e.*, how to make two mutually distrustful participants exchange digital items over open computer networks in a fair way, has been considered as a fundamental problem in electronic transactions. A digital exchange is said to be 'fair' if at the end of exchange, either party receives the other's item or neither party does.

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 10(4):68-87, Dec. 2019
DOI:*10.22667/JOWUA.2019.12.31.068*

*Corresponding author: Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, NSW 2522, Australia, Tel.+61-0242213491

In recent years, the issue of fair exchange of digital items has been extensively addressed and various schemes have been proposed. There were essentially three different approaches to solve the fair exchange problem:

Early solutions were provided by gradual exchange protocols [10, 9, 6] where two parties exchange their commitments in turns "little-by-little". However, these solutions may not provide true fairness, because one party, say B, often has an advantage of (at least) one bit than the other party A does. If B has a stronger computational power, it may terminate the protocol prematurely and compute the remaining parts of A's commitments, while A is not able to do so. Even if both parties have equal computational power, such protocols are normally too interactive and cumbersome with many message flows for some applications.

A new cryptographic primitive called concurrent signatures was introduced by Chen, Kudla and Paterson [7], and it provides a new approach to solve the fair exchange problem. Following the seminal work by Chen et al., there are many subsequent concurrent signature schemes that have been proposed in the literature [18, 20, 15, 19]. Such schemes enables two parties to produce two ambiguous signatures that do not bind to their true signers. Before a vital information called the keystone is released, any third party cannot identify the true signer of the signature. However, after the release of the keystone, both signatures will bind concurrently. Concurrent signature enables building a fair exchange protocol without the involvement of a trusted third party. Nevertheless, the keystone is possessed by the initial party and therefore it has an advantage on deciding when to release the keystone or even whether to release it or not. The initial party might privately show the other party's signature together with the unreleased keystone to outsiders.

The most widely-used approach to solve the fair exchange problem is to resort to a trusted third party (TTP) as an adjudicator, who can be called upon to handle possible disputes between the involved parties [1, 2, 4]. The adjudicator can always resolve the dispute in the case of a network failure between the two involved parties or that one party attempts to cheat. However, the fairness of signature exchange protocols with TTP is only guaranteed when the adjudicator is totally honest and neutral. If the adjudicator colludes with one party, the other would be duped. Hence, an adjudicator that is trusted by both parties must be selected prior to the exchange. The assumption that there always exists such a totally honest and neutral third party over the open networks is somewhat unrealistic since it is difficult for mutually distrustful parties to choose a common third party that they both trust, especially when prevailing information asymmetry exists between two parties from different regions.

To deal with the knotty problem for the excessive reliance on the adjudicator in fair exchange protocols, recently, Shao introduced a new idea of building fair exchange schemes with a semi-trusted adjudicator that only needs to be trusted by one party (the signer) [16, 17]. The two schemes shall be referred to as Shao08 and Shao10 hereafter. Such schemes enforce the adjudicator to sign a voucher (*affidavit*), which stipulates the obligation that the verifier must fulfil to exchange the signer's signature and explicitly promises that if the signer fails to send its signature to the verifier after the verifier has fulfilled its obligation, the adjudicator would do it. This signature from the adjudicator is called a voucher signature. The signer then combines the voucher signature with his own signature, and sends the combined signature to the verifier. This combined signature assures the verifier that the signer has created a signature on the message and the adjudicator has created a signature on the voucher. In case the signer refuses to send his/her own signature to the verifier after the latter has fulfilled the obligation, the adjudicator could extract the signer's signature from the combined signature. The combined signature would also serve as a proof of knavery in case the adjudicator refuses to extract the signer's signature. Consequently, this ensures more accountability of the adjudicator and thus prevents the adjudicator from colluding with the signer against the verifier. The trust placed on the adjudicator from the verifier is thus greatly reduced and this new kind of fair exchange schemes are more practical than previous ones because compared with the case that two mutually distrusted parties have to choose a common adjudicator that they both

trust, it is obviously much easier for the signer to choose a adjudicator unilaterally. In Shao08, aggregate signature was employed for the purpose of combining the signatures while a new primitive called double signature was introduced in Shao10 to achieve the same purpose.

We take a deep look in the schemes proposed by Shao and reveal that some issues are possibly being overlooked. For instance, we believe the original security model of fair exchange should be modified to cope with the introduction of the voucher signature from the adjudicator. In the previous model, a passive attacker can distinguish the signature created by the signer from the signature outputted from the adjudicator as a result of adjudication in Shao08 as opposed by the claim that the two cases should be indistinguishable. Looking ahead, the attack is possible due to the fact that only a weakly secure encryption is used to protect the communication between the signer and the adjudicator. The information leaked from the ciphertext is sufficient for the attacker to launch such an attack. We would like to stress that this attack is outside the security model of normal employed in Shao08. In the model of OFE, the adversary is not given the capability to observe the interactions between the signer and the adjudicator. This is rightfully so as there is no interaction between the signer and adjudicator in ordinary OFE. However, this is not the case in Shao08/Shao10 and this reinforces our belief that an updated model is needed.

Our Contributions. In this paper, we make the following contributions. First, we analyze Shao's scheme in detail and propose a formal definition of a fair exchange scheme with semi-trusted adjudicator (FESTA) which covers the multi-user setting. Second, we propose a generic construction of a FESTA, and provide the security proof of our scheme in the proposed model. From the generic construction, we can construct a FESTA directly from any signature scheme that is existential unforgeable against adaptive chosen message attacks (EUA-CMA), which make it feasible to construct an efficient FESTA in the standard model. We should stress that we do not aim to compare FESTA with the notion of Optimistic Fair Exchange protocols in the literature. While the latter is more widely accepted in the community, our goal in this paper is to provide a formalization of FESTA as well as providing a generic construction that can be instantiated to a secure scheme in the standard model.

2 Preliminaries

2.1 Shao08/Shao10 and Their Implications

We discuss the characteristics exhibited in Shao's schemes. The major difference betweens Shao's schemes and ordinary OFE is an extra round of interaction between the signer and the adjudicator. During this interaction, the signer obtains a signature from the adjudicator on a voucher V. This signature is called a voucher signature, denoted as σ_V. Assume a verifier agrees to fulfill a certain obligation in exchange for the signer's signature on a certain message M. In the exchange protocol, the signer transmits a partial signature to the verifier. The partial signature, denoted as σ_P, is a composition of σ_V and the signer's signature on message M, denoted as σ_M. This composition is an aggregate signature in Shao08 and a double signature in Shao10. Note that this composition exhibits the following properties:

1. Given σ_P, M, V, anyone can verify σ_P is a composition of two valid signatures on M and V respectively.

2. Given σ_P, σ_V, M, V, anyone can compute σ_M which is a valid signature on message M from the signer.

3. Given σ_P, σ_M, M, V, anyone can compute σ_V which is a valid signature on voucher V from the adjudicator.

In case the signer runs away after the verifier has fulfilled his/her obligations, the verifier can still obtain σ_M from the adjudicator since he/she will only fulfill the obligation upon receiving σ_P, M and V from the signer. The adjudicator can use the copy of σ_V to compute and returns σ_M. Note that property 1 and 2 are necessary in the construction of Shao08 and Shao10. One final remark is that not all aggregate signature schemes possess the above properties and consequently it is non-trivial to create FESTA. We discuss the implicated requirement on the definition of FESTA.

Adjudicator/Signer Interaction. In FESTA, signer obtains the voucher signature from the adjudicator for an voucher signature. Implicitly the definition of FESTA should capture this fact. Furthermore, due to the property that given a voucher signature, anyone can compute the signer's signature from the partial signature, the transmission of this voucher signature should be immune from eavesdropper. Shao08 employs a simple encryption mechanism and believed that the transmission could be done in the public channel. We show that this may introduce new vulnerabilities as the encryption employed in Shao08 is only one-way secure. Shao08 claimed that the signature created by the signer is indistinguishable to the signature outputted by the adjudicator through adjudication. It turns out that this claim is not true if attacker is able to observe the communication between the adjudicator and the signer. We illustrate this attack as follows. We would like to stress again that this is outside the model employed by Shao08.

- The attacker observes (S,T) which is the well-known El-Gamal encryption of the voucher signature σ_V under the signer's public key Y_A.
- The attacker observes (W,M,V) which is the partial signature transmitted to the verifier.
- Given any signature σ_M, M, the attacker could test if it is the output of the signer or the adjudicator by testing if the following equation holds:

$$e(P,T-(W-\sigma_M)) \stackrel{?}{=} e(S,Y_A)$$

Note that invisibility is not necessarily a compulsory feature of FESTA. Nonetheless, this attack demonstrates that in the definition of security model of FESTA, we should either provide the capability of observing interactions between the signer and the adjudicator, or assume that these two parties share a secure communication channel. Looking ahead, we choose the later approach between both parties in the system are equipped with a public key. It is relatively straightforward for them to setup a secure channel using any authenticated key exchange protocol. Furthermore, these two parties are going to communicate constantly and thus we further assume they shared a common key, which can be easily achievable after their first key exchange. This decision would certainly simplify the model.

Re-use of Voucher Signature. Both Shao08 and Shao10 claimed that the signer can re-use the same voucher signature to create multiple partial signatures as long as no adjudication request is made. Recall that given a voucher signature, anyone can recover the signer's signature from the partial signature. Thus, if the voucher signature is to be re-used in the creation of multiple signature, it is necessary that the voucher signature should be kept confidential. In this sense property 3 listed above is undesirable, since after completion of the protocol, the verifier can compute the value of the voucher signature based on the signer's signature together with the partial signature.

To prevent this, the signature given to the verifier after her/she fulfills his/her obligation is different from the signature that is used to create the partial signature. As the signature from the signer is *deterministic* in Shao08, a rather abnormal procedure is employed. That is, the signer has to create a

signature on another message M', having the same meaning as M, to be sent to the verifier at the completion of the protocol. Such problem does not appear in Shao10 since the underlying signature scheme is *probabilistic*, meaning that for a particular message M, there are polynomially-many valid signatures.

