A Framework for Universally Composable Non-committing Blind Signatures

Masayuki Abe and Miyako Ohkubo

Information Sharing Platform Laboratories
NTT Corporation, Japan
{abe.masayuki,ookubo.miyako}@lab.ntt.co.jp

Abstract. A universally composable (UC) blind signature functionality demands users to commit to the message to be blindly signed. It is thereby impossible to realize in the plain model. We show that even non-committing variants of UC blind signature functionality remain not realizable in the plain model. We then characterize adaptively secure UC non-committing blind signatures in the common reference string model by presenting equivalent stand-alone security notions. We also present a generic construction based on conceptually simple Fischlin's blind signature scheme.

1 Introduction

Background. Since the introduction of blind signatures [9] vast number of papers are devoted to efficient constructions, security analysis, and extensions. Major applications include untraceable payment systems [9] and anonymous voting [10,13]. The standard notions of security for blind signature schemes in the stand-alone setting are *blindness* and *unforgeability* [9,22,18]. Universal composability (UC) framework [3] offers security in more general setting where other arbitrary protocols are running concurrently. It asserts that the properties provided by an idealized functionality retain even under general composition. A blind signature functionality is first suggested by Canetti in [4] and formally defined by Fischlin in [11] with a round-optimal realization in the common reference string (CRS) model. Kiayias and Zhou study adaptive security in [19].

In known blind signature functionalities, e.g., [11,19], a user *commits* to a message to request a signature. Then a signature is issued by the functionality *remotely* from the view of the signer. In [11], Fischlin pointed out that a UC blind signature protocol that realizes such a functionality implies a UC commitment protocol in the static corruption model and thus impossible to realize in the plain model [7]. A more formal argument is given by Lindell in [20,21]. A common idea for these arguments is that the existence of a simulator implies extraction of the input message and hence contradicts to the blindness.

Is there a hope to circumvent the above impossibility if the functionality is relaxed by giving up the commitment property? In some applications such as a simple e-cash or a coupon system, every message can be a random string that

M. Matsui (Ed.): ASIACRYPT 2009, LNCS 5912, pp. 435–450, 2009.

the users do not need to know or fix in advance. Such applications only concern blindness and unforgeability. In [2], Buan, Gjøsteen, and Kråkmo presented a *non-committing* blind signature functionality where corrupt users no longer deposit messages. Thus there is no need to extract the messages for simulation. It was shown that such a non-committing blind signature functionality is realizable in the plain model and the presented security is equivalent to the unforgeability and weak blindness defined by Juels, Luby and Ostrovsky in [18].

Our contribution. Somewhat contradictory, we show that universally composable non-committing blind signatures are still impossible in the plain model. Our proof shows that if the functionality provides blindness the presence of a simulator contradicts to the unforgeability in the plain model. Importantly, the positive result in [2] stands only in a restricted corruption model where the signer can be corrupted only after the key generation process. As stated in the paper, such a restriction is too strong that it is equivalent to incorporating a trusted party in the protocol. Our result holds for the most general corruption model. It is also pretty robust in the sense that it applies to wide variety of blind signature functionalities that formulate blindness in a reasonable way like all existing functionalities do.

Despite the negative result, non-committing blind signatures remain an interesting cryptographic object to study. The less demanding functionality would allow simple protocol designs in advanced models. This paper presents a thorough characterization of a non-committing blind signature functionality that is secure against adaptive adversaries without secure erasures. We prove that the properties captured by the functionality is equivalent to a pair of stand-alone security notions in the common reference string model, which are the standard unforgeability and a new strong notion of blindness which we call *equivocal simulation blindness*. We then decompose the equivocal simulation blindness to more handy notions called *session equivocality* and *signature equivocality* in a specific setting. We also show a generic construction. The simplicity of our framework can be highlighted when compared to the result on the adaptive security for the committing blind signatures [19].

Due to lack of space, most proofs are moved to the full version [1], which also includes results in the static corruption model.

2 Notations

All algorithms in this paper run in polynomial-time in the security parameter λ. By $y \leftarrow A(x; r)$ we mean that algorithm A is invoked with input x and uniformly chosen randomness r, and outputs something labeled as y. Randomness r may be omitted. By $(a, b) \leftarrow \langle A(x), B(y)\rangle$ we denote an execution of interactive Turing machines A and B on input x and y and with output a and b, respectively. When only one side of the output is of concern, we write $a \leftarrow \langle A(x), B(y)\rangle_L$ for the left side and $b \leftarrow \langle A(x), B(y)\rangle_R$ for the right side. We write $a[\omega] \leftarrow A$ when A has some extra output ω. The meaning of ω depends on the context and will be noted whenever this notation is used. For notations and notions related to the UC framework we refer to [6].

3 Blind Signature Schemes

3.1 Syntax and Standard Security Notions

A blind signature scheme BS in the common reference string model consists of five algorithms $\mathsf{BS} = \mathsf{BS}.\{\mathsf{Crs}, \mathsf{Key}, \mathsf{User}, \mathsf{Signer}, \mathsf{Vrf}\}$. $\mathsf{BS.Crs}$ is a common reference string generator. $\mathsf{BS.Key}$ is a key generator. $\mathsf{BS.User}$ is an interactive signature request algorithm and $\mathsf{BS.Signer}$ is a signing algorithm. Interaction between $\mathsf{BS.User}$ and $\mathsf{BS.Signer}$ forms a signature generation protocol. $\mathsf{BS.Vrf}$ is a signature verification algorithm. A blind signature scheme must provide *completeness* and *consistency*. Roughly, completeness is that for any (m, σ) made faithfully through $\mathsf{BS.Crs}$, $\mathsf{BS.Key}$, $\mathsf{BS.User}$, and $\mathsf{BS.Signer}$, verification algorithm $\mathsf{BS.Vrf}$ outputs 1. Consistency is that $\mathsf{BS.Vrf}$ outputs the same value for the same input (even for keys generated by an adversary). We refer to [14] for details and discussions on these properties. Two standard security notions are unforgeability and blindness as shown below.