In addition, immediately after any adjudication request, the adjudicator has to inform the signer of such request and issue a new voucher signature. The reason is that as a result of this adjudication, the verifier obtained a voucher signature due property 3. This communication between the adjudicator and the signer is necessary since it prevents the signer from creating any partial signature with a voucher that is known to an outsider. One subtlety that is being overlooked arises here: how can the adjudicator contact the signer before the signer issues any new partial signature? Does it imply the signer would have to contact the adjudicator to check if a new voucher signature is to be issued? For the moment we shall put aside this issue and assume it is possible.

The above two measures are still insufficient to guarantee re-using a voucher signature is secure. Thus, another precautionary measure is also introduced. If a verifier fail to fulfill his/her obligation within a certain time period, the signer will abort, meaning that he/she will contact the adjudicator to void the previous voucher signature and have a new one issued. At the same time, adjudicator is required to respond only to adjudication request with respect to the most updated voucher. This assures the signer that at any given time, no one will be in possession of more than one partial signatures created using the same voucher signature of which no obligation has been fulfilled. This avoid the situation that the signer has created multiple partial signatures using the same voucher and is waiting for the respective verifiers to fulfill their obligations. Otherwise, if one malicious verifier turns to the adjudicator for adjudication, he/she can compute the voucher signature using property 3. With that voucher signature, this malicious verifier can obtain the signer's signature from other partial signatures without fulfilling the obligation.

However, this actually introduces a loophole for a malicious signer to cheat. Since the signer is in possession of the previous version of the voucher signature, he is not bounded to use the most up-to-date voucher signature in the creation of the partial signature. The verifier cannot tell if the signer is cheating in this way. The adjudicator will refuse the adjudication request made by this verifier since it has to reject requests with partial signature that is based on previous voucher signatures. Thus, the malicious signer can safely run away after getting the verifier's fulfilling obligation, knowing that his/her signature will not be revealed by the adjudicator. To make the matter worse, the verifier cannot convince the adjudicator that this signer is malicious because from the point of view of the adjudicator it is equally likely that this verifier is malicious and is using a partial signature from previous transactions.

From this analysis, it can be seen that the voucher signature re-use is not possible in the existing constructions. Consequently, we employ a less restrictive definition of FESTA in which the signer can make multiple interactions with the adjudicator so that the definition is general enough to cover existing constructions.

3 Definitions

We provide a formal definition of FESTA together with a suitable security model.

Definition 1. *A fair exchange scheme with semi-trusted adjudicator* (FESTA) *involves two users (a signer and a verifier) and an adjudicator, and is formalized using the following algorithms:*

- ***AdjKeyGen:*** On input 1^k, it generates a secret adjudication key ASK, and a public partial verification key APK.
- ***UserKeyGen:*** On input 1^k and (optionally) APK, it outputs a secret/public key pair $(\mathsf{SK}, \mathsf{PK})$. For a user U_i, we use $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i})$ to denote the user's key pair.

- ***VSigCreate:*** Similar to the signing algorithm of an ordinary digital signature scheme, ***VSigCreate*** $(V, \mathsf{ASK}, \mathsf{PK}_{U_i})$ outputs a voucher signature σ_A, where V is a voucher (affidavit).
- ***VSigVerify:*** Similar to the verification algorithm of an ordinary digital signature scheme, ***VSigVerify*** $(V, \sigma_A, \mathsf{PK}_{U_i}, \mathsf{APK})$ outputs $\top$ (accept) or $\perp$ (reject).
- ***PSigCreate:*** This is the partial signing algorithm. ***PSigCreate***$(M, V, \sigma_A, \mathsf{SK}_{U_i}, \mathsf{APK})$ outputs a partial signature σ'_{U_i}, where M is a message.
- ***PSigVerify:*** This is the partial verification algorithm. ***PSigVerify***$(M, \sigma'_{U_i}, V, \mathsf{PK}_{U_i}, \mathsf{APK})$ outputs $\top$ or $\perp$.
- ***Sign:*** This is the full signing algorithm. ***Sign***$(M, \sigma'_{U_i}, V, \mathsf{SK}_{U_i}, \mathsf{APK})$ outputs a full signature σ_{U_i}.
- ***Verify:*** This is the full verification algorithm. ***Verify***$(M, \sigma_{U_i}, V, \mathsf{PK}_{U_i}, \mathsf{APK})$ outputs $\top$ or $\perp$.
- ***Adjudication:*** This is the adjudication algorithm. ***Adjudication***$(M, \sigma'_{U_i}, V, \mathsf{ASK}, \mathsf{PK}_{U_i})$ outputs a full signature σ_{U_i}, or $\perp$ indicating the failure of resolving a partial signature.

For correctness property, we require that for all security parameters $k \in \mathbb{N}$, for any $(\mathsf{ASK}, \mathsf{APK}) \leftarrow$ **AdjKeyGen**(1^k), $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}) \leftarrow$ **UserKeyGen**(1^k), we have

- **VSigVerify**$(V,$ **VSigCreate**$(V, \mathsf{ASK}, \mathsf{PK}_{U_i}), \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$,
- **PSigVerify**$(M,$ **PSigCreate**$(M, V,$ **VSigCreate**$(V, \mathsf{ASK}, \mathsf{PK}_{U_i}), \mathsf{SK}_{U_i}, \mathsf{APK}), V, \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$,
- **Verify**$(M,$ **Sign**$(M,$ **PSigCreate**$(M, V,$ **VSigCreate**$(V, \mathsf{ASK}, \mathsf{PK}_{U_i}), \mathsf{SK}_{U_i},$ $\mathsf{APK}), V, \mathsf{SK}_{U_i}, \mathsf{APK}), V, \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$, and
- **Verify**$(M,$ **Adjudication**$(M,$ **PSigCreate**$(M, V,$ **VSigCreate**$(V, \mathsf{ASK}, \mathsf{PK}_{U_i}),$ $\mathsf{SK}_{U_i}, \mathsf{APK}), V, \mathsf{ASK}, \mathsf{PK}_{U_i}), V, \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$.

Ambiguity property requires that any "resolved signature" **Adjudication**$(M,$ **PSigCreate**$(M, V,$ **VSigCreate** $(V, \mathsf{ASK}, \mathsf{PK}_{U_i}), \mathsf{SK}_{U_i}, \mathsf{APK}), V, \mathsf{ASK}, \mathsf{PK}_{U_i})$ is *computationally indistinguishable* from an "actual signature" **Sign**$(M,$ **PSigCreate**$(M, V,$ **VSigCreate**$(V, \mathsf{ASK}, \mathsf{PK}_{U_i})$ $\mathsf{SK}_{U_i}, \mathsf{APK}), V, \mathsf{SK}_{U_i}, \mathsf{APK})$.

4 Security Model

Before presenting the security model, we first define the oracles that the adversaries could have access to while conducting an experiment.

Definition 2. *The oracles that the adversaries could have access to about a* FESTA *are as follows:*

- $O_{UserCreate}$: This is user's public key oracle. Taking as input 1^k, it outputs a public key PK_{U_i} for user U_i, where $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}) \leftarrow$ ***UserKeyGen***(1^k).
- $O_{VSigCreate}$: This is the voucher signing oracle. Taking as input (V, PK_{U_i}) where V is a voucher(affidavit) and PK_{U_i} is a user's public key, it outputs a voucher signature $\sigma_A \leftarrow$ ***VSigCreate*** $(V, \mathsf{ASK}, \mathsf{PK}_{U_i})$.
- $O_{PSigCreateA}$: This is the partial signing oracle for the adjudicator. Taking as input $(M, V, \sigma_A, \mathsf{PK}_{U_i})$ where M is a message and PK_{U_i} is a user's public key, it outputs a partial signature $\sigma'_{U_i} \leftarrow$ ***PSigCreate***$(M, V, \sigma_A, \mathsf{SK}_{U_i}, \mathsf{APK})$.

- $O_{PSigCreateV}$: This is the partial signing oracle for verifiers. Taking as input $(M, V, \mathsf{PK}_{U_i})$ where M is a message and PK_{U_i} is a user's public key, it first acquires a voucher signature $\sigma_A \leftarrow O_{VSigCreate}(V, \mathsf{PK}_{U_i})$, and then outputs a partial signature $\sigma'_{U_i} \leftarrow \textbf{\textit{PSigCreate}}(M, V, \sigma_A, \mathsf{SK}_{U_i}, \mathsf{APK})$.
- O_{Adj} : This is the adjudication oracle. Taking as input $(M, \sigma'_{U_i}, V, \mathsf{PK}_{U_i})$, if ***PSigVerify*** $(M, \sigma'_{U_i}, V, \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$, it outputs a full signature $\sigma_{U_i} \leftarrow \textbf{\textit{Adjudication}}(M, \sigma'_{U_i}, V, \mathsf{ASK}, \mathsf{PK}_{U_i})$; otherwise, it outputs $\perp$.
- O_{Sign} : This is the signing oracle. Taking as input $(M, \sigma'_{U_i}, V, \mathsf{PK}_{U_i})$, if ***PSigVerify***$(M, \sigma'_{U_i}, V, \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$, it outputs a full signature $\sigma_{U_i} \leftarrow \textbf{\textit{Sign}}(M, \sigma'_{U_i}, V, \mathsf{SK}_{U_i}, \mathsf{APK})$; otherwise, it outputs $\perp$.

The security of a FESTA consists of four aspects: security against signers (extractability), security against verifiers (opacity), unforgeability against the adjudicator, and unforgeability against signers. The definitions of them in the multi-user and chosen-key model are given as follows:

- **Security against Signers (Extractability)**: Intuitively, we require that no PPT adversary $\mathscr{A}$ should be able to produce a partial signature with non-negligible probability, which looks good to verifiers but cannot be resolved to a full signature by the honest adjudicator. This ensures the fairness for verifiers, that is, if the signer has committed a partial signature, the verifier will always be able to get the full signature of the signer after it has fulfilled its obligation. Formally, we consider the following experiment:
  • **Experiment** `EXT`:

$$\begin{aligned}
\textbf{AdjKeyGen}(1^k) &\rightarrow (\mathsf{ASK}, \mathsf{APK}) \\
(M, \sigma', V, \mathsf{PK}^*) &\leftarrow \mathscr{A}^{O_{UserCreate}, O_{VSigCreate}, O_{Adj}}(\mathsf{APK}) \\
\text{success of } \mathscr{A} &:= [\textbf{PSigVerify}(M, \sigma', V, \mathsf{PK}^*, \mathsf{APK}) = \top \\
&\quad \wedge \textbf{Adjudication}(M, \sigma', V, \mathsf{ASK}, \mathsf{PK}^*) = \perp]
\end{aligned}$$

  In this experiment, the adversary can arbitrarily choose a public key PK^*, and it may not know the corresponding secret key of PK^*. The advantage of $\mathscr{A}$ in the experiment $Adv_{\mathscr{A}}(k)$ is defined to be $\mathscr{A}$'s success probability.

- **Security against Verifiers (Opacity)**: This security notion requires that any PPT adversary $\mathscr{B}$ should not be able to transform a partial signature into a full signature with non-negligible probability if no help has been obtained from the signer or the adjudicator. Formally, we consider the following experiment:
  • **Experiment** `OPA`:

$$\begin{aligned}
\textbf{AdjKeyGen}(1^k) &\rightarrow (\mathsf{ASK}, \mathsf{APK}) \\
\textbf{UserKeyGen}(1^k) &\rightarrow (\mathsf{SK}, \mathsf{PK}) \\
(M, \sigma, V) &\leftarrow \mathscr{B}^{O_{UserCreate}, O_{PSigCreateV}, O_{Sign}, O_{Adj}}(\mathsf{PK}, \mathsf{APK}) \\
\text{success of } \mathscr{B} &:= [\textbf{Verify}(M, \sigma, V, \mathsf{PK}, \mathsf{APK}) = \top \\
&\quad \wedge (M, \cdot, V, \mathsf{PK}) \notin \{Query(\mathscr{B}, O_{Sign}) \cup Query(\mathscr{B}, O_{Adj})\}]
\end{aligned}$$

  where $Query(\mathscr{B}, O_{Sign})$ and $Query(\mathscr{B}, O_{Adj})$ are the sets of queries $\mathscr{B}$ issued to the signing oracle O_{Sign} and the adjudication oracle O_{Adj}, respectively. In the experiment, $\mathscr{B}$ can ask the signer or adjudicator for resolving any partial signature with respect to any public key (adaptively chosen

by $\mathscr{B}$, without the knowledge of the corresponding secret key), with the only limitation described in the experiment. The advantage of $\mathscr{B}$ in the experiment $Adv_{\mathscr{B}}(k)$ is defined to be $\mathscr{B}$'s success probability.

- **Unforgeability against the Adjudicator**: The security notion requires that the adversary $\mathscr{C}$ should not be able to generate with non-negligible probability a partial signature without explicitly asking the signer for generating one. Formally, we consider the following experiment:
  • **Experiment** `UF-ADJ`:

$$\begin{aligned}
\mathbf{UserKeyGen}(1^k) &\rightarrow (\mathsf{SK},\mathsf{PK}) \\
(\mathsf{ASK}^*,\mathsf{APK}) &\leftarrow \mathscr{C}(\mathsf{PK}) \\
(M,\sigma',V) &\leftarrow \mathscr{C}^{O_{UserCreate},O_{PSigCreateA}}(\mathsf{ASK}^*,\mathsf{APK},\mathsf{PK}) \\
\text{success of } \mathscr{C} &:= [\mathbf{PSigVerify}(M,\sigma',V,\mathsf{PK},\mathsf{APK}) = \top \\
&\quad \wedge (M,V,\cdot,\mathsf{PK}) \notin \mathit{Query}(\mathscr{C},O_{PSigCreateA})]
\end{aligned}$$

  where ASK^* is $\mathscr{C}$'s state information, which might not be the corresponding secret key of APK, and $\mathit{Query}(\mathscr{C},O_{PSigCreateA})$ is the set of queries $\mathscr{C}$ issued to the partial signing oracle for the adjudicator $O_{PSigCreateA}$. The advantage of $\mathscr{C}$ in the experiment $Adv_{\mathscr{C}}(k)$ is defined to be $\mathscr{C}$'s success probability.

- **Unforgeability against Signers**: The security notion requires that the adversary $\mathscr{D}$ should not be able to generate with non-negligible probability a partial signature without explicitly asking the adjudicator for generating a voucher signature. Formally, we consider the following experiment:
  • **Experiment** `UF-SIG`:

$$\begin{aligned}
\mathbf{AdjKeyGen}(1^k) &\rightarrow (\mathsf{ASK},\mathsf{APK}) \\
(\mathsf{SK}^*,\mathsf{PK}) &\leftarrow \mathscr{D}(\mathsf{APK}) \\
(M,\sigma',V) &\leftarrow \mathscr{D}^{O_{UserCreate},O_{VSigCreate}}(\mathsf{SK}^*,\mathsf{PK},\mathsf{APK}) \\
\text{success of } \mathscr{D} &:= [\mathbf{PSigVerify}(M,\sigma',V,\mathsf{PK},\mathsf{APK}) = \top \\
&\quad \wedge (V,\mathsf{PK}) \notin \mathit{Query}(\mathscr{D},O_{VSigCreate})]
\end{aligned}$$

  where SK^* is $\mathscr{D}$'s state information, which might not be the corresponding secret key of PK, and $\mathit{Query}(\mathscr{D},O_{VSigCreate})$ is the set of queries $\mathscr{D}$ issued to the voucher signing oracle $O_{VSigCreate}$. The advantage of $\mathscr{D}$ in the experiment $Adv_{\mathscr{D}}(k)$ is defined to be $\mathscr{D}$'s success probability.

Definition 3. *A* FESTA *is said to be* secure in the multi-user setting and chosen-key model *if there is no PPT adversary that wins any of the experiments above with non-negligible advantage.*

5 Generic Construction

In this section, we propose a generic construction of a FESTA. we adopt the definition of pseudo random function (PRF) proposed by [14] without losing generality. The formal defintion of PRF is provided in Appendix C.

5.1 Our Generic Construction of FESTA

Let $\mathbf{SIG_1} = (\mathbf{KG_1}, \mathbf{Sig_1}, \mathbf{Ver_1})$ and $\mathbf{SIG_2} = (\mathbf{KG_2}, \mathbf{Sig_2}, \mathbf{Ver_2})$ be two ordinary signature schemes, respectively. The details of our generic construction of a FESTA are as follows.

- **AdjKeyGen:** The adjudicator runs $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathbf{SIG_1}.\mathbf{KG_1}(1^k)$, and sets $(\mathsf{ASK}, \mathsf{APK}) := (\mathsf{ask}, \mathsf{apk})$.
- **UserKeyGen:** Each user U_i runs $(\mathsf{sk}_{U_i}, \mathsf{pk}_{U_i}) \leftarrow \mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, and sets $(\mathsf{SK}_{U_i}, \mathsf{PK}_{U_i}) := (\mathsf{sk}_{U_i}, \mathsf{pk}_{U_i})$.
- **VSigCreate:** When the user U_i asks the adjudicator to sign a voucher(affidavit) V, which we assume always contains the public key PK_{U_i} of the user U_i, the adjudicator
  1. chooses a random number s, and computes $r = PRF_K(V||s)$ where PRF_K is a pseudo-random function with a seed K, with K being a pre-shared key[1] between the user U_i and the adjudicator.
  2. generates an ordinary signature σ_V on $V||R||s$ using the ordinary signature scheme $\mathbf{SIG_1}$ where $R = G(r)$ and G is a collision-resistant hash function, *i.e.*, $\sigma_V \leftarrow \mathbf{SIG_1}.\mathbf{Sig_1}(\mathsf{ask}, V||R||s)$. The voucher signature on V is set as $\sigma_A := (\sigma_V, s)$.
- **VSigVerify:** On receiving (V, σ_V, s) from the adjudicator, the user U_i
  1. checks whether its public key PK_{U_i} is contained in the voucher V,
  2. computes $R = G(r)$ where $r = PRF_K(V||s)$ and checks whether $\mathbf{SIG_1}.\mathbf{Ver_1}\ (V||R||s, \sigma_V, \mathsf{apk}) = \top$.

  If both hold, it returns $\top$; otherwise, it returns $\perp$.
- **PSigCreate:** On input a message M, the user U_i generates an ordinary signature on $M||V||R||s$ using the ordinary signature scheme $\mathbf{SIG_2}$ *i.e.* $\sigma_M \leftarrow \mathbf{SIG_2}.\mathbf{Sig_2}(\mathsf{sk}_{U_i}, M||V||R||s)$. The partial signature is then set as $\sigma'_{U_i} := (\sigma_M, \sigma_V, R, s)$.
- **PSigVerify:** On input a message M, a partial signature $\sigma'_{U_i} = (\sigma_M, \sigma_V, R, s)$, a voucher V, and a public key PK_{U_i}, the verifier
  1. checks whether the public key PK_{U_i} is contained in the voucher V. If not, it returns $\perp$;
  2. otherwise checks the validity of σ_M and σ_V by running $\mathbf{SIG_2}.\mathbf{Ver_2}(M||V\ ||R||s, \sigma_M, \mathsf{pk}_{U_i})$ and $\mathbf{SIG_1}.\mathbf{Ver_1}(V||R||s, \sigma_V, \mathsf{apk})$ respectively. If both output $\top$, it returns $\top$; otherwise, it returns $\perp$.
- **Sign:** On input a message M, a partial signature $\sigma'_{U_i} = (\sigma_M, \sigma_V, R, s)$, a voucher V and a public key PK_{U_i}, the user U_i
  1. checks whether $\mathbf{PSigVerify}(M, \sigma'_{U_i}, V, \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$,
  2. checks whether $R = G(r)$ where $r = PRF_K(V||s)$.

  If both hold, the signer returns $\sigma_{U_i} := (\sigma_M, \sigma_V, r, s)$ as a full signature; otherwise, it returns $\perp$.
- **Verify:** On input a message M, a full signature $\sigma_{U_i} = (\sigma_M, \sigma_V, r, s)$, a voucher V and a public key PK_{U_i}, the verifier computes $G(r)$ and checks whether $\mathbf{PSigVerify}(M, \sigma'_{U_i}, V, \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$ where $\sigma'_{U_i} = (\sigma_M, \sigma_V, G(r), s)$. If so, it returns $\top$; otherwise, it returns $\perp$.

[1] We note that this pre-shared key can be generated without extra-interaction based on any secure key exchange protocol with respect to the public keys PK_{U_i} and APK.