Definition 1 (Unforgeability: UF). *A blind signature scheme BS is unforgeable if* $\mathbf{Succ}^{uf}_{F^*}(\lambda) = \Pr[\mathbf{Forge}^{\mathsf{BS}}_{F^*}(\lambda) = 1]$ *is negligible in λ for any algorithm F^* where* $\mathbf{Forge}^{\mathsf{BS}}_{F^*}$ *is the experiment shown below. F^* can access to the oracle arbitrary number of times concurrently.*

Experiment $\mathbf{Forge}^{\mathsf{BS}}_{F^*}(\lambda)$:
$\Sigma \leftarrow \mathsf{BS.Crs}(1^\lambda)$
$(vk, sk) \leftarrow \mathsf{BS.Key}(\Sigma)$
$((m_1, \sigma_1), \ldots, (m_{k+1}, \sigma_{k+1})) \leftarrow F^{*\langle \cdot, \mathsf{BS.Signer}(\Sigma, sk)\rangle}(\Sigma, vk)$
Return 1 *iff*
$\mathsf{completed} \leftarrow \langle \cdot, \mathsf{BS.Signer}(\Sigma, sk)\rangle_R$ *happens at most k times, and*
$m_i \neq m_j$ *for all* $1 \leq i < j \leq k+1$, *and*
$\mathsf{BS.Vrf}(\Sigma, vk, m_i, \sigma_i) = 1$ *for all* $1 \leq i \leq k+1$.

Strong unforgeability (sUF) is defined in the same way but requiring $(m_i, \sigma_i) \neq (m_j, \sigma_j)$ instead of $m_i \neq m_j$, This paper focuses on the above relatively weaker notion as it suffices for major applications.

Definition 2 (Blindness: BL). *A blind signature scheme BS is blind if* $\mathbf{Adv}^{bl}_{B^*}(\lambda) = |\Pr[\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, 0) = 1] - \Pr[\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, 1) = 1]|$ *is negligible in λ for any algorithm B^* where* $\mathbf{Blind}^{\mathsf{BS}}_{B^*}$ *is the experiment shown below.*

$\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, b)$:
$\Sigma \leftarrow \mathsf{BS.Crs}(1^\lambda)$
$(vk, m_0, m_1) \leftarrow B^*(\Sigma)$
$\sigma_b \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m_b), B^*\rangle_L$
$\sigma_{1-b} \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m_{1-b}), B^*\rangle_L$
If $\sigma_0 = \perp$ *or* $\sigma_1 = \perp$ *then set* $\sigma_0 = \sigma_1 = \perp$.
Return $\tilde{b} \leftarrow B^*(\sigma_1, \sigma_0)$

For ease of notation, we represent algorithm B^* as stateful so that it implicitly takes over its internal state from the previous invocation every time it is invoked by the experiment. Only new inputs are explicitly presented in the description. This convention is applied to all algorithms denoted with asterisk (*) throughout the paper.

As observed in [17], the above definition captures the case where the adversary attempts to get useful information by aborting the sessions. [12] extends the notion in such a way that, when adversary B^* is given $(\perp, \perp)$ at the end, it is given an extra piece of information that tells which session (the first or second or both) actually yields $\perp$ in the user side. The results in this paper also holds with respect to the stronger notion of blindness.

4 UC Non-committing Blind Signatures

4.1 Functionality $\mathcal{F}_{\mathbf{ncb}}$

Figure 1 illustrates our non-committing blind signature functionality $\mathcal{F}_{\mathrm{ncb}}$. In the figure, v is a *deterministic* signature verification algorithm. Π is a description of a stateless signing algorithm. See [6] for remarks on running arbitrary algorithms in a functionality. As well as the ordinary signature functionality in [5] we formulate $\mathcal{F}_{\mathrm{ncb}}$ not to provide any security properties if an unregistered verification key is given as input to the signature generation and verification phases. See the discussion about the key management below.

The idea of using counters to enforce the unforgeability is the same as that in [2]. Due to the difference of the timing that the counters are increased, our formulation can live with the general communication model thoroughly controlled by the adversary while the one in [2] needs authenticated communication in its realization. Note that the bare signature functionality in [5] can be realized without authenticated channel because there is no link between the public-key and the identity of the signer and it is not a matter who issues a signature as long as the signature is valid.

Non-committing Property. Observe that input message m from a corrupt user is sent nowhere nor stored in the functionality. Thus $\mathcal{S}$ working on behalf of a corrupt user can complete the signature generation process whatever m is. This formulation results in avoiding the need of extracting the message from the corrupt users.

Unforgeability. This property holds only while signer P_s is honest. Counter C_{cmpl} counts the number of completed signature generations in the signer's side while counter C_{valid} counts the number of valid signatures on distinct messages received by honest users with legitimate verification. The verification process accepts signatures on new messages only if $C_{\mathsf{cmpl}} > C_{\mathsf{valid}}$. From the specification, it is clear that $C_{\mathsf{cmpl}} \geq C_{\mathsf{valid}}$ always holds as long as the signer is honest. Thus unforgeability is guaranteed in the absolute sense. To capture weak unforgeability, C_{valid} is incremented only for unique messages in the signature generation

Key Generation : Given $(\texttt{KeyGen}, sid)$ from a party P_s, verify that $sid = (P_s, sid')$ for some sid'. If not, then ignore. Else, forward $(\texttt{KeyGen}, sid)$ to simulator $\mathcal{S}$. Then, on receiving $(\texttt{Generated}, sid, v, \Pi)$ from $\mathcal{S}$, send $(\texttt{Generated}, sid, v)$ to P_s and record (P_s, v, Π). Let $C_{\mathsf{cmpl}} = C_{\mathsf{valid}} = 0$, and Γ be empty. This phase must be completed only once and before other phases.

Signature Generation : On receiving $(\texttt{Request}, sid, ssid, v', m)$ for some m from P_u, send $(\texttt{Request}, sid, ssid, v')$ to $\mathcal{S}$ and do the following.

i. On receiving $(\texttt{Signed}, sid, ssid)$ from $\mathcal{S}$, forward it to P_s. Set $C_{\mathsf{cmpl}} \leftarrow C_{\mathsf{cmpl}} + 1$.

ii. On receiving $(\texttt{Received}, sid, ssid)$ from $\mathcal{S}$, do as follows:

- If P_u is honest and $v' = v$, then do as follows. If $(m, *, 1) \notin \Gamma$, set $C_{\mathsf{valid}} \leftarrow C_{\mathsf{valid}} + 1$. Compute $\sigma \leftarrow \Pi(m)$ and record $(m, \sigma, 1)$ to Γ. If $(m, \sigma, 0) \in \Gamma$, send an error message to signer P_s and halt. Send $(\texttt{Received}, sid, ssid, \sigma)$ to P_u.
- Else if P_u is corrupt or $v' \neq v$, ask $\mathcal{S}$ and forward P_u whatever received from $\mathcal{S}$.