- **Adjudication:** On input a message M, a partial signature $\sigma'_{U_i} = (\sigma_M, \sigma_V, R, s)$ purportedly produced by user U_i, a voucher V and a public key PK_{U_i}, the adjudicator

  1. checks whether $\mathbf{PSigVerify}(M, \sigma'_{U_i}, V, \mathsf{PK}_{U_i}, \mathsf{APK}) = \top$,
  2. checks whether $R = G(r)$ where $r = PRF_K(V||s)$.

  If both hold, it returns $\sigma_{U_i} := (\sigma_M, \sigma_V, r, s)$ as a full signature. Otherwise, it rejects the input by outputting $\perp$.

5.2 Security Analysis of Our Generic Construction

The proofs of the following theorems are provided in Appendix B.

Theorem 1. *Our scheme is secure against signers (extractability) if* $\mathbf{SIG_1}$ *is existential unforgeable against adaptive chosen message attacks.*

Theorem 2. *Our scheme is secure against verifiers (opacity) if* $\mathbf{SIG_2}$ *is existential unforgeable against adaptive chosen message attacks and G is a collision-resistant hash function.*

Theorem 3. *Our scheme is unforgeable against the adjudicator if* $\mathbf{SIG_2}$ *is existential unforgeable against adaptive chosen message attacks.*

Theorem 4. *Our scheme is unforgeable against signers if* $\mathbf{SIG_1}$ *is existential unforgeable against adaptive chosen message attacks.*

5.3 Instantiations

In the following, we provide several examples to demonstrate the flexibility of our generic construction. To build FESTA based on well-established complexity assumptions, we can choose $\mathbf{SIG_1}$, $\mathbf{SIG_2}$ to be the signature scheme proposed in [12] and the factorization-based PRF due to [13] (**Instantiation** $\mathscr{I}_1$). Another possibility is to adopt the signature scheme due to [21] and the decisional Deffie-Hellman (DDH) assumption-based PRF due to [13][2] (**Instantiation** $\mathscr{I}_2$). **Instantiation** $\mathscr{I}_1$ is secure under the RSA assumption while **Instantiation** $\mathscr{I}_2$ is secure under the computational Diffie-Hellman (CDH) assumption and the DDH assumption, all of which being well-established assumptions on which many cryptographic primitives are based. On the other hand, one could construct efficient scheme based on possibly stronger assumptions. An example is to employ the short signature from [3] and the PRF presented in [8] (**Instantiation** $\mathscr{I}_3$), which is secure under the q decisional bilinear Diffie-Hellman inversion (q-DBDHI) assumption[3]. Table 1 illustrates the three suggested instantiations with the resulting full signature size assuming a security level equivalent to 80-bit symmetric key cryptography. We would like like remark that while **Instantiation** $\mathscr{I}_2$ is space-efficient, its public parameter size is linear in the length of the message.

6 Conclusion

In the paper, we revisited Shao's idea on fair exchange of digital signatures with semi-trusted adjudicator (FESTA). Our investigations leads to a refined definition of FESTA, which takes into account the subtle issues that are possibly overlooked, such as the re-use of voucher signatures, in the existing constructions. Also, we proposed a generic construction of a FESTA. Compared with Shao's two concrete schemes, our

[2]Note that the DDH assumption may not hold in the group of which the signature [21] is built and thus two independent groups will have to be chosen.

[3]The strong Diffie-Hellman assumption of which the signature scheme from [3] is based on is subsumed into the q-DBDHI assumption.

Table 1: Summary of possible instantiations.

	Underlying Signature	Underlying PRF	Security Assumption	Full Signature Size
$\mathscr{I}_1$	Hohenberger-Waters	Naor-Reingold	RSA	6144
$\mathscr{I}_2$	Waters	Naor-Reingold	CDH/DDH	964
$\mathscr{I}_3$	Boneh-Boyen	Dodis-Yampolskiy	q-DBDHI	962

generic construction enables us to construct a FESTA directly from any ordinary signature scheme that is existential unforgeable against adaptive chosen message attacks, which make it feasible to construct an efficient FESTA in the standard model.

References

[1] N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocols for fair exchange," in *Proc. of the 4th ACM conference on Computer and communications security (CCS'97), Zurich, Switzerland*. ACM, April 1997, pp. 7–17.

[2] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *Proc. of the 17th International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'98), Espoo, Finland*, ser. Lecture Notes in Computer Science, vol. 1403. Springer-Verlag, May-June 1998, pp. 591–606.

[3] D. Boneh and X. Boyen, "Short signatures without random oracles and the sdh assumption in bilinear groups," *Journal of Cryptology*, vol. 21, no. 2, pp. 149–177, April 2008.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. of the 22nd International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'03), Warsaw, Poland*, ser. Lecture Notes in Computer Science, vol. 2656. Springer-Verlag, May 2003, pp. 416–432.

[5] D. Boneh, B. Lynn, and H. Shacham, "Short sigmatures from the weil pairing," in *Proc. of the 7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'01), Gold Coast, Australia*, ser. Lecture Notes in Computer Science, vol. 2248. Springer-Verlag, December 2001, pp. 514–532.

[6] E. F. Brickell, D. Chaum, I. B. Damgård, and J. van de Graaf, "Gradual and verifiable release of a secret," in *Proc. of the 4th CRYPTO: Conference on the Theory and Application of Cryptographic Techniques (CRYPTO'87), Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 293. Springer-Verlag, August 1987, pp. 156–166.

[7] L. Chen, C. Kudla, and K. G. Paterson, "Concurrent signatures," in *Proc. of the 23rd International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'04), Interlaken, Switzerland*, ser. Lecture Notes in Computer Science, vol. 3027. Springer-Verlag, May 2004, pp. 278–305.

[8] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Proc. of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Les Diablerets, Switzerland*, ser. Lecture Notes in Computer Science, vol. 3386. Springer-Verlag, January 2005, pp. 416–431.

[9] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, vol. 28, no. 6, pp. 637–647, June 1985.

[10] O. Goldreich, "A simple protocol for signing contracts," in *Proc. of the Advances in Cryptology (CRYPTO'83), Boston, Massachusetts, USA*, ser. Lecture Notes in Computer Science. Springer-Verlag, 1983, pp. 133–136.

[11] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM (JACM)*, vol. 33, no. 4, pp. 792–807, October 1986.

[12] S. Hohenberger and B. Waters, "Short and stateless signatures from the rsa assumption," in *Proc. of the 29th Annual International Cryptology Conference (CRYPTO'09), Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 5677. Springer-Verlag, August 2009, pp. 654–670.

[13] M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudo-random functions," *Journal of the ACM (JACM)*, vol. 51, no. 2, pp. 231–262, March 2004.

[14] M. Naor, O. Reingold, and A. Rosen, "Pseudo-random functions and factoring," *SIAM Journal on Computing*, vol. 31, no. 5, pp. 1383–1404, 2002.

[15] K. Nguyen, "Asymmetric concurrent signatures," in *Proc. of the 7th International Conference on Information and Communications Security (ICICS'05), Beijing, China*, ser. Lecture Notes in Computer Science, vol. 3783. Springer-Verlag, December 2005, pp. 181–193.

[16] Z. Shao, "Fair exchange protocol of signatures based on aggregate signatures," *Computer Communications*, vol. 31, no. 10, pp. 1961–1969, June 2008.

[17] Z. Shao, "Fair exchange protocol of schnorr signatures with semi-trusted adjudicator," *Computers and Electrical Engineering*, vol. 36, no. 6, pp. 1035–1045, November 2010.

[18] W. Susilo, Y. Mu, and F. Zhang, "Perfect concurrent signature schemes," in *Proc. of the 6th International Conference on Information and Communications Security (ICICS'04), Malaga, Spain*, ser. Lecture Notes in Computer Science, vol. 3269. Springer-Verlag, October 2004, pp. 14–26.

[19] D. Tonien, W. Susilo, and R. Safavi-Naini, "Multi-party concurrent signatures," in *Proc. of the 9th International Conference on Information Security (ISC'06), Samos Island, Greece*, ser. Lecture Notes in Computer Science, vol. 4176. Springer-Verlag, August-September 2006, pp. 131–145.

[20] K. N. Wang, F. Bao, and J. Zhou, "The fairness of perfect concurrent signature," in *Proc. of the 7th International Conference on Information and Communications Security (ICICS'06), Raleigh, North Carolina, USA*, ser. Lecture Notes in Computer Science, vol. 4307. Springer-Verlag, December 2006, pp. 435–451.

[21] B. Waters, "Efficient identity-based encryption without random oracles," in *Proc. of the 24th International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'05), Aarhus, Denmark*, ser. Lecture Notes in Computer Science, vol. 3494. Springer-Verlag, May 2005, pp. 114–127.

Author Biography

Yang Wang received his Ph.D. degree from the University of Wollongong (UOW), Australia in 2014. His research interests include applied cryptography, fair exchange and security analysis of protocols.

Willy Susilo is a Senior Professor in the School of Computing and Information Technology, Faculty of Engineering and Information Sciences in University of Wollongong, Australia. He is the director of Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong. Willy is an innovative educator and researcher. Currently, he is the Head of School of Computing and Information Technology at UOW (2015 - now). Prior to this role, he was awarded the prestigious Australian Research Council Future Fellowship in 2009. He was the former Head of School of Computer Science and Software Engineering (2009 - 2010) and the Deputy Director of ICT Research Institute at UOW (2006 - 2008). He is the Editor in Chief of the Information journal. Willy obtained his PhD from the University of Wollongong in 2001. He has published

more than 300 papers in journals and conference proceedings in cryptography and network security. He has served as the program committee member of several international conferences. In 2016, he was awarded the "Researcher of the Year" at UOW, due to his research excellence and contributions. His work on the creation of short signature schemes has been well cited and it is part of the IETF draft.

Joonsang Baek is a Senior Lecturer in the School of Computer Science and Information Technology and a member of the Institute of Cybersecurity and Cryptology, University of Wollongong (UOW), Australia. He was a Research Scientist in the Institute for Infocomm Research, Singapore, and an Assistant Professor in the Khalifa University of Science and Technology, United Arab Emirates. Joonsang received his PhD from Monash University, Australia, in 2004. His PhD thesis was on security analysis of signcryption, and has received great attention from the research community. He has published his work in numerous reputable journals and conference proceedings. His current research interests are in the field of applied cryptography and cybersecurity. He has also served as a Program Committee Member and the Chair for a number of renowned conferences on information security and cryptography.

Jongkil Kim received his PhD in Computer Science from the University of Wollongong (UOW), Wollongong, Australia in 2016. He is currently a lecturer in the School of Computing and Information Technology and a member of Institute of Cybersecurity and Cryptology in UOW. His main research interest is in the area of applied cryptography and cyber security. He has authored multiple publications in an information security area.

Intae Kim received his BS degree in Computer and Information Communication Engineering in 2010 from Hongik University, and his MS and PhD degrees in Electronics and Computer Engineering in 2012 and 2017 from Hongik University, respectively. He is currently a postdoc in the School of Computing and Information Technology, Faculty of Engineering and Information Sciences in University of Wollongong, Australia. His research interests include cryptography and network security.

A Shao's Schemes Revisited

In this section, we revisit the two FESTA schemes proposed by Shao (Shao08 and Shao10), respectively. We point out some issues that are previously overlooked. These issues affect the formal definition of syntax and security games of FESTA. Finally, we show that one claim made by Shao is inaccurate. Specifically, contrary to the claim made by Shao that interaction between the signer and the adjudicator is limited to one per failed exchange of signature, we observe that the signer is actually required to contact the adjudicator every time he/she wishes to create a new partial signature in practical situations.

A.1 Shao08

This scheme takes the GDH (Gap Diffie-Hellman) signature [5] as a building block. There are three parties involved in the scheme: a signer, a verifier and an adjudicator chosen by the signer unilaterally. The public parameters are the additive group G_1 and multiplicative group G_2 of the same prime order q, a generator P for G_1, a full-domain hash function $H_1 : \{0,1\}^* \to G_1$, and a bilinear pairing $e : G_1 \times G_1 \to G_2$. The private keys for the signer and the adjudicator are denoted by x_A, x_C, respectively. The public keys are computed as $Y_C = x_C P$.

First, the adjudicator composes a voucher V which stipulates the obligation the verifier must fulfill to exchange the signer's message signature in an aggregate signature and explicitly promises that the adjudicator itself would do it if the signer fails to send the message signature to exchange the verifier's fulfilling obligation. The adjudicator generates a GDH signature $\sigma_V = x_C H_1(V)$ on this voucher V as its voucher signature.

To prevent the voucher signature from being eavesdropped while being transmitted to the signer, the adjudicator chooses a random integer $r \in \mathbb{Z}_q^*$ and computes $S = rP$, $T = \sigma_V + rY_A$, and sends (S,T) to the signer. The signer recovers σ_V by computing $\sigma_V = T - x_A S$ and uses σ_V as the current voucher signature to compute an aggregate signature afterwards.

Then the signer generates its message signature $\sigma_M = x_A H_1(M)$, computes the aggregate signature $W = \sigma_M + \sigma_V$, and sends (W, M, V) to the verifier. Normally the verifier accepts the aggregate signature and fulfills its obligation if $e(W,P) = e(H_1(M), Y_A) e(H_1(V), Y_C)$, where $M \neq V$.

After receiving from the verifier the obligation with regards to the message M, the signer generates a new message M' that has the same meaning as the message M, computes its new message signature $\sigma'_M = x_A H_1(M')$, and sends back (σ'_M, M'). In case of a network failure or the signer refuses to send the new message signature, the verifier turns to the adjudicator for a resolution by sending the aggregate signature (W, M, V) and a proof that it has fulfilled the obligation. The adjudicator checks whether V is the current voucher and whether $e(W,P) = e(H_1(M), Y_A) e(H_1(V), Y_C)$. If both hold, the adjudicator returns $\sigma_M = W - x_C H_1(V)$. Besides, the adjudicator generates a new voucher signature $\sigma'_V = x_C H_1(V')$ for a new voucher V', and sends (σ'_V, V') to the signer as the current voucher signature.

When the signer sends the aggregate signature, if the reply of the verifier does not arrive after a reasonable amount of time (the signer chooses itself how long it decides to wait for a given message), the signer asks the adjudicator to execute an abort algorithm to generate a new voucher signature and set it as the current voucher signature.

A.2 Shao10

Shao10 incorporates Schnorr signature as a building block. The public parameters are the primes p and q of appropriate size, such that $q|(p-1)$, a generator g for the subgroup $\mathbb{Z}_p^*$ of order q, and a collision-free hash function $H_2 : \{0,1\}^* \times \mathbb{Z}_p^* \to \mathbb{Z}_q^*$. The private keys for the signer and the adjudicator are denoted by x_A, x_C, respectively. The public keys are computed as $Y_C = g^{x_C}$.

First, the adjudicator composes a voucher V which stipulates the obligation the verifier must fulfill to exchange the signer's message signature in a double signature and explicitly promises that the adjudicator itself would do it if the signer fails to send the message signature to exchange the verifier's fulfilling obligation. The adjudicator generates a Schnorr signature $\sigma_V = (f,u,V)$ on this voucher V as its voucher signature by choosing a random $t \in \mathbb{Z}_q^*$, and computing $v = g^t$, $f = H_2(V,v)$, $u = t - x_C f$.

To prevent the voucher signature from being eavesdropped while transmitted to the signer, the adjudicator chooses a random integer $r \in \mathbb{Z}_q^*$ and computes $h = g^r, z = u \oplus H_2(V, Y_A^r)$, and sends (h,z,f,V) to the signer. The signer recovers σ_V by computing $u = z \oplus H_2(V, h^{x_A})$ and uses σ_V as the current voucher signature to compute a double signature afterwards.

Then the signer generates a Schnorr signature $\sigma_M = (e,s,M)$ as its message signature by choosing a random $k \in \mathbb{Z}_q^*$ and computing $l = g^k, e = H_2(M,l), s = k - x_A e$, and the double signature is set as $\sigma = (w,e,v,M,V)$ where $w = sf + ue, v = g^u Y_C^f$. Note that normally the verifier accepts the double signature and fulfills its obligation if $e = H_2(M, g^{w/f}(Y_C Y_A)^e v^{-e/f})$, where $f = H_2(V,v)$.

After receiving from the verifier the obligation with respect to the message M, the signer sends back a new message signature $\sigma'_M = (e',s',M)$ by choosing a random $k' \in \mathbb{Z}_q^*$ and computing $l' = g^{k'}, e' = H_2(M,l'), s' = k' - x_A e'$. In case of a network failure or the signer refuses to send the new message signature, the verifier turns to the adjudicator for a resolution by sending the double signature $\sigma = (w,e,v,M,V)$ and a proof that it has fulfilled the obligation. With the current voucher signature (f,u,V), the adjudicator computes $v = g^u Y_C{}^f$ and checks whether $f = H_2(V,v)$ and $e = H_2(M, g^{w/f}(Y_C Y_A)^e v^{-e/f})$. If both hold, the adjudicator returns $s = (w - ue)/f$ and thus the verifier obtains the message signature (e,s,M). Besides, the adjudicator generates a new voucher signature σ'_V for a new voucher V', and sends (σ'_V, V') to the signer as the current voucher signature.

When the signer sends the double signature, if the reply of the verifier does not arrive after a reasonable amount of time (the signer chooses itself how long it decides to wait for a given message), the signer asks the adjudicator to execute an abort algorithm to generate a new voucher signature and set it as the current voucher signature.

B Pseudorandom Functions

Pseudorandom functions have been defined by [11]. Here, we adopt the definition proposed by [14] without losing generality. Loosely speaking, pseudorandom functions are efficient distributions of functions which cannot be distinguished from the uniform distribution by any efficient procedure that can get the value of the function at arguments of its choice. To formalize the notion of pseudorandom functions, we need to consider ensembles of functions.

Definition 4. *Let l and k be any two $\mathbb{N} \to \mathbb{N}$ functions. An $\{0,1\}^l \to \{0,1\}^k$ function ensemble is a sequence $F = \{F_n\}_{n\in\mathbb{N}}$ of random variables, so that the random variable F_n assumes values in the set of $\{0,1\}^{l(n)} \to \{0,1\}^{k(n)}$ functions. The uniform $\{0,1\}^l \to \{0,1\}^k$ function ensemble, denoted $R = \{R_n\}_{n\in\mathbb{N}}$, has R_n uniformly distributed over the set of $\{0,1\}^{l(n)} \to \{0,1\}^{k(n)}$ functions.*

Definition 5. *A function ensemble, $F = \{F_n\}_{n\in\mathbb{N}}$, is called efficiently computable if there exist probabilistic polynomial time algorithms, $\mathscr{I}$ and $\mathscr{V}$, and a mapping from strings to functions, ϕ, so that $\phi(\mathscr{I}(1^n))$ and F_n are identically distributed and $\mathscr{V}(i,x) = (\phi(i))(x)$.*
f_i is denoted as the function assigned to i (i.e. $f_i \stackrel{\text{def}}{=} \phi(i)$). i and $\mathscr{I}$ are referred to as the key of f_i and the key-generating algorithm of F, respectively.

The distinguisher is defined to be an (probabilistic polynomial time) oracle machine that could make queries to a function (which is either sampled from the pseudorandom function ensemble or from the

uniform function ensemble.) We assume that on input 1^n the oracle machine makes only n-bit queries. For any probabilistic oracle machine, $\mathscr{M}$, and any $\{0,1\}^n \to \{0,1\}^{t(n)}$ function, O, we denote by $\mathscr{M}^O(1^n)$ the distribution of $\mathscr{M}$'s output on input 1^n and with access to O.

Definition 6. *An efficiently computable $\{0,1\}^n \to \{0,1\}^{t(n)}$ function ensemble, $F = \{F_n\}_{n\in\mathbb{N}}$, is pseudorandom if for any probabilistic polynomial time oracle machine $\mathscr{M}$, any polynomial $p(\cdot)$, and all sufficiently large n's*

$$|Pr[\mathscr{M}^{F_n}(1^n) = 1] - Pr[\mathscr{M}^{R_{n,t}}(1^n) = 1]| < \frac{1}{p(n)}$$

where $R = \{R_{n,t}\}_{n\in\mathbb{N}}$ is the uniform $\{0,1\}^n \to \{0,1\}^{t(n)}$ function ensemble.

The term "pseudorandom functions" is hereafter used as an abbreviation for "efficiently computable pseudorandom function ensemble".