Signature Verification : On receiving $(\texttt{Verify}, sid, ssid, v', m, \sigma)$ from some party P_v, set $\varphi = v'(m, \sigma)$ and do as follows.

1. If $v' \neq v$, set $f = \varphi$.
2. Else if $(m, \sigma, f') \in \Gamma$ for any f', then set $f = f'$.
3. Else if P_s is corrupt or $(m, *, 1) \in \Gamma$, then set $f = \varphi$ and record (m, σ, f) to Γ.
4. Otherwise:
   (a) If $C_{\mathsf{cmpl}} > C_{\mathsf{valid}}$, then set $f = \varphi$ and $C_{\mathsf{valid}} \leftarrow C_{\mathsf{valid}} + f$.
   (b) Otherwise, set $f = 0$.
   Then record (m, σ, f) to Γ.

Output $(\texttt{Verified}, sid, ssid, f)$ to P_v.

Player Corruption : On receiving corruption to P_u, send all inputs and outputs exchanged with P_u to simulator $\mathcal{S}$. Also send all randomness used in the evaluations of Π with respect to P_u.

Fig. 1. Non-committing blind signature functionality $\mathcal{F}_{\mathrm{ncb}}$

process (see step (ii)). Strong unforgeability can be captured by removing conditions "if $(m, *, 1) \notin \Gamma$" and "or $(m, *, 1) \in \Gamma$" from the signature generation and verification phases respectively.

Completeness and Consistency. If the signer and a user are not corrupted and the registered key is given as input to the signature generation phase, $(m, \sigma, 1)$ is recorded. The verification phase for such faithfully generated (m, σ) and registered v finds that record and always outputs $f = 1$. Thus completeness is captured. Consistency holds for free since algorithm v is deterministic. Limiting v to be deterministic loses generality but makes the exposition considerably simpler. For issues with respect to probabilistic verification algorithms see [5,6,14].

Blindness. Important observations are; 1. Π is fixed *before* any sub-session for signature generation starts, 2. Π takes nothing but message m as input, and 3. Message m and $\Pi(m)$ are never sent to $\mathcal{S}$ or P_s during the signature

generation phase. This formulation thereby assures that remotely computed σ is independent of the signature generation viewed by the signer. Such a mechanism, which we call *remote signing*, is suggested in [4] and employed by all known blind signature functionalities.

On Key Management. The "bare" signature functionality in [4,6] is formulated in such a way that it stores a single public-key in every session and the security properties are guaranteed only for the registered public-key. The functionality enjoys concise presentation and high modularity. We take over his approach to define $\mathcal{F}_{\mathrm{ncb}}$. Namely, if unregistered v' is given as input to the signature generation or verification phase, $\mathcal{F}_{\mathrm{ncb}}$ behaves just as $\mathcal{S}$ intended. So even though a user is honest, no security is guaranteed in such a case. (Recall that the environment can pass arbitrary v' to an honest user.) Accordingly, upper-level protocols that uses $\mathcal{F}_{\mathrm{ncb}}$ must be responsible to provide registered v to the honest users.

An alternative formulation would be to let $\mathcal{F}_{\mathrm{ncb}}$ to explicitly reject unregistered v'. It however results in incorporating a mechanism for distributing the correct public-key *within the blind signature protocol*. For instance, the protocol realizing $\mathcal{F}_{\mathrm{ncb}}$ may be constructed in $\mathcal{F}_{\mathrm{ca}}$-hybrid model where $\mathcal{F}_{\mathrm{ca}}$ is the certificate-authority functionality [5] that serves *only* for the blind signature protocol. Though this kind of issue can be handled by the theorem of universal composition with joint state [8], we prefer $\mathcal{F}_{\mathrm{ncb}}$ to be basic for the sake of higher modularity.

In the literates, [11,2] implicitly follow the same approach as ours. They however define their functionality only for the case of receiving the registered public-key as input to the signature generation phase. It results in simpler presentation but eventually the details need to be provided with care. [19] shows more extended functionality such that it handles several public-keys under the same session-id and guarantees blindness for every set of signatures issued with the same public-key. This approach however suffers high complexity in its presentation.

Variations. $\mathcal{F}_{\mathrm{ncb}}$ in Fig. 1 notifies only the end of the signature generation process to the environment. It can be extended so that the environment can give the signer explicit approval or denial for starting the process by adding another round of interaction among $\mathcal{S}$, $\mathcal{F}_{\mathrm{ncb}}$, and P_s. It is also possible to let the environment know about the abnormal termination of the protocol in the same way. These modifications do not affect to the results in this paper since they can be incorporated only by modifying the protocol wrapper in Section 5.2 accordingly.

4.2 Impossibility in the Plain Model

This section shows that $\mathcal{F}_{\mathrm{ncb}}$ cannot be realized without accessing to extra ideal functionalities or assuming some help from incorruptible parties. To make the statement meaningful, we consider non-trivial protocols where honest parties running the protocol with right inputs terminate and output something with noticeable probability.

Theorem 1. *There exists no non-trivial protocol that securely realizes* $\mathcal{F}_{\mathrm{ncb}}$ *in the plain model.*

Proof. We use $\mathcal{S}$ to extract the remote signing function Π and use it to break the unforgeability in the real protocol. Recall that a forgery could never happen in the ideal model. Thus $\mathcal{Z}$ can distinguish the ideal process and a real protocol execution by observing a successful forgery.

Suppose that there exists a non-trivial protocol π that realizes $\mathcal{F}_{\mathrm{ncb}}$ in the plain model. Recall that $\mathcal{F}_{\mathrm{ncb}}$ is invoked when it receives (`KeyGen`, *sid*) from a signer. It then outputs (`Generated`, *sid*, v) to the signer. Protocol π works in the same way since it realizes $\mathcal{F}_{\mathrm{ncb}}$. Let π_{KG} denote such a part of π that receives (`KeyGen`, *sid*) as input and outputs (`Generated`, *sid*, v).