C Proofs of Theorems

C.1 Proof of Theorem 1

To show security against signers (extractability), we convert any adversary $\mathscr{A}$ that wins the experiment EXT in running time t into a forger $\tilde{\mathscr{A}}$ for the underlying signature scheme $\mathbf{SIG_1} = (\mathbf{KG_1}, \mathbf{Sig_1}, \mathbf{Ver_1})$. Recall that $\tilde{\mathscr{A}}$ gets apk as input and has access to $\mathbf{SIG_1}$'s ordinary signature signing oracle O_{SIG_1}. The forger $\tilde{\mathscr{A}}$ wins if it forges a signature on a new voucher which has not been queried to O_{SIG_1}. On the other hand, $\mathscr{A}$ expects APK as input and then returns an verification public key PK and has access to $O_{UserCreate}$, $O_{VSigCreate}$ and O_{Adj} oracles. $\mathscr{A}$ wins if it forges a partial signature σ' on message M and voucher V such that $\mathbf{PSigVerify}(M, \sigma', V, \mathsf{PK}, \mathsf{APK}) = \top$ but $\mathbf{Adjudication}(M, \sigma', V, \mathsf{ASK}, \mathsf{PK}) = \bot$.

On input apk, the forger $\tilde{\mathscr{A}}$ begins simulating the attack environment of $\mathscr{A}$. It feeds $\mathsf{APK} := \mathsf{apk}$ as inputs to $\mathscr{A}$, which then returns an verification public key PK. To simulate $O_{UserCreate}$ to user's public key query of $\mathscr{A}$, the forger $\tilde{\mathscr{A}}$ generates $(\mathsf{sk}_{U_i}, \mathsf{pk}_{U_i}) \leftarrow \mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, and returns $\mathsf{PK}_{U_i} := \mathsf{pk}_{U_i}$ to $\mathscr{A}$.

To simulate $O_{VSigCreate}$ to a voucher signing query (V_i, PK_i) of $\mathscr{A}$, where $\mathsf{PK}_i \in \{\text{outputs of } O_{UserCreate}\} \cup \{\mathsf{PK}\}$, the forger $\tilde{\mathscr{A}}$ checks whether the public key PK_i is contained in the voucher V_i. If not, $\tilde{\mathscr{A}}$ returns $\bot$ to $\mathscr{A}$; otherwise, it chooses a random number s_i, computes $r_i = PRF_{K_i}(V_i||s_i)$ and $R_i = G(r_i)$ where PRF_{K_i} is a pseudo-random function with a seed K_i,, with K_i being a pre-shared key between the adjudicator and the user whose public key is PK_i, and makes a signature query $V_i||R_i||s_i$ to its own oracle O_{SIG_1}. The answer from O_{SIG_1} is σ_{V_i}. $\tilde{\mathscr{A}}$ returns (σ_{V_i}, s_i) to $\mathscr{A}$.

To simulate O_{Adj}'s response to $\mathscr{A}$'s adjudication query $(M_i, \sigma_i', V_i, \mathsf{PK}_i)$ where $\sigma_i' = (\sigma_{M_i}, \sigma_{V_i}, R_i, s_i)$, the forger $\tilde{\mathscr{A}}$ checks whether σ_i' is valid (*i.e.*, $\mathbf{PSigVerify}$ $(M_i, \sigma_i', V_i, \mathsf{PK}_i, \mathsf{APK}) = \top$), and then checks whether $R_i = G(r_i)$ where $r_i = PRF_{K_i}(V_i||s_i)$.

- If both hold, $\tilde{\mathscr{A}}$ returns $\sigma_i = (\sigma_{M_i}, \sigma_{V_i}, r_i, s_i)$ to $\mathscr{A}$.
- Otherwise, $\tilde{\mathscr{A}}$ returns $\bot$ to $\mathscr{A}$.

Finally, adversary $\mathscr{A}$ returns a successful forgery (M^*, σ'^*, V^*), where $\sigma'^* = (\sigma_{M^*}, \sigma_{V^*}, R^*, s^*)$ is the partial signature such that $\mathbf{PSigVerify}(M^*, \sigma'^*, V^*, \mathsf{PK}, \mathsf{APK}) = \top$ but $\mathbf{Adjudication}$ $(M^*, \sigma'^*, V^*, \mathsf{ASK}, \mathsf{PK}) = \bot$. It means that $R^* \neq G(PRF_K(V^*||s^*))$ where PRF_K is a pseudo-random function with a seed K, with K being a pre-shared key between the adjudicator and the user whose public key is PK. Thus, $\tilde{\mathscr{A}}$ has never issued a query to its signing oracle on input $V^*||R^*||s^*$, and σ_{V^*} becomes an existential forgery of $\mathbf{SIG_1}$ on new message $V^*||R^*||s^*$. $\tilde{\mathscr{A}}$ succeeds in breaking the unforgeability of scheme $\mathbf{SIG_1}$.

Suppose adversary $\mathscr{A}$ made at most q_u, q_v and q_a adaptive queries to user's public key oracle $O_{UserCreate}$, the voucher signing oracle $O_{VSigCreate}$ and the adjudication oracle O_{Adj}, respectively. Assume t_U, t_G, t_C and t_V are the execution time of algorithm $\mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, the time to compute a pseudo-random function value and a G value, the time to query oracle O_{SIG_1}, and the execution time of algorithm **PSigVerify**, respectively. Then the time taken to answer $O_{UserCreate}$ queries is $q_u \cdot t_U$, the time taken to answer $O_{VSigCreate}$ queries is $q_v \cdot (t_G + t_C)$ and the time taken to answer O_{Adj} queries is $q_a \cdot (t_G + t_V)$. The running time t' of forger $\tilde{\mathscr{A}}$ is that of adversary $\mathscr{A}$ plus time taken to respond to q_u user's public key queries, q_v voucher signing queries and q_a adjudication queries. Hence, $t' \approx t + q_u \cdot t_U + q_v \cdot (t_G + t_C) + q_a \cdot (t_G + t_V)$. □.

C.2 Proof of Theorem 2

To show security against verifiers (opacity), we convert any adversary $\mathscr{B}$ that wins the experiment OPA in running time t into an forger $\tilde{\mathscr{B}}$ for the underlying scheme $\mathbf{SIG_2}$ or the collision-resistant hash function G. Recall that $\tilde{\mathscr{B}}$ gets pk and $\tilde{R}$ as input, and wins if it forges a signature of $\mathbf{SIG_2}$ on a new message which has not been queried to $\mathbf{SIG_2}$'s ordinary signature signing oracle O_{SIG_2}, or it outputs two values r_1 and r_2 such that $r_1 \neq r_2$ but $G(r_1) = G(r_2)$, or it obtains a pre-imagine r such that $G(r) = \tilde{R}$. On the other hand, $\mathscr{B}$ expects $(\mathsf{PK}, \mathsf{APK})$ as input and has access to $O_{UserCreate}$, $O_{PSigCreateV}$, O_{Sign} and O_{Adj} oracles. $\mathscr{B}$ wins if it forges a full signature σ on message M and voucher V, without asking a query $(M, \cdot, V, \mathsf{PK})$ to O_{Sign} or O_{Adj}.

The forger $\tilde{\mathscr{B}}$ begins simulating the attack environment of $\mathscr{B}$. It generates $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathbf{SIG_1}.\mathbf{KG_1}(1^k)$ and sets $\mathsf{APK} := \mathsf{apk}$, $\mathsf{PK} := \mathsf{pk}$. To simulate $O_{UserCreate}$ to user's public key query of $\mathscr{B}$, the forger $\tilde{\mathscr{B}}$ generates $(\mathsf{sk}_{U_i}, \mathsf{pk}_{U_i}) \leftarrow \mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, and returns $\mathsf{PK}_{U_i} := \mathsf{pk}_{U_i}$ to $\mathscr{B}$.

To simulate $O_{PSigCreateV}$ to a partial signing query $(M_i, V_i, \mathsf{PK}_i)$ of $\mathscr{B}$ where PK_i is a user's public key, we assume each query is valid (*i.e.*, public key PK_i is contained in voucher V_i), and divide the queries to this oracle into two types: Type I queries ($\mathsf{PK}_i = \mathsf{PK}$) and Type II queries ($\mathsf{PK}_i \in \{$outputs of $O_{UserCreate}\}$). Suppose adversary $\mathscr{B}$ made q_p times Type I queries, and $q_{p'}$ times Type II queries, respectively. After choosing j randomly in the interval of $(1, 2, \cdots, q_p)$, the forger $\tilde{\mathscr{B}}$ simulates $O_{PSigCreateV}$'s response to the i-th Type I query (M_i, V_i, PK) of $\mathscr{B}$ as follows.

- If $i = j$, $\tilde{\mathscr{B}}$ chooses a random number s_j, generates $\sigma_{V_j} \leftarrow \mathbf{SIG_1}.\mathbf{Sig_1}(\mathsf{ask}, V_i || \tilde{R} || s_j)$, and makes a signature query $M_j || V_j || \tilde{R} || s_j$ to its own oracle O_{SIG_2}. The answer from O_{SIG_2} is σ_{M_j}. $\tilde{\mathscr{B}}$ returns $\sigma'_j = (\sigma_{M_j}, \sigma_{V_j}, \tilde{R}, s_j)$ to $\mathscr{B}$, and adds the tuple $(M_j, V_j, \tilde{R}, s_j)$ to the P-list.

- If $i \neq j$, $\tilde{\mathscr{B}}$ chooses a random number s_i, computes $r_i = PRF_K(V_i || s_i)$ and $R_i = G(r_i)$ where PRF_K is a pseudo-random function with a seed K, with K being a pre-shared key between the adjudicator and the user whose public key is PK, generates $\sigma_{V_i} \leftarrow \mathbf{SIG_1}.\mathbf{Sig_1}(\mathsf{ask}, V_i || R_i || s_i)$, and makes a signature query $M_i || V_i || R_i || s_i$ to its own oracle O_{SIG_2}. The answer from O_{SIG_2} is σ_{M_i}. $\tilde{\mathscr{B}}$ returns $\sigma'_i = (\sigma_{M_i}, \sigma_{V_i}, R_i, s_i)$ to $\mathscr{B}$, and adds the tuple (M_i, V_i, R_i, s_i) to the P-list.