Consider a particular $\mathcal{A}^*$ and $\mathcal{Z}^*$ that behave in $\mathrm{EXEC}_{\pi,\mathcal{A}^*,\mathcal{Z}^*}$ as follows. $\mathcal{Z}^*$ first asks $\mathcal{A}^*$ to corrupt the signer. $\mathcal{Z}^*$ then runs π_{KG} with input (`KeyGen`, *sid*) and obtains (`Generated`, *sid*, v). (Here, without loss of generality, we assume that π_{KG} can be run solely by the signer up to the moment (`Generated`, *sid*, v) is output. See the discussion after the proof for generalization.) $\mathcal{Z}^*$ then sends (`KeyGen`, *sid*) and v to $\mathcal{A}^*$ and receives (`Generated`, *sid*, v) from $\mathcal{A}^*$ working on behalf of the corrupt signer. $\mathcal{Z}^*$ then asks a signature on a message m by sending (`Request`, *sid*, *ssid*, v, m) to an honest user. If $\mathcal{A}^*$ is to join π on behalf of the signer to generate a signature, $\mathcal{Z}^*$ takes over the role and completes the protocol by faithfully following π. The user eventually outputs (`Received`, *sid*, *ssid*, σ). Finally $\mathcal{Z}^*$ sends (`Verify`, *sid*, *ssid*, v, m, σ) to a user and receives (`Verified`, *sid*, *ssid*, f) as a result of verification. Observe that, even though the signer is corrupted, $\mathcal{Z}^*$ simulates an honest signer by following π. Furthermore, due to the completeness and terminating property of π, $\mathcal{Z}^*$ can complete signature generation with noticeable probability. If $\mathcal{Z}^*$ completes, $f = 1$ appears at the end. Since π realizes $\mathcal{F}_{\mathrm{ncb}}$, there exists a simulator $\mathcal{S}^*$ for such $\mathcal{A}^*$ and $\mathcal{Z}^*$. To successfully simulate $\mathcal{A}^*$, simulator $\mathcal{S}^*$ has to send Π to $\mathcal{F}_{\mathrm{ncb}}$ before $\mathcal{Z}^*$ sends (`Request`, *sid*, *ssid*, v, m) to an honest user. Furthermore, with noticeable probability, $\Pi(m)$ must yield a valid signature accepted by protocol π.

Now we construct $\mathcal{Z}$ that distinguishes $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}$ by using above $\mathcal{S}^*$ as a subroutine. $\mathcal{Z}$ first sends (`KeyGen`, *sid*) to the honest signer and receives (`Generated`, *sid*, v). Then $\mathcal{Z}$ starts simulating $\mathcal{Z}^*$. It asks $\mathcal{S}^*$ to corrupt the simulated signer. Then it sends (`KeyGen`, *sid*) and v to $\mathcal{S}^*$ and receives (`Generated`, *sid*, v, Π) from $\mathcal{S}^*$ on behalf of $\mathcal{F}_{\mathrm{ncb}}$. Now $\mathcal{Z}$ computes $\sigma \leftarrow \Pi(m)$ for some m. It then sends (`Verify`, *sid*, *ssid*, v, m, σ) to a verifier and receives (`Verified`, *sid*, *ssid*, f). The output of $\mathcal{Z}$ is f.

Let us evaluate $\mathcal{Z}$. Suppose that $\mathcal{Z}$ is in $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. $\mathcal{Z}$ simulates $\mathcal{Z}^*$ perfectly for $\mathcal{S}^*$. In particular v in this case is generated honestly by π just as $\mathcal{Z}^*$ does. So $\mathcal{S}^*$ outputs (`Generated`, *sid*, v, Π) as expected. Then with noticeable probability such Π yields σ that passes the verification protocol of π. Thus $f = 1$ happens with noticeable probability in this case. Next suppose that $\mathcal{Z}$ is in $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}$. In this case, v is generated by $\mathcal{S}$. If it is distinguishable from the one observed in $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$, $\mathcal{Z}$ distinguishes $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}$ on that basis. If it is indistinguishable, $\mathcal{S}^*$ outputs (`Generated`, *sid*, v, Π) as well

as in the previous case. Since no signature generation process is completed in $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}$ and $\mathcal{F}_{\mathrm{ncb}}$ provides absolute unforgeability, $\mathcal{F}_{\mathrm{ncb}}$ rejects σ generated by Π. Thus $f = 0$ for this case. Accordingly $\mathcal{Z}$ distinguishes $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}$ with noticeable probability. ∎

An essential point is that $\mathcal{F}_{\mathrm{ncb}}$ demands $\mathcal{S}$ to extract Π even from a corrupt signer for the sake of blindness. But the successful extraction of Π contradicts to the unforgeability. The situation is very similar to the case of UC commitments [7] where the message from a corrupt committer must be extracted for the sake of binding property, and the successful extraction contradicts to the hiding property.

The proof does not go through if protocol π involves incorruptible trusted parties or any extra ideal functionalities. The point is that $\mathcal{Z}^*$ should be able to run π_{KG} by itself so that the distribution of v is solely under the control. This allows $\mathcal{Z}$ to simulate $\mathcal{Z}^*$ simply by sending v generated outside of $\mathcal{Z}$. If π_{KG} involves parties other than the signer, $\mathcal{Z}^*$ corrupts them before they send off any message and simulate them honestly by following π_{KG}. When $\mathcal{Z}$ simulates $\mathcal{Z}^*$, these corrupted parties are simulated by following the behavior of the real uncorrupted players $\mathcal{Z}$ is working with.

5 Characterization

5.1 Blindness Based on Simulatability

The following new notion called *simulation blindness* assures that the signature generation protocol can be executed without knowing the message. Similarly, the resulting signature can be generated without involving any information from the protocol run. To capture adaptive security, we require *state reconstruction property*. We use the term *equivocal* when a notion involves state reconstruction property.