To simulate $O_{PSigCreateV}$'s response to the i-th Type II query $(M_i, V_i, \mathsf{PK}_i)$ of $\mathscr{B}$ where $\mathsf{PK}_i \in \{$outputs of $O_{UserCreate}\}$, the forger $\tilde{\mathscr{B}}$ knows the corresponding secret key sk_i during the user's public key query. $\tilde{\mathscr{B}}$ chooses a random number s_i, computes $r_i = PRF_{K_i}(V_i || s_i)$ and $R_i = G(r_i)$ where PRF_{K_i} is a pseudo-random function with a seed K_i, with K_i being a pre-shared key between the adjudicator and the user whose public key is PK_i, generates $\sigma_{V_i} \leftarrow \mathbf{SIG_1}.\mathbf{Sig_1}(\mathsf{ask}, V_i || R_i || s_i)$ and $\sigma_{M_i} \leftarrow \mathbf{SIG_2}.\mathbf{Sig_2}(\mathsf{sk}_i, M_i || V_i || R_i || s_i)$, respectively. $\tilde{\mathscr{B}}$ returns $\sigma'_i = (\sigma_{M_i}, \sigma_{V_i}, R_i, s_i)$ to $\mathscr{B}$, and adds the tuple (M_i, V_i, R_i, s_i) to the P-list.

To simulate O_{Sign}'s or O_{Adj}'s response to $\mathscr{B}$'s i-th signing or adjudication query $(M_i, \sigma'_i, V_i, \mathsf{PK}_i)$ where $\sigma'_i = (\sigma_{M_i}, \sigma_{V_i}, R_i, s_i)$ and $\mathsf{PK}_i \in \{$outputs of $O_{UserCreate}\} \cup \{\mathsf{PK}\}$, the adversary $\tilde{\mathscr{B}}$ first checks

whether σ_i' is valid (*i.e.*, **PSigVerify**$(M_i, \sigma_i', V, \mathsf{PK}_i, \mathsf{APK}) = \top$), and then checks whether (M_i, V_i, R_i, s_i) is in the P-list.

- If σ_i' is valid, (M_i, V_i, R_i, s_i) is in the P-list and $R_i \neq \tilde{R}$, $\tilde{\mathscr{B}}$ computes $r_i = PRF_{K_i}(V_i||s_i)$ where PRF_{K_i} is a pseudo-random function with a seed K_i, with K_i being a pre-shared key between the adjudicator and the user whose public key is PK_i, and returns $\sigma_i = (\sigma_{M_i}, \sigma_{V_i}, r_i, s_i)$ to $\mathscr{B}$.
- If σ_i' is valid, (M_i, V_i, R_i, s_i) is in the P-list and $R_i = \tilde{R}$, returns $\perp$ to $\mathscr{B}$ and aborts the simulation.
- If σ_i' is not valid or (M_i, V_i, R_i, s_i) is not in the P-list, $\tilde{\mathscr{B}}$ returns $\perp$ to $\mathscr{B}$.

Denote that the i-th element in the P-list is (M_i, V_i, R_i, s_i), and suppose that $(M_{j^*}, V_{j^*}, R_{j^*} = \tilde{R}, s_{j^*})$ where j^* is a fixed number in the interval of $(1, 2, \cdots, q_p + q_{p'})$ corresponds to the element that was added to the P-list when $\mathscr{B}$ made the j-th Type I $O_{PSigCreateA}$ query before. Since s_i is chosen uniformly at random by the simulator, the probability that there exists an $i \in \{1, 2, \cdots, q_p + q_{p'}\} \backslash \{j^*\}$ such that $(V_i, s_i) = (V_{j^*}, s_{j^*})$ is negligible. Thus, we can safely assume $(V_i, s_i) \neq (V_{j^*}, s_{j^*})$ for any $i \in \{1, 2, \cdots, q_p + q_{p'}\} \backslash \{j^*\}$. If there exists an $i \in \{1, 2, \cdots, q_p + q_{p'}\} \backslash \{j^*\}$ such that $R_i = \tilde{R}$, $\tilde{\mathscr{B}}$ can simply output the value $r = PRF_{K_i}(V_i||s_i)$, which is a pre-imagine of $\tilde{R}$, and thus wins by breaking the one-way property of hash function G.

If there is an $i_1 \in \{1, 2, \cdots, q_p + q_{p'}\} \backslash \{j^*\}$ and an $i_2 \in \{1, 2, \cdots, q_p + q_{p'}\} \backslash \{j^*\}$ such that $i_1 \neq i_2$ but $R_{i_1} = R_{i_2}$, where $(M_{i_1}, V_{i_1}, R_{i_1}, s_{i_1})$ and $(M_{i_2}, V_{i_2}, R_{i_2}, s_{i_2})$ are the i_1-th and i_2-th elements in the P-list, respectively, then due to the fact that both s_{i_1} and s_{i_2} are random numbers and both $PRF_{K_{i_1}}$ and $PRF_{K_{i_2}}$ are pseudo-random functions, it is with overwhelming probability that $PRF_{K_{i_1}}(V_{i_1}||s_{i_1}) \neq PRF_{K_{i_2}}(V_{i_2}||s_{i_2})$. Thus, $\tilde{\mathscr{B}}$ can simply output two values $r_1 = PRF_{K_{i_1}}(V_{i_1}||s_{i_1})$ and $r_2 = PRF_{K_{i_2}}(V_{i_2}||s_{i_2})$. Since $r_1 \neq r_2$ but $G(r_1) = G(r_2)$, $\tilde{\mathscr{B}}$ wins by breaking the collision-resistant property of G.

If a query $(M_{i'}, \sigma_{i'}', V_{i'}, \mathsf{PK}_{i'})$ where $\sigma_{i'}' = (\sigma_{M_{i'}}, \sigma_{V_{i'}}, R_{i'}, s_{i'})$ to O_{Sign} or O_{Adj} oracle is valid (*i.e.*, **PSigVerify**$(M_{i'}, \sigma_{i'}', V_{i'}, \mathsf{PK}_{i'}, \mathsf{APK}) = \top$), but $(M_{i'}, V_{i'}, R_{i'}, s_{i'})$ is not in the P-list, then it is easy to see that $\sigma_{M_{i'}}$ becomes an existential forgery of $\mathbf{SIG_2}$ on $M_{i'}||V_{i'}||R_{i'}||s_{i'}$. $\tilde{\mathscr{B}}$ can simply output $(M_{i'}||V_{i'}||F_{i'}, \sigma_{M_{i'}})$, and thus wins by breaking the unforgeability of scheme $\mathbf{SIG_2}$.

Due to the analysis above, we only need to consider the occasion that each element (M_i, V_i, R_i, s_i) in the P-list has a unique R_i, and each query $(M_{i'}, \sigma_{i'}', V_{i'}, \mathsf{PK}_{i'})$ where $\sigma_{i'}' = (\sigma_{M_{i'}}, \sigma_{V_{i'}}, R_{i'}, s_{i'})$ to O_{Sign} or O_{Adj} oracle is valid and $(M_{i'}, V_{i'}, R_{i'}, s_{i'})$ is in the P-list.

Finally, the adversary $\mathscr{B}$ forges a full signature (M^*, V^*, σ^*) where $\sigma^* = (\sigma_{M^*}, \sigma_{V^*}, r^*, s^*)$, without making the query $(M^*, \cdot, V^*, \mathsf{PK})$ to oracle O_{Sign} or O_{Adj}. $\tilde{B}$ computes $R^* = G(r^*)$. If (M^*, V^*, R^*, s^*) is not in the P-list, from the analysis above, σ_{M^*} becomes an existential forgery of $\mathbf{SIG_2}$ on $M^*||V^*||R^*||s^*$. On the other hand, (M^*, V^*, R^*, s^*) in the P-list implies $R^* = \tilde{R}$ with non-negligible probability. Thus, if $R^* = \tilde{R}$, the adversary $\tilde{\mathscr{B}}$ can simply output the exact r^* and thus wins by breaking the one-way property of function G.

Suppose adversary $\mathscr{B}$ made at most q_u, q_s and q_a adaptive queries to user's public key oracle $O_{UserCreate}$, the signing oracle O_{Sign} and the adjudication oracle O_{Adj}, respectively. Assume t_{U_1} t_{U_2}, t_G, t_{C_1}, t_{C_2}, t_C and t_V are the execution time of algorithm $\mathbf{SIG_1}.\mathbf{KG_1}(1^k)$, the execution time of algorithm $\mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, the time to compute a pseudo-random function value and a G value, the execution time of algorithm $\mathbf{SIG_1}.\mathbf{Sig_1}$, the execution time of algorithm $\mathbf{SIG_2}.\mathbf{Sig_2}$, the time to query oracle O_{SIG_2} and the execution time of algorithm **PSigVerify**, respectively. Then the time taken to answer $O_{UserCreate}$ queries is $q_u \cdot t_{U_2}$, the time taken to answer $O_{PSigCreateV}$ queries is $(q_p - 1) \cdot t_G + q_p \cdot (t_{C_1} + t_C) + q_{p'} \cdot (t_G + t_{C_1} + t_{C_2})$, the time taken to answer O_{Sign} queries is $q_s \cdot (t_G + t_V)$ and the time taken to answer O_{Adj} queries is $q_a \cdot (t_G + t_V)$. The time taken to generate the adjudicator's public key is t_{U_1} and the time taken to transform $\mathscr{B}$'s forgery into $\tilde{\mathscr{B}}$'s forgery is less than t_G. The running time t' of adversary $\tilde{\mathscr{B}}$ is that of adversary $\mathscr{B}$ plus time taken to generate the adjudicator's public key and time taken to respond to q_u user's public

key queries, $(q_p + q_{p'})$ partial signing queries, q_s signing queries, q_a adjudication queries and the time taken to transform $\mathscr{B}$'s forgery into $\tilde{\mathscr{B}}$'s forgery. Thus $t' \approx t + t_{U_1} + q_u \cdot t_{U_2} + q_p \cdot (t_G + t_{C_1} + t_C) + q_{p'} \cdot (t_G + t_{C_1} + t_{C_2}) + (q_s + q_a) \cdot (t_G + t_V)$. □.