Definition 3 (Equivocal Simulation Blindness: $\mathsf{EqSimBLND}$). *A blind signature scheme BS is equivocal simulation blind if there exists a set of algorithms $\mathsf{SIM} = \mathsf{SIM}.\{\mathsf{Crs}, \mathsf{User}, \mathsf{Sig}, \mathsf{State}\}$ such that $\mathsf{SIM}.\mathsf{User}$ and $\mathsf{SIM}.\mathsf{State}$ can be stateful and $\mathsf{SIM}.\mathsf{Sig}$ must be stateless, and advantage $\mathbf{Adv}^{\mathsf{eqsib}}_{D^*}(\lambda) = |\Pr[\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0) = 1] - \Pr[\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1) = 1]|$ is negligible in λ for any D^*, where $\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, b)$ is the following experiment. Oracles are accessible in arbitrary manner.*

$\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$:
$\Sigma \leftarrow \mathsf{BS}.\mathsf{Crs}(1^\lambda)$
$vk \leftarrow D^*(\Sigma)$
$\tilde{b} \leftarrow D^{*\mathcal{O}_1(\Sigma, vk, \cdot)}$
Return $\tilde{b}$

$\mathcal{O}_1(\Sigma, vk, m)$
$\sigma \leftarrow \langle \mathsf{BS}.\mathsf{User}(\Sigma, vk, m; r), D^* \rangle_L$
Output (σ, r)

$\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$:
$(\Sigma, t) \leftarrow \mathsf{SIM.Crs}(1^\lambda)$
$vk \leftarrow D^*(\Sigma)$
$\tilde{b} \leftarrow D^{*\mathcal{O}_0(\Sigma, vk, \cdot, t)}$
Return $\tilde{b}$

$\mathcal{O}_0(\Sigma, vk, m, t)$
$\delta[\omega_u] \leftarrow \langle \mathsf{SIM.User}(\Sigma, vk, t), D^* \rangle_L$
$\sigma[\omega_s] \leftarrow \mathsf{SIM.Sig}(\Sigma, vk, m, t)$
$r \leftarrow \mathsf{SIM.State}(\omega_u, \omega_s)$
If $\delta = 0$, *then set* $\sigma = \perp$.
Output (σ, r)

Denoted by ω_u *and* ω_s *are the state information of* SIM.User *and* SIM.Sig, *respectively.*

Note that SIM.State is supposed to simulate the randomness even for the case where the interaction between SIM.User and D^* is terminated abnormally. SIM.State can see how the interaction is terminated by seeing the state information ω_u.

It would be more useful if we could present separate notions of simulatability for simulating the view of sessions by SIM.User and the signatures by SIM.Sig. We call the notions *session equivocality* and *signature equivocality*. It is however not a proper way in general. Since SIM.User and SIM.Sig uses the same trapdoor as input and they may give negative influence each other when they are used at the same time. We thus consider a special case where trapdoors are separated like (t_1, t_2), and SIM.User (and SIM.Sig) can be run only with t_1 (and t_2, respectively). With respect to the separate trapdoor generator we define two notions of simulatability.

Definition 4 (Separable Trapdoor Generator). *SIM.Crs is a separable trapdoor generator if it outputs* $(\Sigma, (t_1, t_2))$ *such that* Σ *is indistinguishable from those generated by BS.Crs with negligible advantage, say* $\mathbf{Adv}^{\mathsf{crs}}_{C^*}$, *for any algorithm* C^*.

Definition 5 (Signature Equivocality: SigEq). *A blind signature scheme BS is signature equivocal if there exists algorithms SIM.Sig and SIM.SigState such that advantage function* $\mathbf{Adv}^{\mathsf{sigeq}}_{A^*}(\lambda) = |\Pr[\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, 0) = 1] - \Pr[\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, 1) = 1]|$ *is negligible in security parameter* λ *for any* A^*, *where* $\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, b)$ *is the following experiment.*

$\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, b)$:
$(\Sigma, (t_1, t_2)) \leftarrow \mathsf{SIM.Crs}(1^\lambda)$
$vk \leftarrow A^*(\Sigma)$
$\tilde{b} \leftarrow A^{*\mathcal{O}_b(\Sigma, vk, \cdot, t_1)}$
Return $\tilde{b}$

$\mathcal{O}_1(\Sigma, vk, m, t_1)$
$\sigma \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m; r_1 || r_2), A^* \rangle_L$
Output $(\sigma, r_1 || r_2)$

$\mathcal{O}_0(\Sigma, vk, m, t_1)$
$\sigma[\theta] \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m; r_1 || r_2), A^* \rangle_L$
$\sigma'[\omega_s] \leftarrow \mathsf{SIM.Sig}(\Sigma, vk, m, t_1)$
$r'_1 \leftarrow \mathsf{SIM.SigState}(\theta, \omega_s)$
If $\sigma = \perp$, *then* $\sigma' = \perp$, $r'_1 = r_1$.
Output $(\sigma', r'_1 || r_2)$

Symbol θ *is the transcript observed by* BS.User, *and* ω_s *is a state information of* SIM.Sig.

Definition 6 (Session Equivocality: SesEq). *A blind signature scheme BS is session equivocal if there exists algorithms SIM.User and SIM.SesState such that advantage function* $\mathbf{Adv}^{seseq}_{E^*}(\lambda) = |\Pr[\mathbf{SesEQ}^{BS}_{E^*}(\lambda,0)=1] - \Pr[\mathbf{SesEQ}^{BS}_{E^*}(\lambda,1)=1]|$ *is negligible in* λ *for any algorithm* E^**, where experiment* $\mathbf{SesEQ}^{BS}_{E^*}$ *is the following.*

$\mathbf{SesEQ}^{BS}_{E^*}(\lambda, b)$:
$(\Sigma, (t_1, t_2)) \leftarrow SIM.Crs(1^\lambda)$
$vk \leftarrow E^*(\Sigma, t_1)$
$\tilde{b} \leftarrow E^{*\mathcal{O}_b(\Sigma, vk, \cdot, t_2)}$
Return $\tilde{b}$

$\mathcal{O}_1(\Sigma, vk, m, t_2)$:
$\langle BS.User(\Sigma, vk, m; r_1||r_2), E^*\rangle$
Return r_2

$\mathcal{O}_0(\Sigma, vk, m, t_2)$:
$\delta[\omega_u] \leftarrow \langle SIM.User(\Sigma, vk, t_2), E^*\rangle_L$
$r_2 \leftarrow SIM.SesState(\omega_u, m)$
Return r_2

Oracle $\mathcal{O}_b$ *receives a message* m *from* E^* *and interacts with* E^**. Symbol* ω_u *is the state information of SIM.User.*

In Definition 5 it is assumed that randomness r used in BS.User can be separated into two parts r_1 and r_2. An intuition is that r_2 is used while interacting with the signer and r_1 is used after receiving the final message from the signer for computing the output signature. This treatment does not lose generality as one can set either part as empty. Regarding Definition 6 we stress that the messages and the resulting signatures are not given to E^*. Also note that trapdoor t_1 is given to E^*.

We now show relations between the standard blindness and simulation blindness. Since simulation blindness captures blindness in a very strong way, it seems natural that the following lemma holds. Proofs for the following lemmas are in [1].

Lemma 1 (EqSimBLND $\Rightarrow$ BL). *If BS is equivocal simulation blind then it is blind.*

Proof is done in a standard way. We construct D^* that successfully breaks equivocal simulation blindness by using B^* that breaks blindness.

Regarding the reverse direction, we do not know if blindness solely implies simulation blindness or not. We however can show that there exists a scheme that is blind *and unforgeable* but not simulation blind. Namely, for the schemes that provide both blindness and unforgeability the simulation blindness is a strictly stronger notion than blindness. This implication is limited but sufficiently meaningful since we are interested in schemes that provide both blindness and unforgeability. Proof can be done in the similar way as that of Theorem 1.

Lemma 2 (BL $\wedge$ UF $\not\Rightarrow$ EqSimBLND). *There exists BS that is blind and unforgeable but not equivocal simulation blind.*

The following lemma states that it suffices to consider simulatability about sessions and signatures individually when trapdoors are separable for each purpose.

Lemma 3 (SesEq $\wedge$ SigEq $\Rightarrow$ EqSimBLND). *If BS has a separable trapdoor generator and is signature equivocal and session equivocal with respect to the generator then BS is equivocal simulation blind.*

Proof is done through three steps of game transformations starting from $\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$ to $\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$.

5.2 Protocol Wrapper Wrap()

In Fig. 2, we show how to transform a blind signature scheme BS into a blind signature protocol by applying a simple wrapper algorithm, Wrap(). The resulting protocol Wrap(BS) is in the $\mathcal{F}_{\mathrm{crs}}$-hybrid model where $\mathcal{F}_{\mathrm{crs}}$ is the CRS generation and distribution functionality whose output distribution is defined by BS.

Blind Signature Protocol Wrap(BS) **in** $\mathcal{F}_{\mathbf{crs}}$**-model**

Key Generation: Upon receiving $(\texttt{KeyGen}, sid)$ from the environment $\mathcal{Z}$, a party P_s verifies that $sid = (P_s, sid')$ for some sid'. If not, do nothing. Else, P_s derives CRS Σ from $\mathcal{F}_{\mathrm{crs}}$, computes $(vk, sk) \leftarrow \mathsf{BS.Key}(\Sigma)$ and outputs $(\texttt{Generated}, sid, v)$ where $v(m, \sigma) = \mathsf{BS.Vrf}(\Sigma, vk, \sigma, m)$.

Blind Signature Generation: Party P_u and P_s do the following.

- P_u**-side:** On receiving $(\texttt{Request}, sid, ssid, v', m)$ from $\mathcal{Z}$, derive Σ from $\mathcal{F}_{\mathrm{crs}}$, send $(\texttt{Request}, sid, ssid, v')$ to P_s, invoke $\mathsf{BS.User}(\Sigma, vk', m)$, and interact with P_s. Take vk' out from v'. If BS.User outputs σ such that $\mathsf{BS.Vrf}(\Sigma, vk', \sigma, m) = 1$, then output $(\texttt{Received}, sid, ssid, \sigma)$.
- P_s**-side:** On receiving $(\texttt{Request}, sid, ssid, v')$ from a user P_u, get Σ from $\mathcal{F}_{\mathrm{crs}}$, invoke $\mathsf{BS.Signer}(\Sigma, sk)$ and interacts with P_u. If BS.Signer outputs completed, then output $(\texttt{Signed}, sid, ssid)$.

Signature Verification: On receiving $(\texttt{Verify}, sid, ssid, v', m, \sigma)$ from $\mathcal{Z}$, a party P_v derives Σ from $\mathcal{F}_{\mathrm{crs}}$, takes vk' from v', computes $f \leftarrow \mathsf{BS.Vrf}(\Sigma, vk', \sigma, m)$, and outputs $(\texttt{Verified}, sid, ssid, f)$.

Common Reference Functionality $\mathcal{F}_{\mathbf{crs}}$

CRS Generation: On receiving $(\texttt{CrsGen}, sid)$, $\mathcal{F}_{\mathrm{crs}}$ computes $\Sigma \leftarrow \mathsf{BS.Crs}(1^\lambda)$ for the first time and returns Σ. Simply return the same Σ for further requests.

Fig. 2. UC blind signature protocol transformed from stand-alone scheme BS

Note that the resulting protocol does not implement any mechanism to verify the given verification algorithm v'. It works as intended if $v' = v$ but no security is guaranteed for the user if $v' \neq v$. Also note that the signer ignores v' given from the user and uses the genuine secret key sk.

5.3 Equivalence

Theorem 2 ($\mathsf{UF} \wedge \mathsf{EqSimBLND} \Leftrightarrow \mathcal{F}_{\mathrm{ncb}}$)**.** *Protocol* Wrap(*BS*) *securely realizes* $\mathcal{F}_{\mathrm{ncb}}$ *with respect to adaptive adversaries if and only if BS is unforgeable and equivocal simulation blind.*

"If" direction is proven by constructing a simulator, $\mathcal{S}$, that uses $\mathcal{A}$ as a black-box. To run $\mathcal{A}$ properly, $\mathcal{S}$ simulates entities and their communication in

$\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}$. We then apply the game transformation technique starting from $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a)$ as Game 0. Game 1 removes the use of simulation algorithms $\mathsf{SIM.Crs}$, $\mathsf{SIM.User}$, $\mathsf{SIM.Sig}$, and $\mathsf{SIM.State}$ from $\mathcal{F}_{\mathrm{ncb}}$ and $\mathcal{S}$. The difference is negligible due to the simulation blindness. Game 2 then modifies the verification process of $\mathcal{F}_{\mathrm{ncb}}$ so that it no longer care for the counters. This modification is justified by the unforgeability. Game 3 further modifies the verification process so that it completely follows the verification function. Justification is due to the completeness and consistency. Game 4 then modifies $\mathcal{F}_{\mathrm{ncb}}$ so that it does not record the signed messages any more. It is justified by the completeness and consistency again. Finally, Game 5 removes unused actions in $\mathcal{F}_{\mathrm{ncb}}$ and $\mathcal{S}$. This is just cosmetic to make sure that $\mathcal{F}_{\mathrm{ncb}}$ and $\mathcal{S}$ do nothing but executing the real protocol. Thus Game 5 is equivalent to $\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)$.