C.3 Proof of Theorem 3

Proof. To show unforgeability against the adjudicator, we convert any adversary $\mathscr{C}$ that wins the experiment UF-ADJ in running time t into a forger $\tilde{\mathscr{C}}$ for the underlying signature scheme $\mathbf{SIG_2} = (\mathbf{KG_2}, \mathbf{Sig_2}, \mathbf{Ver_2})$. Recall that $\tilde{\mathscr{C}}$ gets pk as input and has access to $\mathbf{SIG_2}$'s ordinary signature signing oracle O_{SIG_2}. The forger $\tilde{\mathscr{C}}$ wins if it forges a signature on a new message which has not been queried to O_{SIG_2}. On the other hand, $\mathscr{C}$ expects PK as input and then returns an adjudicator's public key APK and has access to $O_{UserCreate}$ and $O_{PSigCreateA}$ oracles. $\mathscr{C}$ wins if it forges a partial signature σ' on message M and voucher V without asking a query $(M, V, \cdot, \mathsf{PK})$ to $O_{PSigCreateA}$.

On input pk, the forger $\tilde{\mathscr{C}}$ begins simulating the attack environment of $\mathscr{C}$. It feeds $\mathsf{PK} := \mathsf{pk}$ as input to $\mathscr{C}$, which then returns the adjudicator's public key APK. To simulate $O_{UserCreate}$ to user's public key query of $\mathscr{C}$, the forger $\tilde{\mathscr{C}}$ generates $(\mathsf{sk}_{U_i}, \mathsf{pk}_{U_i}) \leftarrow \mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, and returns $\mathsf{PK}_{U_i} := \mathsf{pk}_{U_i}$ to $\mathscr{C}$.

To simulate $O_{PSigCreateA}$ to a partial signature query $(M_i, V_i, \sigma_{V_i}, s_i, \mathsf{PK}_i)$ of $\mathscr{C}$, where $\mathsf{PK}_i \in \{\text{outputs of } O_{UserCreate}\} \cup \{\mathsf{PK}\}$, the forger $\tilde{\mathscr{C}}$ checks the validity of (V_i, σ_{V_i}, s_i) w.r.t. PK_i (*i.e.* $\mathbf{VSigVerify}(V_i, \sigma_{V_i,}, s_i, \mathsf{PK}_i, \mathsf{APK}) = \top$) and then whether $\mathsf{PK}_i = \mathsf{PK}$ or not.

- If it is valid and $\mathsf{PK}_i = \mathsf{PK}$, $\tilde{\mathscr{C}}$ computes $r_i = PRF_K(V_i||s_i)$, $R_i = G(r_i)$, where PRF_K is a pseudo-random function with a seed K, with K being a pre-shared key between the adjudicator and the user whose public key is PK, and makes a signature query $M_i||V_i||R_i||s_i$ to its own oracle O_{SIG_2}. The answer from O_{SIG_2} is σ_{M_i}. $\tilde{\mathscr{C}}$ returns $(\sigma_{M_i}, \sigma_{V_i}, R_i, s_i)$ to $\mathscr{C}$.

- If it is valid and $\mathsf{PK}_i \neq \mathsf{PK}$, $\tilde{\mathscr{C}}$ knows the corresponding secret key sk_i during the user's public key query. $\tilde{\mathscr{C}}$, computes $r_i = PRF_{K_i}(V_i||s_i)$, $R_i = G(r_i)$, where PRF_{K_i} is a pseudo-random function with a seed K_i, with K_i being a pre-shared key between the adjudicator and the user whose public key is PK_i, generates $\sigma_{M_i} \leftarrow \mathbf{SIG_2}.\mathbf{Sig_2}(\mathsf{sk}_i, M_i||V_i||R_i||s_i)$ and returns $(\sigma_{M_i}, \sigma_{V_i}, R_i, s_i)$ to $\mathscr{C}$.

- If it is not valid, $\tilde{\mathscr{C}}$ returns $\perp$ to $\mathscr{C}$.

The simulation is perfect. Finally, adversary $\mathscr{C}$ returns a successful forgery (M^*, σ'^*, V^*), where $\sigma'^* := (\sigma_{M^*}, \sigma_{V^*}, R^*, s^*)$, such that $\mathbf{PSigVerify}(M^*, \sigma'^*, V^*, \mathsf{PK}, \mathsf{APK}) = \top$ and $(M^*, V^*, \cdot, \mathsf{PK}) \notin Query(\mathscr{C}, O_{PSigCreateA})$. Since $(M^*, V^*, \cdot, \mathsf{PK}) \notin Query(\mathscr{C}, O_{PSigCreateA})$, it is easy to see that $\tilde{\mathscr{C}}$ has never issued a query to its signing oracle on input $M^*||V^*||R^*||s^*$. Therefore, σ_{M^*} is a valid ordinary signature on the new message $M^*||V^*||R^*||s^*$. $\tilde{\mathscr{C}}$ simply outputs $(M^*||V^*||R^*||s^*, \sigma_{M^*})$. Obviously $(M^*||V^*||R^*||s^*, \sigma_{M^*})$ is an existential forgery of signature scheme $\mathbf{SIG_2}$.

Suppose adversary $\mathscr{C}$ made at most q_u and q_p adaptive queries to user's public key oracle $O_{UserCreate}$ and the partial signing oracle $O_{PSigCreateA}$, respectively. Assume t_U, t_G, t_C and t_V are the execution time of algorithm $\mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, the time to compute a pseudo-random function value and a G value, the longer one between the time to query oracle O_{SIG_2} and the execution time of algorithm $\mathbf{SIG_2}.\mathbf{Sig_2}$, and the execution time of algorithm $\mathbf{VSigVerify}$, respectively. Then the time taken to answer $O_{UserCreate}$ queries is $q_u \cdot t_U$ and the time taken to answer $O_{PSigCreateA}$ queries is $q_p \cdot (t_G + t_C + t_V)$, respectively. The running time t' of adversary $\tilde{\mathscr{C}}$ is that of adversary $\mathscr{C}$ plus time taken to respond to q_u user's public key queries and q_p partial signature queries. Thus, $t' \approx t + q_u \cdot t_U + q_p \cdot (t_G + t_C + t_V)$. □.

C.4 Proof of Theorem 4

To show unforgeability against signers, we convert any adversary $\mathscr{D}$ that wins the experiment UF-SIG in running time t into a forger $\tilde{\mathscr{D}}$ for the underlying signature scheme $\mathbf{SIG_1} = (\mathbf{KG_1}, \mathbf{Sig_1}, \mathbf{Ver_1})$. Recall that $\tilde{\mathscr{D}}$ gets apk as input and has access to $\mathbf{SIG_1}$'s ordinary signature signing oracle O_{SIG_1}. The forger $\tilde{\mathscr{D}}$ wins if it forges an ordinary signature on a new voucher which has not been queried to O_{SIG_1}. On the other hand, $\mathscr{D}$ expects APK as input and then returns its public key PK and has access to $O_{UserCreate}$ and $O_{VSigCreate}$ oracles. $\mathscr{D}$ wins if it forges a partial signature σ' on message M and voucher V without asking a query (V, PK) to $O_{VSigCreate}$

On input apk, the forger $\tilde{\mathscr{D}}$ begins simulating the attack environment of $\mathscr{D}$. It feeds $\mathsf{APK} := \mathsf{apk}$ as inputs to $\mathscr{D}$, which then returns its public key PK. To simulate $O_{UserCreate}$ to a public key query of $\mathscr{D}$, the forger $\tilde{\mathscr{D}}$ generates $(\mathsf{sk}_{U_i}, \mathsf{pk}_{U_i}) \leftarrow \mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, and returns $\mathsf{PK}_{U_i} := \mathsf{pk}_{U_i}$ to $\mathscr{D}$.

To simulate $O_{VSigCreate}$ to a voucher signing query (V_i, PK_i) of $\mathscr{D}$, where $\mathsf{PK}_i \in \{\text{outputs of } O_{UserCreate}\} \cup \{\mathsf{PK}\}$, the forger $\tilde{\mathscr{D}}$ checks whether PK_i is contained in voucher V. If not, $\tilde{\mathscr{D}}$ returns $\perp$ to $\mathscr{D}$. Otherwise, it chooses a random number s_i, computes $r_i = PRF_{K_i}(V_i||s_i)$, $R_i = G(r_i)$, where PRF_{K_i} is a pseudo-random function with a seed K_i, with K_i being a pre-shared key between the adjudicator and the user whose public key is PK_i, and makes a signature query $V_i||R_i||s_i$ to its own oracle O_{SIG_1}. The answer from O_{SIG_1} is σ_{V_i}. $\tilde{\mathscr{D}}$ returns (σ_{V_i}, s_i) to $\mathscr{D}$.

The simulation is perfect. Finally, adversary $\mathscr{D}$ returns a successful forgery (M^*, σ'^*, V^*), where $\sigma'^* := (\sigma_{M^*}, \sigma_{V^*}, R^*, s^*)$, such that $\mathbf{PSigVerify}(M^*, \sigma'^*, V^*, \mathsf{PK}, \mathsf{APK}) = \top$. Since $(V^*, \mathsf{PK}) \notin Query(\mathscr{D}, O_{VSigCreate})$, $\tilde{\mathscr{D}}$ has never issued a query to its signing oracle on input $V^*||R^*||s^*$. Therefore, σ_{V^*} is a valid ordinary signature on the new message $V^*||R^*||s^*$. We simply let $\tilde{\mathscr{D}}$ output $(V^*||R^*||s^*, \sigma_{V^*})$. Obviously $(V^*||R^*||s^*, \sigma_{V^*})$ is an existential forgery of signature scheme $\mathbf{SIG_1}$.

Suppose adversary $\mathscr{D}$ made at most q_u and q_v adaptive queries to user's public key oracle $O_{UserCreate}$ and the voucher signing oracle $O_{VSigCreate}$, respectively. Assume t_U, t_G, t_C are the execution time of algorithm $\mathbf{SIG_2}.\mathbf{KG_2}(1^k)$, the time to compute a pseudo-random value and a G value, and the time to query oracle O_{SIG_1}, respectively. Then the time taken to answer $O_{UserCreate}$ queries is $q_u \cdot t_U$ and the time taken to answer $O_{VSigCreate}$ queries is $q_v \cdot (t_G + t_C)$, respectively. The running time t' of adversary $\tilde{\mathscr{D}}$ is that of adversary $\mathscr{D}$ plus time taken to respond to q_u user's public key queries and q_p partial signature queries. Thus, $t' \approx t + q_u \cdot t_U + q_v \cdot (t_G + t_C)$. □.