"Only if" direction is more intricate. First, assuming that BS is not simulation blind, we show that, for any $\mathcal{S}$, there exists successful $\mathcal{Z}$. Second, assuming that BS is simulation blind but forgeable, we construct successful $\mathcal{Z}$ that is not fooled by any $\mathcal{S}$. For the first part, we construct simulation algorithms $\mathsf{SIM.Crs}$, $\mathsf{SIM.User}$, $\mathsf{SIM.Sig}$ and $\mathsf{SIM.State}$ by using $\mathcal{S}$ as a subroutine. For such simulation algorithms there exists adversary D^* that breaks simulation blindness since we assumed that BS is not simulation blind. Then we use such D^* to construct $\mathcal{Z}$. A tricky issue in constructing these simulation algorithms is that they do not share the internal state. Since individual copy of $\mathcal{S}$ is run independently in these functions, it would output different CRS-es and public-keys. Our idea is to use the trapdoor as a container of the randomness given to $\mathcal{S}$ so that every simulation algorithm can give the same randomness to $\mathcal{S}$. In this way, every copy of $\mathcal{S}$ works on the same CRS and public-key so that all simulation algorithms work consistently. A formal proof is given in [1].

6 A Generic Construction

6.1 Overview

Our starting point is the "basic" blind signature scheme by Fischlin [11]. In his scheme, a user commits to message m by sending a commitment c and the signer returns a bare signature s on c. Then the user computes a final signature σ which actually is a non-interactive zero-knowledge proof of knowledge about the message m and the valid signature s. Unforgeability is based on the binding property of the commitment and the unforgeability of the bare signature scheme and the knowledge soundness of NIZK. Blindness is from the hiding property of the commitment scheme and the zero-knowledge property of NIZK. By BS_{G} we denote this generic scheme. When transformed by our wrapper, $\mathsf{Wrap}(\mathsf{BS}_{\mathsf{G}})$ securely realizes non-committing blind signature functionality $\mathcal{F}_{\mathrm{ncb}}$ with respect to *static* adversaries. (See [1] for details.) It is a surprise that such a conceptually simple scheme can provide universal composability even though the adversary is limited to be static.

An essential issue to handle adaptive security is the state reconstruction. Looking at the structure of BS_{G}, the session equivocality can be easily achieved

by replacing the commitment scheme with a *trapdoor* commitment scheme. (In fact, with such a small modification to BS_{G}, the resulting $\mathsf{Wrap}(\mathsf{BS}_{\mathsf{G}})$ provides adaptive UC security *in the erasure model*.) On the other hand, the signature equivocability is not generally possible there. Recall that a signature is simulated by the zero-knowledge simulator. It therefore can be the case that there exists no randomness that is consistent to a real witness. To overcome this problem, we consider eliminating the use of zero-knowledge simulator by providing a correct witness to the proof system through the simulation of the bare signature in the signer-side. Namely, we make the signer's signing algorithm to be simulatable by using a signature scheme in the CRS model so that valid signatures can be created with the trapdoor of the CRS. In this way, we can always provide a witness to the proof system used in the user-side algorithm. Now, witness indistinguishability of the proof system assures that the same proof could have been created from any other witnesses. Accordingly, a consistent randomness always exists. This particular structure is suggested in [17] for the purpose of removing the CRS in the stand-alone model. We will take advantage of the structure for achieving adaptive security.

6.2 Building Blocks

−NIWI (*Non-interactive Witness Indistinguishable Proof System*). It is a non-interactive witness indistinguishable proof system of knowledge when the CRS is generated in the regular way. By NIWI.Crs, NIWI.Prf and NIWI.Vrf, we denote the CRS generation function, the proof generation function and the verification function, respectively. Additionally it must allow state reconstruction when the CRS is simulated. Namely, one can reconstruct a consistent randomness for a given witness and a valid transcript. The Groth-Sahai proof system [16], the GS proof system for short, meets these requirements under SXDH or DLIN assumption. It unfortunately does not work for any NP statement but works efficiently for relations represented by bilinear products. We thus need to choose other building blocks so that they fit to the GS proof system for instantiation.

−TC (*Trapdoor Commitment Scheme*). It is a standard trapdoor commitment scheme. By TC.Key, TC.Com and TC.Vrf, we denote the key generation function, the commitment function, and the verification function. There are two more functions such that one generates a random commitment and the other opens the commitment to an arbitrary value by using the trapdoor generated by TC.Key. See [1] an instantiation that works well with GS proof system under the SXDH assumption.

−SSIG (*Simulatable Signature Scheme*). It is a signature scheme in the CRS model with a special property such that valid signatures can be computed from the public-key and the trapdoor bind the CRS. By SSIG.Crs and SSIG.Key, we denote the CRS generation function and the key generation function. SSIG.Key takes the CRS and outputs a signing key and a verification key. Besides the signature generation function SSIG.Sign, there is a signature simulation function SSIG.Sim that generates valid signatures by using the public-key and the

trapdoor generated by $\mathsf{SSIG.Crs}$. It is stressed that the simulated signatures must pass the verification by the verification function $\mathsf{SSIG.Vrf}$ but it is not demanded that they are indistinguishable from the real ones. Similarly, unforgeability is the standard unforgeability against chosen message attacks. In particular, the adversary is not given simulated signatures.

Any standard signature scheme can be turned into a simulatable one in an unconditional way as follows. Generate two key pairs by running the key generation algorithm twice independently. The first key pair is used as the CRS and the trapdoor while the second pair is used as the verification and signing key. Normal signing is done by using the second key. Simulation is done by the first key. A signature is accepted if it passes the original verification predicate with respect to either of the keys.

To fit to the other building blocks, SSIG must be able to sign group elements and the verification predicate must be represented as a product of pairings. For such a signature scheme a feasibility result based on DLIN assumption can be seen in [15].

6.3 The Scheme

The CRS generation function $\mathsf{BS.Crs}$ computes $(\Sigma_{\mathsf{wi}}, t_{\mathsf{wi}}) \leftarrow \mathsf{NIWI.Crs}(1^\lambda)$, $(\Sigma_{\mathsf{tc}}, t_{\mathsf{tc}}) \leftarrow \mathsf{TC.Key}(1^\lambda)$, and $(\Sigma_{\mathsf{ssig}}, t_{\mathsf{ssig}}) \leftarrow \mathsf{SSIG.Crs}(1^\lambda)$, and outputs $\Sigma = (\Sigma_{\mathsf{wi}}, \Sigma_{\mathsf{bc}}, \Sigma_{\mathsf{ssig}})$. Key generation function $\mathsf{BS.Key}$ is the same as $\mathsf{SIG.Key}$, which outputs vk and sk. The signature generation protocol is illustrated in Fig. 3. The proof system NIWI proves the following relation between witness $w = (s, c, z)$ and instance $x = (vk, \Sigma_{\mathsf{tc}}, \Sigma_{\mathsf{ssig}}, m)$:

$$\mathsf{TC.Vrf}(\Sigma_{\mathsf{tc}}, c, m, z) = 1 \wedge \mathsf{SSIG.Vrf}(\Sigma_{\mathsf{ssig}}, vk, c, s) = 1$$

Verification function $\mathsf{BS.Vrf}$ takes $((\Sigma_{\mathsf{wi}}, \Sigma_{\mathsf{tc}}, \Sigma_{\mathsf{ssig}}), vk, \sigma, m)$ as input and outputs $\varphi \in \{0, 1\}$ such that $\varphi \leftarrow \mathsf{NIWI.Vrf}(\Sigma_{\mathsf{wi}}, (vk, \Sigma_{\mathsf{tc}}, \Sigma_{\mathsf{ssig}}, m), \sigma)$.

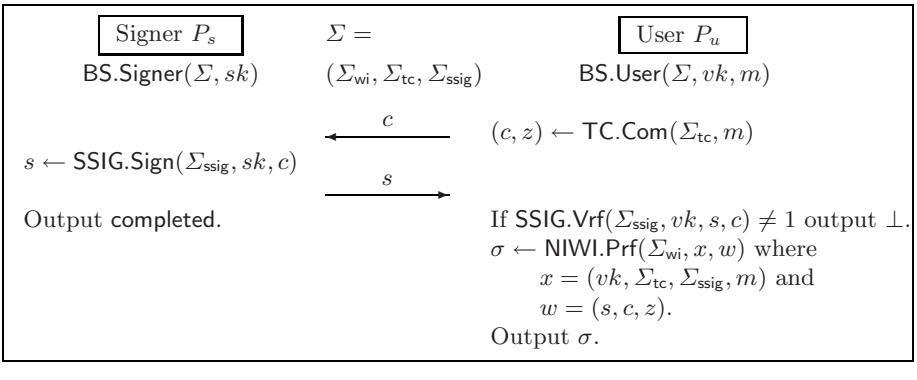

Fig. 3. Generic blind signature scheme $\mathsf{BS_S}$. The signature generation protocol.

Theorem 3. *Protocol* $\mathsf{Wrap}(\mathsf{BS}_\mathsf{S})$ *securely realizes* $\mathcal{F}_{\mathrm{ncb}}$ *in the* $\mathcal{F}_{\mathrm{crs}}$*-hybrid model with respect to adaptive adversaries without erasures.*

We claim that the scheme is session equivocal and signature equivocal. Observe that setting $t_1 = (t_{\mathsf{wi}}, t_{\mathsf{ssig}})$ and $t_2 = (t_{\mathsf{tc}})$ forms separated trapdoors. Session equivocality is proven by constructing SIM.User and SIM.SesState by using the trapdoor property of TC. Signature equivocality can be shown by constructing SIM.Sig and SIM.SigState by using the simulation property of SSIG and state reconstractability of NIWI. Thus from Lemma 3, we can say that the scheme is equivocal simulation blind. We then argue that the scheme is unforgeable due to the binding property of TC, the unforgeability of SSIG and the proof of knowledge property of NIWI. Finally Theorem 3 is applied to complete the proof of Theorem 2.

References

1. Abe, M., Ohkubo, M.: A framework for universally composable non-committing blind signatures. IACR ePrint Archive 2009 (2009)
2. Buan, A.B., Kråkmo, K.G.L.: Universally composable blind signatures in the plain model. IACR ePrint Archive 2006/405 (2006)
3. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145 (2001)
4. Canetti, R.: On universally composable notions of security for signature, certification and authentication. IACR ePrint Archive 2003/239 (2003)
5. Canetti, R.: Universally composable signatures, certification and authentication. In: 17th Computer Security Foundations Workshop, CSFW (2004); Revised version available in IACR ePrint archive 2003/239
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive 2000/067. 2nd version updated on 13 Dec (2005)
7. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
8. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
9. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO 1982, pp. 199–204. Prenum Publishing Corporation (1982)
10. Chaum, D.L.: Elections with unconditionally-secret ballots and disruptions equivalent to breaking RSA. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 177–182. Springer, Heidelberg (1988)
11. Fischlin, M.: Round-optimal composable blind signatures in the common reference model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
12. Fischlin, M., Schröder, D.: Security of blind signatures under aborts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 297–316. Springer, Heidelberg (2009)
13. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
14. Garay, J., Kiayias, A., Zhou, H.-S.: Sound and fine-grain specification of cryptographic tasks. IACR ePrint Archive 2008/132 (2008)

15. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
16. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
17. Hazay, C., Katz, J., Koo, C., Lindell, Y.: Concurrently-secure blind signatures without random oracles or setup assumptions. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 323–341. Springer, Heidelberg (2007)
18. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997)
19. Kiayias, A., Zhou, H.: Equivocal blind signatures and adaptive UC-security. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 340–355. Springer, Heidelberg (2008)
20. Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: STOC, pp. 683–692. ACM, New York (2003)
21. Lindell, Y.: Lower bounds and impossibility results for concurrent self composition. Journal of Cryptology 21(2), 200–249 (2008)
22. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 339–360 (2000)